UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO – SALAMANCA

DIVISIÓN DE INGENIERÍAS

Interacción del cuerpo humano con objetos virtuales

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRIA EN ADMINISTRACIÓN DE TECNOLOGÍAS

LÍNEA DE INVESTIGACIÓN: GESTIÓN Y DESARROLLO DE SISTEMAS DE INFORMACIÓN

PRESENTA:

I.S.C. Elías Salazar Martínez

DIRECTOR: Dr. Igor V. Guryev

CODIRECTOR: Dr. Juan Manuel López Hernández

YURIRIA, GTO ENERO, 2021

Campus Irapuato-Salamanca | División de Ingenierías

Asunto: *Notificación de Dictamen de Modalidad de Titulación*

C. ELÍAS SALAZAR MARTÍNEZ
Estudiante de la Maestría en Administración de Tecnologías
PRESENTE.

Una vez concluido el análisis y discusión sobre la propuesta del proyecto de tesis titulada: **"Interacción del cuerpo humano con objetos virtuales"** bajo la revisión del *Asesor(es): GURYEV Igor, Dr.* y *LÓPEZ HERNÁNDEZ Juan Manuel* en su tercera reunión extraordinaria del 12 de agosto del 2019, el comité académico de la Maestría en Administración de Tecnologías acordó la siguiente resolución:

> Con fundamento en las fracciones primera a cuarta del Artículo 67 y el Artículo 68 del Estatuto Académico 2008 y el Artículo 78 del Reglamento Académico 2019, este comité acordó **POR UNANIMIDAD DE VOTOS ACEPTAR SU PROPUESTA DE TESIS** designando el siguiente jurado:
>
> ***Presidente:*** *HERNÁNDEZ GÓMEZ Geovanni, Dr.*
> ***Secretario:*** *GURYEV Igor, Dr.*
> ***Vocal:*** *LÓPEZ RAMÍREZ Misael, Dr.*

Se extiende la presente notificación de la resolución el 19 de agosto del 2019.

A T E N T A M E N T E
"LA VERDAD OS HARÁ LIBRES"
DIRECTORA SUPLENTE

Rocio Lizárraga

Dra. ROCIO ALFONSINA LIZARRAGA MORALES

c.c.p. Secretaría Académica de la División de Ingenierías
GURYEV Igor, Dr. y LÓPEZ HERNÁNDEZ Juan Manuel, Dr. – Asesor(es)

DIVISIÓN DE INGENIRÍAS – CAMPUS IRAPUATO-SALAMANCA
Domicilio Conocido, Comunidad de Palo Blanco, Carretera Salamanca-Valle de Santiago km 3.5 +1.8
Slamanca, Gto., C.P. 36885, Tel(464) 647-9940, FAX ext. 2311

Campus Irapuato-Salamanca

Coordinación de Asuntos Escolares

Formato de Titulación

Llenar en computadora con ayuda del oficio de modalidad.

La modalidad de tesis es única para los posgrados

Nivel:	
Licenciatura	
Maestría	X
Doctorado	

Marcar con una X

Modalidad:	Tesis

Año:	2021

Poner el número de año p.e. 2015

Información sobre Obtención de Grado Académico:

Nombre	Elías Salazar Martínez
NUA	146734
Programa	Maestría en Administración de Tecnologías

Para modalidades con Jurado completar la siguiente información:

Lugar, hora y fecha de la presentación

Lugar		Firma y sello de autorización de reservación de lugar.
Hora	10:00 a.m.	
Fecha	29/01/2021	

Título del trabajo	Interacción del cuerpo humano con objetos virtuales

Jurado

	Nombre con grado académico completo: p.e. Doctor en Informática Industrial Nombre Apellido Paterno Apellido Materno	Firma de autorización para realización de examen de grado o titulación.
Presidente	Doctor en Ciencias (Óptica) Geovanni Hernández Gómez	
Secretario	Doctor en Ciencias Fisicomatemáticas con especialidad en Óptica y Física Láser Igor V. Guryev	
Vocal (1)	Doctor en Ingeniería Eléctrica Misael López Ramírez	
Vocal 2 (Doctorado)		
Vocal 3 (Doctorado)		

Asesoría

Director del trabajo	Dr. Igor V. Guryev
Codirector	Dr. Juan Manuel López Hernández

(No llenar para uso exclusivo de la Coordinación.)

Valida (nombre y firma): _____

Una vez terminado de llenar imprimir en dos tantos (uno para entregar al iniciar el trámite de autorización del examen de grado o titulación y otro para firma de recibido).

Yuriria, Gto., a 18 de Enero del 2021.

M. en I. HERIBERTO GUTIÉRREZ MARTIN
JEFE DE LA UNIDAD DE ADMINISTRACIÓN ESCOLAR
PRESENTE-

Por medio de la presente, se otorga autorización para proceder a los trámites de impresión, empastado de tesis y titulación al alumno(a) Elías Salazar Martínez del ***Programa de Maestría en*** Administración de Tecnologías y cuyo número de ***NUA*** es: 146734 del cual soy director. El título de la tesis es: Interacción del cuerpo humano con objetos virtuales

Hago constar que he revisado dicho trabajo y he tenido comunicación con los sinodales asignados para la revisión de la tesis, por lo que no hay impedimento alguno para fijar la fecha de examen de titulación.

ATENTAMENTE

NOMBRE Y FIRMA
DIRECTOR DE TESIS
SECRETARIO
Dr. Igor Guryev

NOMBRE Y FIRMA
DIRECTOR DE TESIS
Dr. Igor Guryev

NOMBRE Y FIRMA
PRESIDENTE
Dr. Geovanni Hernández Gómez

NOMBRE Y FIRMA
VOCAL
Dr. Misael López Ramírez

Dedicatoria

A mis padres y hermanos por siempre estar presentes en los momentos difíciles y ofrecer su apoyo incondicional.

A mis amigos y profesores que me acompañaron y apoyaron durante mis estudios de posgrado.

Agradecimientos

El presente trabajo pudo realizarse gracias al apoyo otorgado por el Consejo Nacional de Ciencia y Tecnología CONACYT, a través de la subdirección de becas nacionales con la contribución de la beca al CVU: **895024** asignada en la convocatoria con número de referencia **005058**.

A la Universidad de Guanajuato, por brindarme la oportunidad de pertenecer al Departamento de Estudios Multidisciplinarios sede Yuriria, perteneciente a la División de Ingenierías del Campus Irapuato-Salamanca; al permitirme realizar mis estudios de posgrado en la Maestría en Administración de Tecnologías.

A mi asesor de tesis, el Dr. Igor V Guryev, por brindarme la oportunidad de trabajar con él, por brindarme la orientación, herramientas y conocimientos necesarios para la elaboración de este proyecto de tesis.

A mi codirector de tesis, el Dr. Juan Manuel López Hernández, por brindar ayuda adicional en el desarrollo de este trabajo de investigación, por compartir material de apoyo, ideas, sugerencias y conocimientos.

Resumen

En el presente proyecto de investigación se propone una metodología para el desarrollo de un sistema computacional de reconocimiento de los movimientos del cuerpo humano, así como su interacción con objetos virtuales que pueden ser mostrados a través de la pantalla de una computadora o sobre una pared utilizando un proyector de imágenes. Para el reconocimiento del cuerpo humano, el sistema utiliza un Kinect v2.0 de las consolas de videojuegos Xbox ONE de Microsoft, asimismo, emplea técnicas de visión por computadora a través de la librería de código abierto de OpenCV, con la cual se crean los objetos virtuales con los que el usuario puede interactuar.

En primera instancia, el sistema desarrollado le permite a un usuario mover los objetos virtuales que se le muestran a través de la pantalla de la computadora al colocar y empuñar la mano sobre uno de ellos. Por otro lado, parte del desarrollo del sistema se realizó con la intención de que pueda ser implementado en los entrenamientos en un gimnasio con muros de escalada, en este caso, el funcionamiento del sistema consiste en validar que un escalador cumpla satisfactoriamente con una ruta de entrenamiento; utilizando un proyector de imágenes se le indica al escalador la ruta de entrenamiento a seguir al mostrar un objeto virtual sobre cada una de las rocas de escalada que necesita sujetar, cuando el escalador sujeta la roca de escalada indicada, el objeto virtual cambia de color, lo cual permite comprobar el desempeño que tuvo el escalador sobre el muro de escalada.

Índice General

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

1. Introducción.

La visión por computadora es un campo de la inteligencia artificial que permite obtener, procesar y analizar imágenes del mundo real con la finalidad de que puedan ser interpretadas por una computadora. Una de las áreas de estudio que comprende la visión por computadora es el reconocimiento de patrones de movimiento, el cual puede ser empleado para detectar y reconocer el movimiento de objetos o de seres vivos como el ser humano, como se muestra en [1], [2], donde se desarrolló un algoritmo para la detección y reconocimiento de patrones de movimiento del cuerpo humano por medio de un sistema de captura de movimientos al realizar tareas de manipulación de objetos, ya sea al levantar o aventar un objeto.

El reconocimiento de patrones de movimiento de personas puede aplicarse en diversas áreas, como por ejemplo: en sistemas de videovigilancia donde se busca detectar patrones de movimiento anómalos con el fin de prevenir situaciones de peligro [3], en el área médica donde se busca asegurar una mejor calidad en el proceso de rehabilitación de los pacientes con alguna discapacidad motora [4], o en el área deportiva, donde el reconocimiento de patrones de movimiento se ha utilizado para analizar el comportamiento del cuerpo humano en jugadores de basquetbol al realizar tiros libres [5].

Los avances tecnológicos que se han dado en los últimos años en las cámaras digitales con sensores de profundidad (RGB-D), como es el caso de Kinect de Microsoft, cuya función principal es el reconocimiento de los movimientos del cuerpo humano y su área de aplicación principal está orientada a las consolas de videojuegos, es un sistema versátil, rápido, de bajo costo, fácil instalación y completamente automático que permite el reconocimiento y reconstrucción en tiempo real del cuerpo completo de un ser humano [6]. Estas características han permitido que estudiantes e investigadores puedan realizar proyectos a un bajo costo [7] y extender su uso más allá de los videojuegos [8], [9].

En la industria de la moda, cada vez más empresas han optado por utilizar espejos inteligentes en sus tiendas de ropa, los cuales se utilizan como probadores virtuales, donde los clientes al situarse frente a uno de estos espejos, se les muestra un menú donde pueden

seleccionar por medio de sus manos, diversas combinaciones de ropa, lo cual les permite observar cómo lucirían vistiendo esa ropa sin necesidad de probarse físicamente cada combinación. Esta interacción humano-computadora se realiza por medio de la tecnología empleada por Kinect o dispositivos similares que detectan el cuerpo y los movimientos del usuario, asimismo emplean un sistema conocido como NUI (*Natural User Interface*), los cuales tienen la característica de permitir al usuario interactuar con algún dispositivo usando movimientos, señas, gestos y lenguajes cotidianos de la vida diaria.

1.1. Antecedentes.

Existen diversos proyectos de investigación que exploran la viabilidad de implementar sistemas NUI en distintas áreas utilizando Kinect como dispositivo principal, una de esas áreas es en el ámbito educativo, tal es el caso del artículo [10] donde se explora la idea de utilizar Kinect como tecnología interactiva y analizan como podría facilitar y mejorar el proceso de aprendizaje y enseñanza, sin embargo, Kinect no se puede utilizar por sí solo, requiere de ser integrado con otros dispositivos como una computadora, un proyector de imágenes y software compatible, asimismo, también es necesario disponer de suficiente espacio en el salón de clase y el tiempo de calibración de los dispositivos puede tomar un tiempo considerable de la clase para su inicialización.

En [11] se investiga el uso de Kinect para la creación de un pizarrón interactivo, empleando un bolígrafo de LED infrarrojo, un proyector de imágenes y una computadora. El proyector de imágenes muestra el escritorio de la computadora sobre la superficie de un tablero montado en la pared, y utilizando los datos de profundidad y el sensor infrarrojo de Kinect permiten localizar de manera precisa los elementos señalados con el bolígrafo de LED infrarrojo.

En el ámbito deportivo, se han explorado diversos métodos para evaluar el desempeño de escaladores en gimnasios de escalada en espacios cerrados, en los que se monitorean sus patrones de movimiento en los entrenamientos en muros de gimnasios de escalada, como el propuesto en [12], proyecto en el que se utilizan cinco sensores de medición inercial (IMU), los cuales están distribuidos en la pelvis, pies y manos del escalador, con el objetivo de obtener información útil acerca de su desempeño durante su ascenso por el muro de escalada.

Otros proyectos [13], [14] buscan hacer más entretenido, divertido e inmersivo el entrenamiento en muros de escalada al proyectar gráficos sobre los muros simulando estar dentro de un videojuego. Entre otras propuestas [15] está el utilizar una aplicación de realidad aumentada para teléfonos inteligentes, donde los escaladores pueden compartir con sus compañeros sus estilos, técnicas y desafíos de entrenamiento así como la ruta que siguieron para alcanzar la cima del muro de escalada.

Asimismo, se ha buscado mejorar el reconocimiento del movimiento de los escaladores al considerar su información anatómica [16], por la cual se puede tener una mejor interacción con un juego virtual proyectado sobre un muro de escalada, brindando información acerca de que parte del cuerpo está en contacto con la proyección de los objetos virtuales.

1.2. Planteamiento del Problema.

Los entrenamientos en gimnasios con muros de escalada en espacios cerrados requieren de una gran destreza, habilidad y técnica por parte de los escaladores, cualidades que obtienen a base de mucha practica y perseverancia. Una de las principales problemáticas a las que se pueden enfrentar los escaladores es la concurrencia de alumnos que puede haber en el gimnasio, lo cual puede provocar que al entrenador le tome tiempo adicional en atenderlos, ocasionando retrasos en las practicas. De igual manera, suele ser común que un escalador pueda olvidar la ruta que debe seguir para ascender por el muro de escalada, sobre todo si se trata de un escalador principiante en sus primeras clases. Contar con un sistema de entrenamiento asistido por computadora que le muestre al escalador la ruta a seguir y le pueda proveer retroalimentación en tiempo real acerca de su desempeño mediante la proyección de gráficos sobre el muro de escalada, podría ayudar a disminuir estas problemáticas.

1.3. Justificación

Con el presente proyecto de investigación se pretende desarrollar un sistema de bajo costo que integre el reconocimiento de patrones de movimiento de las extremidades del cuerpo humano y la interacción con objetos virtuales proyectados a través de un proyector de imágenes utilizando la cámara con sensores de profundidad de Kinect.

Con este sistema, se busca asistir a los escaladores en sus entrenamientos en el gimnasio con muros de escalada mediante la proyección de objetos virtuales sobre el muro de escalada, los cuales indican la ruta que deben seguir para completar una rutina de entrenamiento, de igual manera, el sistema brinda retroalimentación en tiempo real al escalador acerca de su desempeño en el muro de escalada al señalar, mediante un cambio de color del objeto virtual, las rocas que ha sujetado durante su acenso por el muro de escalada.

Como se ha visto anteriormente, existen algunos proyectos de investigación donde se propone el uso de sensores de medición inercial (IMU) [12] para monitorear el desempeño de los escaladores en gimnasios de muros de escalada, o utilizar la cámara integrada en los teléfonos inteligentes para generar patrones de escalada, empleando la realidad aumentada con el fin de hacer más eficiente y entretenido el entrenamiento en muros de escalada [15]. En el presente proyecto, el utilizar la cámara con sensores de profundidad de Kinect permitirá reconocer los patrones de movimiento de los escaladores sin necesidad de recurrir a hardware adicional, como sensores de medición inercial (IMU) u otro tipo de sensores de movimiento, asimismo el escalador podrá observar que ruta debe seguir a través de la proyección de los gráficos en el muro de escalada sin depender de un asistente personal o entrenador que le tenga que indicar constantemente donde debe colocar sus manos y piernas.

1.4. Objetivos de Investigación.

1.4.1. Objetivo General:

Desarrollar un sistema automatizado de reconocimiento de patrones de movimiento de las extremidades del cuerpo humano, implementando tecnologías de visión por computadora e integrando componentes de hardware como cámaras digitales con sensores de profundidad, proyectores de imágenes; y software con la finalidad de detectar y reconocer la interacción del cuerpo humano con objetos virtuales.

1.4.2. Objetivos Específicos:

- Desarrollar un algoritmo de reconocimiento de patrones de movimiento de las extremidades del cuerpo humano.
- Identificar mediante software las posiciones relativas entre las extremidades del cuerpo humano y objetos virtuales en diferentes condiciones de iluminación.

- Desarrollar algoritmos de detección de colisiones para analizar la interacción del cuerpo humano con objetos virtuales.
- Probar el sistema de reconocimiento de patrones de movimiento del cuerpo humano y su interacción con objetos virtuales en un gimnasio con muros de escalada.

1.5. Descripción del documento.

El presente documento está estructurado en cuatro capítulos: en el capítulo 1 se hace una introducción describiendo algunos proyectos relacionados con el área de estudio de la visión por computadora y proyectos similares al expuesto en este trabajo de investigación; se exponen las razones que motivaron la realización del proyecto expuesto en este documento, así como los objetivos general y específicos de la investigación.

En el capítulo 2 se presenta la metodología del proyecto de investigación donde se describe el proceso general para desarrollar el sistema que permite la interacción del cuerpo humano con objetos virtuales, explicando en cada fase del proceso, los fundamentos teóricos necesarios para comprender las diversas secciones que conforman el capítulo.

En el capítulo 3 se describen los experimentos realizados para evaluar el sistema desarrollado que permite la interacción del cuerpo humano con los objetos virtuales, asimismo se presentan los resultados obtenidos de las pruebas realizadas en tres entornos experimentales: cubículo de oficina, salón de clase y en el gimnasio de escalada.

Finalmente, en el capítulo 4 se presentan las conclusiones, alcances y limitaciones del proyecto de investigación, de igual manera, se exponen los lineamientos a seguir en un trabajo futuro para mejorar el sistema de interacción del cuerpo humano con objetos virtuales aplicado a los entrenamientos en muros de gimnasio de escalada.

CAPÍTULO 2

2. Metodología.

El objetivo de este capítulo es presentar el procedimiento general que se ha propuesto en este proyecto de investigación para el desarrollo e implementación de un sistema de reconocimiento de los movimientos del cuerpo humano y su interacción con objetos virtuales.

El sistema desarrollado tiene como propósito probar y analizar la interacción del cuerpo humano con objetos virtuales que se visualizan a través del monitor de una computadora o por medio de un proyector de imágenes. La interacción con estos objetos se realiza por medio de las manos del usuario.

Asimismo, el desarrollo del sistema giró en torno a la idea de que pueda ser implementado en un gimnasio con muros de escalada, donde el escalador pueda observar, mediante la proyección de objetos virtuales sobre el muro de escalada, cuáles son las rocas de escalada o *climbing holds* que debe sujetar para completar una ruta de entrenamiento previamente establecida por su entrenador. El esquema de la arquitectura del sistema propuesto para este propósito se muestra en la Figura 2.1. En general, el funcionamiento del sistema consiste en reconocer el momento en que el escalador sujeta o interactúa con una *climbing hold* iluminada con un objeto virtual, indicando mediante un cambio de color las *climbing holds* que ha sujetado para completar la ruta de entrenamiento.

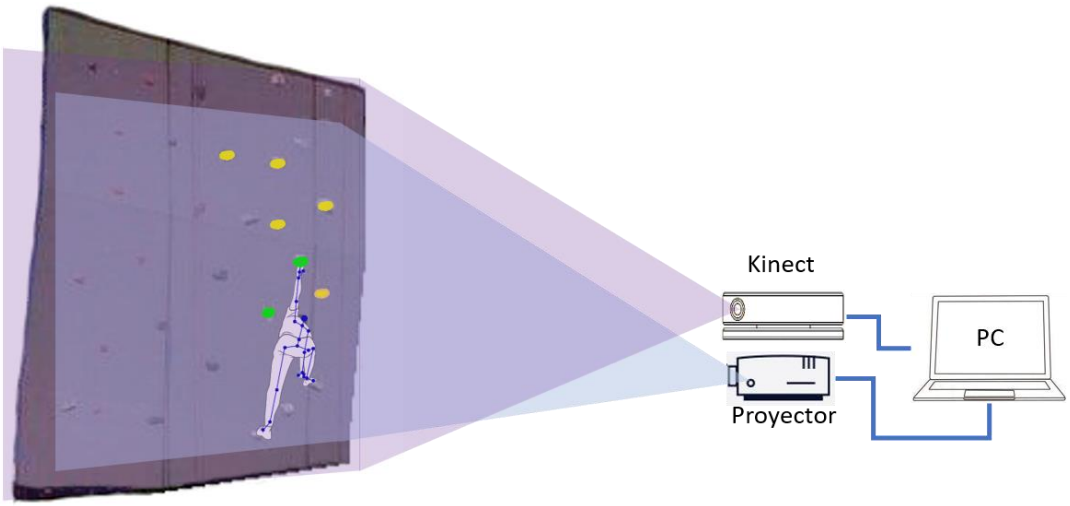


Figura 2.1. Esquema de la arquitectura del sistema propuesto para un gimnasio de escalada.

El sistema computacional se desarrolló en el entorno de programación de Microsoft Visual Studio 2017, utilizando C++ como lenguaje de programación y como dispositivo principal para llevar a cabo el reconocimiento de los movimientos del cuerpo humano se utilizó un Kinect v2.0 de Microsoft [17]. Asimismo, se utilizó el kit de desarrollo de software (Software Development Kit) Kinect for Windows SDK 2.0 [17], la librería NtKinect [18] y la librería de código abierto empleada para desarrollar aplicaciones de visión por computadora OpenCV v4.0 [19]. En cuanto a los dispositivos para la proyección de los objetos virtuales sobre el muro de escalada, se emplearon dos proyectores de imágenes, uno de los cuales fue un proyector de bolsillo Philips PicoPix, cuya resolución de imagen es de 640x360 píxeles, con una potencia de brillo de hasta 40 lúmenes y conexión USB; el segundo proyector, fue un mini proyector portátil GP9 PONER SAUND, con una potencia de brillo de 2400 lúmenes, entrada HDMI y con resolución nativa de 720p (1280x720).

En la Figura 2.2 se muestra el procedimiento general que se ha propuesto en este proyecto de investigación para realizar el reconocimiento de los movimientos del cuerpo humano y su interacción con objetos virtuales que son visualizados en tiempo real a través del monitor de una computadora o un proyector de imágenes.

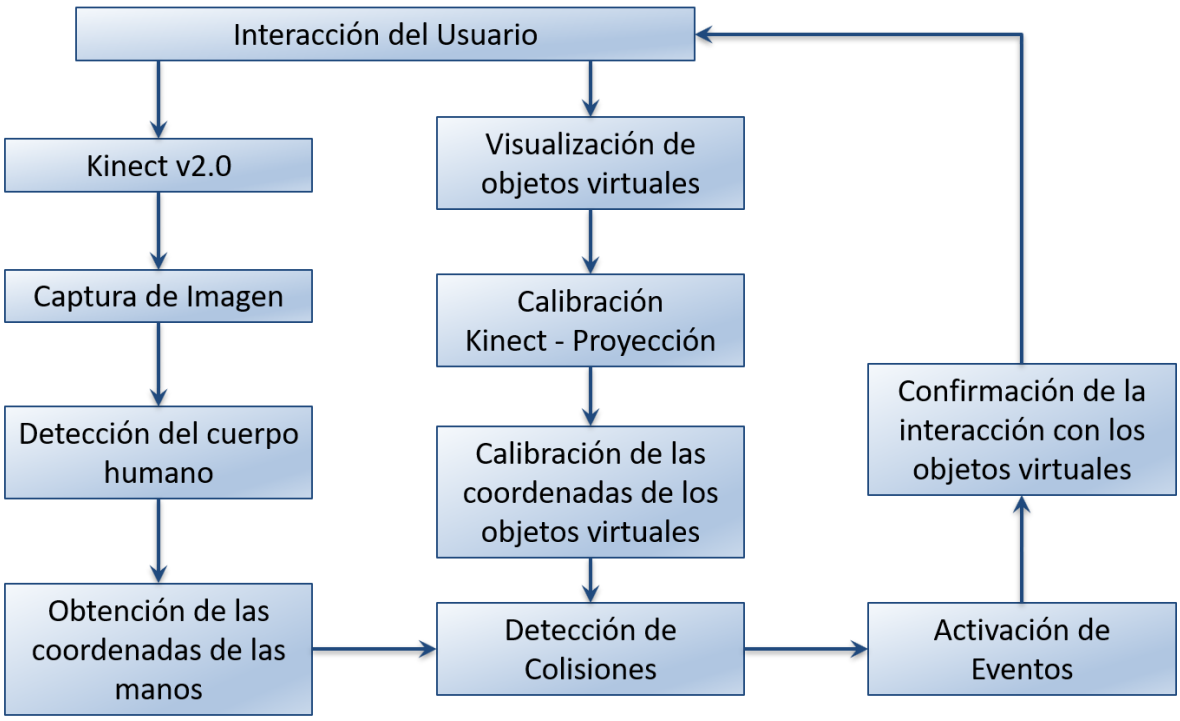
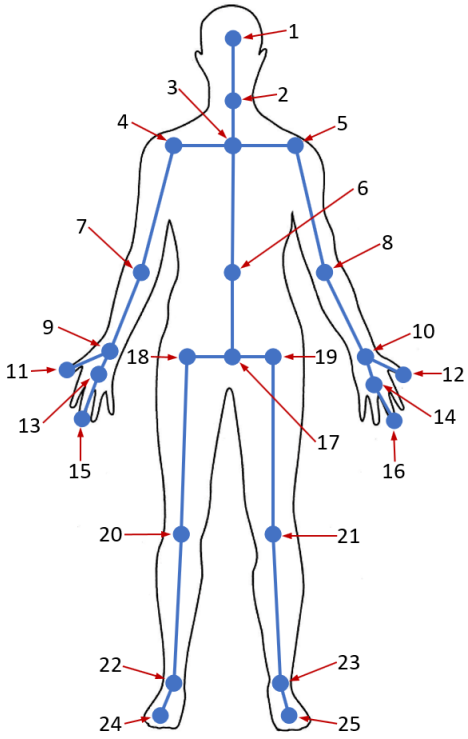


Figura 2.2. Proceso general de la interacción del cuerpo humano con objetos virtuales.

2.1. Interacción del Usuario

La interacción del cuerpo humano con objetos virtuales se realiza utilizando un Kinect v2.0 para Windows, el cual, tiene la capacidad de monitorear hasta seis personas de manera simultánea dentro de su campo de visión y puede detectar hasta 25 articulaciones para cada una de ellas, las articulaciones que Kinect puede reconocer se muestran en la Figura 2.3.



1. Cabeza.
2. Cuello.
3. Espina dorsal superior (altura de los hombros).
4 - 5. Hombro derecho - izquierdo.
6. Espina dorsal media.
7 - 8. Codo derecho - izquierdo.
9 - 10. Muñeca derecha - izquierda.
11 - 12. Pulgar derecho - izquierdo.
13 - 14. Mano (palma) derecha - izquierda.
15 - 16. Dedo medio derecho - izquierdo.
17. Espina dorsal inferior (altura de la cintura).
18 - 19. Cintura derecha - izquierda.
20 - 21. Rodilla derecha - izquierda.
22 - 23. Tobillo derecho – izquierdo.
24 - 25. Pie derecho – izquierdo.

Figura 2.3. Articulaciones del cuerpo humano que Kinect puede reconocer.

El cuerpo del usuario puede ser detectado por Kinect estando de pie o sentado. La distancia óptima para una detección correcta del cuerpo humano por medio de Kinect es de 0.5 metros como mínimo hasta 4.5 metros como máximo, el usuario no será reconocido si se posiciona fuera de ese rango de visión de Kinect.

Para interactuar con el sistema el usuario se debe posicionar frente al dispositivo Kinect, ya sea de pie o sentado a la distancia antes mencionada de 4.5 metros como máximo, por lo tanto, la ubicación de los dispositivos con respecto al muro en el gimnasio de escalada no debe superar dicha distancia (Figura 2.4).

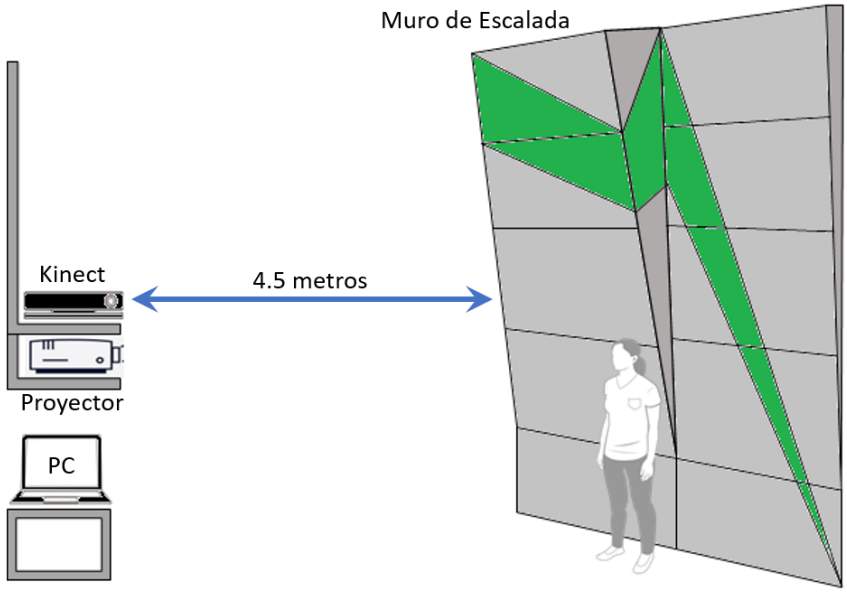


Figura 2.4. Ubicación de los dispositivos respecto al muro en el gimnasio de escalada.

2.2. Kinect v2.0

Kinect es un dispositivo de detección de movimiento creado por Microsoft para los juegos de la consola XBOX ONE y las computadoras personales con Windows. La versatilidad de Kinect le permite ver los movimientos de un cuerpo humano completo, así como detectar pequeños gestos hechos con las manos, algunos gestos faciales y recibir comandos de voz.

El sensor de Kinect provee cuadros de imagen a color desde su cámara RGB (*Red, Green, Blue* – Rojo, Verde, Azul), posee un emisor infrarrojo que, en conjunto a un sensor de profundidad, puede medir la profundidad de las imágenes capturadas a una resolución milimétrica. Tiene un arreglo de cuatro micrófonos que transfiere datos de audio hacia el kit de desarrollo SDK [17], en la Figura 2.5 se muestra la ubicación de los componentes básicos de Kinect.

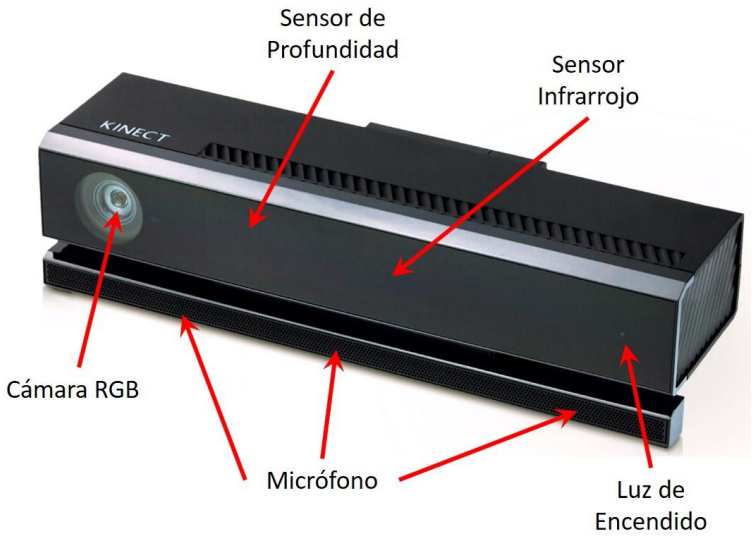


Figura 2.5. Componentes básicos de Microsoft Kinect v2.0

Los requisitos de Hardware necesarios para desarrollar aplicaciones para Kinect pueden variar en gran medida de acuerdo a lo que se quiere desarrollar y a la complejidad del procesamiento de imagen a realizar, sin embargo, como requisitos mínimos recomendados podemos listar las siguientes especificaciones [20]:

- Sistema Operativo de 64 bits, procesador x64.
- CPU con procesador de doble núcleo a 3.1 GHz.
- 4 GB de RAM
- Puerto USB 3.0 integrado.
- Tarjeta gráfica con soporte para DirectX 11
- Sistema operativo Windows 8, 8.1 o Windows 10.
- Adaptador de Kinect para Windows PC (Figura 2.6).

Figura 2.6. Adaptador de Kinect para Windows PC.

2.3. Captura de imagen

La cámara RGB de Kinect puede adquirir imágenes a 30 cuadros por segundo (FPS, *frames per second*) con una resolución de 1920x1080 pixeles y tiene un ángulo de visión en horizontal de 70° y en vertical de 60° [17]. Debido a que la librería de OpenCV utiliza el formato BGR o BGRA por default, se ha optado por utilizar la librería NtKinect [18] que cuenta con funciones que hace la conversión de formato de BGRA a RGB de forma automática sin necesidad de que el programador desarrolle código adicional. Por lo tanto, para capturar las imágenes obtenidas con la cámara RGB de Kinect se utilizó la función *setRGB()* de NtKinect, la cual almacena las características de la imagen capturada por medio de la variable *rgbImage*, cuyos parámetros se describen en la Tabla 2.1 [21].

Tabla 2.1. *rgbImage* variable miembro de NtKinect para la imagen RGB de Kinect.

Tipo	Nombre	Descripción

cv::Mat	rgbImage	Contiene los parámetros para manipular la imagen RGB de la cámara de Kinect con una resolución de 1920x1080.
		Parámetros utilizados:
		• rgbImage.cols: Resolución en la dirección horizontal de la imagen (1920)
		• rgbImage.rows: Resolución en la dirección vertical de la imagen (1080)
		• rgbImage.at<cv::Vec4b>(y , x) --- Accesa al pixel en la coordenada (x , y) de la imagen.
		
		`cv::Vec4b pixel = rgbImage.at<cv::Vec4b>(y,x);`
		`pixel[0] // Azul`
		`pixel[1] // Verde`
		`pixel[2] // Rojo`
		`pixel[3] // Alpha`

A razón de que Kinect está diseñado para que el usuario interactúe con el dispositivo situándose frente a él y frente a la pantalla de la computadora, Kinect adquiere los datos de la detección del usuario e imagen invirtiendo el eje ***x***, es decir, captura las imágenes como si se tratará del reflejo en un espejo, si levantamos la mano derecha, en la imagen reflejada capturada por la cámara de Kinect, estaremos levantando la mano izquierda. Por consiguiente, al utilizar un proyector para realizar una proyección de la imagen adquirida, esta imagen se mostrará invertida (Figura 2.7). Este es un aspecto importante que considerar

al momento de adquirir las coordenadas de los objetos virtuales y detectar las interacciones con el cuerpo humano cuando se utiliza un proyector de imágenes.

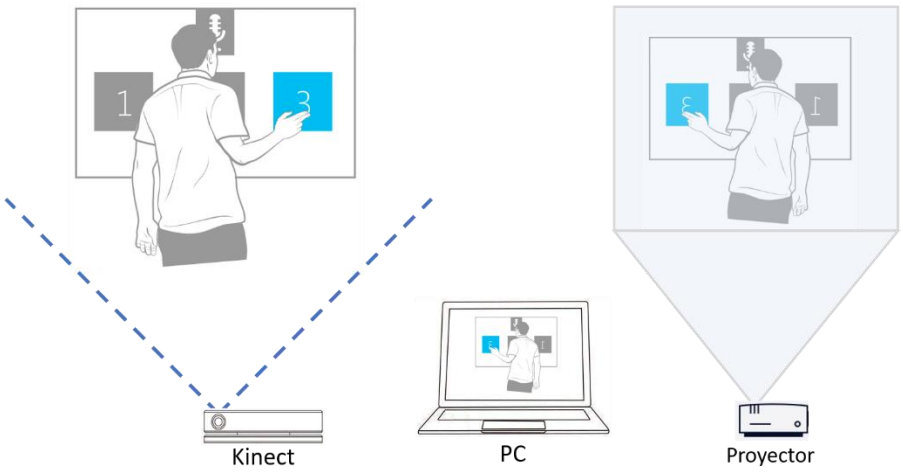


Figura 2.7. Captura de imágenes con Kinect v2.0.

2.4.Detección del cuerpo humano

La detección del cuerpo humano se realiza por medio del sensor de profundidad y el sensor infrarrojo de Kinect. El sensor de profundidad incorpora una cámara ToF (*Time of Flight* - Tiempo de Vuelo), esta cámara tiene la capacidad de medir el tiempo que tarda la luz infrarroja emitida por Kinect en alcanzar un objeto y regresar al origen al ser reflejada por el objeto alcanzado. Obteniendo este tiempo de vuelo y considerando la velocidad de la luz emitida por el emisor infrarrojo, Kinect puede calcular la distancia existente entre sí mismo y los elementos que se encuentran dentro de su rango de visión (Figura 2.8) [20], [22].

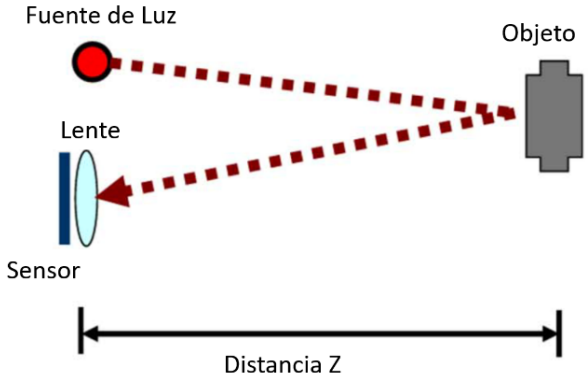


Figura 2.8. Funcionamiento sensor de profundidad de Kinect.

Con los datos resultantes, Kinect crea una imagen de profundidad (En la Figura 2.9 se muestra un ejemplo de una imagen de profundidad tomada con Kinect [20]) con una resolución de 512x424 pixeles que posteriormente puede ser utilizada para detectar y monitorear el cuerpo humano y sus movimientos.

Figura 2.9. Imagen de profundidad creada con Kinect.

Es importante mencionar que, aunque el rango de la distancia medible del sensor de profundidad es de 0.5 hasta 8 metros, un ser humano solo puede ser detectado si se encuentra en un rango de 0.5 a 4.5 metros [17].

Como se ha mencionado anteriormente, el sensor de profundidad puede detectar hasta 6 personas simultáneamente con 25 articulaciones cada una estando de pie (Figura 2.3) y 10 articulaciones estando sentadas.

Cuando Kinect detecta un cuerpo humano, obtiene diversos datos de las articulaciones que son representadas por la estructura de datos *Joint*, esta estructura de datos forma parte de la librería Kinect.h que pertenece a Kinect for Windows SDK v2.0 y se compone de 3 variables miembro (Tabla 2.2), las cuales se describen a continuación:

- Joint_Type: Tipo o nombre de la articulación (Tabla 2.3).
- Position: Coordenadas en 3D que representa la posición de la articulación.

- TrackingState: Valor empleado para indicar el estado de seguimiento (*tracking*) de la articulación: sin seguimiento (*NotTracked*), Seguimiento inferido (*Inferred*) y realizando seguimiento (*Tracked*) (Tabla 2.4).

Tabla 2.2. Declaración de la estructura de datos *Joint* en la librería Kinect.h.

Estructura de datos *Joint*

```
typedef struct _Joint {
    JointType JointType;
    CameraSpacePoint Position;
    TrackingState TrackingState;
} Joint;
```

Tabla 2.3. Declaración de *JointType* variable miembro de *Joint* en la librería Kinect.h.

Variable miembro *JointType*

```
enum _JointType {
    JointType_SpineBase= 0,          JointType_KneeLeft= 13,
    JointType_SpineMid= 1,           JointType_AnkleLeft= 14,
    JointType_Neck= 2,               JointType_FootLeft= 15,
    JointType_Head= 3,               JointType_HipRight= 16,
    JointType_ShoulderLeft= 4,       JointType_KneeRight= 17,
    JointType_ElbowLeft= 5,          JointType_AnkleRight= 18,
    JointType_WristLeft= 6,          JointType_FootRight= 19,
    JointType_HandLeft= 7,           JointType_SpineShoulder= 20,
    JointType_ShoulderRight= 8,      JointType_HandTipLeft= 21,
    JointType_ElbowRight= 9,         JointType_ThumbLeft= 22,
    JointType_WristRight= 10,        JointType_HandTipRight= 23,
    JointType_HandRight= 11,         JointType_ThumbRight= 24,
    JointType_HipLeft= 12,           JointType_Count= ( JointType_ThumbRight + 1 )
                                 };
```

Tabla 2.4. Declaración de *TrackingState* variable miembro de *Joint* en la librería Kinect.h.

Variable miembro *TrackingState*

```
enum _TrackingState {
    TrackingState_NotTracked= 0,
    TrackingState_Inferred= 1,
    TrackingState_Tracked= 2
};
```

Para acceder a la información de las articulaciones en la estructura de datos *Joint* por medio de la librería NtKinect se puede utilizar la función *setSkeleton()*, la cual asigna los datos de las articulaciones *Joint* a las variables miembro: *skeleton*, *skeletonId*, *skeletonTrackingId*, cuyo funcionamiento se describe en la Tabla 2.5.

Tabla 2.5. Funcionamiento de las variables miembro de NtKinect referentes al cuerpo humano.

Tipo	Variable miembro	Descripción
vector<Joint> vector<vector<Joint>>	skeleton	Vector que almacena la información del cuerpo humano. La información del conjunto de articulaciones del cuerpo se define como tipo vector<Joint>. Para controlar la información de múltiples personas se define un vector de vectores de la forma: vector<vector<Joint>> . Componentes de la variable miembro *skeleton* : • skeleton.size(): Devuelve el número de personas que se han detectado. • skeleton[*X*]: Información del cuerpo de *X* persona. • skeleton[*X*].size(): Número de articulaciones detectadas de *X* persona. • skeleton[*X*][*jointType*]: Obtiene información de una articulación en específico (*jointType*) de *X* persona. • skeleton[*X*][*jointType*].Position: Obtiene las coordenadas de una articulación en específico (*jointType*) de *X* persona. • skeleton[*X*][*jointType*].TrackingState: Estado de seguimiento de una articulación en específico (*jointType*) de *X* persona.
vector<int>	skeletonId	Vector de índices de los cuerpos correspondientes a las personas (*skeletons*). El indice de skeleton[*X*] es almacenado en skeletonId[*index*].
vector<UINT64>	skeletonTrackingId	Vector de identificadores de seguimiento de las personas (*skeletons*). El identificador de seguimiento de skeleton[*X*] es almacenado en skeletonTrackingId[*index*].

2.5. Obtención de las coordenadas de las manos

Como pudimos observar en la Figura 2.5, la posición de la cámara RGB, la cámara de profundidad y el emisor infrarrojo es diferente, asimismo, como hemos visto en la sección 2.3 y 2.4, la cámara RGB captura imágenes a una resolución de 1920x1080 pixeles, mientras que la cámara de profundidad captura imágenes a una resolución de 512x424 pixeles. Debido a estas diferencias, Kinect maneja tres sistemas de coordenadas (Figura 2.10) [20], [21], [23]:

ColorSpace: Sistema de coordenadas en dos dimensiones (2D) usado por la cámara RGB, con la que obtiene imágenes a color RGB. El punto en el eje de coordenadas ***(x, y)*** donde se encuentra el valor RGB de la imagen se conoce como *ColorSpacePoint.*

DepthSpace: Sistema de coordenadas en dos dimensiones (2D) usado por la cámara de profundidad, con la que obtiene imágenes de profundidad e imágenes del espectro infrarrojo. El punto en el eje de coordenadas ***(x, y)*** donde se encuentra el valor de profundidad de la imagen se conoce como *DepthSpacePoint.*

CameraSpace: Sistema de coordenadas en tres dimensiones (3D) que Kinect utiliza para conocer la posición de las articulaciones del cuerpo humano. El punto en el eje de coordenadas ***(x, y, z),*** donde se ubica la información de la articulación del cuerpo humano se conoce como *CameraSpacePoint*, donde ***x, y*** es la posición de las articulaciones en el plano en dos dimensiones de la imagen obtenida por la cámara de profundidad, y ***z*** es el valor de profundidad de las articulaciones obtenida por el sensor de profundidad, esto es, la distancia existente entre Kinect y el cuerpo humano (Sección 2.4).

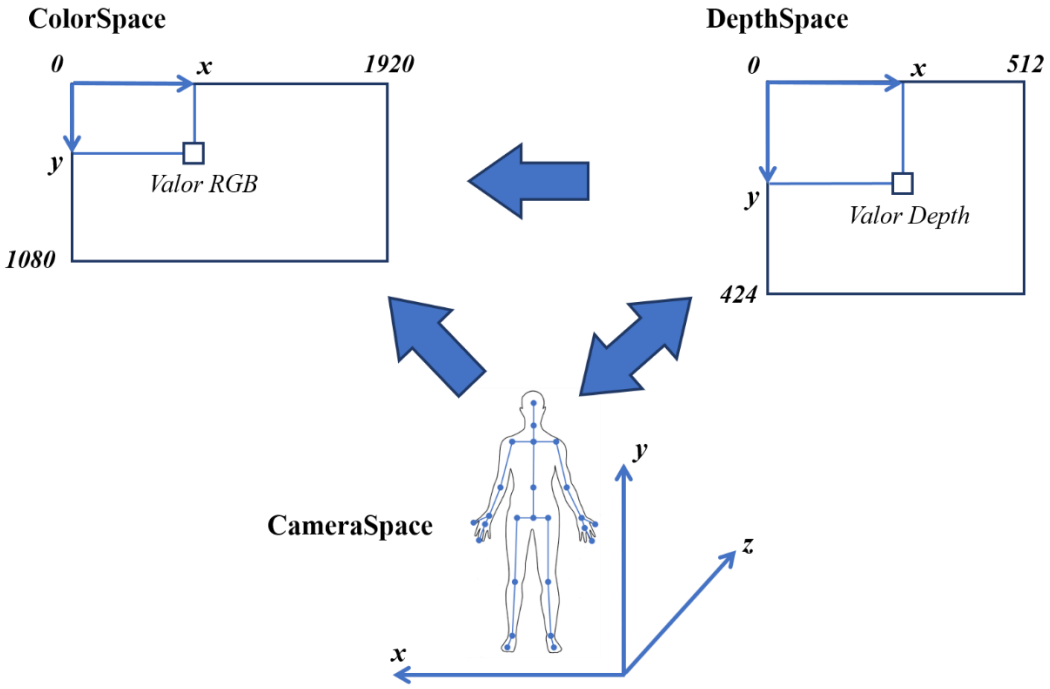


Figura 2.10. Sistema de coordenadas de Kinect v2.0

Derivado de lo anterior, resulta evidente que no es posible obtener las coordenadas de las articulaciones del cuerpo humano empleando únicamente la cámara RGB de Kinect, debido a que es la cámara de profundidad junto al emisor infrarrojo quienes generan los datos de posición de las articulaciones en un sistema de tres dimensiones, mientras que la cámara RGB maneja los datos en un plano de dos dimensiones; asimismo, hemos visto que no hay

una concordancia en la resolución de las imágenes que pueden capturar ambas cámaras, por lo tanto, para conocer la posición que tendrían las articulaciones del cuerpo humano en un plano bidimensional y de esta manera, poder visualizarlas a través de la cámara RGB, es necesario realizar una conversión del sistema de coordenadas de *CameraSpace* al sistema *ColorSpace*.

Kinect for Windows SDK v2.0 contiene funciones de conversión que permiten mapear o alinear los puntos de un sistema de coordenadas a otro. Estas funciones son miembros de la clase *ICoordinateMapper* [20] y se describen en la Tabla 2.6.

Tabla 2.6. Funciones de conversión entre sistemas de coordenadas de Kinect v2.0.

Tipo de Valor de Retorno	**Función**	**Descripción**
HRESULT	MapCameraPointToColorSpace(CameraSpacePoint *sp* , ColorSpacePoint **cp*)	Convierte las coordenadas *sp* en *CameraSpace* a las coordenadas *cp* en *ColorSpace* . El valor de retorno es S_OK o un código de error.
HRESULT	MapCameraPointToDepthSpace(CameraSpacePoint *sp* , DepthSpacePoint **dp*)	Convierte las coordenadas *sp* en *CameraSpace* a las coordenadas *dp* en *DepthSpace* . El valor de retorno es S_OK o un código de error.
HRESULT	MapDepthPointToColorSpace(DepthSpacePoint *dp* , UINT16 *depth* , ColorSpacePoint **cp*)	Convierte las coordenadas *dp* en *DepthSpace* y distancia *depth* a las coordenadas *cp* en *ColorSpace* . El valor de retorno es S_OK o un código de error.
HRESULT	MapDepthPointToCameraSpace(DepthSpacePoint *dp* , UINT16 *depth* , CameraSpacePoint **sp*)	Convierte las coordenadas *dp* en DepthSpace y distancia *depth* a las coordenadas *sp* en *CameraSpace* . El valor de retorno es S_OK o un código de error.

Una instancia de la clase *ICoordinateMapper*, empleada para mapear los sistemas de coordenadas, está incluida como variable miembro de la librería NtKinect llamada *coordinateMapper*.

Al principio de esta sección hemos visto que las coordenadas de las articulaciones del cuerpo humano se encuentran en el sistema de coordenadas *CameraSpace*, por lo tanto, para poder visualizarlas por medio de la cámara RGB, es necesario convertir el sistema de coordenadas al sistema *ColorSpace*, de esta manera, también es posible obtener la posición de las articulaciones del cuerpo humano correspondientes a las coordenadas ***(x, y)*** de la imagen RGB.

Ahora solo es necesario distinguir cuales coordenadas corresponden a las articulaciones de las manos, para esto, utilizamos la función *setSkeleton()* que obtiene la información de las articulaciones del cuerpo humano y la almacena en la variable miembro *skeleton* de NtKinect, la cual es un vector de tipo *Joint*, estructura que a su vez tiene como variable miembro a *JointType* que contiene el tipo o nombre de las articulaciones, donde *JointType_HandLeft* corresponde a la mano izquierda y *JointType_HandRight* corresponde a la mano derecha (ver Tabla 2.3 de la Sección 2.4). En la Figura 2.1 se muestra una detección de las articulaciones realizada con Kinect, marcando con un color distinto las articulaciones correspondientes a las manos.

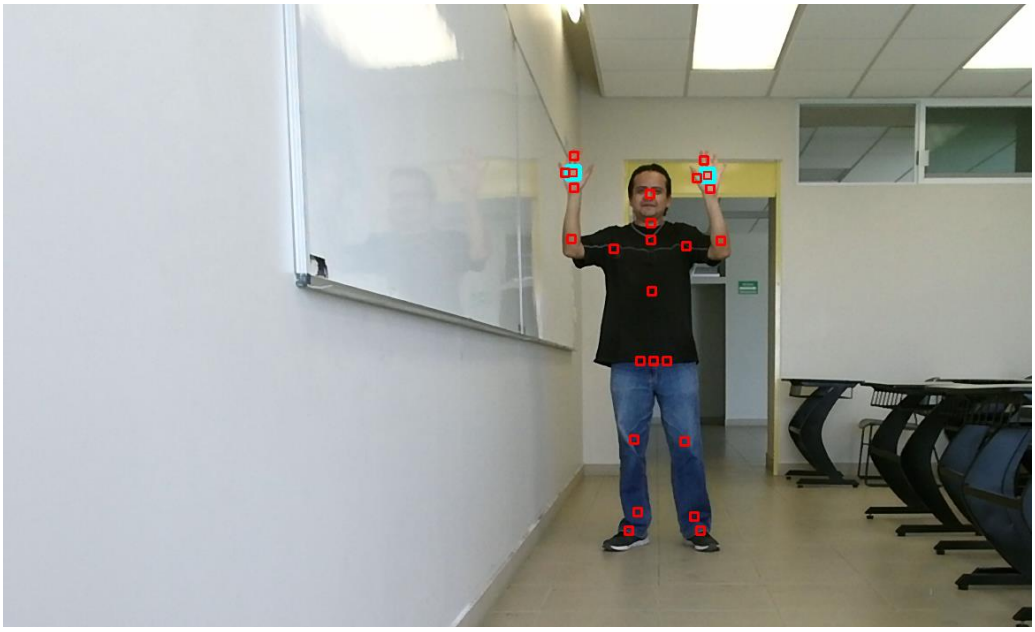


Figura 2.11. Obtención de las coordenadas de las articulaciones de las manos con Kinect v2.0.

2.6. Visualización de objetos virtuales

La creación de los objetos virtuales se puede realizar utilizando la librería de Vision por Computadora OpenCV y posteriormente visualizarlos a través de la pantalla de la computadora, o bien, visualizarlos en una pared o en un muro por medio de un proyector de imágenes.

2.6.1. Visualización de objetos virtuales a través de la pantalla de la computadora.

En el presente proyecto, la visualización de los objetos virtuales a través de la pantalla de computadora se realizó por medio de la creación de círculos utilizando la función *circle()* de OpenCV, cuyos parámetros y funcionamiento se describen en la Tabla 2.7 [24].

Tabla 2.7. Función *circle* de OpenCV.

Función	**Tipo de Variable**	**Parametros**
void cv::circle	(InputOutputArray	img,
	Point	center,
	int	radius,
	const Scalar &	color,
	int	thickness = 1, (Opcional)
	int	lineType = LINE_8, (Opcional)
	int	shift = 0 (Opcional)
	)	
Uso		
cv::circle = (img, center, radius, color[, thickness[, lineType[, shift]]])		
Parametros	**Descripción**	
img	Imagen donde se creará el círculo.	
center	Punto central del círculo. Point(*x*,*y*)	
radius	Radio del círculo.	
color	Color del círculo.	
thickness	Grosor del contorno del círculo (si el valor es positivo). Si el valor es negativo o se usa *FILLED*, se creará un círculo lleno.	
lineType	Tipo de línea del contorno del círculo.	
shift	Número de bits fraccionarios en las coordenadas del centro y en el valor del radio.	

Los círculos se dibujan sobre *rgbImage* (ver Sección 2.3), que es la representación de la imagen RGB obtenida con la cámara RGB de Kinect, dando como resultado la imagen que se muestra en la Figura 2.12.

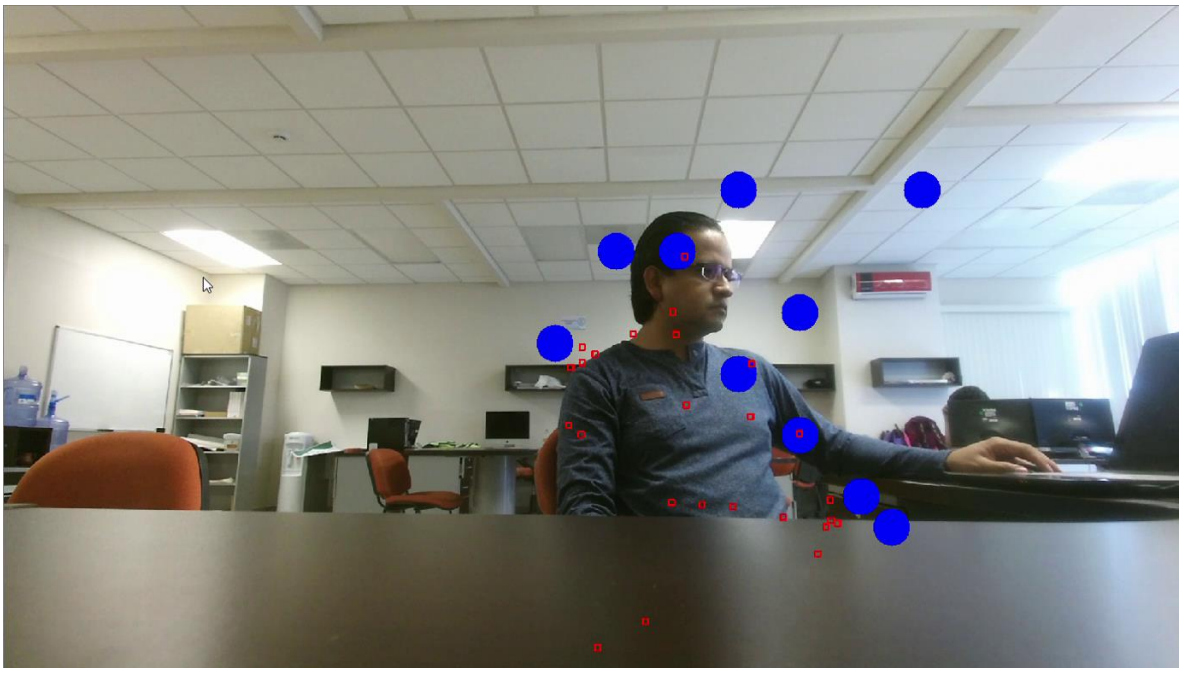


Figura 2.12. Creación de círculos como objetos virtuales sobre la imagen RGB de Kinect.

2.6.2. Visualización de objetos virtuales utilizando un proyector de imágenes.

Uno de los objetivos de este proyecto de investigación es probar la interacción del cuerpo humano con objetos virtuales en un gimnasio con muros de escalada, por este motivo, la visualización de objetos virtuales por medio de un proyector de imágenes consiste en mostrar sobre un muro de escalada, una representación virtual por medio de imágenes de las rocas de escalada o *climbing holds* con la finalidad de indicar al escalador cuales *climbing holds* debe sujetar para completar una ruta de entrenamiento.

Para indicar la ruta que debe seguir el escalador, mediante el proyector de imágenes se muestra una *climbing hold* virtual sobre la *climbing hold* real que el escalador debe sujetar, al sujetarla, la *climbing hold* virtual cambia de color, esto con el objetivo de indicar al escalador si ha completado exitosamente la ruta de entrenamiento, o si el escalador cae, él o el entrenador puedan observar cuantas *climbing holds* le faltaron por sujetar para concluir la ruta.

A diferencia del método de visualizar los objetos virtuales a través de la pantalla de la computadora donde los círculos se dibujan sobre la imagen RGB de Kinect, en la

visualización de objetos virtuales con el proyector de imágenes, las *climbing holds* virtuales se dibujan en una imagen independiente a la imagen RGB que captura la cámara RGB de Kinect. Esta imagen se crea a partir de la detección del área de proyección del proyector de imágenes utilizando diversas técnicas de procesamiento y segmentación de imágenes con OpenCV; primero se aplica una sustracción de fondo, el cual es un método de procesamiento de imágenes que consiste en detectar cambios en una secuencia de imágenes realizando una resta entre ellas con el fin de detectar la localización de un objeto o identificar si este objeto ha cambiado de posición [25]; en la Figura 2.13 se tiene una imagen sin proyección (a) y una imagen con una proyección (b) en blanco para resaltar el área de proyección, al aplicar la sustracción de fondo entre ambas imágenes se obtiene la ubicación del área de proyección (c).

Figura 2.13. Sustracción de fondo de imágenes. (a) Imagen sin proyección, (b) Imagen con proyección y (c)Resultado de aplicar la sustracción de fondo a las imágenes (a) y (b).

Posteriormente se aplica una umbralización a la imagen resultante de la sustracción de fondo (imagen (c) de la Figura 2.13Figura 2.2). La umbralización es un método de segmentación de imágenes que consiste en resaltar los bordes y contornos de los objetos mediante la creación de una imagen binaria, la cual se obtiene al analizar cada uno de los pixeles de una imagen de entrada para reemplazarlos por un pixel negro o blanco de acuerdo a un valor de intensidad constante predeterminado, si la intensidad del pixel de la imagen de entrada es menor al valor de intensidad predeterminado, este pixel es reemplazado por un pixel negro, y viceversa, si la intensidad del pixel de la imagen de entrada es mayor al valor de intensidad predeterminado, el pixel es reemplazado por un pixel blanco [25]. En la Figura 2.14 se muestra la imagen binaria como resultado de la umbralización de la imagen (c) de la Figura 2.13.

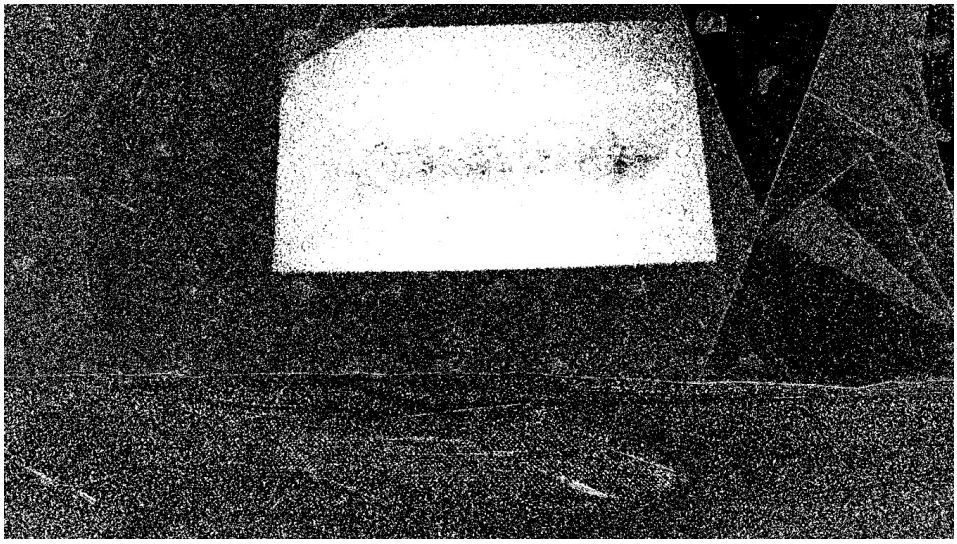


Figura 2.14. Umbralización de imagen.

Después de generar la imagen binaria, se utiliza la función *findContours()* de OpenCV (Tabla 2.8) [25] para obtener el contorno del área de proyección, como paso siguiente se crea un rectángulo que encapsula el contorno del área de proyección empleando la función *boundingRect()* de OpenCV (Tabla 2.9) [25], esta función devuelve como valor de retorno el ancho, alto y las coordenadas ***x, y*** del punto del vértice superior izquierdo del rectángulo, estos datos corresponderían al área de proyección (Figura 2.15).

Tabla 2.8. Función *findContours* de OpenCV.

Función	**Tipo de Variable**	**Parametros**
void cv::findContours	(InputOutputArray	image,
	OutputArrayOfArrays	contours, (Opcional)
	OutputArray	hierarchy, (Opcional)
	int	mode,
	int	method,
	Point	offset = Point() (Opcional)
	)	
Descripción		
Encuentra los contornos desde una imagen binaria.		
Uso		
image, contours, hierarchy = cv::findContours(image, mode, method[, contours[, hierarchy[, offset]]])		
Parametros	**Descripción**	
image	Imagen binaria de entrada.	
contours	Vector de puntos de salida donde se almacenan los contornos detectados	
hierarchy	Vector opcional de salida que contiene información acerca de la topología de la imagen. Contiene igual cantidad de elementos como número de contornos.	
mode	Modo de busqueda de los contornos, por ejemplo:, RETR_EXTERNAL, solo obtiene los contornos exteriores extremos.	
method	Método de aproximación de contorno, por ejemplo: CHAIN_APROX_NONE, almacena todos los puntos de los contornos.	
offset	Desplazamiento opcional por el que se desplaza cada punto del contorno.	

Tabla 2.9. Función *boundingRect* de OpenCV.

Función	**Tipo de Variable**	**Parametros**
Rect cv::boundingRect	(InputArray)	array,
Descripción		
La función calcula y devuelve el rectángulo delimitador superior derecho para un conjunto de puntos especificado o píxeles distintos de cero de una imagen en escala de grises.		
Uso		
retval = cv::boundingRect(array)		
Parametros	**Descripción**	
array	Imagen de entrada en escala de grises o conjunto de puntos bidimensionales almacenados en un vector	

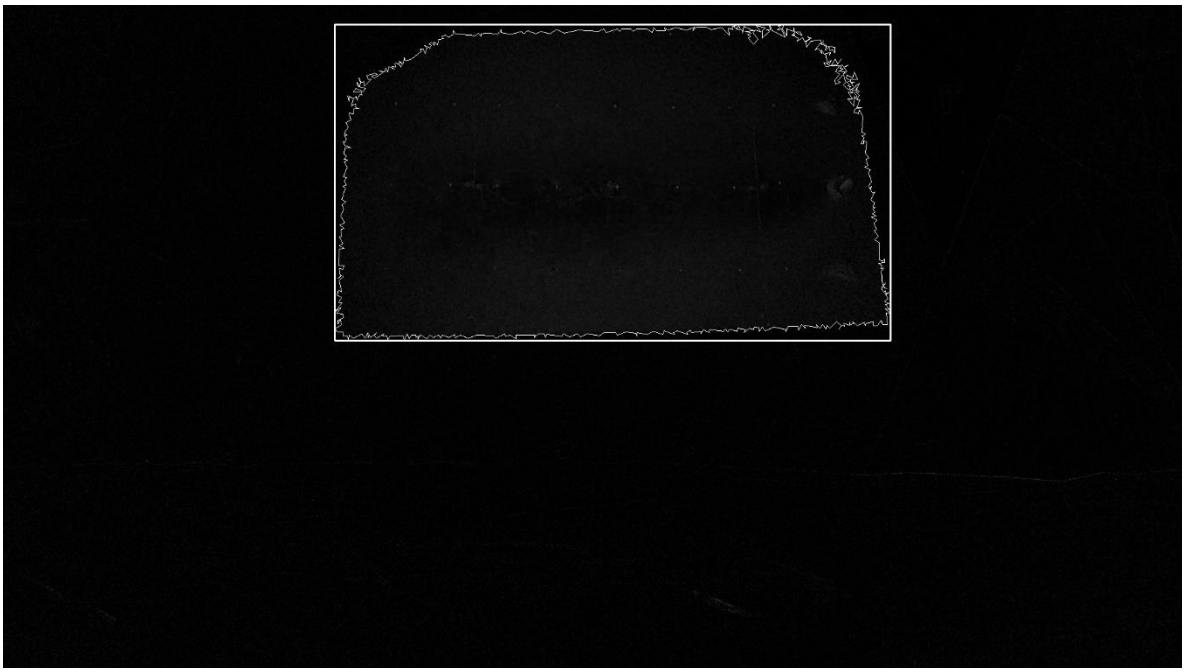


Figura 2.15. Detección del área de proyección.

Finalmente, con los datos del tamaño y ubicación del área de proyección obtenidos con la función *boundingRect()*, se utiliza la función *drawContours()* de OpenCV (Tabla 2.10) [25] para dibujar las *climbing holds* virtuales (Figura 2.16) que serán mostradas al escalador mediante el proyector de imágenes (Figura 2.17) .

Tabla 2.10. Función *drawContours* de OpenCV.

Función	**Tipo de Variable**	**Parametros**
void cv::drawContours	(InputOutputArray	image,
	InputArrayOfArrays	contours,
	int	contourIdx
	const Scalar &	color
	int	thickness = 1, (Opcional)
	int	lineType = LINE_8, (Opcional)
	InputArray	hierarchy = noArray(), (Opcional)
	int	maxLevel = INT_MAX, (Opcional)
	Point	offset = Point() (Opcional)
	)	
Descripción		
Dibuja contornos delineados o rellenos.		
Uso		
cv::drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]])		
Parametros	**Descripción**	
image	Imagen de destino.	
contours	Todos los contornos de entrada que estan almacenados como un vector de puntos.	
contourIdx	Parámetro que indica el contorno a dibujar, si es negativo, todos los contornos son	
color	Color de los contornos.	
thickness	Grosor de línea de los contornos, si es negativo o igual a FILLED, el interior de los contornos también es dibujado.	
lineType	Tipo de línea de los contornos.	
hierarchy	Información opcional acerca de la jerarquía, es necesario solo si se quieren dibujar algunos contornos.	
maxLevel	Nivel máximo para el dibujado de contornos, si es 0, solo los contornos especificados son dibujados, si es uno, se dibujan los contornos con todos sus contornos anidados.	
offset	Parámetro opcional para el desplazamiento de los contornos.	

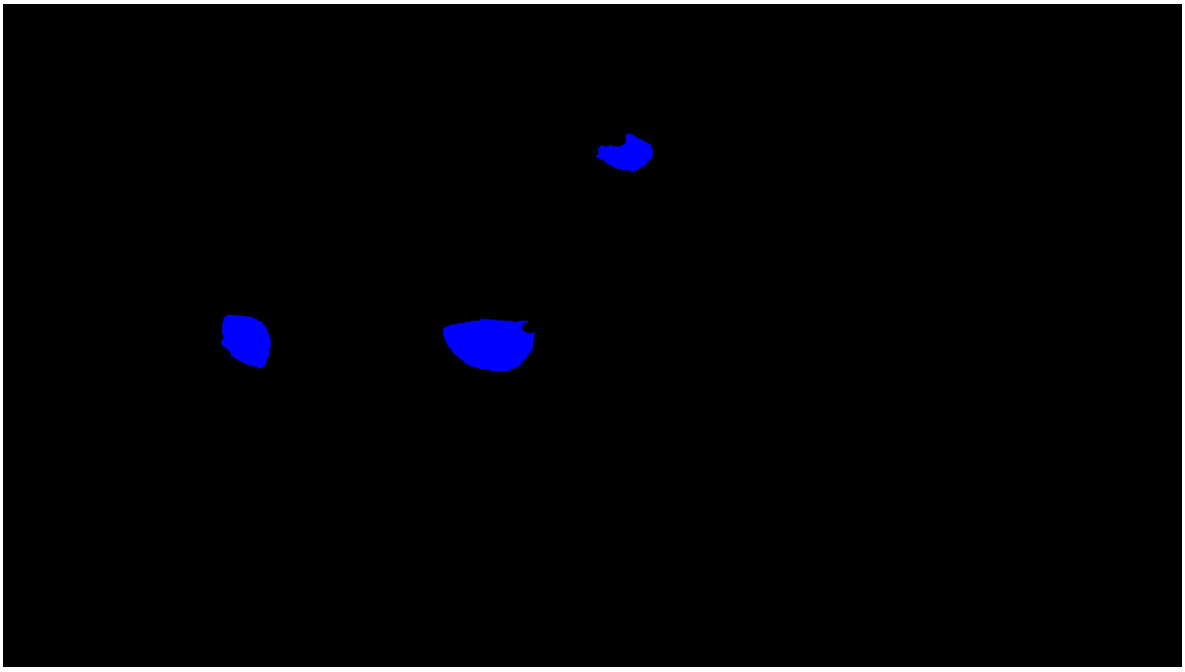


Figura 2.16. Dibujado de las *climbing holds* virtuales que son visualizadas con el proyector de imágenes.

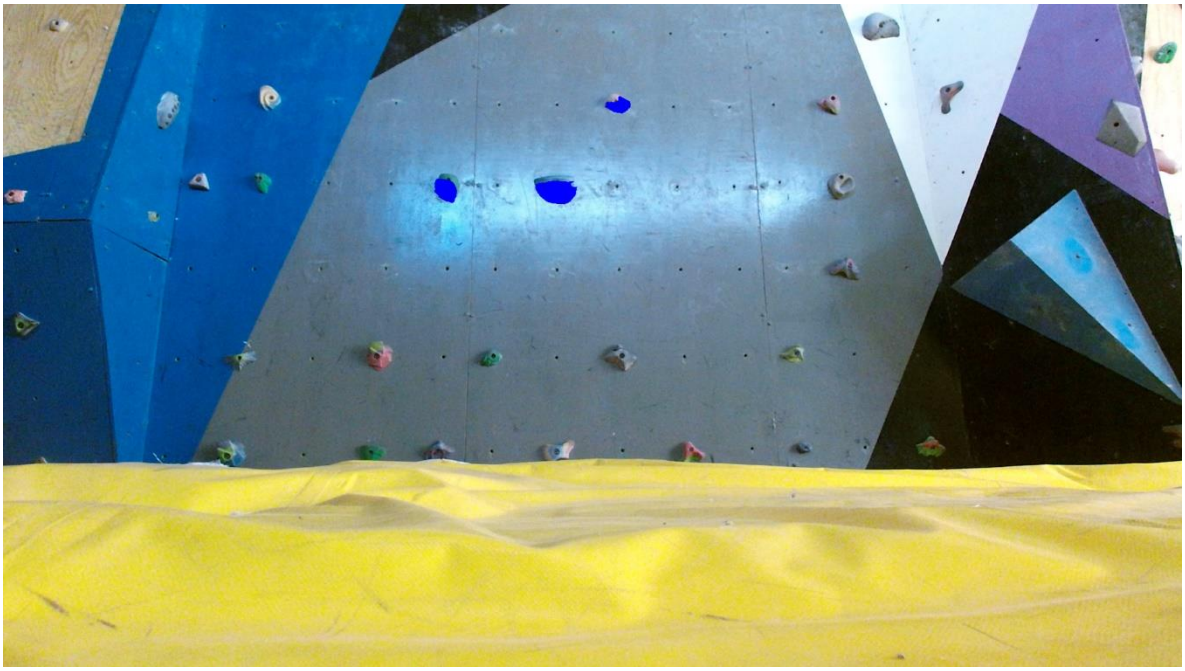


Figura 2.17. Proyección de las *climbing holds* virtuales sobre el muro de escalada mediante un proyector de imágenes.

2.7. Calibración de las coordenadas de los objetos virtuales

En la Sección 2.6 hemos visto que los objetos virtuales se pueden visualizar a través de la pantalla de la computadora o utilizando un proyector de imágenes.

2.7.1. Obtención de las coordenadas de los objetos virtuales que se muestran a través de la pantalla de la computadora.

En el sistema que se desarrolló para este proyecto de investigación, se definió una cantidad fija de círculos que se mostrarán en pantalla utilizando la función *circle()* de OpenCV, también se definieron las coordenadas *P(x, y)* donde se ubicarán los puntos centrales de los círculos y el tamaño del radio de cada uno de ellos. Tanto el radio, como las coordenadas *P(x, y)* del punto central de los círculos se almacenan en un arreglo o vector para posteriormente ser dibujados en la imagen RGB de Kinect.

Adicionalmente, el sistema también puede generar de manera aleatoria las coordenadas *P(x, y)* del punto central y el tamaño del radio de los círculos; y los almacena de forma automática en un arreglo o vector conforme va generando estos datos.

2.7.2. Calibración de las coordenadas de los objetos virtuales que se muestran utilizando un proyector de imágenes.

Retomando el objetivo de probar la interacción del cuerpo humano con objetos virtuales en un gimnasio con muros de escalada. Para la calibración de las coordenadas de las *climbing holds* virtuales, primero es necesario conocer la posición relativa de las *climbing holds* reales del muro de escalada que se encuentren dentro del área de proyección, para obtener estas posiciones se utiliza un sistema de reconocimiento de *climbing holds* en muros de escalada en tiempo real [26], [27], el cual emplea diversos algoritmos y métodos para el reconocimiento y detección de objetos, generando como resultado las coordenadas de los contornos de las *climbing holds* dentro del área de proyección.

La ubicación de los contornos de las *climbing holds* dentro del área de proyección corresponde a las coordenadas de un punto de contorno $C\,(x_2, y_2)$ que se muestra en la Figura 2.18. Con estas coordenadas se pueden crear las *climbing hold*s virtuales que se mostrarán al escalador mediante un proyector de imágenes, tal como se describió en la sección 2.6.2. Sin embargo, para reconocer la interacción del escalador con las *climbing holds* virtuales es necesario realizar un proceso de calibración de imágenes, el cual consiste en alinear las coordenadas de los contornos (x_2, y_2) de las *climbing holds* virtuales del área de proyección con el área de la imagen RGB de Kinect.

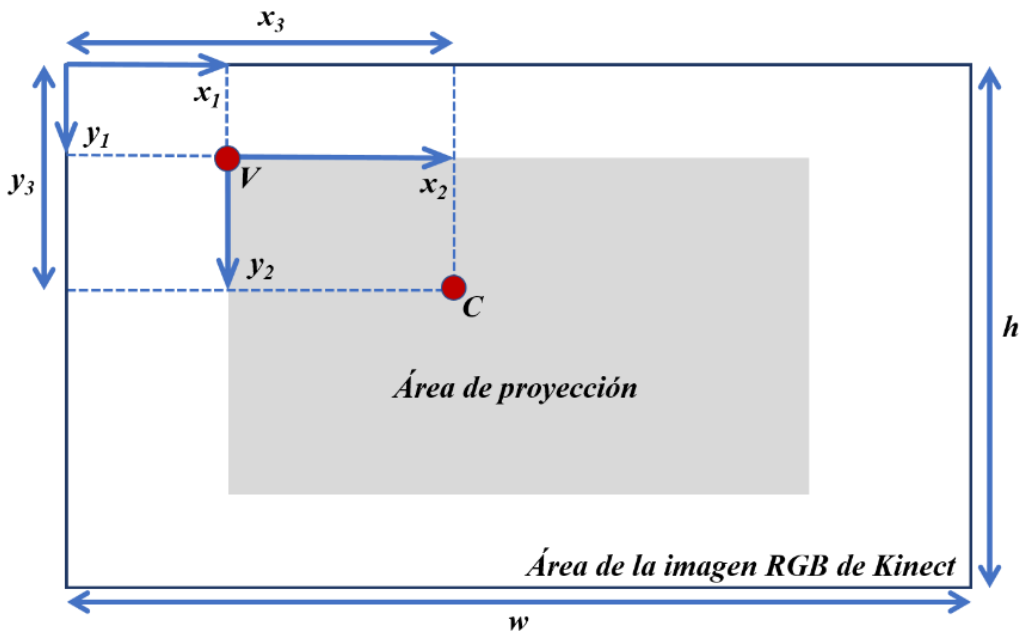


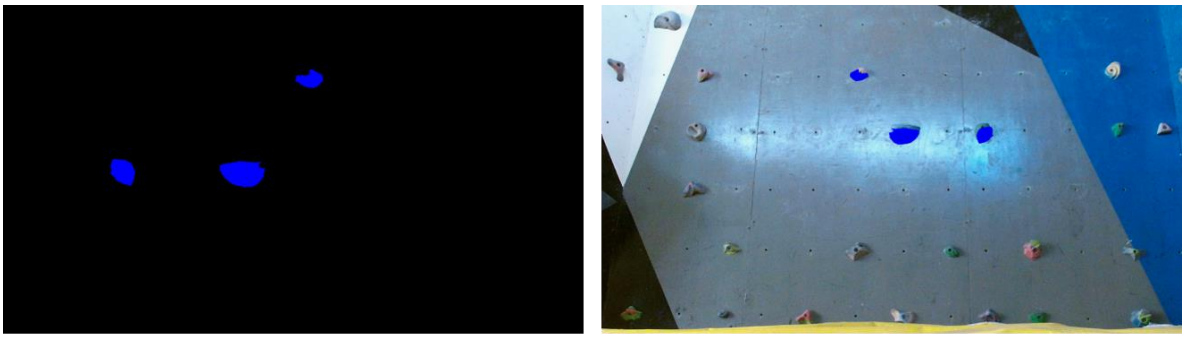
Figura 2.18. Calibración de la imagen en el área de proyección con la imagen RGB de Kinect.

Por lo tanto, tomando como referencia la Figura 2.18, también es necesario obtener las coordenadas del vértice V(x_1, y_1) mediante la detección del área de proyección cuyo proceso se explicó en la sección 2.6.2, asimismo, es importante considerar, de acuerdo a lo visto en la sección 2.3, que Kinect captura las imágenes con la cámara RGB en modo espejo y dado que el escalador se sitúa frente al muro de escalda dando la espalda a Kinect, cuando el escalador se mueve a su lado izquierdo, en el campo de visión de Kinect se estará desplazando al lado derecho, del mismo modo, si las *climbing holds* reales se ubican del lado derecho, es necesario posicionar las *climbing holds* virtuales del lado izquierdo de la imagen RGB de Kinect invirtiendo las coordenadas en el eje ***x***. Considerando estos factores, las fórmulas para calcular las coordenadas de los contornos de las *climbing holds* virtuales son las siguientes:

$$x_3 = w - (x_1 + x_2)$$

$$y_3 = y_1 + y_2$$

En la Figura 2.19 se muestra el resultado de la calibración de una imagen que es visualizada por medio del proyector de imágenes (a) con la imagen RGB de Kinect (b).



(a) (b)

Figura 2.19.Calibración de imágenes. (a) Imagen visualizada con el proyector de imágenes, (b) Resultado de la calibración en la imagen RGB de Kinect.

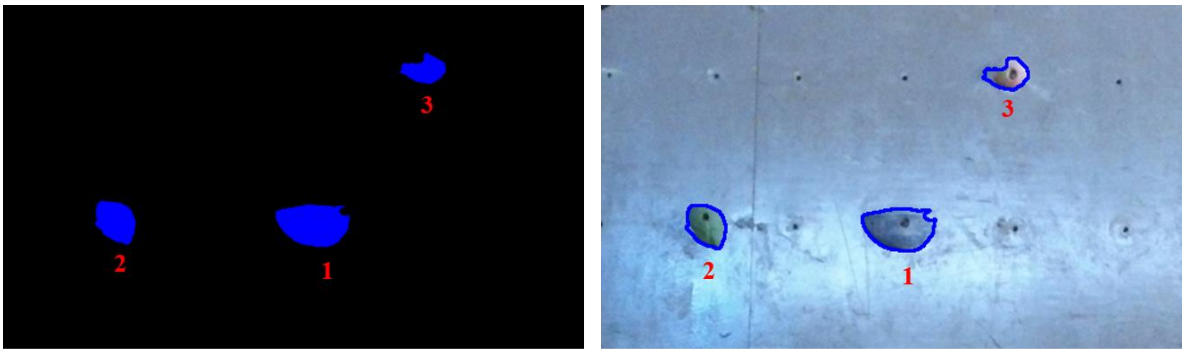
2.8.Detección de colisiones

Para reconocer la interacción de las manos con los objetos virtuales una vez obtenidas las coordenadas de ambos elementos, se implementó un algoritmo básico de detección de colisiones. Básicamente, una colisión es la intersección o choque entre dos o más cuerpos donde al menos uno de ellos está en movimiento [28], por lo tanto, la detección de una

colisión consiste en comparar las coordenadas de los objetos o cuerpos con el fin de comprobar si hay una intersección entre sus coordenadas.

Para seleccionar el algoritmo de detección apropiado, es importante considerar la forma o el tipo de representación geométrica de los objetos y, para este caso en particular, la forma de las manos. Sin embargo, aunque es posible comparar geométricamente las coordenadas de dos objetos para comprobar si existe una colisión entre ellos, esto no suele ser una buena idea si la representación geométrica de los objetos consiste en cientos o incluso miles de polígonos, dando como resultado un alto consumo de memoria y recursos.

En la Figura 2.20 se muestra la representación geométrica (a) de las climbing holds reales (b), cuyas coordenadas que conforman sus contornos se muestran en la Tabla 2.11, las cuales se obtienen al utilizar el sistema de reconocimiento de *climbing holds* en muros de escalada en tiempo real [27].



(a) (b)

Figura 2.20. (a) Representación geométrica, (b) *Climbing holds* reales.

Tabla 2.11. Coordenadas de los contornos de las *climbing holds*.

Contornos		
Objeto 1	**Objeto 2**	**Objeto 3**
[336, 238; 335, 239; 330, 239; 329, 240; 326, 240; 325, 241; 320, 241; 319, 242; 315, 242; 314, 243; 312, 243; 310, 245; 310, 250; 311, 251; 311, 253; 312, 254; 312, 255; 313, 256; 313, 257; 316, 260; 316, 261; 322, 267; 323, 267; 326, 270; 327, 270; 328, 271; 329, 271; 330, 272; 332, 272; 333, 273; 335, 273; 336, 274; 342, 274; 343, 275; 356, 275; 357, 274; 359, 274; 360, 273; 361, 273; 363, 271; 364, 271; 365, 270; 365, 269; 369, 265; 369, 264; 372, 261; 372, 260; 373, 259; 373, 254; 374, 253; 374, 249; 375, 248; 374, 247; 373, 247; 370, 244; 370, 240; 369, 239; 363, 239; 362, 240; 359, 240;	[159, 235; 156, 238; 156, 240; 155, 241; 155, 253; 154, 254; 154, 255; 155, 256; 155, 257; 156, 257; 161, 262; 161, 264; 162, 264; 165, 267; 166, 267; 167, 268; 168, 268; 169, 269; 170, 269; 171, 270; 172, 270; 173, 271; 176, 271; 177, 272; 179, 272; 180, 273; 183, 273; 185, 271; 185, 270; 186, 269; 186, 267; 187, 266; 187, 265; 188, 264; 188, 259; 189, 258; 189, 252; 188, 251; 188, 249; 187, 248; 187, 246; 186, 245; 186, 244; 184, 242; 184, 241; 183, 240; 182, 240; 181, 239; 180, 239; 179, 238; 178, 238; 177, 237; 176, 237; 175, 236; 173, 236; 172, 235]	[440, 105; 438, 107; 438, 111; 435, 114; 433, 114; 432, 115; 429, 115; 428, 114; 427, 114; 426, 115; 423, 115; 422, 114; 421, 114; 418, 117; 418, 121; 417, 122; 418, 123; 419, 123; 420, 124; 422, 124; 426, 128; 427, 128; 428, 129; 429, 129; 430, 130; 431, 130; 432, 131; 435, 131; 436, 132; 441, 132; 442, 133; 444, 133; 445, 132; 446, 132; 449, 129; 451, 129; 456, 124; 456, 123; 457, 122; 457, 117; 456, 116; 456, 114; 447, 105]

Tal como se observa en la Tabla 2.11, la cantidad de comparaciones para comprobar si las coordenadas de las manos se encuentran dentro del área de colisión de las *climbing holds*, puede llegar a ser considerablemente alta si la complejidad de los contornos y la cantidad de *climbing holds* aumentan, por lo tanto, en lugar de usar la misma forma geométrica utilizada para mostrar las *climbing holds* virtuales al escalador, es recomendable sustituirla por una forma geométrica más simple para detectar las colisiones. En videojuegos, por ejemplo, es común utilizar figuras geométricas simples, tales como círculos o rectángulos, para sustituir o representar un objeto en el juego independientemente de su complejidad, si las figuras colisionan, es correcto asumir que los objetos igualmente están colisionando. A estas figuras geométricas simples se les conoce como volúmenes delimitadores o *bounding volumes* [28].

La función de los volúmenes delimitadores consiste en encapsular uno o más objetos de naturaleza o forma más compleja y se utilizan frecuentemente para acelerar las comprobaciones de colisión entre objetos independientemente de la representación geométrica utilizada para mostrarlos. La Figura 2.21 muestra cinco de los volúmenes delimitadores más utilizados [28].

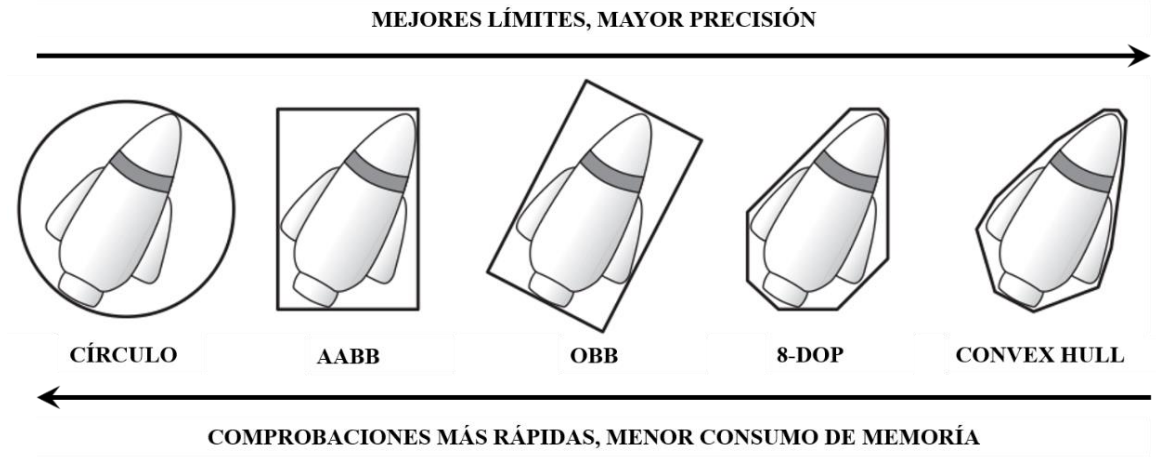


Figura 2.21. Tipos de volumen delimitador: círculo o esfera, *axis-aligned bounding box* (AABB), *oriented bounding box* (OBB), *eight-direction discrete orientation polytope* (8-DOP) y *convex hull*.

El tipo de volumen delimitador que se seleccionó para detectar las colisiones entre las manos del escalador y las climbing holds virtuales fue el círculo, debido a que, teóricamente, es el más eficiente en cuanto a velocidad de comprobación de colisiones y consumo de

memoria se refiere [28], además, tomando como referencia la forma de la mayoría de las *climbing holds* (Figura 2.22), el círculo es el que mejor se adapta a su forma y contorno.

Figura 2.22. Formas de *climbing holds*.

Para la creación de los círculos como volumen delimitador sobre las climbing holds virtuales se utilizó la función *minEnclosingCircle()* de OpenCV (Tabla 2.12) [25], la cual genera un círculo de área mínima que encapsula un conjunto de puntos en dos dimensiones, obteniendo como valor de salida, el radio y el punto central del círculo. En este caso, el círculo generado encapsula los contornos de las *climbing holds* virtuales como se muestra en la Figura 2.23.

Tabla 2.12. Función *minEnclosingCircle()* de OpenCV.

Función	**Tipo de Variable**	**Parametros**
void cv::minEnclosingCircle	(InputArray Point2f & float &)	points, center, radius
Descripción		
Encuentra el círculo envolvente mínimo de un conjunto de puntos 2D utilizando un algoritmo iterativo.		
Uso		
cv::minEnclosingCircle(points, center, radius)		
Parametros	**Descripción**	
points	Vector de entrada con los puntos de los contornos en 2D.	
center	Valor de salida del centro del círculo.	
radius	Valor de salida del radio del círculo.	

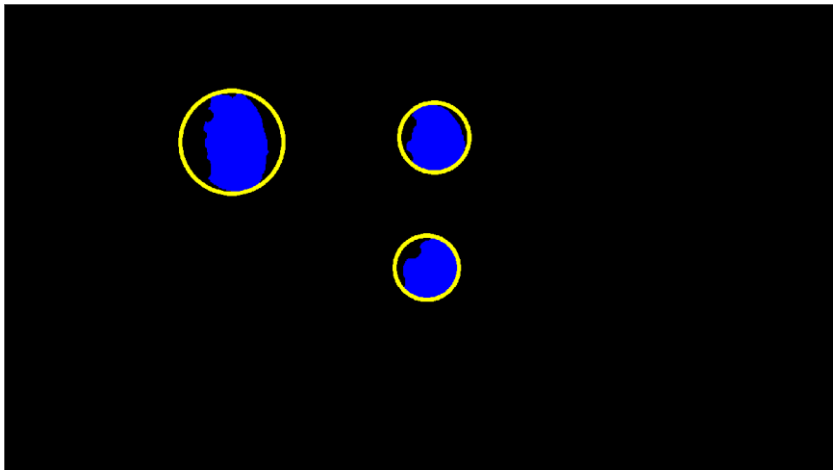


Figura 2.23. Creación de volúmenes delimitadores sobre las *climbing holds* virtuales.

En cuanto a las manos del escalador, también se utilizaron círculos como volumen delimitador con el fin de corregir la detección de colisiones, debido a que, en algunos casos, las *climbing holds* son muy pequeñas y no alcanzan a hacer contacto con las coordenadas de las manos en las articulaciones *JointType_HandRight* y *JointType_HandLeft* que se ubican en la palma de las manos. Para crear estos círculos, se utilizó la función *circle()* de OpenCV (Tabla 2.7, Sección 2.6.1), donde, como parámetro del centro de los círculos se utilizaron las coordenadas *JointType_HandRight* (mano derecha) y *JointType_HandLeft* (mano izquierda); y como parámetro del radio, se calculó la distancia entre el punto $P_A(x_1, y_1)$ y el punto $P_B(x_2, y_2)$ como se muestra en la Figura 2.24, donde el punto $P_A(x_1, y_1)$ corresponde a las coordenadas *JointType_HandRight* y *JointType_HandLeft* (palma de la mano derecha e izquierda); y el punto $P_B(x_2, y_2)$ corresponde a las coordenadas *JointType_HandTipRight* y *JointType_HandTipLeft* (punto del dedo medio de la mano derecha e izquierda).

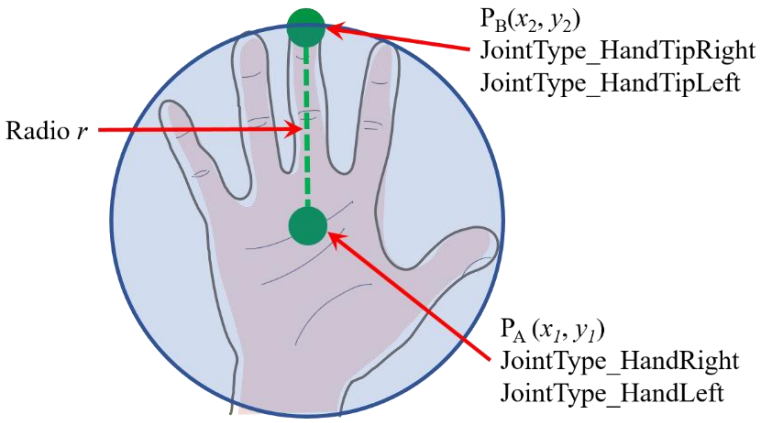


Figura 2.24. Creación de volúmenes delimitadores sobre las manos.

Una vez que se han creado los círculos como volumen delimitador para cada uno de los contornos de las *climbing holds* virtuales y las manos del escalador, se implementa un algoritmo de detección de colisiones entre dos círculos, el cual consiste en calcular la distancia entre los centros de los círculos para, posteriormente, comparar esta distancia contra la suma de sus radios. Si la distancia entre los centros de los círculos es menor a la suma de sus radios, significa que hay una intersección en sus circunferencias y, por ende, existe una colisión [28].

El cálculo de la distancia entre los centros de los círculos se basa en la aplicación del teorema de Pitágoras [29], el cual establece que: en todo triángulo rectángulo, el cuadrado de la hipotenusa es igual a la suma de los cuadrados de los catetos. Esto es: $a^2 + b^2 = c^2$ (Figura 2.25).

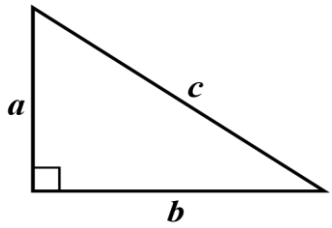


Figura 2.25. Teorema de Pitágoras

De acuerdo a la Figura 2.26, aplicando el teorema de Pitágoras a una pendiente entre dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ en un sistema de coordenadas cartesiano, se tiene un triángulo rectángulo de hipotenusa $\boldsymbol{d}$ y catetos de longitud $\boldsymbol{x_2 - x_1}$ y $\boldsymbol{y_2 - y_1}$, por consiguiente, se obtienen las siguientes formulas [30]:

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Estas fórmulas representan la longitud de un segmento de recta determinado por los puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$. Es decir, representan la distancia entre dichos puntos. Entonces se puede deducir que una colisión entre dos círculos es verdadera si se cumple con cualquiera de las siguientes condiciones:

$$d^2 < (r_1 + r_2)^2$$

$$d < r_1 + r_2$$

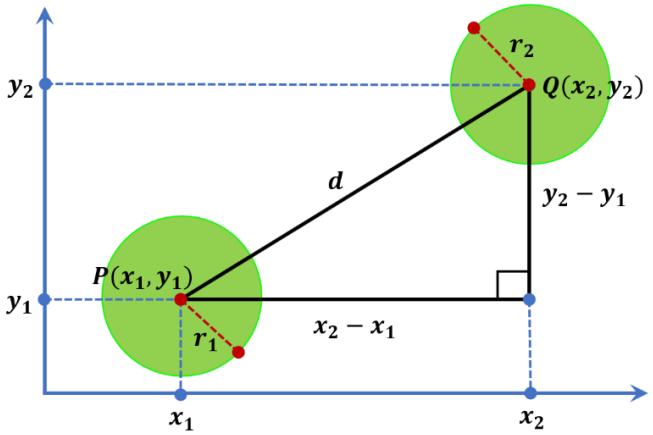


Figura 2.26. Distancia entre dos círculos.

2.9. Activación de eventos

En esta fase se define que sucede después de que se ha detectado una colisión entre la mano del usuario y un objeto virtual mediante la activación de un evento que realice alguna función en concreto, o bien, que confirme al usuario su interacción con el objeto virtual. Un evento es una acción que se produce como resultado de la interacción del usuario con un objeto virtual [20].

En el sistema para reconocer la interacción del cuerpo humano con objetos virtuales se implementó un *Timer* (Temporizador) como evento que se activa al momento de presentarse una colisión. Cuando el usuario coloca su mano sobre un objeto virtual provocando una colisión, el *Timer* mide el tiempo que dura el usuario en mantener la mano sobre la posición del objeto, al cumplirse un determinado lapso, el sistema toma como valida la colisión y se activa otro evento que consiste en cambiar de color al objeto virtual, de esta manera, se le confirma al usuario su interacción con el objeto.

Uno de los objetivos de utilizar un *Timer* es reducir o evitar la incidencia de falsos positivos, por ejemplo, si se muestra un menú interactivo en la pantalla de la computadora, al aplicar un *Timer* el usuario tiene que mantener su mano sobre la opción deseada para seleccionarla, de esta manera se puede evitar que seleccione una opción incorrecta al desplazar su mano sobre otras opciones del menú. Asimismo, para reconocer la interacción del escalador con las *climbing holds* virtuales, se aplicó un *Timer* como método para validar que el escalador está sujetando la *climbing hold* indicada, en lugar de únicamente pasar su mano sobre la *climbing hold* virtual sin realmente estar sujetando la *climbing hold* real.

Adicionalmente, en la interacción de objetos virtuales que se muestran en la pantalla de una computadora, se implementó otro modo de interacción que consiste en permitir al usuario poder mover los objetos virtuales al empuñar la mano.

Kinect, en su versión 2.0, tiene la característica de poder detectar las siguientes señas o gestos realizados con la mano (Figura 2.27) [20]:

- Open: Abriendo la palma de la mano juntando los dedos.
- Closed: Cerrando la mano formando un puño.
- Lasso: Gesto de señalar o apuntar con los dedos medio e índice.

Figura 2.27. Gestos realizados con la mano detectados por Kinect. (a) *Open*, (b) *Closed*, (c) *Lasso*.

Si el usuario no realiza ningún gesto, Kinect detecta el estado de la mano como desconocido (*Unknown*). En cambio, si Kinect no puede detectar correctamente el estado o posición de la mano, entonces el estado de la mano lo clasifica como *NotTracked*. La declaración del estado de la mano en la librería Kinect for Windows SDK 2.0 se muestra en la Tabla 2.13.

Tabla 2.13. Declaración del estado de la mano en la librería Kinect.h.

Librería Kinect.h

```
enum _HandState {
    HandState_Unknown= 0,
    HandState_NotTracked= 1,
    HandState_Open= 2,
    HandState_Closed= 3,
    HandState_Lasso= 4
};

enum _TrackingConfidence {
    TrackingConfidence_Low= 0,
    TrackingConfidence_High= 1
};
```

Para obtener los datos del estado de la mano. Primero se utiliza la función *setSkeleton()* de NtKinect para obtener la información de la detección de las articulaciones del cuerpo humano y, posteriormente, se utiliza la función *handState()* de NtKinect (Tabla 2.14) [18] para obtener el estado de la mano.

Tabla 2.14. Función *handState()* de NtKinect.

Función	**Tipo de Variable**	**Parametros**
pair handState	(Int bool)	id, isLeft,
Descripción		
Obtiene el estado de la mano después de utilizar la función *setSkeleton()*		
Uso		
pair<int, int> handState = kinect.handState(id, isLeft)		
Parametros	**Descripción**	
id	Indice de la persona detectada (*skeleton[id]*)	
isLeft	Utilizar verdadero para obtener el estado de la mano izquierda y falso para el estado de la mano derecha.	

Entonces, si el estado de la mano del usuario es igual a *HandState_Closed* y además existe una colisión entre la mano del usuario y un objeto virtual, las coordenadas (x, y) del punto central del objeto virtual serán igual a las coordenadas (x, y) de la articulación de la palma de la mano (*JointType_HandRight* o *JointType_HandLeft*), permitiendo de esta manera, que el objeto virtual se desplace junto con el movimiento de la mano del usuario. Por lo tanto, si el estado de la mano del usuario cambia o es diferente a *HandState_Closed*, el objeto deja de moverse.

CAPÍTULO 3

3. Pruebas y resultados

En el presente capítulo se describen las pruebas realizadas al sistema para reconocer la interacción del cuerpo humano con objetos virtuales, las cuales se llevaron acabó en tres entornos experimentales:

- Cubículo de oficina.
- Salón de Clase.
- Gimnasio de escalada.

Posteriormente se presenta un análisis de los resultados obtenidos en las pruebas realizadas en dichos entornos experimentales.

En cuanto a los dispositivos que se emplearon para realizar las pruebas en el cubículo de oficina, se utilizó un Kinect v2.0, y una computadora portátil con las especificaciones técnicas descritas en la Tabla 3.1, adicionalmente, para las pruebas realizadas en el salón de clase y en el gimnasio de escalada, se utilizaron un proyector de imágenes de bolsillo y un proyector de imágenes portátil respectivamente, cuyas especificaciones técnicas se describen en la Tabla 3.2.

Tabla 3.1. Especificaciones técnicas de la computadora portátil.

Especificaciones técnicas	
Procesador	Intel Core i5-3427U (2 núcleos, 1.8 GHz)
Gráficos	Intel HD Graphics 4000, DirectX 12
RAM	6 GB DDR3 SDRAM
Disco	SSD 480 GB
Sistema Operativo	Microsoft Windows 10 Home Single Language (x64)
Puerto	USB 3.0

Tabla 3.2. Especificaciones técnicas de los proyectores de imágenes.

Epecificaciones técnicas	**Proyector de bolsillo**	**Proyector Portatil**
Modelo	Philips PicoPix	GP9 Poner Saund
Resolución	640x360 píxeles	720p (1280x720)
Potencia de brillo	40 lúmenes	2400 lúmenes
Conexión	USB	VGA, HDMI

Es importante mencionar que, como preparativo inicial, para las pruebas realizadas en el salón de clase y en el gimnasio de escalada fue necesario hacer un reconocimiento de las *climbing holds* reales con el fin de conocer su tamaño y posición, para posteriormente, crear las *climbing holds* virtuales que se proyectarían sobre las *climbing holds* reales utilizando el proyector de imágenes. Asimismo, fue necesario detectar el contorno del área de proyección con la finalidad de realizar una calibración entre la imagen que se está visualizando con el proyector de imágenes y la imagen de la cámara RGB de Kinect, para de esta manera, conocer la posición (coordenadas x, y) que tendrían las *climbing holds* en el campo de visión de Kinect. El procedimiento de este preparativo inicial esta descrito en las secciones 2.6.2 y 2.7.2.

Referente a la detección del cuerpo humano, con el propósito de verificar la correcta detección de las articulaciones y monitorear su movimiento, se marcaron las articulaciones detectadas con un pequeño cuadro de color rojo, tal como se muestra en la Figura 3.1.

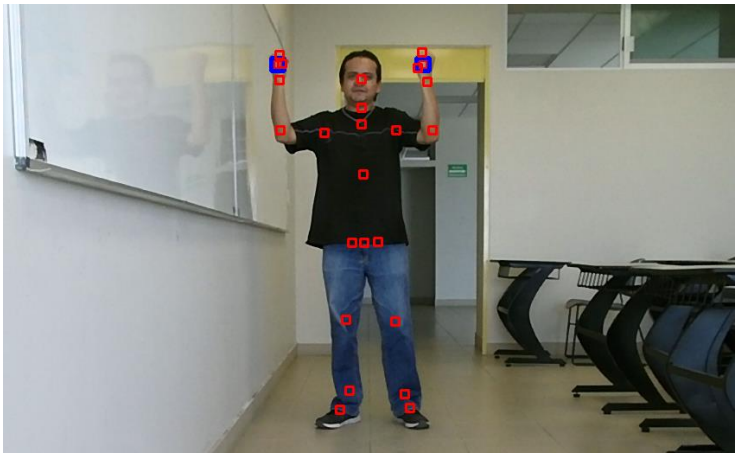


Figura 3.1. Detección de las articulaciones del cuerpo humano.

3.1. Primer entorno experimental: Cubículo de oficina

El primer entorno experimental fue en un cubículo de oficina, lugar donde fue desarrollado el sistema para reconocer la interacción del cuerpo humano con objetos virtuales y se realizaron las pruebas básicas de funcionamiento.

En este entorno experimental se analizó la interacción con objetos virtuales que se visualizan en la pantalla de la computadora. Para llevar a cabo las pruebas, se colocaron el Kinect y la computadora portátil sobre un escritorio, mientras que el usuario se ubicó, ya sea

de pie o sentado, de frente y con la mirada hacia los dispositivos a una distancia aproximada de entre 0.5 a 1.5 metros. Una representación gráfica de la ubicación de los elementos que conformaron el primer entorno experimental se muestra en la Figura 3.2.

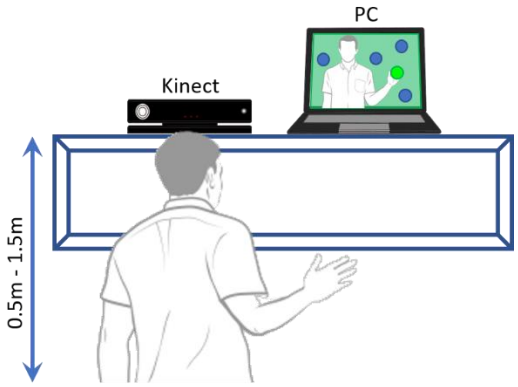


Figura 3.2. Primer entorno experimental.

La prueba consistió en mostrar de 6 a 10 objetos virtuales en forma de círculos de color azul en la pantalla de la computadora, con los cuales el usuario puede interactuar con sus manos. Cuando el sistema detecta una interacción, es decir, cuando el usuario coloca la palma de su mano sobre un círculo, este círculo cambia a color verde, lo cual indica que el usuario interactuó de forma exitosa con el objeto virtual, asimismo cuando el usuario empuña la mano sobre un objeto, el usuario puede desplazarlo hacia otra posición en la pantalla, si el usuario abre su mano, entonces el objeto deja de moverse.

3.1.1. Resultados.

Durante las pruebas se presentaron algunas inconsistencias en la detección de las articulaciones del cuerpo humano, principalmente en las pruebas donde el usuario interactuó con los objetos virtuales estando sentado. En la Figura 3.3 se pueden observar algunos de los errores en la detección de las articulaciones, los cuales están señalados con una flecha roja. La razón de estas inconsistencias podría ser que las manos no son del todo visibles desde el inicio de la prueba, por lo que Kinect trata de inferir la ubicación de las articulaciones con base a la porción del cuerpo que si esta visible.

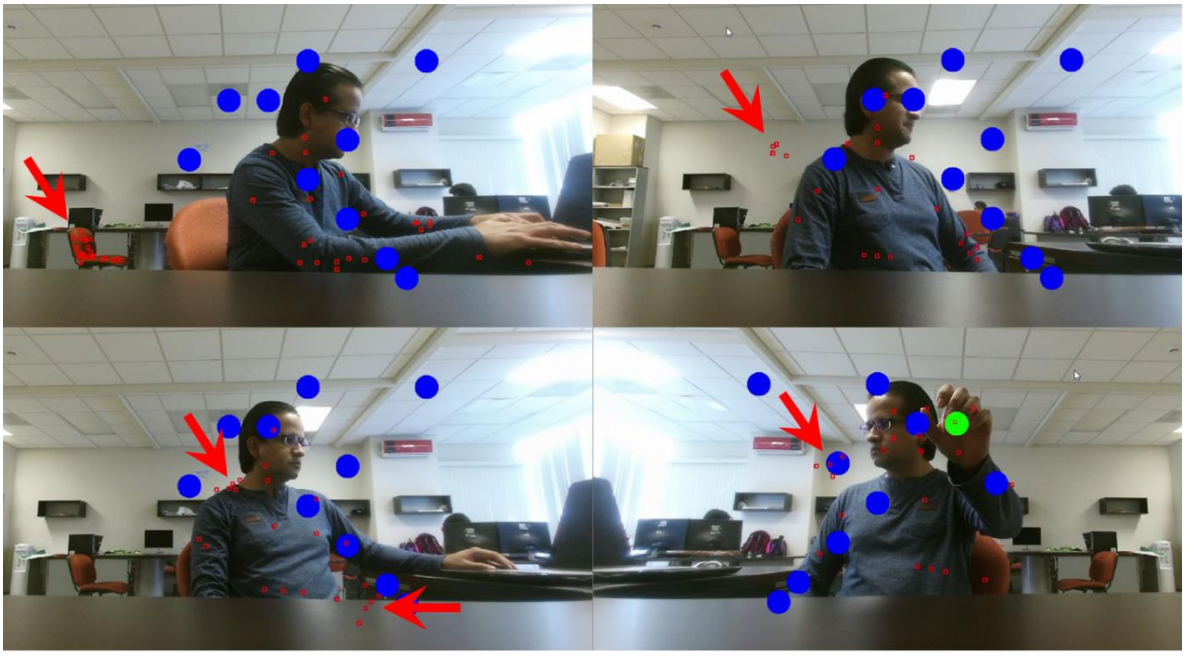


Figura 3.3. Errores en la detección de articulaciones.

Además de los errores en la detección de las articulaciones del cuerpo humano, los resultados también arrojaron algunas interacciones fallidas y falsos positivos. Una interacción fallida sucede cuando el sistema no reconoce la interacción entre la mano del usuario y un objeto virtual, como puede observarse en la Figura 3.4, estos errores suceden cuando Kinect no detecta correctamente las articulaciones de la mano, por lo tanto, no se cuenta con las coordenadas necesarias para detectar una interacción con el objeto virtual.

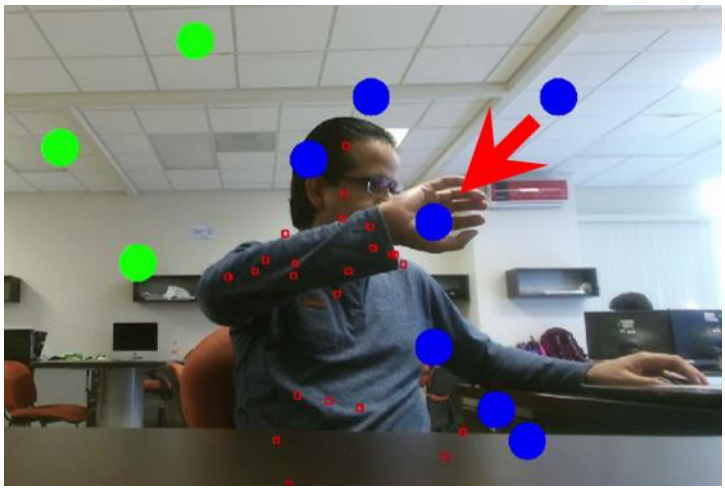


Figura 3.4. Interacción fallida entre la mano del usuario y un objeto virtual.

Por otra parte, los falsos positivos son aquellas interacciones donde el usuario toca o mueve accidentalmente un objeto virtual, sin que el usuario haya querido interactuar con el objeto. En la Figura 3.5 se puede observar que un objeto se movió de manera involuntaria cuando el usuario paso su mano sobre este objeto. Para reducir la incidencia de este tipo de falsos positivos, se implementó un Timer que obliga al usuario a mantener su mano sobre un objeto virtual durante un tiempo determinado, al cumplir con ese tiempo, el sistema toma como valida su interacción y entonces el usuario podrá mover el objeto o se podrá activar otro tipo de evento.

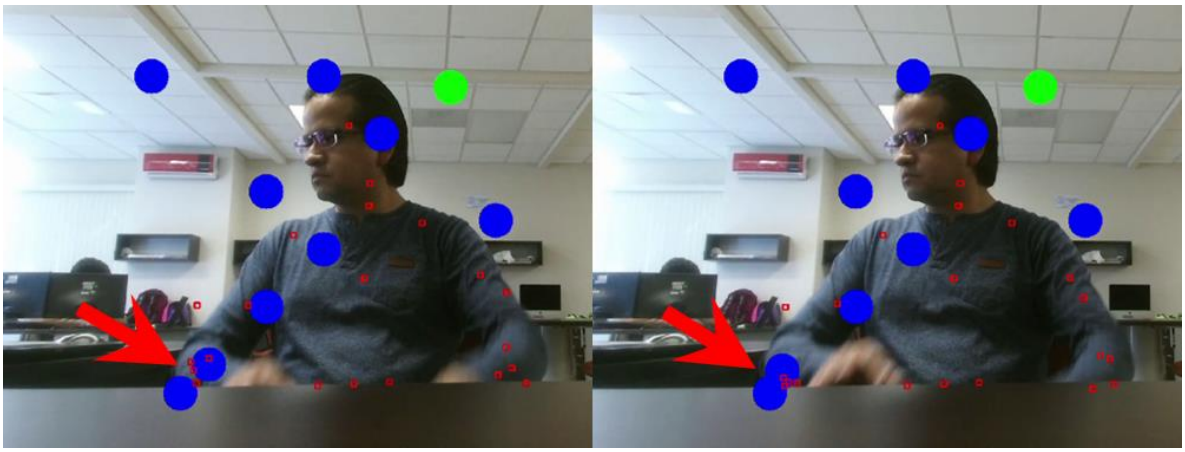


Figura 3.5. Falso positivo al mover de forma involuntaria un objeto virtual.

Otro tipo de falso positivo sucede cuando se detecta una interacción con otra parte del cuerpo distinta a la mano, tal como se observa en la Figura 3.6, donde al presentarse una detección errónea de las articulaciones, se detectó una interacción al tocar un objeto virtual con el hombro.

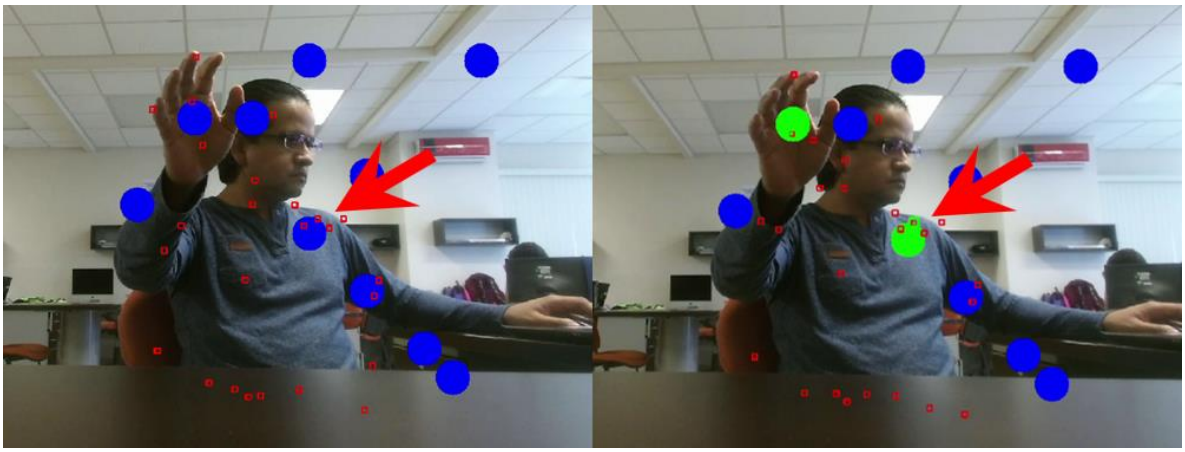


Figura 3.6. Falso positivo al detectarse una interacción entre el hombro del usuario y un objeto virtual.

En cuanto a resultados positivos, en la figura se puede observar una correcta interacción entre la mano del usuario y un objeto virtual, además se observa una detección exitosa de las articulaciones del cuerpo humano.

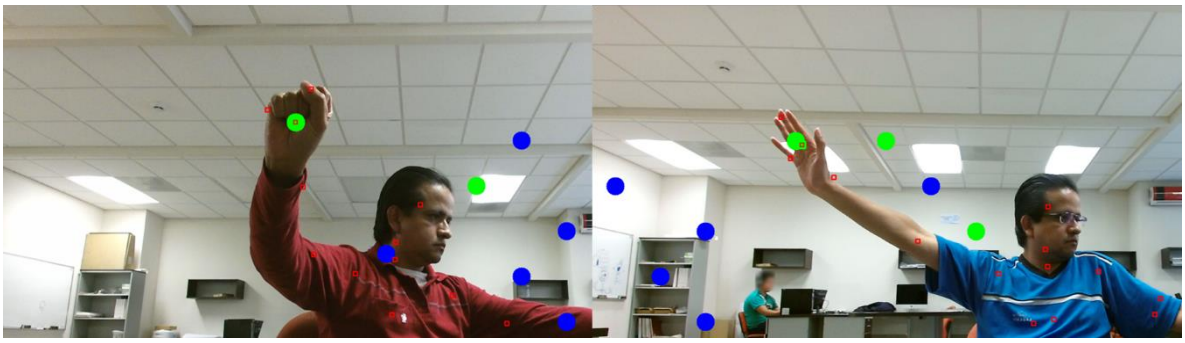


Figura 3.7. Interacción correcta entre la mano del usuario y un objeto virtual.

Con respecto a la interacción que tuvo el usuario con el sistema estando de pie, no se presentaron errores en la detección de las articulaciones del cuerpo humano, por lo tanto, tampoco se presentaron interacciones fallidas. En la Figura 3.8 se muestra una pruebas donde el usuario interactuó con los objetos virtuales estando de pie.

Figura 3.8. Interacción del usuario con objetos virtuales estando de pie.

En resumen, los resultados obtenidos en el primer entorno experimental cuando el usuario interactuó con los objetos virtuales estando sentado se muestran en la Tabla 3.3, en la que se puede observar que una correcta detección de las articulaciones del cuerpo humano contribuye de manera significativa en reducir la incidencia de falsos positivos e interacciones fallidas, siendo de suma importancia asumir una posición y distancia correcta al momento de interactuar con el Kinect.

Tabla 3.3. Resultados de la interacción del usuario con objetos virtuales estando sentado.

Entorno experimental	Total de pruebas	No. de prueba	Cantidad de objetos virtuales	Detección exitosa de las articulaciones del cuerpo humano	Interacciones exitosas	Interacciones fallidas	Falsos positivos	% Interacciones exitosas	% Interacciones fallidas	% Falsos positivos
1.- Cubículo de oficina	15	1	6	Si	5	0	1	83.33%	0.00%	16.67%
		2	6	Si	6	0	0	100.00%	0.00%	0.00%
		3	7	Si	6	0	1	85.71%	0.00%	14.29%
		4	7	Si	7	0	0	100.00%	0.00%	0.00%
		5	7	No	4	1	2	57.14%	14.29%	28.57%
		6	8	Si	6	0	2	75.00%	0.00%	25.00%
		7	8	No	5	2	1	62.50%	25.00%	12.50%
		8	8	No	4	2	2	50.00%	25.00%	25.00%
		9	10	No	6	3	1	60.00%	30.00%	10.00%
		10	10	Si	9	1	0	90.00%	10.00%	0.00%
		11	10	No	6	3	1	60.00%	30.00%	10.00%
		12	10	No	6	2	2	60.00%	20.00%	20.00%
		13	10	Si	9	1	0	90.00%	10.00%	0.00%
		14	10	Si	10	0	0	100.00%	0.00%	0.00%
		15	10	Si	9	0	1	90.00%	0.00%	10.00%
							Total	**77.58%**	**10.95%**	**11.47%**

En lo que concierne a los resultados obtenidos cuando el usuario interactuó con los objetos virtuales estando de pie, se muestran en la Tabla 3.4, en la que se observa que se obtuvo un 93 % de interacciones exitosas.

Tabla 3.4. Resultados de la interacción del usuario con objetos virtuales estando de pie.

Entorno experimental	Total de pruebas	No. de prueba	Cantidad de objetos virtuales	Interacciones exitosas	Interacciones fallidas	Falsos positivos	% Interacciones exitosas	% Interacciones fallidas	% Falsos positivos
1.- Cubículo de oficina	8	1	6	6	0	0	100.00%	0.00%	0.00%
		2	7	5	0	2	71.43%	0.00%	28.57%
		3	7	6	0	1	85.71%	0.00%	14.29%
		4	8	8	0	0	100.00%	0.00%	0.00%
		5	8	7	0	1	87.50%	0.00%	12.50%
		6	10	10	0	0	100.00%	0.00%	0.00%
		7	10	10	0	0	100.00%	0.00%	0.00%
		8	10	10	0	0	100.00%	0.00%	0.00%
						Total	**93.08%**	**0.00%**	**6.92%**

3.2.Segundo entorno experimental: Salón de clase

El segundo entorno experimental fue en un salón de clase, donde se analizó la interacción del cuerpo humano con objetos virtuales que se visualizan en una pared utilizando un proyector de imágenes.

Para efectuar las pruebas en el salón de clase y con el fin de simular un entorno similar a una sección de un muro de escalada, se montó una superficie de cartón sobre un pizarrón blanco en la que se colocaron *climbing holds* (rocas de escalar) reales, las cuales se muestran en la Figura 3.9.

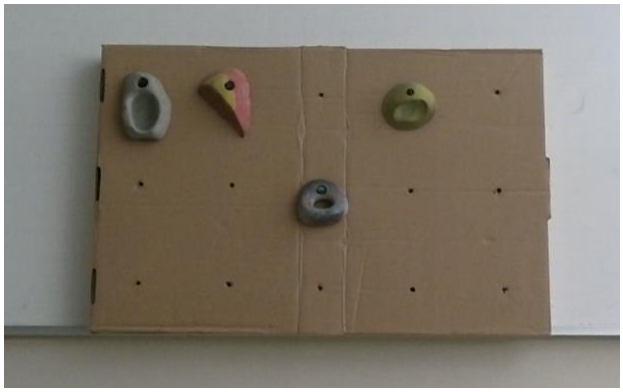


Figura 3.9. Superficie de cartón con *climbing holds* reales.

Los dispositivos como el Kinect, la computadora y el proyector de imágenes se colocaron sobre un escritorio que se ubicó a una distancia aproximada de 2.5 metros con respecto a la superficie de cartón, mientras que el usuario se ubicó justo enfrente de la superficie con las *climbing holds*, dando la espalda al Kinect y los demás dispositivos. Una representación gráfica de la ubicación de los elementos que conformaron el segundo entorno experimental se muestra en la Figura 3.10.

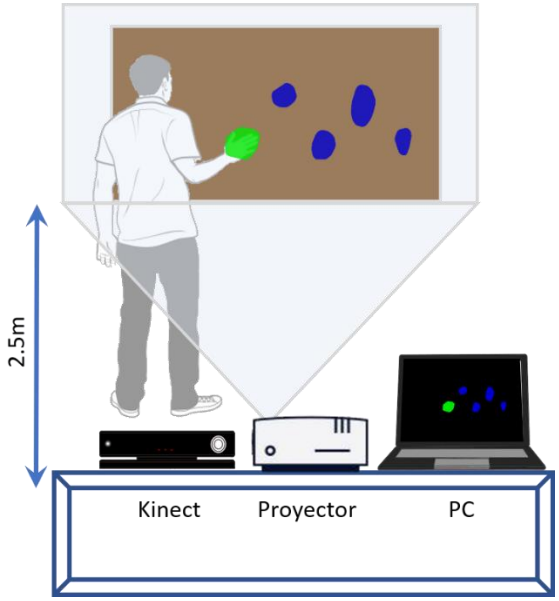


Figura 3.10. Segundo entorno experimental.

La prueba en este entorno experimental consistió en proyectar *climbing holds* virtuales de color azul sobre las *climbing holds* reales utilizando el proyector de imágenes, La interacción del usuario con las *climbing holds* se lleva acabo cuando coloca su mano sobre una de ellas y mantiene esa posición durante 0.5 segundos, al término de ese tiempo, el

sistema toma como valida la interacción y la *climbing hold* virtual cambia a color verde, la prueba se da por concluida hasta que el usuario ha interactuado con todas las *climbing holds*.

3.2.1. Resultados.

Como se mencionó al inicio de este capítulo, como preparativo inicial para realizar las pruebas en el segundo entorno experimental, se hizo una detección del área de proyección, cuyas imágenes y resultados del proceso de detección se muestran en la Figura 3.11.

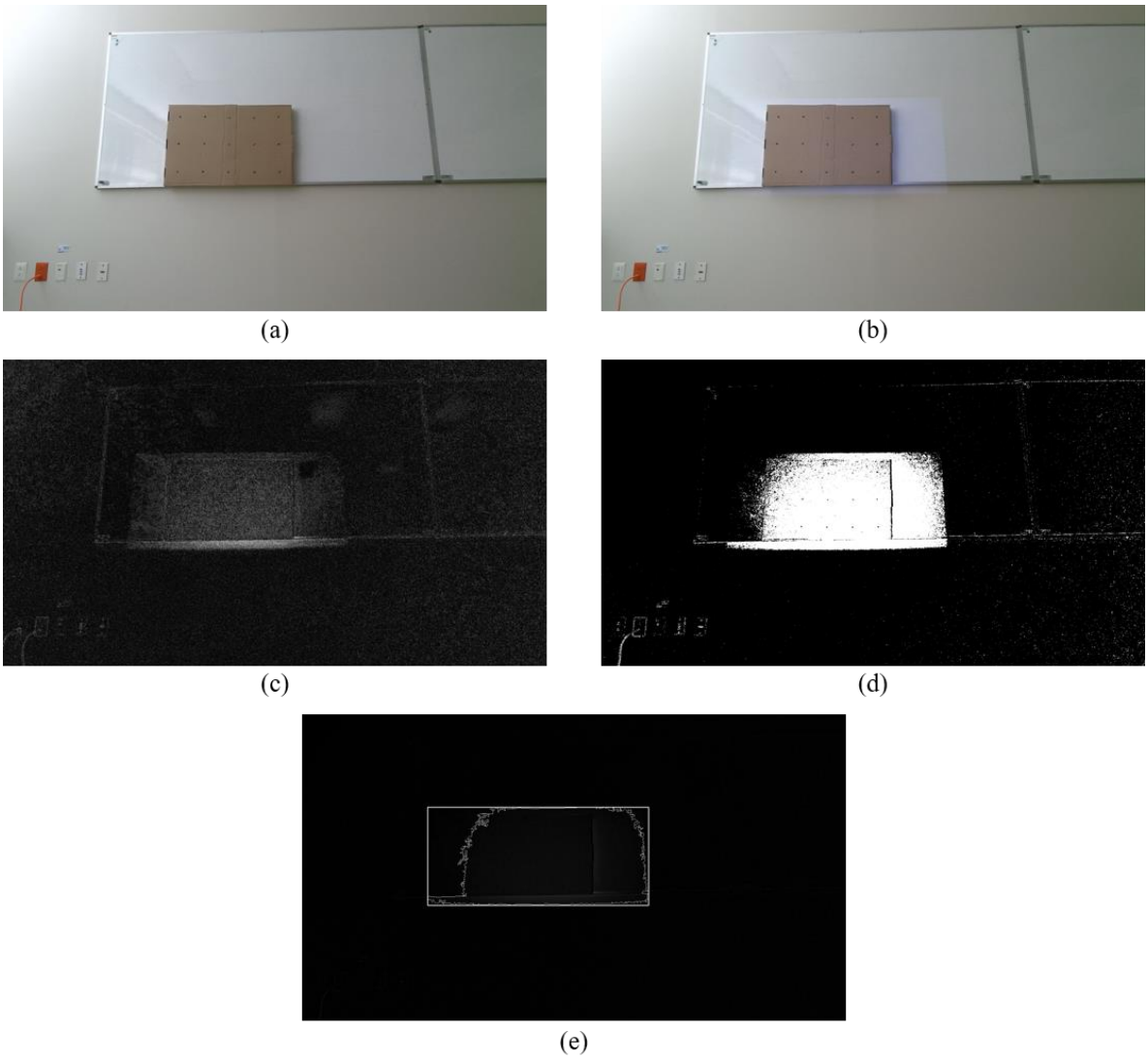
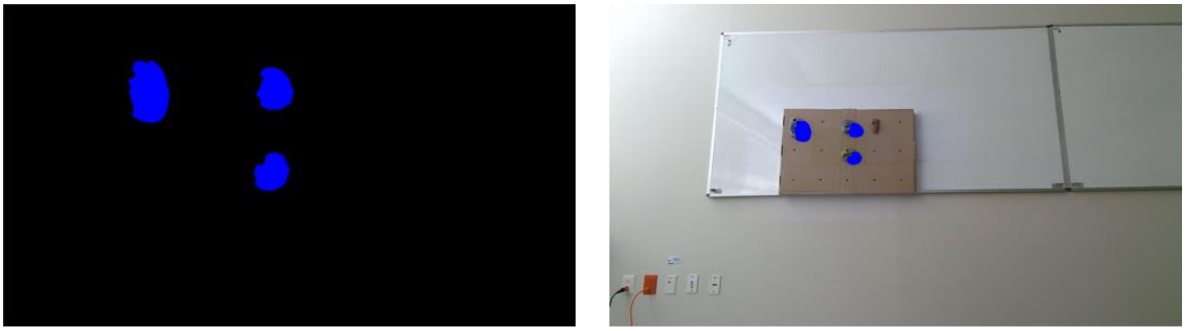


Figura 3.11. Detección del área de proyección en el segundo entorno experimental. (a) Imagen de fondo, (b) Imagen de la pantalla del proyector, (c) Sustracción de fondo, (d) Umbralización, (e) Contorno del área de proyección.

Posteriormente se llevó a cabo el proceso de calibración de imágenes, el cual tiene como propósito calcular las coordenadas de los objetos virtuales que se observan en el campo

de visión de la cámara RGB de Kinect con base a los objetos virtuales mostrados con el proyector de imágenes. El resultado del proceso de calibración se muestra en la Figura 3.12.



(a) (b)

Figura 3.12. Calibración de imágenes en el segundo entorno experimental, (a) Imagen mostrada con el proyector de imágenes, (b) Imagen calibrada al campo de visión de la cámara RGB de Kinect.

Una vez concluido el proceso de calibración se realizaron las pruebas necesarias para analizar la interacción del usuario con los objetos virtuales que, en este caso, son representaciones de las *climbing holds* de un muro de escalada. En la Figura 3.13 se muestra una secuencia de imágenes de una de las pruebas realizadas, en la cuales se puede apreciar la interacción del usuario con una *climbing hold* virtual.

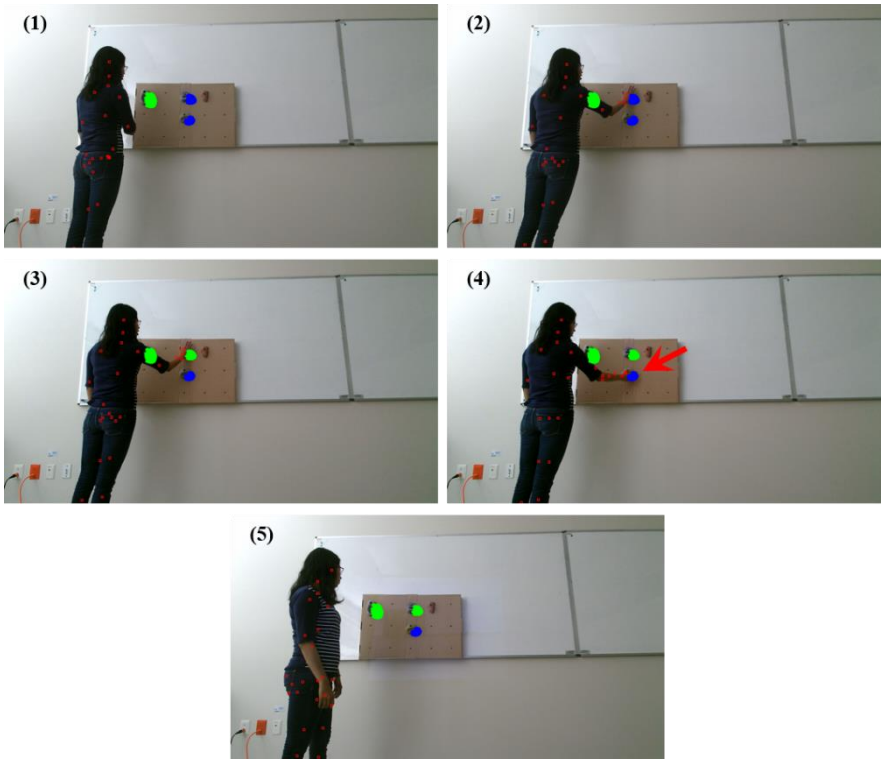


Figura 3.13. Secuencia de la interacción del usuario con objetos virtuales.

Como se puede observar en la imagen número (4) de la Figura 3.13, el sistema no detectó la interacción de la mano con la *climbing hold* virtual debido a una detección errónea de las articulaciones de la mano, las cuales están muy cercanas entre sí. En la Figura 3.14 se observa un acercamiento a esa zona, donde al dibujar los círculos delimitadores, se puede apreciar que no existe una colisión entre ambos volúmenes delimitadores.

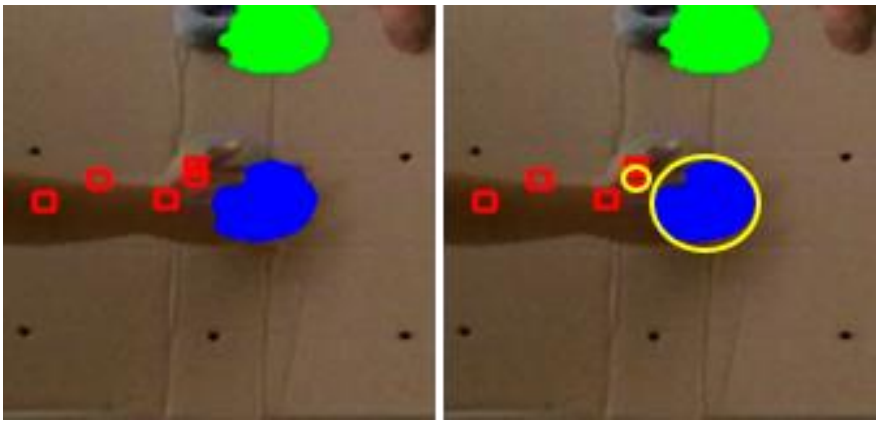


Figura 3.14. Interacción fallida entre la mano del usuario y una *climbing hold* virtual.

Una posible alternativa para dar solución a este tipo de errores sería realizar una detección inicial de las articulaciones de la mano del usuario, en la cual se genere un volumen delimitador que se mantenga con un tamaño constante durante la ejecución del programa, de esta manera si las articulaciones de la mano se juntan, el volumen delimitador mantendría su tamaño inicial.

Los resultados obtenidos en el segundo entorno experimental se muestran en la Tabla 3.5, en la cual se observa que las pruebas de la interacción del cuerpo humano con objetos virtuales que se visualizan en una pared utilizando un proyector de imágenes arrojaron un 88.89% de interacciones exitosas.

Tabla 3.5. Resultados de las pruebas realizadas en el segundo entorno experimental.

Entorno experimental	Total de pruebas	No. de prueba	Cantidad de objetos virtuales	Interacciones exitosas	Interacciones fallidas	Falsos positivos	% Interacciones exitosas	% Interacciones fallidas	% Falsos positivos
2.- Salón de clase	6	1	4	4	0	0	100.00%	0.00%	0.00%
		2	4	4	0	0	100.00%	0.00%	0.00%
		3	3	2	1	0	66.67%	33.33%	0.00%
		4	3	3	0	0	100.00%	0.00%	0.00%
		5	6	4	1	1	66.67%	16.67%	16.67%
		6	5	5	0	0	100.00%	0.00%	0.00%
						Total	**88.89%**	**8.33%**	**2.78%**

3.3. Tercer entorno experimental: Gimnasio de Escalada

El tercer entorno experimental fue en un gimnasio de escalada, donde se analizó la interacción del cuerpo humano con objetos virtuales que se visualizan sobre las rocas de escalada o *climbing holds* de un muro de escalada utilizando un proyector de imágenes.

Para realizar las pruebas en el gimnasio de escalada, se acondiciono una superficie de madera en la que se colocaron los dispositivos como el Kinect, la computadora y el proyector de imágenes, los cuales se colocaron a una distancia aproximada de 3 metros con respecto al muro de escalada, mientras que el usuario se ubicó justo enfrente del muro de escalda, dando la espalda al Kinect y los demás dispositivos. Una representación gráfica de la ubicación de los elementos que conformaron el tercer entorno experimental se muestra en la Figura 3.15.

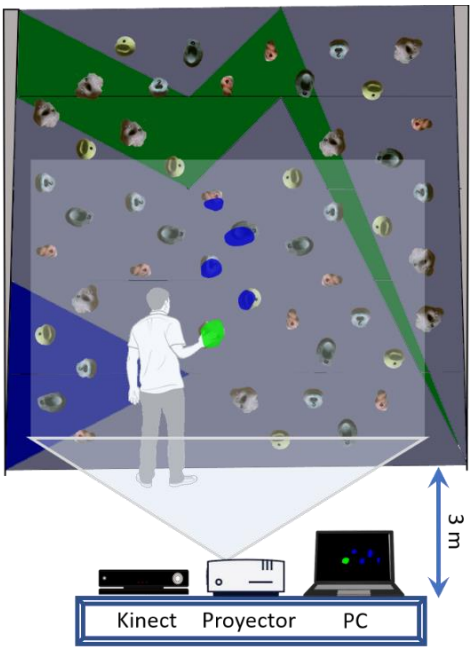


Figura 3.15. Tercer entorno experimental.

Al igual que en el segundo entorno experimental, la prueba en el tercer entorno experimental consistió en proyectar *climbing holds* virtuales de color azul sobre las *climbing holds* reales utilizando el proyector de imágenes, La interacción del usuario con las *climbing holds* se lleva acabo cuando coloca su mano sobre una de ellas y mantiene esa posición durante 0.5 segundos, al término de ese tiempo, el sistema toma como valida la interacción

y la *climbing hold* virtual cambia a color verde, la prueba se da por concluida hasta que el usuario ha interactuado con todas las *climbing holds* virtuales.

3.3.1. Resultados.

Para llevar a cabo las pruebas en el tercer entorno experimental, primero se hizo una detección del área de proyección, cuyas imágenes y resultados del proceso de detección se muestran en la Figura 3.16.

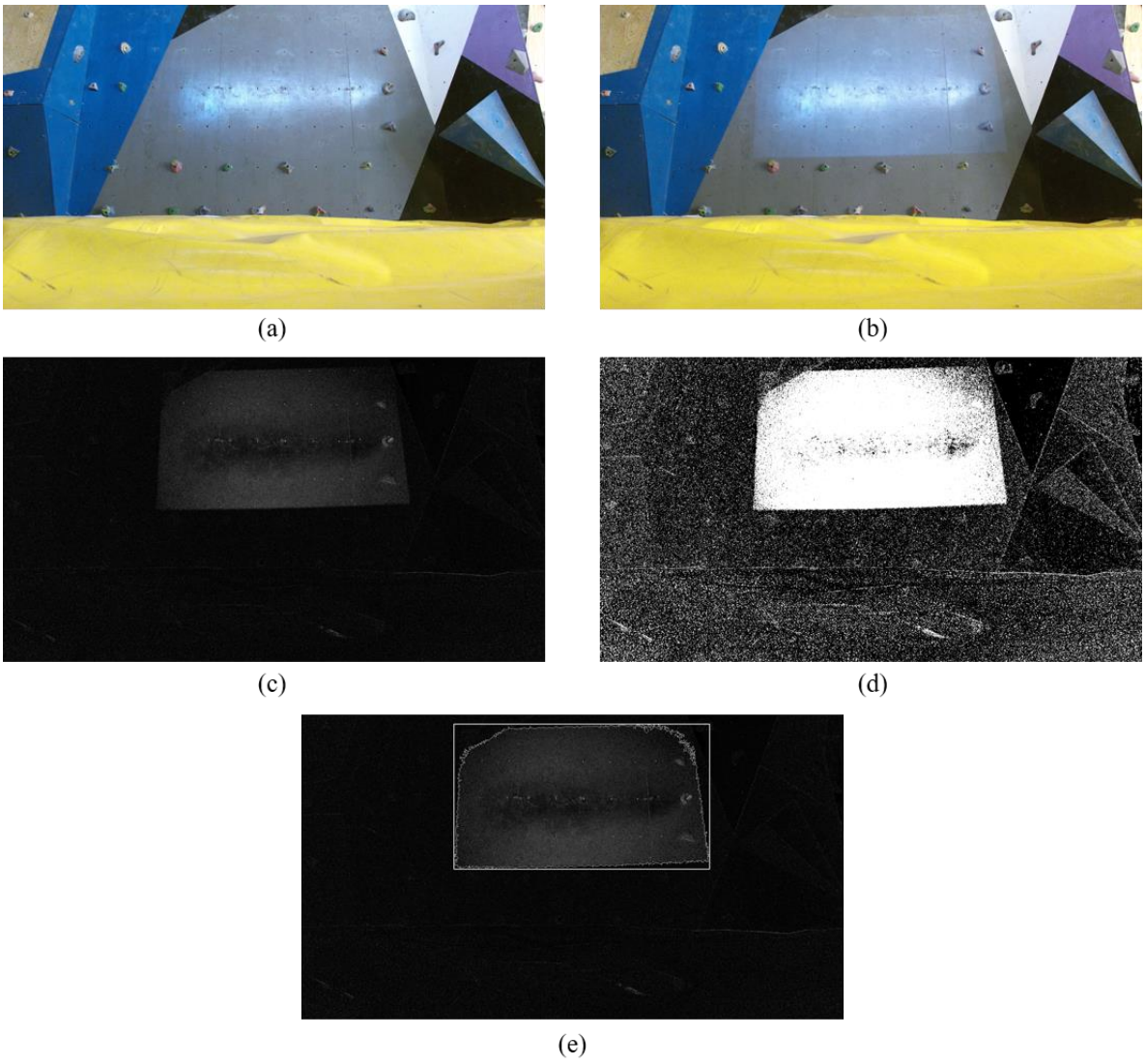


Figura 3.16. Detección del área de proyección en el tercer entorno experimental. (a) Imagen de fondo, (b) Imagen de la pantalla del proyector, (c) Sustracción de fondo, (d) Umbralización, (e) Contorno del área de proyección.

Posteriormente se llevó a cabo el proceso de calibración de imágenes, el cual tiene como propósito calcular las coordenadas de las *climbing holds* virtuales que se observan en

el campo de visión de la cámara RGB de Kinect con base a las *climbing holds* virtuales mostradas con el proyector de imágenes. El resultado del proceso de calibración se muestra en la Figura 3.17.

(a) (b)

Figura 3.17. Calibración de imágenes en el tercer entorno experimental, (a) Imagen mostrada con el proyector de imágenes, (b) Imagen calibrada al campo de visión de la cámara RGB de Kinect

Después de concluir el proceso de calibración se realizaron las pruebas necesarias para analizar la interacción del usuario con las *climbing holds* virtuales. En la Figura 3.18 Figura 3.13se muestra una secuencia de imágenes de una de las pruebas realizadas, en la cuales se puede apreciar la interacción del usuario con las *climbing holds* virtuales.

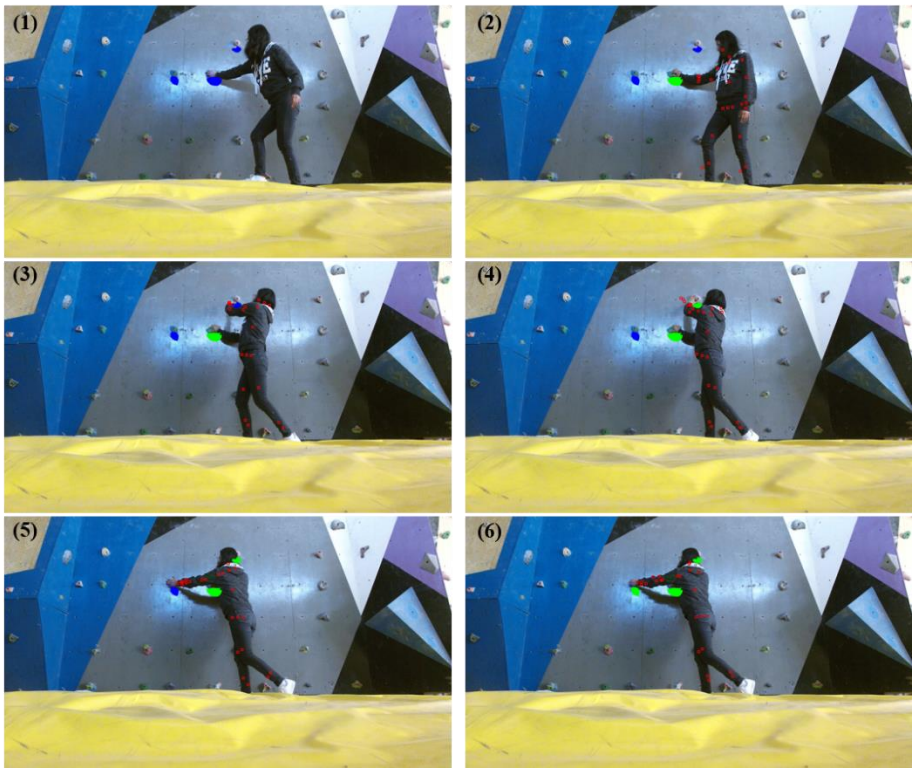


Figura 3.18. Secuencia de la interacción del usuario con *climbing holds* virtuales.

En general, en este tercer entorno experimental se obtuvieron resultados similares a los obtenidos en el salón de clase, teniendo un 91.11% de interacciones exitosas tal como se observa en la Tabla 3.6. De igual manera, se presentaron los errores donde no hay una separación entre las articulaciones de las manos, apareciendo muy juntas entre ellas, lo que propicia que no se detecten correctamente las colisiones entre los volúmenes delimitadores de las manos y las *climbing holds* reales, lo cual representa una oportunidad de mejora para un trabajo a futuro.

Tabla 3.6. Resultados de las pruebas realizadas en el tercer entorno experimental.

Entorno experimental	Total de pruebas	No. de prueba	Cantidad de objetos virtuales	Interacciones exitosas	Interacciones fallidas	Falsos positivos	% Interacciones exitosas	% Interacciones fallidas	% Falsos positivos
3.- Gimnasio de escalada	6	1	5	4	1	0	80.00%	20.00%	0.00%
		2	4	4	0	0	100.00%	0.00%	0.00%
		3	5	5	0	0	100.00%	0.00%	0.00%
		4	4	4	0	0	100.00%	0.00%	0.00%
		5	3	2	1	0	66.67%	33.33%	0.00%
		6	3	3	0	0	100.00%	0.00%	0.00%
						Total	**91.11%**	**8.89%**	**0.00%**

CAPÍTULO 4

4. Conclusiones

En el presente proyecto de investigación se desarrolló un sistema computacional básico para interactuar con objetos virtuales que se muestran en la pantalla de una computadora o que pueden ser visualizados utilizando un proyector de imágenes. Se encontró que Kinect tiende a presentar ciertas imprecisiones si no se tiene la distancia y la posición del cuerpo adecuada al situarse frente a este dispositivo, lo cual provoca que ocurran falsos positivos cuando otra parte del cuerpo toca un objeto virtual y se toma como valido, aunque no haya sido tocado con las manos.

Asimismo, se ha explorado una de las muchas aplicaciones que puede tener la visión por computadora en el área recreativa, educativa, deportiva y como avance tecnológico, al analizar la posibilidad de realizar entrenamientos en gimnasio con muros de escalada asistidos por computadora, facilitando la tarea del entrenador y haciendo las sesiones de entrenamiento más dinámicas y entretenidas. El sistema desarrollado en este proyecto puede ser utilizado como base para otros proyectos similares que pueden hacer uso del dispositivo Kinect, cuya versatilidad y bajo costo ha permitido a estudiantes, profesores e investigadores obtener grandes avances en la robótica, inteligencia artificial, reconocimiento de patrones de movimiento del cuerpo humano, entre otras áreas.

La finalidad de este proyecto de investigación es que pueda servir como base en el desarrollo de aplicaciones donde el usuario pueda manipular los objetos virtuales mostrados en la computadora usando las manos, o interactuar con objetos virtuales que se muestran utilizando un proyector de imágenes como, por ejemplo: en la creación de juegos interactivos o educativos, aplicaciones orientadas al entrenamiento de algún deporte, para la manipulación de gráficas, esquemas, diagramas u otros elementos.

Derivado de los resultados obtenidos en este proyecto de investigación, ha sido posible observar los alcances y limitaciones del sistema desarrollado.

4.1. Alcances:

- Se implementó un algoritmo de reconocimiento de movimientos de las extremidades del cuerpo humano, el cual permite obtener los datos de las articulaciones del cuerpo humano mediante el Kinect for Windows SDK 2.0.
- Se identificaron las posiciones relativas entre las extremidades del cuerpo humano y objetos virtuales en diferentes condiciones de iluminación, encontrando que Kinect suele tener un buen rendimiento bajo diferentes condiciones de iluminación.
- Se desarrolló un algoritmo básico de detección de colisiones para analizar la interacción del cuerpo humano con objetos virtuales.
- Se realizó un prueba piloto del sistema en un gimnasio de escalada.

4.2. Limitaciones:

- Para una mejor detección de las coordenadas de las manos es necesario mantenerlas hacia los lados y a una cierta separación respecto al cuerpo, debido a que, si las mantenemos junto al cuerpo, al frente o a la espalda, los datos de las coordenadas obtenidas por Kinect pueden ser poco confiables.
- Debido a la tasa de fotogramas de 30 FPS que utiliza Kinect, pueden presentarse imprecisiones en la detección de las articulaciones del cuerpo humano si se realizan movimientos muy rápidos o bruscos.
- Un exceso de luz artificial o natural puede afectar la detección del área de proyección, lo cual puede provocar que la posición de los objetos virtuales no coincida con la posición de las rocas de escalar (*climbing holds*) repercutiendo negativamente en la detección de las colisiones entre la mano del usuario y los objetos virtuales. Asimismo, puede afectar la visibilidad de los objetos virtuales si el proyector de imágenes carece de potencia lumínica.

4.3. Trabajo a futuro.

Como se ha mencionado anteriormente, el sistema desarrollado en este proyecto de tesis para reconocer la interacción del cuerpo humano con objetos virtuales es una base para

la creación de nuevos proyectos, por lo que la oportunidad de mejora es bastante amplia, a continuación, se mencionan algunos aspectos a incluir o mejorar en el trabajo a futuro.

- Investigar la posibilidad de mejorar o corregir la detección de las articulaciones del cuerpo humano mediante software, ya sea empleando otro kit de desarrollo de terceros además del Kinect for Windows SDK 2.0, o bien, analizar diversos software de código abierto.
- Analizar el desempeño de Kinect al utilizar otros gestos realizados con las manos para manipular los objetos virtuales que se muestran en la pantalla de la computadora, con el fin de indicar comandos o funciones mediante gestos.

Referente a la implementación del sistema para reconocer la interacción del cuerpo humano con objetos virtuales en los entrenamientos en gimnasios con muros de escalada se tienen contemplados las siguientes adiciones o modificaciones:

- Realizar una detección inicial de las articulaciones de las manos para configurar los volúmenes delimitadores de las manos, con el fin de establecer un valor constante para que mantengan su tamaño durante la sesión de entrenamiento.
- Mejorar el algoritmo de detección del área de proyección para escenarios donde la luz del proyector sea poco visible, o en su defecto, dar la posibilidad de que el usuario pueda seleccionar manualmente el área de proyección al dibujar un rectángulo sobre dicha área.
- Agregar un cronometro en las rutinas de entrenamiento, ya sea para que el escalador realice pruebas contrarreloj o simplemente para medir el tiempo que tardo en completar una ruta de entrenamiento. La idea es que el cronometro inicie cuando el escalador sujete la primer roca de escalar y se detenga al sujetar la última roca de escalar de una ruta de entrenamiento.
- Anadir la función de grabar las sesiones entrenamiento a elección del usuario.
- En lugar de proyectar los contornos de las rocas de escalar o *climbing holds* como objetos virtuales, utilizar círculos y permitir configurar los colores para facilitar la visibilidad de las *climbing holds* que el escalador debe sujetar para completar una ruta de entrenamiento.

- Emplear diversos tipos de volúmenes delimitadores que pudieran ser diferentes al círculo, con la finalidad de utilizar aquellos que mejor se adapten a la forma de las *climbing holds* y analizar su impacto en el rendimiento del sistema.

REFERENCIAS

[1] A. Arya and F. Farhadi-Niaki, “An Implementation on Object Move Detection Using OpenCV,” *Department of System and Computer Engineering, Carleton Univ., Ottawa*, pp. 1–5, 2010.

[2] L. Gutzeit and E. A. Kirchner, “Automatic Detection and Recognition of Human Movement Patterns in Manipulation Tasks.,” in *PhyCS*, 2016, pp. 54–63.

[3] I. S. J. A. Gómez J. P. Chavat, “Algoritmos de inteligencia computacional para la deteccion de patrones de movimiento de personas,” Universidad de la República, 2016.

[4] A. Fern’ndez-Baena, A. Susín, and X. Lligadas, “Biomechanical Validation of Upper-Body and Lower-Body Joint Movements of Kinect Motion Capture Data for Rehabilitation Treatments,” in *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems*, 2012, pp. 656–661.

[5] A. Schmidt, “Movement pattern recognition in basketball free-throw shooting,” *Human Movement Science*, vol. 31, no. 2, pp. 360–382, Apr. 2012.

[6] X. Wei, P. Zhang, and J. Chai, “Accurate realtime full-body motion capture using a single depth camera,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 188, 2012.

[7] T. M. Takala and M. Matveinen, “Full body interaction in virtual reality with affordable hardware,” in *2014 IEEE Virtual Reality (VR)*, 2014, pp. 157–157.

[8] R. E. M. d. Oliveira and J. C. d. Oliveira, “A Kinect-based Oil Platform Training Application,” in *2014 XVI Symposium on Virtual and Augmented Reality*, 2014, pp. 135–138.

[9] S. Utku and M. H. Özcanhan, “Virtual reality measurement tool: Kinect MedVision,” in *2016 20th National Biomedical Engineering Meeting (BIYOMUT)*, 2016, pp. 1–4.

[10] J. Hsu, “The Potential of Kinect as Interactive Educational Technology,” *2nd International Conference on Education and Management Technology*, vol. 13, 2011.

[11] S. Zhang, W. He, Q. Yu, and X. Zheng, “Low-cost interactive whiteboard using the Kinect,” in *2012 International Conference on Image Analysis and Signal Processing*, 2012, pp. 1–5.

[12] L. Seifert, V. Dovgalecs, J. Boulanger, D. Orth, R. Hérault, and K. Davids, “Full-body movement pattern recognition in climbingast,” *Sports Technology*, vol. 7, no. 3–4, pp. 166–173, Oct. 2014.

[13] R. Kajastila, L. Holsti, and P. Hämäläinen, “The Augmented Climbing Wall: High-Exertion Proximity Interaction on a Wall-Sized Interactive Surface,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 758–769.

[14] R. Kajastila and P. Hämäläinen, “Augmented Climbing: Interacting with Projected Graphics on a Climbing Wall,” in *Proceedings of the Extended Abstracts of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, 2014, pp. 1279–1284.

[15] F. Daiber, F. Kosmalla, and A. Krüger, “BouldAR: using augmented reality to support collaborative boulder training,” in *CHI’13 Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 949–954.

[16] J. Kim, D. Chung, and I. Ko, “A climbing motion recognition method using anatomical information for screen climbing games,” *Human-centric Computing and Information Sciences*, vol. 7, no. 1, p. 25, 2017.

[17] Microsoft, *Kinect for Windows, Human Interface Guidelines v2.0*. Microsoft Corporation, 2014.

[18] 新田善久 and others, “NtKinect: C++ Class Library for Kinect V2,” *研究報告ヒューマンコンピュータインタラクション(HCI)*, vol. 2017, no. 24, pp. 1–6, 2017.

[19] R. L. J. Caro, *Opencv Computer Vision Application Programming Cookbook (2nd Edition)*. PACKT PUB, 2013.

[20] M. Rahman, *Beginning Microsoft Kinect for Windows SDK 2.0*. Apress, 2017.

[21] Y. Nitta, “NtKinect: C++ Class Library for KinectV2,” in *the 172-th conference SIG Human-Computer Interaction*, 2017.

[22] C. S. Bamji, P. OtextquotesingleConnor, T. Elkhatib, S. Mehta, B. Thompson, L. A. Prather, D. Snow, O. C. Akkaya, A. Daniel, A. D. Payne, T. Perry, M. Fenton, and V.-H. Chan, “A 0.13 m CMOS System-on-Chip for a 512 texttimes 424 Time-of-Flight Image Sensor With Multi-Frequency Photo-Demodulation up to 130 MHz and 2 GS/s ADC,” *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 303–319, Jan. 2015.

[23] Y. Nitta and Y. Murayama, “Privacy-aware Remote Monitoring System by Skeleton Recognition,” in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[24] G. B. Adrian Kaehler, *Learning OpenCV 3*. O’Reilly UK Ltd., 2017.

[25] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. O"Reilly Media, Inc., 2016.

[26] N. Gurieva, I. Guryev, R. P. Sánchez, and E. S. Martınez, “Augmented Reality for Personalized Learning Technique: Climbing Gym Case Study,” *Open Journal for Information Technology*, vol. 2, no. 2, pp. 21–34, Dec. 2019.

[27] R. Pacheco-Sánchez, “Reconocimiento de climbing holds en muros de escalada en tiempo real.,” Universidad de Guanajuato, 2020.

[28] C. Ericson, *Real-time collision detection*. Amsterdam Boston: Elsevier, 2005.

[29] P. Strathern, *Pitágoras y su teorema*. Siglo XXI de Espana Editores, 2014.

[30] G. Tinoco, “Antecedentes: Distancia entre dos puntos,” 2018.