UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"SISTEMA DE MONITOREO DE MOVIMIENTOS DEL CUERPO HUMANO, MEDIANTE EL USO DE UNA CÁMARA DE PROFUNDIDAD"

AUTOR: JORGE ALBERTH ESCANTA ANRRANGO

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA – ECUADOR
Mayo 2019

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA
AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003087879		
APELLIDOS Y NOMBRES:	ESCANTA ANRRANGO JORGE ALBERTH		
DIRECCIÓN:	Salinas – Ibarra		
EMAIL:	jaescantaa@utn.edu.ec – jorge_escanta@hotmail.com		
TELÉFONO FIJO:	062665041	**TELÉFONO MÓVIL:**	0996433842

DATOS DE LA OBRA	
TÍTULO:	"SISTEMA DE MONITOREO DE MOVIMIENTOS DEL CUERPO HUMANO, MEDIANTE EL USO DE UNA CÁMARA DE PROFUNDIDAD"
AUTOR (ES):	JORGE ALBERTH ESCANTA ANRRANGO
FECHA: DD/MM/AAAA	17 de Mayo de 2019
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA
ASESOR /DIRECTOR:	CARLOS XAVIER ROSERO C.

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 17 días del mes de Mayo de 2019

EL AUTOR:

(Firma).....

Nombre: Jorge Escanta A.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado "SISTEMA DE MONITOREO DE MOVIMIENTOS DEL CUERPO HUMANO, MEDIANTE EL USO DE UNA CÁMARA DE PROFUNDIDAD", presentado por el egresado JORGE ALBERTH ESCANTA ANRRANGO, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, mayo de 2019

Carlos Xavier Rosero
DIRECTOR DE TESIS

Agradecimiento

A Dios por haberme dado salud, paciencia y ser mi fortaleza en cada etapa de mi vida.

A la Universidad Técnica del Norte, en especial a mi facultad por haberme permitido alcanzar mi meta de ser un profesional, a todo el personal docente por brindarme todos sus conocimientos y enseñanzas.

Al Ing. Carlos Xavier Rosero por su apoyo y guía en el desarrollo del presente trabajo.

Jorge Escanta

Dedicatoria

El presente proyecto de titulación lo dedico a mi familia, que son un pilar muy importante en mi vida, agradezco todo su apoyo incondicional, gracias a ellos he podido alcanzar esta meta profesional; a mi hermana Sandra, por sus cuidados y todo el cariño que me ha sabido brindar en todo momento, a su comprensión y sobre todo gracias por su paciencia.

A mi madre Diana, que siempre me ha guiado, y es un ejemplo de superación; a mi padre Jorge, que me ha inculcado muchos valores y sobre todo ha promovido en mí la constancia, a mi esposa Lorena, por haberme brindado su apoyo en todo momento, a mi hijos, Camila y Jorge, quienes han sido mi mayor motivación en cada momento.

A todos mis compañeros y amigos, que de una u otra manera influyeron para la culminación de este proyecto.

Jorge Escanta

Resumen

El presente trabajo de investigación nace de la necesidad de realizar un monitoreo de los movimientos que realiza el cuerpo humano, permitir su análisis y posterior tratamiento de los resultados obtenidos, ya que el análisis de dichos movimientos forman parte de la Robótica y la Biomecánica. Este tipo de investigaciones tienen también como finalidad el contribuir al desarrollo de nuevas destrezas en un investigador, y sea una base para el desarrollo de nuevas aplicaciones que permitan el análisis y estudio de la movilidad de las articulación humanas.

La aplicación a desarrollarse está elaborada en un lenguaje de programación de código abierto "open source", lo cual hace posible que esta aplicación pueda ser modificada, mejorada y adaptada según la necesidad del investigador, además muestra un sistema de monitoreo de los diferentes movimiento corporales a través de los cálculos de los ángulos de flexión de las extremidades del cuerpo. La aplicación está basada en el uso de una cámara de profundidad y los atributos que ésta posee, la cual fue desarrollado por la empresa Microsoft, está, permite captar los movimientos del cuerpo humano y virtualizarlos, a su vez se generara un esqueleto "skeleton", el que será mostrado en pantalla y reproducirá los movimientos realizados por el usuario que se posicioné frente al sistema.

Abstract

The present research work arises from the need to perform a monitoring of the movements performed by the human body, allowing its analysis and subsequent treatment of the results obtained, since the analysis of these movements are part of the Robotics and Biomechanics. This type of research also aims to contribute to the development of new skills in a researcher, and be a basis for the development of new applications that allow the analysis and study of the mobility of human articulation.

The application to be developed is developed in an open source programming language "open source", which makes it possible for this application to be modified, improved and adapted according to the needs of the researcher, in addition it shows a monitoring system of the different body movements through the calculations of the flexion angles of the extremities of the body. The application is based on the use of a depth camera and the attributes that it has, which was developed by Microsoft, is able to capture the movements of the human body and virtualize them, in turn will generate a "skeleton", which will be shown on the screen and will reproduce the movements made by the user who positioned himself in front of the system.

Índice general

Índice de figuras

Índice de diagramas de flujo

Índice de tablas

Capítulo 1

1. Introducción

1.1. Problema

El cuerpo humano es una maravilla de versatilidad debido a que tiene la capacidad de moverse con precisión, potencia y velocidad. El ser humano a diferencia de la mayoría de mamíferos, puede moverse en una amplia gama de posiciones, lo que permite la realización de una variedad infinita de tareas en todas las partes del cuerpo [1].

El estudio del movimiento humano forma parte del campo de la Biomecánica. El análisis del movimiento humano requiere de la descripción detallada de los cambios de posición del cuerpo, o de sus segmentos, así como de la identificación de las causas que lo producen [2]. Un área de extensa aplicación en visión artificial es la de los sistemas de la monitorización de movimientos del cuerpo humano [3]. Con la invención de cámaras de profundidad, se facilitó en gran medida este tipo de análisis, ya que estas proporcionan una imagen con mayor información. Esta información resulta muy útil para la detección y seguimiento de objetos, pues cuenta con la posibilidad de hacer cortes por distancia, abstrayendo el fondo por ejemplo, o situando el foco a una distancia exacta que podemos elegir [4].

En la actualidad, existen diversas aplicaciones de análisis del movimiento humano, pero estas aplicaciones son de difícil acceso para los investigadores, ya que tiene un elevado costo, o también, dichas aplicaciones poseen limitantes, lo que imposibilitan el desarrollo de nuevos estudios.

Este proyecto se enfoca en desarrollar una aplicación que realice el seguimiento y monitoreo de movimientos del cuerpo humano, a realizarse en un entorno de programación de libre distribución. Para ello se utilizará una cámara de profundidad, la cual basa su funcionamiento en el uso de una Interfaz Natural de Usuario, la que reconoce gestos, comandos de voz y objetos e imágenes [5].

1.2. Objetivos

1.2.1. Objetivo Principal

- Desarrollar una aplicación que permita el monitoreo de movimientos del cuerpo humano, mediante el uso de una cámara de profundidad.

1.2.2. Objetivos Específicos

- Recopilar información sobre el estado del arte acerca de sistemas de monitoreo de movimientos del cuerpo humano.
- Determinar los requerimientos de software y hardware necesarios para el desarrollo de la aplicación.
- Diseñar una aplicación con una interfaz natural de usuario que permita el monitoreo de los movimientos del cuerpo humano (skeleton tracking).
- Implementar la aplicación y someterla a condiciones reales de funcionamiento.

1.3. Antecedentes

En la Universidad Técnica del Norte se encontró un tema similar en el cual, se realizó un sistema para el monitoreo corporal de extremidades inferiores. En el que se utilizó un sensor Kinect para capturar y esquematizar el cuerpo humano captando sus movimientos, virtualizarlos y visualizarlos, el sistema muestra la esquematización de las extremidades inferiores del cuerpo humano y el seguimiento de sus movimientos basadas en puntos denominados "Joints" los cuales interpretan y emulan las rotaciones y flexiones que realizan las articulaciones [6].

En la Universidad Tecnológica Equinoccial se desarrolló una aplicación remota de interacción entre el profesor y el PC, que agilite el proceso enseñanza aprendizaje en las aulas de dicha universidad. En el que se usó una Interfaz Natural de Usuario (NUI) del dispositivo "Kinect". Tambien se desarrolló una aplicación en un software de programación, simulación y control. En Simulink de Matlab, se formuló una serie de algoritmos para diseñar un sistema de visión artificial, el cual permite el reconocimiento de movimientos que realiza el usuario y con esto poder convertirlos en órdenes tales como: a) avanzar de diapositiva, b) retroceder de diapositiva, c) aumentar zoom, y, c) disminuir zoom [7].

En la Universidad Superior Politécnica de Chimborazo, se realizó una investigación la cual consiste en evaluar algoritmos de tracking 3D, aplicado a la simulación de un brazo robótico, usando Kinect. La evaluación se realizó a tres algoritmos: 1) CAMShift, 2) Algoritmo por Color, y, 3) Algoritmo por Umbralización, con los cuales se implementó la misma aplicación de seguimiento de objeto (tracking). Para evaluar los resultados se aplicaron métodos estadísticos como medición y error. El siguiente paso fue realizar un estudio matemático sobre la cinemática de un brazo robótico. Este estudio se realizó en el Software Matlab, mismo que arrojó los

parámetros buscados para el uso del robot. Luego, se implementó la simulación del brazo robótico imprimiendo los resultados sobre la imagen capturada en tiempo real [8].

En la Universidad Politécnica de Madrid se realizó un documento en el cual, se presenta el diseño e implementación de un programa software en un entorno de libre distribución (GNU). Este programa extrae un modelo cinemático de una persona en movimiento y extrae patrones de movimiento, mediante el uso del dispositivo Kinect. El programa desarrollado permite tanto almacenar dicha información como enviarla a un cliente mediante una arquitectura cliente servidor [9].

Además, en la Universidad Tecnológica de Pereira se realizó un proyecto, en el cual, realizan la integración del Kinect con una línea bastante robusta como es la Robótica, utilizando el software ROS, en el que se hace uso de controladores y paquetes para obtener la esqueletización y visualización de joints (articulaciones). Para la realización de este proyecto se implementó el paquete Skeleton_Marker, también se hizo uso del controlador Nite para obtener la imagen de profundidad que garantizó la estabilidad de los joints. Una vez se accedió a la imagen, se creó un nuevo paquete para recibir la información generada por los puntos sensados; para esta tarea se implementó Transform Frame (TF) de ROS, que hace relación a una transformación de los datos generados para ser enviados; en esta secuencia se utilizó los mensajes de ROS (listener) para enviar y recibir la información de un paquete a otro [10].

1.4. Justificación

El movimiento de las partes del cuerpo humano siempre ha sido el centro de atención de muchos investigadores. Los que han buscado con diferentes dispositivos y aplicaciones poder imitar las

acciones que este realiza. Esto se ha logrado con la constante evolución tecnológica y la aparición de nuevos y mejores sistemas de captación de gestos y movimientos corporales.

Para la realización de este proyecto, y el análisis de los movimientos del cuerpo humano, se utilizará un método no invasivo, el cual involucra una cámara de profundidad. Esto se logrará gracias a que la cámara a usarse, entre una de sus características, permite el seguimiento y la captura del movimiento del cuerpo humano, lo que lo convierte en la herramienta correcta para el desarrollo de la investigación.

El presente proyecto también tendrá una gran cantidad de beneficios, entre los cuales se encuentran: a) observar y analizar los movimientos que realiza una persona, b) realizar un monitoreo de la misma, y, c) crear una plataforma para el desarrollo de nuevas aplicaciones dirigidas a niños, adultos mayores, personas con capacidades diferentes y entre otros.

Además, se pretende que este sistema sea una herramienta base para futuras investigaciones y que se convierta en una plataforma de impulso para docentes, estudiantes e investigadores. Y que contribuya al desarrollo integral e intelectual de un Ingeniero Mecatrónico.

1.5. Alcance

El proyecto a realizarse, consistirá en el desarrollo de una aplicación en un entorno de desarrollo "open source". Esta aplicación se orientará a realizar el seguimiento y monitoreo de las articulaciones más importantes del cuerpo humano.

En base a las características de las cámaras de profundidad se realizará una aplicación, la cual localizará e identificará a un usuario y trazará un esqueleto sobre este, que será visualizado en pantalla, Este esqueleto repetirá los movimientos que el usuario realice, y a su vez mostrará los diferentes ángulos de flexión de las articulaciones del cuerpo humano.

Capítulo 2

2. Revisión Literaria

2.1. Estudio de los Movimientos del Cuerpo Humano

El movimiento humano, resultado final de una actividad sinérgica[1] entre músculos y articulaciones, depende estrechamente de una dinámica multicorporal sumamente complicada. Caminar, correr, saltar, asir o lanzar son movimientos simples interrelacionados por una sucesión continua de cambios complejos de lugar y postura del cuerpo. En todos los casos, el sistema musculo esquelético se mantiene completamente controlado mediante mecanismos no explícitos que necesitan exploraciones adicionales para que se pueda simular el movimiento humano de manera realista [11].

2.2. Análisis del Movimiento del Cuerpo Humano

El análisis automático del movimiento humano es un área de investigación cuyo interés ha crecido en los últimos años [12], [13]. El análisis y síntesis del movimiento humano tiene numerosas aplicaciones de interés, como en vídeo-vigilancia, desarrollo de Interfaces de Usuario Avanzadas o en medicina [14], dicho análisis visual es de especial interés en Biomecánica.[2]

[1] Acción conjunta de varios elementos en la realización de una función.

[2] Disciplina que estudia el comportamiento del cuerpo humano sujeto a las cargas mecánicas, desde el punto de vista de la mecánica clásica.

Con el propósito de describir y analizar el movimiento del cuerpo humano, es frecuente dividirlo en segmentos funcionales, como los muestra la Figura 2.1, estos son: cabeza, cuello, tronco, muslo, pierna, pie, cintura escapular, brazo, antebrazo y mano [2], estos segmentos suelen ser llamados eslabones [15].

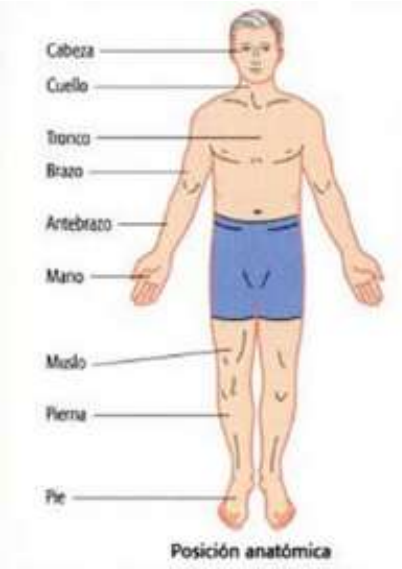

Figura 2.1. Segmentos funcionales del cuerpo [2]

2.2.1. Planos y Ejes de Movimiento del Cuerpo Humano

Para facilitar el conocimiento de la relación de estructuras entre sí y el movimiento de unos segmento respecto a otro, se ha concebido planos y ejes imaginarios de referencia que atraviesan el cuerpo de forma que unos son perpendiculares a otros, véase Figura 2.2., [15], [16].

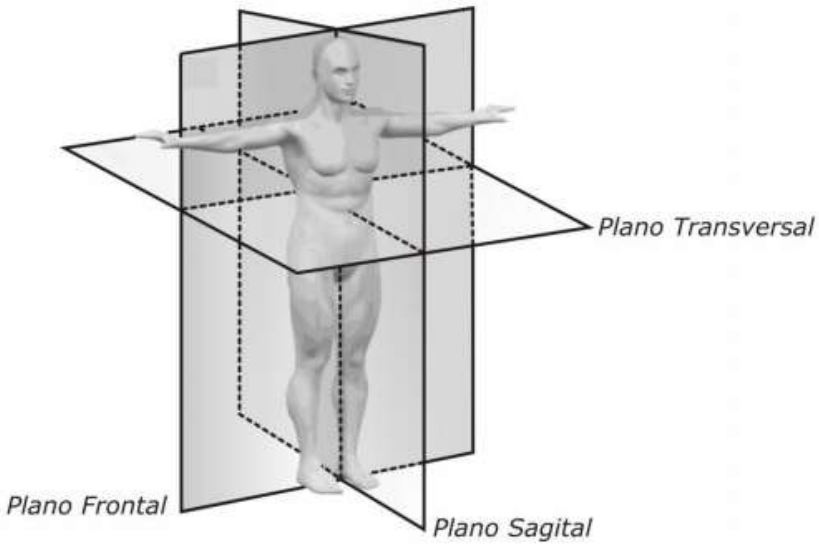

Figura 2.2. Planos y ejes anatómicos [15] [16]

2.2.2. Tipos de Movimientos Corporales

Los principales movimientos del cuerpo[3] son:

- Flexión
- Extensión
- Abducción
- Aducción
- Rotación

El cambio de posición de un cuerpo puede ser clasificado según la trayectoria que describa, estos pueden ser de traslación, de rotación y la combinación entre traslación y rotación [2].

[3] Principales movimientos del cuerpo [65], [64].

2.2.3. Limitaciones del Movimiento del Cuerpo Humano

El tipo y la amplitud de movimiento en cualquier articulación, dependen de la estructura de la articulación y del número de sus ejes, las restricciones están impuestas por los ligamentos y músculos, así como también por la masa del tejido adyacente [15].

2.3. Algoritmos de Dimensionamiento del Cuerpo Humano

La captura de los movimientos del cuerpo humano en su conjunto o en subconjuntos es una de las técnicas más versátiles usadas, no sólo en diagnósticos clínicos, o estudios biomecánicos, sino en la industria del entretenimiento o del diseño y que además puede llevarse a cabo de muchas formas, dentro de las cuales las más comunes son [17], [18]:

- Sistemas mecánicos
- Sistemas de captura ópticos
- Seguimiento magnético

La investigación dedicada a este campo en particular se puede agrupar en dos grandes categorías. En la primera categoría, los científicos investigan el seguimiento esquelético, nos referiremos a esto como estimación de postura, ya que su objetivo es lograr una aproximación de articulaciones esqueléticas más rápida o más precisa. La investigación en la segunda categoría se llama reconocimiento de actividad, ya que avanza para reconocer la actividad semántica de un ser humano en el contexto de diversas aplicaciones [19].

2.3.1. Modelos Matemáticos del Cuerpo Humano

Estos modelos consisten en la reducción de la complejidad de la geometría del cuerpo humano a sus componentes más esenciales, su finalidad es la de ofrecer una aproximación general y

consisten en su mayoría en una reducción de las diferentes geometrías que componen el cuerpo humano a formas geométricas simples y relaciones matemáticas que rigen sus dimensiones. Estas funciones matemáticas pueden partir de uno o varios parámetros antropométricos, lo que permite ajustarlos a un individuo (o familia de individuos) particular. Dentro de esta categoría, algunos modelos son [17]:

- Modelos bidimensionales
- Modelos de volúmenes
- Modelos digitales del cuerpo humano

2.3.1.1. Modelo de parámetros geométricos

2.3.1.1.1. Modelo de segmentos

Como punto de partida, se propone un modelo de segmentos basado en el de Nikolova y Toshev [20], como lo muestra la Figura 2.3., este modelo originalmente cuenta con dieciséis volúmenes, tres para cada extremidad, la cabeza, el torso superior, medio e inferior. Para esta investigación, la cabeza, el torso superior, medio e inferior se unifican en un solo volumen debido a que no son el foco del estudio y en el análisis de la marcha se pueden tomar como un solo cuerpo sin afectar el resultado de manera significativa [17].

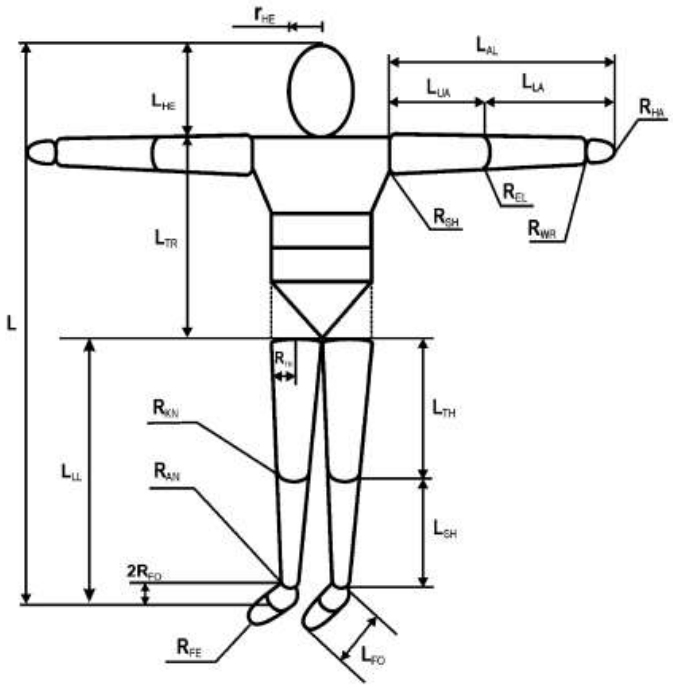

Figura 2.3. Modelo de volúmenes [17]

2.3.1.1.2. Modelo geométrico

El objetivo del modelo geométrico consiste en establecer una relación entre las medidas realizadas sobre las imágenes (bidimensionales) con el espacio de estados (de tantas dimensiones como grados de libertad posea el modelo). Con ayuda de un modelo geométrico, se puede construir una imagen sintética que representa el estado descrito por una solución perteneciente al espacio de estados, véase Figura 2.4.A [21].

2.3.1.1.3. Modelo de medida

La comparación entre la medida (imagen procedente de la cámara, que es procesada para extraer sus características interesantes) y predicción (imagen sintética, que se construye con ayuda del modelo y que simula una medida) permite obtener el grado de similitud entre ambas y asignar un peso (o medida de verosimilitud) a la solución evaluada, véase Figura 2.4.B [21].

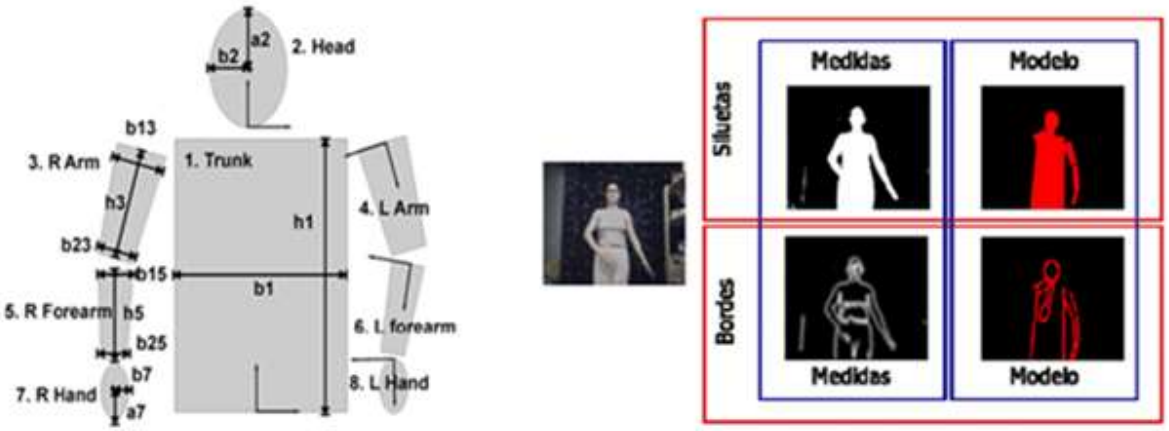

Figura 2.4. A. Modelo geométrico de las estimaciones establecidas por los métodos, B. Modelo de medida en las distintas distribuciones de las estimaciones capturadas [21]

2.3.1.2. Modelo de estimación secuencial y filtro de partículas

Existen diferentes métodos para realizar el seguimiento visual de objetos basándose en predicciones y medidas sobre el sistema. En el popular enfoque del Filtro de Kalman [22], se predice un estado con un determinado nivel de confianza y se estima la diferencia entre los datos sintéticos y las medidas [13]. El filtro de partículas (Particle Filter - PF) propone un enfoque basado en múltiples hipótesis para construir un conjunto de estados probables. El algoritmo PF permite modelar un proceso estocástico[4] mediante una función de densidad de probabilidad arbitraria, describiéndola numéricamente por medio de un conjunto de puntos, llamados partículas, pertenecientes al espacio de estados del proceso [23].

[4] Es un conjunto de variables aleatorias que depende de un parámetro o de un argumento. En el análisis de series temporales, ese parámetro es el tiempo.

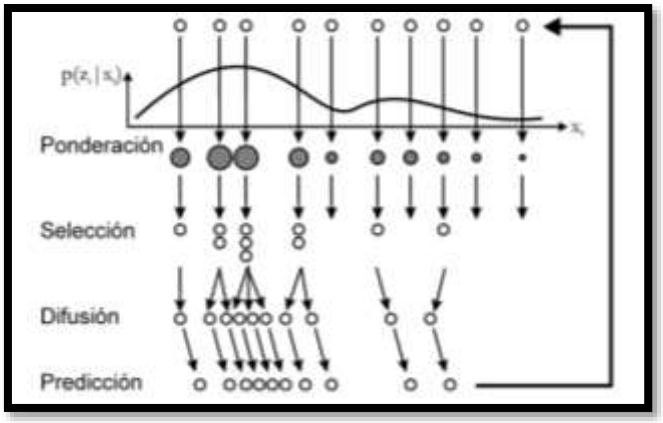

Figura 2.5. Estimación de la filtración por punto en cada movimiento [21]

La estimación secuencial y los filtros de partículas, desde un punto de vista bayesiano[5], consiste en el cálculo recursivo, con un cierto grado de confianza, del estado del sistema x_t en el instante t, utilizando para ello las observaciones z_t. Para ello, es necesario calcular la $pdf\ p(x_t|z_t)$, en dos etapas:

1. Predicción: supóngase que se dispone de $p(x_t|z_{t-1})$. La etapa de predicción implica el uso del modelo del sistema para obtener de manera recursiva la pdf a priori $p(x_t|z_{t-1})$ en el instante siguiente t, mediante la ecuación 2.1[6]:

$$p(x_t|z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})\,dx_{t-1} \tag{2.1}$$

2. Evaluación: En el instante t, se dispone de una nueva medida z_t que se puede utilizar para la actualización del estado del sistema mediante el uso de la ecuación 2.2[7]:

[5] En probabilidad, determina el cambio de una hipótesis en función de la aparición de nueva información.

[6] Ecuación de Chapman-Kolmogorov, la cual expresa: "la probabilidad de que dos hechos debidos al azar (y que cumplen unas condiciones determinadas), pasen conjuntamente es muy pequeña"

[7] Teorema de Bayes, que expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A.

$$p(x_t|z_t) = \frac{p(z_t|x_t)p(x_t|z_{t-1})}{p(z_t|z_{t-1})} \quad (2.2)$$

Estas ecuaciones conforman la base de la solución bayesiana óptima. Sin embargo, constituyen únicamente un resultado conceptual, dado que, en general, no se puede determinar analíticamente [24].

En la Figura 2.5., se muestra un esquema del funcionamiento del algoritmo PF. Su objetivo es la estimación recursiva de la pdf a posteriori $p(x_t|z_t)$, a través de un conjunto de muestras ponderadas, llamadas partículas $\{(x_{0t}, \pi_{0t}), \dots, (x_{Nt}, \pi_{Nt})\}$, donde los pesos πt están normalizados. PF comienza eligiendo una población inicial de partículas utilizando una pdf conocida. El vector de medidas z_t en el instante t, se utiliza para calcular los pesos de las partículas. A continuación, los pesos se normalizan y se calcula la población para el instante siguiente, mediante la selección de partículas con probabilidad proporcional a su peso. Como las partículas con mayor peso pueden elegirse varias veces, se aplica una etapa de difusión de forma que se evita la pérdida de diversidad en el conjunto de partículas. Finalmente, se predice la evolución de las partículas utilizando una regla de actualización [25].

2.4. Microsoft Kinect

KINECT es un sensor RGB-D que proporciona imágenes sincronizadas de color y profundidad, inicialmente fue utilizado como dispositivo de entrada por Microsoft para la consola de juegos Xbox. Con un algoritmo de captura de movimiento humano 3D, permite las interacciones entre los usuarios y un juego sin la necesidad de tocar un controlador. Recientemente, la sociedad de visión por computadora descubrió que la tecnología de detección de profundidad de Kinect podría extenderse mucho más allá de los juegos y a un costo mucho menor que las cámaras 3D

tradicionales como cámaras estéreo y cámaras time-of-flight (ToF) [26]. Además, la naturaleza complementaria de la profundidad y la información visual (RGB) proporcionada por Kinect arranca nuevas soluciones potenciales para problemas clásicos en la visión por computadora. En solo dos años después del lanzamiento de Kinect, ya han aparecido una gran cantidad de artículos científicos y demostraciones técnicas en diversas conferencias / publicaciones sobre la visión.

2.4.1. Funcionamiento del sensor Kinect

Desde el punto de vista funcional, la cámara de rango Kinect es muy similar a las cámaras ToF [26], ya que ambas estiman la Geometría 3D de escenas dinámicas, pero la tecnología Kinect actual es totalmente diferente. La cámara de rango KinectTM se basa en el chip PrimesensorTM producido por PrimeSense. Aunque Kinect es el consumidor más popular producto electrónico basado en dicho chip, no es el único, Asus X-tion Pro y X-tion Pro Live son otros productos con una cámara de rango basada en el mismo chip. Todas estas cámaras de rango, admiten una cámara de video IR y un proyector IR que proyecta en la escena patrones de luz IR para obtener una matricial implementación del principio de triangulación activa para estimar la profundidad de la escena y geometría 3D [26].

2.4.2. Diagrama de funcionamiento

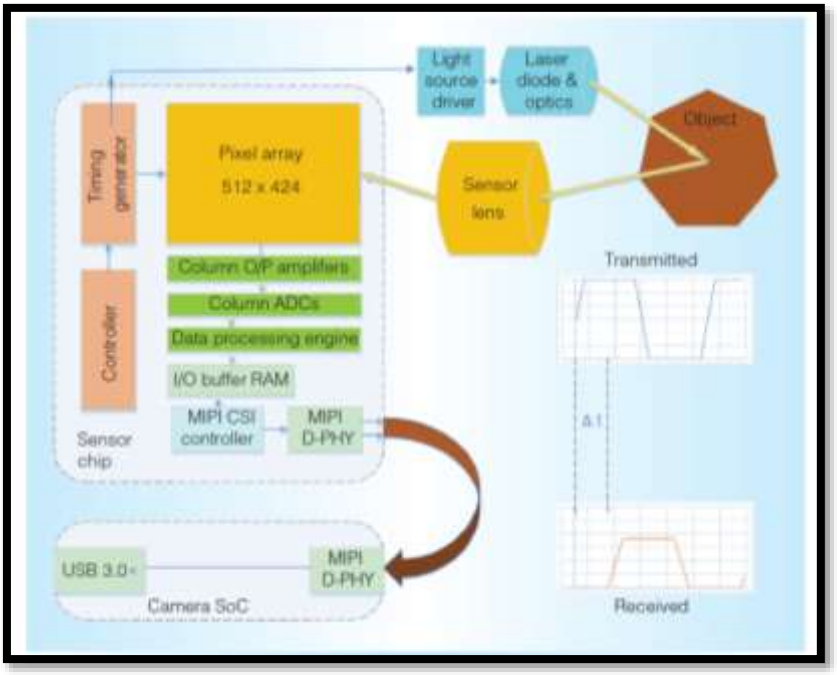

Figura 2.6. Sistema de sensor de imagen 3D [27]

La Figura 2.6., muestra el sistema de sensor de imagen 3D. El sistema consiste en el chip sensor, una cámara SoC, iluminación y óptica del sensor. El SoC gestiona el sensor y las comunicaciones con la consola Xbox One. El sistema de tiempo de vuelo modula una fuente de luz de la cámara con una onda cuadrada. Eso usa la detección de fase para medir el tiempo toma la luz para viajar desde la fuente de luz a el objeto y de vuelta al sensor, y calcula distancia de los resultados. El generador de temporización crea una modulación ola cuadrada. El sistema usa esta señal para modular tanto la fuente de luz local (transmisor) y el píxel (receptor) [27].

2.4.3. Modelo matemático del Sensor Kinect

La figura 2.7., ilustra la relación entre la distancia de un punto de objeto k al sensor con respecto a un plano de referencia y la disparidad d medida. Para expresar las coordenadas 3D de los puntos

del objeto, consideramos un sistema de coordenadas de profundidad con su origen en el centro de perspectiva de la cámara infrarroja. El eje Z es ortogonal al plano de la imagen hacia el objeto, el eje X perpendicular al eje Z en la dirección de la línea de base b entre el centro de la cámara infrarroja y el proyector láser, y el eje Y ortogonal a X y Z girando a la derecha sistema de coordenadas entregado [28].

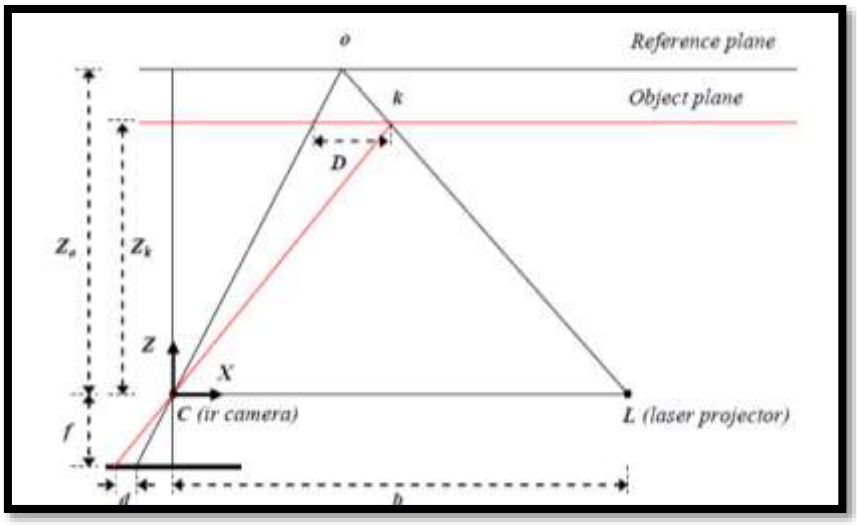

Figura 2.7. Relación entre profundidad relativa y disparidad medida [28]

Suponga que un objeto está en el plano de referencia a una distancia Zo del sensor, y se captura un punto en el objeto en el plano de la imagen de la cámara infrarroja. Si el objeto se desplaza más cerca (o más lejos) del sensor, la ubicación del punto en el plano de la imagen se desplazará en la X dirección. Esto se mide en el espacio de la imagen como la disparidad d correspondiente a un punto k en el espacio del objeto [28].

$$Z_t = \frac{Z_o}{1+\frac{Z_o}{fbd}} \tag{2.3}$$

La ecuación 2.3, es el modelo matemático básico para la derivación de la profundidad de la disparidad observada siempre que los parámetros constantes Z_o, f y b puedan determinarse

mediante calibración. La coordenada Z de un punto junto con f define la escala de imagen para ese punto. Las coordenadas del objeto planimétrico de cada punto se puede calcular a partir de sus coordenadas de imagen y la escala,

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x) \tag{2.4}$$

$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y) \tag{2.5}$$

donde X_k y Y_k son las coordenadas de la imagen del punto, x_o y y_o son las coordenadas del principal punto, y δx y δy son correcciones para la distorsión de la lente, para lo cual varios modelos con diferentes coeficientes existen. Tenga en cuenta que aquí suponemos que el sistema de coordenadas de la imagen es paralelo a la línea base y, por lo tanto, al sistema de coordenadas de profundidad.

2.5. Estructura de Captura Kinect

La figura 2.8., muestra la disposición de un sensor Kinect, que consiste en un proyector de infrarrojos (IR), una cámara IR y una cámara RGB. El sensor de profundidad comprende el proyector IR y la cámara IR, para su funcionamiento el proyector IR arroja un patrón de puntos en la escena 3D, mientras que la cámara IR captura el reflejado de las manchas del IR, Kinect es por lo tanto un sensor de profundidad de luz. La relación geométrica entre el IR proyector y la cámara IR se obtiene a través de un procedimiento de calibración fuera de línea en el cual. La profundidad de la escena es invisible a la cámara RGB, pero puede ser vista por la cámara IR. Como cada patrón local de puntos proyectados es único, entre los patrones de puntos locales observados en la imagen con los patrones de puntos calibrados del proyector es factible. La profundidad de un punto se

puede deducir por la relativa traducción izquierda-derecha del patrón de puntos. Esta traducción cambia, depende de la distancia del objeto al plano de la cámara-proyector [19].

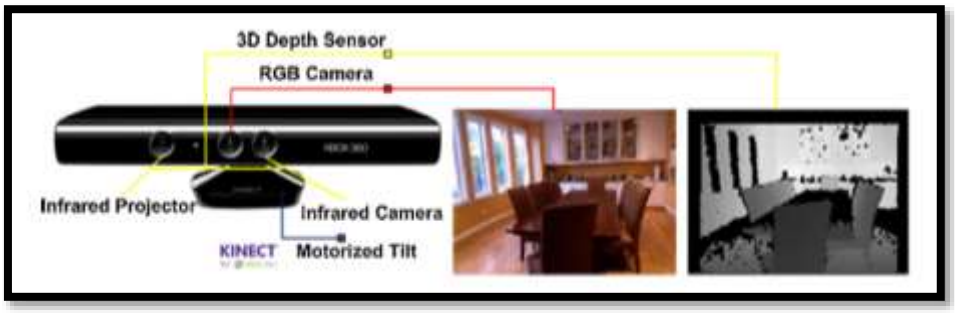

Figura 2.8. Configuración de hardware de Kinect [19]

La estructura de registro del Kinect puede otorgar un abanico de distintos tipos de capturas como se puede ver en la Figura 2.9., a continuación:

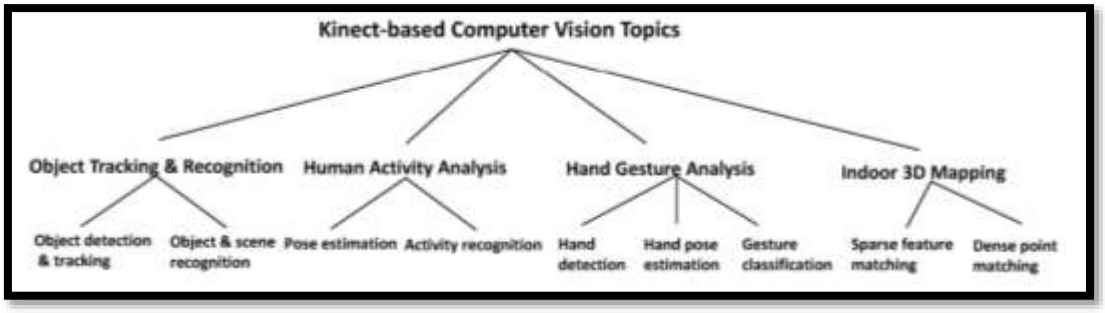

Figura 2.9. Tipos de visión del sensor Kinect [19]

2.6. Propuesta de desarrollo

En la Figura 2.10, se presenta una propuesta de desarrollo del sistema de monitoreo de los movimientos del cuerpo humano, en la cual se muestran los diferentes subprocesos que intervienen para obtener el resultado final,

El primer paso para la realización del sistema es la obtención de imágenes RGB y de profundidad, para luego pasar al segundo subproceso en el que se detecta al Usuario, el cual se

convierte en la persona en analizar los diferentes movimientos que realice, en el tercer bloque se obtienen los puntos de interés o punto a seguir del usuario detectado, estos puntos se utilizan en el cuarto bloque, el cual consiste en trazar líneas rectas que unen estos puntos formando un esqueleto de seguimiento, en el quinto bloque se procede a calcular los ángulos formados por las líneas que conforman el esqueleto de seguimiento, para finalmente en el sexto y último bloque unir estos dos datos y visualizarlos en pantalla.

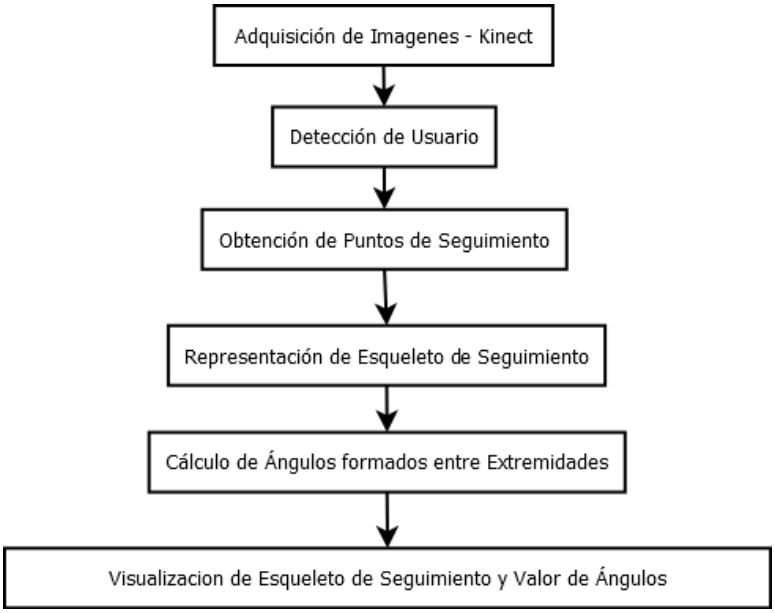

Diagrama de Flujo 2.1. Sistema de Monitoreo de Movimiento del Cuerpo Humano

Capítulo 3

3. Metodología

Para el desarrollo del sistema de monitoreo es necesario considerar los diferentes aspectos que intervienen en este proceso, entre ellos tenemos; el hardware utilizado (sensor de profundidad, equipo computacional), los requerimientos de software (sistema operativo, lenguaje de programación, librerías), así como también determinar las condiciones de funcionamiento (escenario, ambiente, iluminación, cantidad de usuarios, etc.).

3.1. Requerimientos y configuración de hardware y software

3.1.1. Hardware Utilizado

- Kinect XBOX 360 [29], [30], [31]
- Conector USB Kinect para PC
- Computadora

3.1.2. Requerimientos de Software

Para la implementación del sistema de monitoreo se emplearan distintas herramientas de desarrollo:

- Ubuntu 14.04 – Sistema Operativo GNU/Linux

- Python 2.7 – Lenguaje de Programación [32][8]
- PyCharm – (IDE -Integrated Development Environment), Entorno de Desarrollo de la aplicación [33][9]
- Librerías empleadas:
    - OpenKinect – Conjunto de herramientas para uso del Kinect. [34][10]
    - OpenCV – Conjunto de herramientas para visión por computador [35][11]
    - OpenNI/Nite – Conjunto de herramientas que permite la interacción natural con el dispositivo [36][12]

3.2. Condiciones de funcionamiento del sistema de monitoreo

Hay que tomar en cuenta parámetros importantes para el correcto funcionamiento del sistema, tales como:

- Distancia máxima de reconocimiento
- Resolución de datos – imágenes
- Procesamiento de datos de seguimiento
- Carga computacional del sensor Kinect
- Alimentación eléctrica.

De la misma manera el funcionamiento de sistema de monitoreo dependerá de factores externos como:

[8] Proceso de instalación del lenguaje de programación Python 2.7: [71]
[9] Proceso de instalación de IDE – PyCharm: [72]
[10] Proceso de instalación librería OpenKinect/libfreenect: [74]
[11] Proceso de instalación librería OpenCV: [75]
[12] Proceso de instalación librería OpenNI: [73]

- Cantidad de luz presente
- Reflexión sobre el ambiente de trabajo
- Objetos que no permitan la interacción entre el emisor y receptor infrarrojo del Kinect.

3.2.1. Implementación del sistema

Para la implementación del sistema de monitoreo es necesario tomar en cuenta algunas condiciones para que funcione con normalidad, entre ellas se menciona el escenario en donde se desarrollaran las pruebas, la distancia usuario – cámara de profundidad y el posicionamiento de la cámara.

3.2.1.1. Escenario

Para realizar la fase de implementación y pruebas de funcionamiento del sistema de monitoreo se eligió un escenario de aproximadamente $10m^2$, el cual contaba con iluminación tanto artificial como natural, véase Figura 3.1., en la cual se obtienen diferentes perspectivas del escenario en donde se realizó las diferentes pruebas de funcionamiento del sistema de monitoreo.

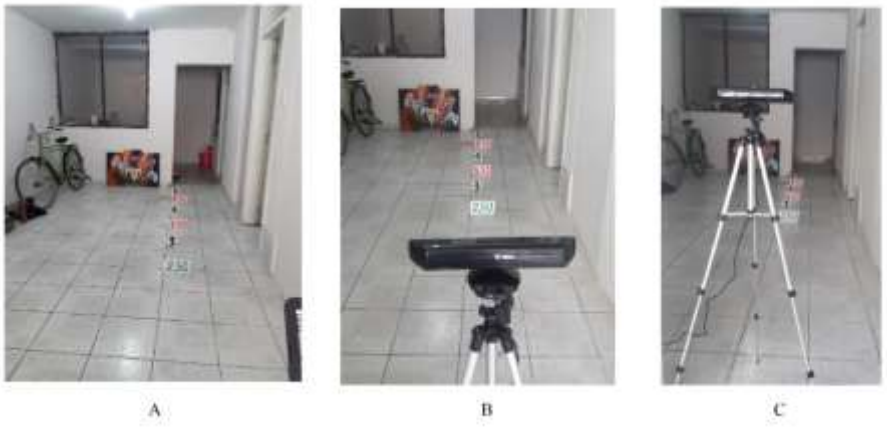

Figura 3.1. Escenario para la realización de pruebas de funcionamiento

3.2.2. Distancia usuario cámara de profundidad

El rango de funcionamiento óptimo de la cámara de profundidad es entre ± 1m - ± 4m, además de un ángulo de visión horizontal de 70° y visión vertical 60° [37] , como se muestra en la Figura 3.2.

Figura 3.2. a. Rango de funcionamiento de la cámara RGB; b. Rango de funcionamiento de la cámara de profundidad [37]

3.2.2.1. Ubicación de la cámara de profundidad – usuario

Luego de la realización de varias pruebas, se determinó que la distancia recomendable para el óptimo funcionamiento del sistema es de alrededor de 2.5 m con respecto al usuario, a una altura de 1.10 m con respecto al piso, y una inclinación de 10° con respecto al eje horizontal, como lo muestra la Figura 3.3.

Figura 3.3. A. Distancia Cámara de profundidad – Usuario; B. Ubicación cámara de profundidad respecto al piso.

3.2.3. Configuración general de la cámara de profundidad – Kinect

La configuración general de la cámara de profundidad es una librería que contiene las propiedades importantes para realizar las operaciones de trabajo con OpenNI algunas de estas propiedades se las considera como constantes durante la ejecución del sistema, véase Anexo 1.

3.3. Desarrollo del sistema de monitoreo de movimiento del cuerpo humano

3.3.1. Diagrama de bloques

Este punto nos permite tener una apreciación más simple del funcionamiento del sistema de trabajo, además de los elementos que lo componen. El sistema consta principalmente de una cámara de profundidad – Kinect conectado a un computador, el mismo que se encargara de procesar y analizar los datos obtenidos por parte de la cámara de profundidad.

Diagrama de Flujo 3.1. Diagrama del sistema de trabajo

Como se puede apreciar en el Diagrama de Flujo 3.1., se presentan 4 etapas antes de obtener el resultado final, a continuación se detalla cada una de estas:

- El primer bloque consta del sensor Kinect que será conectado mediante su adaptador a la fuente de alimentación y a la vez al computador con su adaptador de Xbox a USB.
- El segundo bloque consta de un ordenador con un sistema operativo GNU/Linux - "Ubuntu 14.04" y el IDE - "PyCharm" que hace usos de las librerías "OpenKinect – Libfreenect", "OpenCV" y "OpenNI" usadas para la adquisición y manipulación de datos entregados por la cámara de profundidad.

- El tercer bloque consiste en la obtención de puntos de seguimiento y en la visualización de esqueleto de seguimiento del usuario.
- En el cuarto bloque se muestran los ángulos formados entre las diferentes extremidades, mismos que se visualizan en pantalla.

3.3.2. Adquisición de imágenes

Para la adquisición de imágenes por parte de la cámara de profundidad se emplean los aplica la configuración básica, mediante la cual se adquieren imágenes RGB y de profundidad los cuales permiten digitalizar imágenes y mediante los módulos de OpenCV se crea las pantallas, y además se usa el módulo Freenect[13] para la inicialización de la obtención de datos.

3.3.2.1. Obtención de imágenes RGB

En el Diagrama de Flujo 3.2., se observa el proceso para la adquisición de imágenes RGB por parte de la cámara, en el cual luego de iniciar el programa la cámara adquiere imágenes esperando a que el programa sea interrumpido o finalizado por completo, para lo cual hace uso del código de programa en el Anexo 2.

[13] Libfreenect, es una biblioteca multiplataforma que proporciona las interfaces necesarias para activar, inicializar y comunicar datos con el hardware de Kinect.

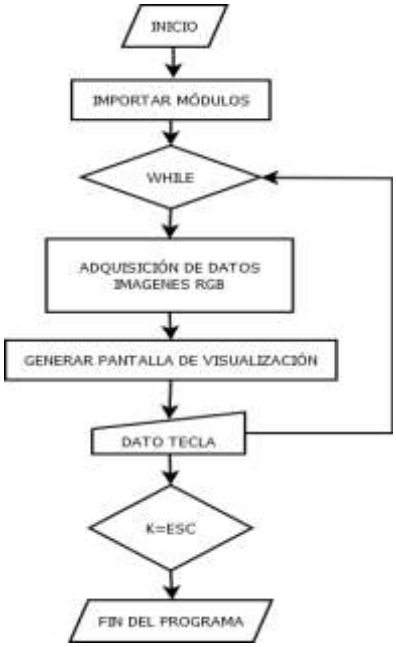

Diagrama de Flujo 3.2. Obtención de imágenes RGB

En la Figura 3.4., se observa la imagen RGB obtenida de un escenario de prueba, en donde se muestra una imagen bidimensional, es decir solamente se muestran dos planos de trabajo.

Figura 3.4. Imagen RGB obtenida

3.3.2.2. Obtención de imágenes con profundidad

En la obtención de imágenes con profundidad se hace referencia al Anexo 3, en donde se muestra el código de programa utilizado, además en el Diagrama de Flujo 3.3., se muestran los subprocesos que intervienen en la obtención de imágenes con profundidad, en los cuales se importan los módulos necesarios para la adquisición de imágenes y el programa se mantiene corriendo hasta que sea interrumpido o finalizado.

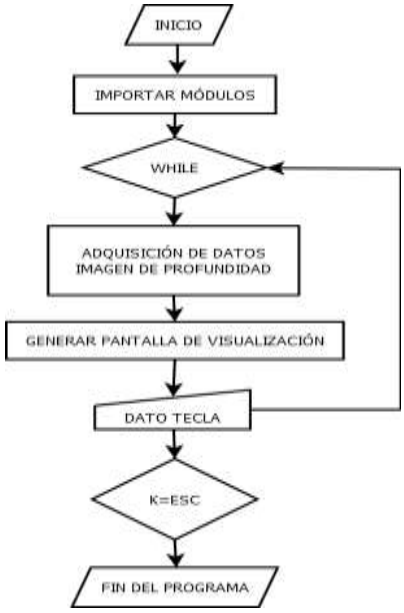

Diagrama de Flujo 3.3. Obtención de imágenes con profundidad

En la Figura 3.5., se puede observar el resultado de la captura de imágenes con profundidad, en donde en no se observan colores RBG, más bien se notan coloraciones grisáceas, de las cuales dependiendo de la distancia hacia el objeto varia su intensidad, siendo más oscura para aquellos objetos que se encuentran a distancias cortas y más tenues o blanquecinas para objetos con distancias mayores.

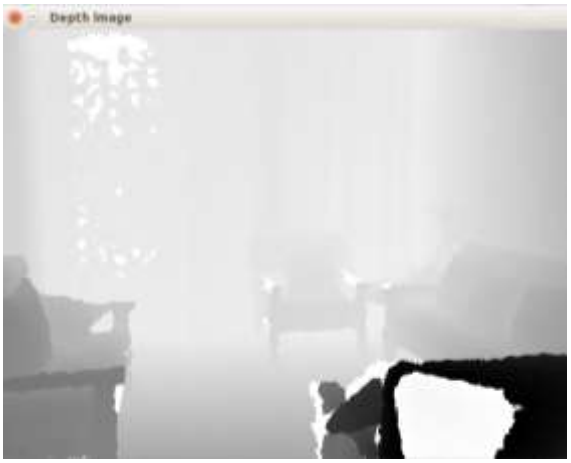

Figura 3.5. Imagen con profundidad

3.3.2.3. Obtención de imagen RGB y de profundidad por la cámara KINECT

Para la obtención de las imágenes RGB y con profundidad se utilizó el Diagrama de Flujo 3.4., el cual describe el comportamiento del código del programa mostrado en el Anexo 4.

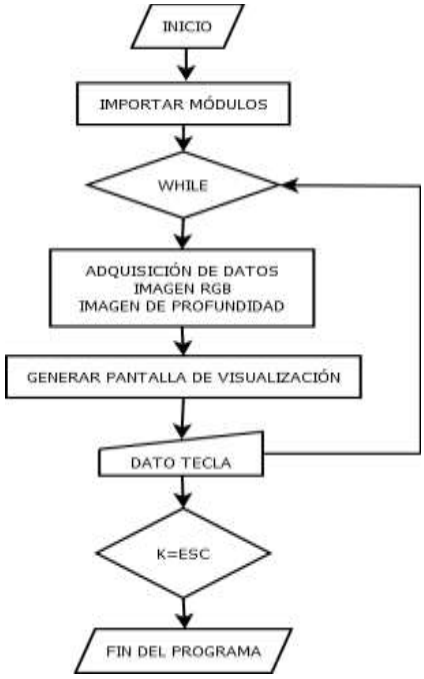

Diagrama de Flujo 3.4. Obtención de imágenes RGB y con profundidad

En la Figura 3.6., se puede observar conjuntamente las imágenes RGB (véase Figura 3.6.A) y con profundidad (véase Figura 3.6.B) obtenidas, en donde la Figura 3.6.B, muestra la variación de coloración de los objetos dependiendo de la distancia que se encuentre con respecto a la cámara.

Figura 3.6. A. Imagen RGB, B. Imagen con profundidad

3.3.3. Detección de usuario

La cámara de profundidad permite la detección de más de un usuario, para lo cual se partió de la detección de hasta cuatro usuarios, en el Anexo 5, se muestra el código utilizado para la detección y perdida de varios usuarios, además el Diagrama de Flujo 3.5., muestra los diferentes subprocesos que intervienen en la detección del usuario.

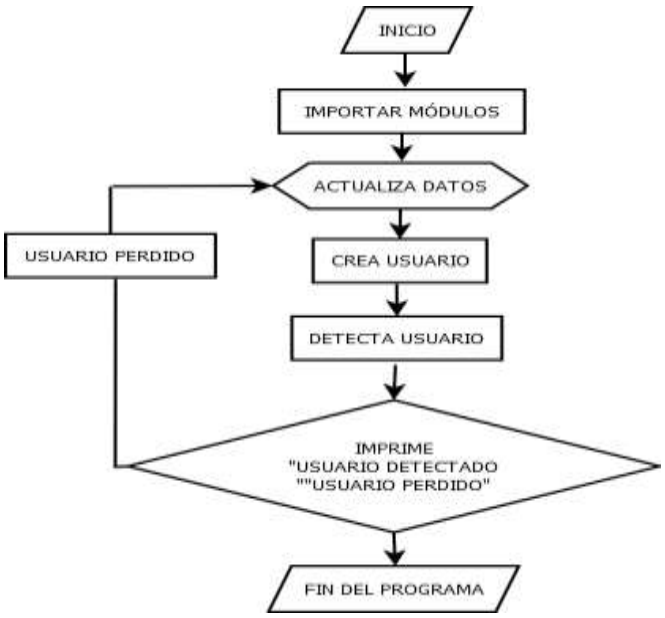

Diagrama de Flujo 3.5. Detección de usuario

Para la realización de la detección de usuarios, se utilizó un usuario, dado como resultado lo mostrado en la Figura 3.7., cuando el usuario es detectado por el sistema y lo mostrado en la Figura 3.8., cuando el usuario es perdido o desaparece del rango de funcionamiento de la cámara, mismo resultado se muestra en pantalla, indicando la situación actual del usuario.

Figura 3.7. Detección de Usuario

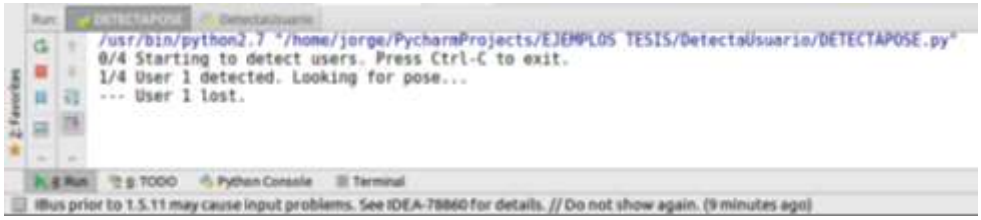

Figura 3.8. Usuario perdido

3.3.4. Obtención de puntos de seguimiento

Usando el demo de OpenNI – Skeleton, donde se muestra cómo identificar cuándo se detecta un nuevo usuario, buscar una pose para ese usuario, calibrar a los usuarios cuando están en la pose y rastrearlos. Específicamente, imprime la ubicación de la cabeza de los usuarios, ya que son rastreados. Este código al igual que los otros realizados en pruebas anteriores sirven como referencia para realizar el código de Skeleton tracking.

3.3.4.1. Pose "PSI" en OpenNI

El rastreo del esqueleto consta de procesar las imágenes de profundidad obtenidas con la cámara de profundidad para detectar formas humanas e identificar las partes del cuerpo del usuario presente en la imagen. El rastreo del esqueleto de OpenNI /NITE se inicia al realizar la pose de calibración "PSI"[14] levantando los brazos durante dos segundos como se muestra en la Figura 3.9.,

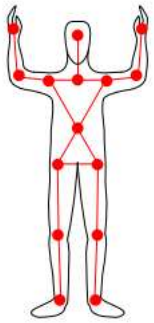

Figura 3.9. Pose "psi" de inicio de rastreo del esqueleto OpenNI /NITE [38]

Para la detección y calibración de un usuario, se utilizó el código de programa mostrado en el Anexo 6, el cual permite la detección y calibración de hasta 4 usuarios, véase Figura 3.10, cuando el usuario es detectado y Figura 3.11., cuando el usuario se encuentra en Posición "PSI" para inicio

[14] PSI (Platform for Situated Intelligence)

de calibración, además en el Diagrama de Flujo 3.5., se muestra el proceso de detección y calibración de un usuario.

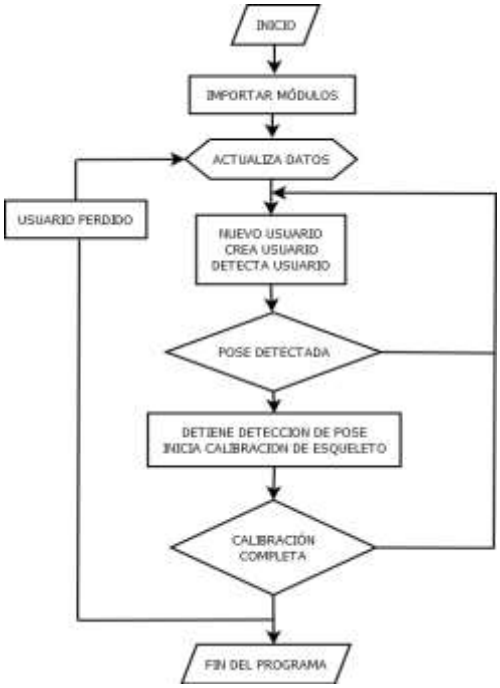

Diagrama de Flujo 3.6. Detección y calibración de usuario con pose “PSI”

Figura 3.10. Detección de usuario

Figura 3.11. Usuario en posición inicial calibrado

3.3.5. Representación del esqueleto de seguimiento

Una vez detectado y calibrado el usuario, se obtienen los punto de interés los cuales se visualizan en pantalla mediante la unión de líneas rectas entre estos, formado un esqueleto de seguimiento, el cual imita los diferentes movimientos de las extremidades del usuario, en el Diagrama de Flujo 3.7, se muestra el proceso de esqueletización bloque a bloque, en donde se utilizan las variables de los procesos anteriores para ello y además muestra la constante actualización de datos del sistema, el código de programa utilizado se encuentra en el Anexo 7.

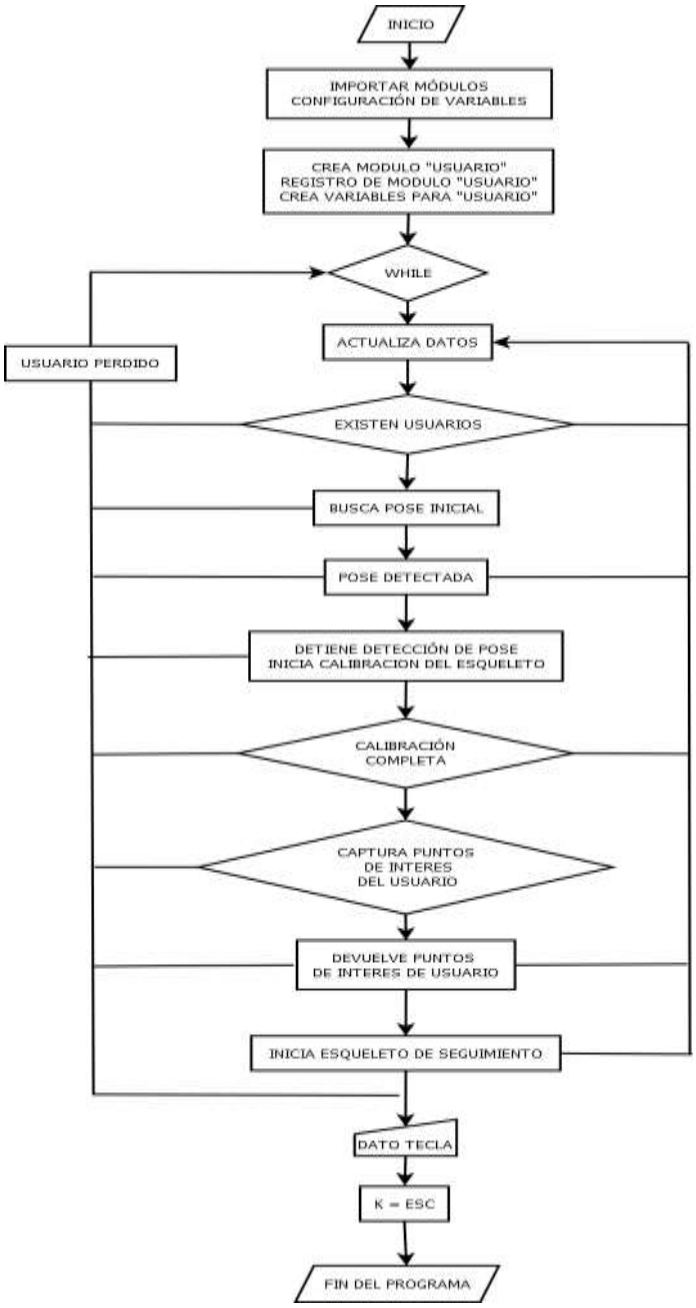

Diagrama de Flujo 3.7. Esqueleto de seguimiento

El resultado obtenido muestra dos pantallas (véase Figura 3.12.), en la primera se observa al usuario realizando diferentes movimientos, mientras que en la segunda se muestra al usuario más un esqueleto de seguimiento, el cual representa a los punto obtenidos luego de la calibración del usuario, además es capaz de imitar los movimientos del usuario en tiempo real.

Figura 3.12. Generación de esqueleto de seguimiento

3.3.6. Cálculo del ángulo entre articulaciones

El código de programa utilizado para el cálculo del ángulo formado entre dos rectas (véase Anexo 8), nos permite obtener los valores para la visualización en la pantalla, para realizar este cálculo, se hace uso de las pendientes de las rectas en que forman el ángulo a medir, para lo cual se obtiene el valor de la pendiente de las rectas a analizar utilizando la ecuación de la pendiente de la recta (3.1),

$$m = \frac{y2-y1}{x2-x1} \tag{3.1}$$

una vez obtenidos los valores de las pendientes de las rectas, se utilizan los valores encontrados para el cálculo del ángulo formado entre dichas rectas, se lo hace utilizando la ecuación (3.3), para el cálculo de ángulos entre dos rectas.

$$Tan \propto = \left|\frac{m2-m1}{1+m2*m1}\right| \tag{3.3}$$

Como resultado se obtiene lo mostrado en la Figura 3.13, en donde se observan los alores obtenidos luego del proceso de cálculo de pendientes de rectas de ángulos formados entre dos rectas.

Figura 3.13. Visualización de esqueleto de seguimiento y valores de ángulos formados entre extremidades

3.4. Sistema de monitoreo de movimientos del cuerpo humano

Una vez determinadas todas las variables que intervienen en el desarrollo del sistema de monitoreo, se obtiene un código de programa (véase Anexo 9), el cual es capaz de monitorear los movimiento que realice un usuario, además permite la visualización en pantalla de los ángulos formados entre las principales extremidades del cuerpo humano, como lo muestra a Figura 3.14., a continuación en el Diagrama de Flujo 3.8., en el cual se muestra los diferentes subprocesos que interviene hasta la visualización en pantalla del esqueleto de seguimiento conjuntamente con los valores de los ángulos que forman las extremidades, estos procesos son:

- Importación los módulos y la configuración inicia para la cámara de profundidad,

- Creación de una base de datos en donde se guardarán y actualizaran los datos que entregue la cámara de profundidad, en este caso se le llama “USUARIO”,
- Detección de usuario, calibración y búsqueda de pose que dé inicio al sistema de monitoreo,
- Captura y esquematización de puntos de interés del usuario,
- Inicialización de esqueleto de seguimiento haciendo uso de los puntos obtenidos,
- Cálculo de los ángulos formados entre las líneas rectas que unen los puntos de interés del usuario.
- Visualización de esqueleto de seguimiento y de valores de ángulos formados entre rectas del esqueleto de seguimiento, en este caso se visualizan 6 los valores de ángulos formados entre las extremidades:
    - Hombro – Antebrazo izquierdo
    - Hombro – Antebrazo derecho
    - Antebrazo – Brazo izquierdo
    - Antebrazo – Brazo derecho
    - Muslo – tobillo izquierdo
    - Muslo – tobillo derecho

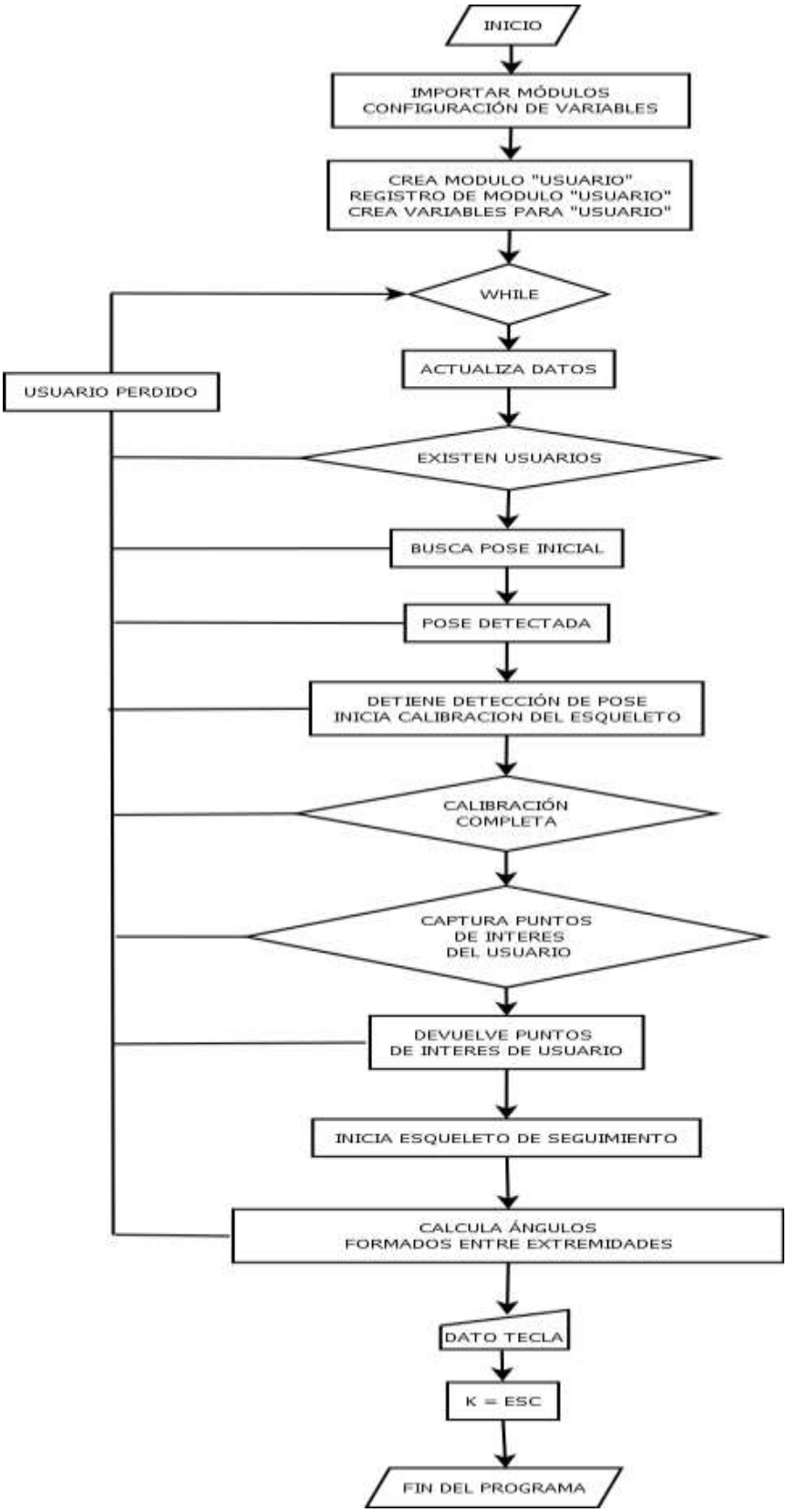

Diagrama de Flujo 3.8. Sistema de monitoreo de movimiento del cuerpo humano

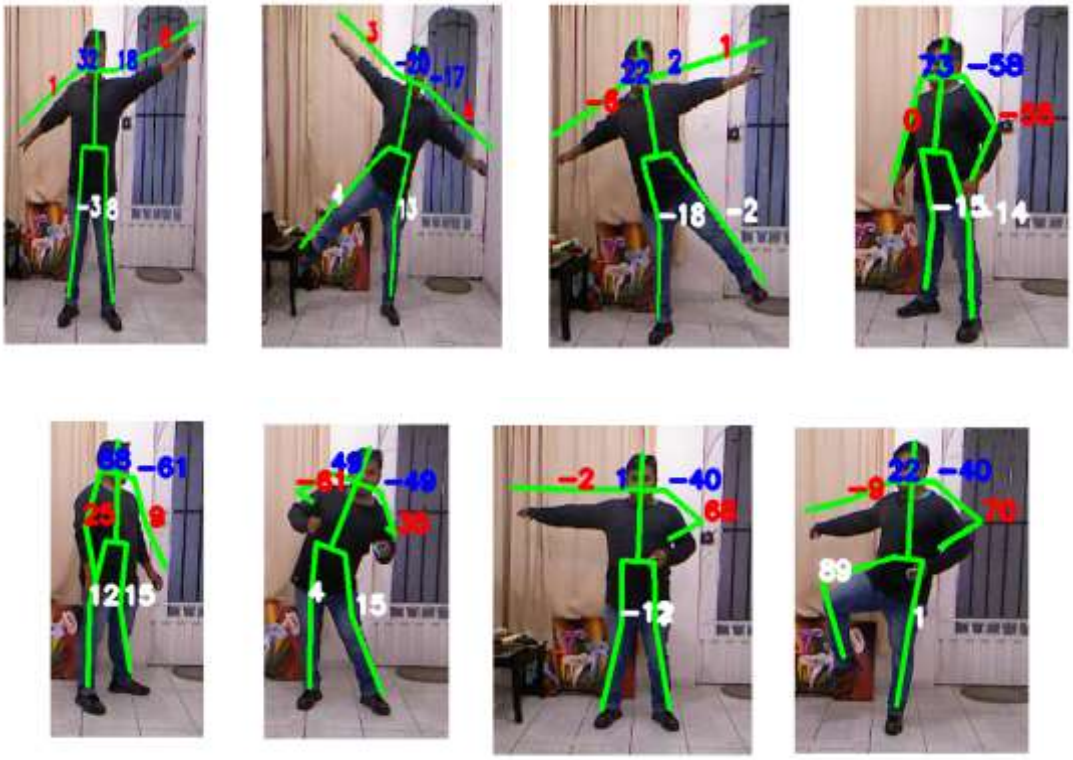

Figura 3.14. Esqueleto de seguimiento con valores de ángulos formados entre extremidades

Capítulo 4

4. Pruebas y resultados

Con la finalidad de comprobar el correcto funcionamiento del sistema de monitoreo de movimientos del cuerpo humano, se realizaron diferentes pruebas las que permitan determinar:

- Rango de funcionamiento del sistema de monitoreo,
- Efectividad de seguimiento de esqueleto,
- Efectividad medida de ángulos formados entre extremidades

Para lo cual se procedió a la toma de diferentes imágenes a distintas distancias entre usuario – cámara de profundidad las que permitan determinar el correcto funcionamiento y la fiabilidad en la medida de los ángulos mostrados por el sistema de monitoreo.

4.1. Pruebas de rango funcionamiento del sistema de monitoreo

Para la realización de esta prueba se tomaron imágenes a diferentes distancias entre la cámara de profundidad (véase Figura 4.1.) y la posición del usuario, las cuales permitan determinar el rango óptimo de funcionamiento del sistema, descartando las distancias en las cuales el sistema no funcione correctamente, la toma de imágenes se realizó en un rango de separación entre los 2 metros hasta los 5 metros.

Figura 4.1. Distancias para la toma de imágenes de prueba de funcionamiento

4.1.1. Resultados obtenidos

En base a las pruebas realizadas (véase anexo 10) del rango de funcionamiento del sistema se obtuvieron los resultados mostrados en la Tabla 4.1.

Tabla 4.1. Resultados de rango de funcionamiento del sistema de monitoreo

Numero de muestra	Distancia Cámara - Usuario	Funcionamiento del sistema
1	2 metros	NO
2	2 metros	NO
3	2 metros	NO
4	2 metros	NO
5	2 metros	NO
6	2.50 metros	SI
7	2.50 metros	NO

8	2.50 metros	SI
9	2.50 metros	SI
10	2.50 metros	SI
11	3 metros	SI
12	3 metros	SI
13	3 metros	SI
14	3 metros	SI
15	3 metros	SI
16	3.50 metros	SI
17	3.50 metros	SI
18	3.50 metros	SI
19	3.50 metros	SI
20	3.50 metros	SI
21	4 metros	SI
22	4 metros	SI
23	4 metros	SI
24	4 metros	SI
25	4 metros	SI
26	4.50 metros	NO
27	4.50 metros	NO
28	4.50 metros	SI
29	4.50 metros	NO
30	4.50 metros	NO

31	5 metros	NO
32	5 metros	NO
33	5 metros	NO
34	5 metros	NO
35	5 metros	NO

4.1.2. Análisis de resultados

En base a la Tabla 4.1., se obtiene el siguiente análisis de resultados:

- El rango para la detección de usuarios, pose inicial, calibración e inicio del esqueleto de seguimiento esta entre los 2.50 metros hasta los 4 metros.
- En rangos menores a los 2.50 metros y mayores a los 4 metros de distancia el óptimo funcionamiento del sistema de monitoreo se ve afectado, debido principalmente a la no detección del usuario, ya que en distancias menores a los 2.50 metros la cámara de profundidad no es capaz de realizar la detección del cuerpo completo del usuario y a su vez en distancias mayores a los 4 metros la cámara excede el rango de funcionamiento de la misma (véase Figura 3.2.A).

4.2. Pruebas de efectividad de seguimiento de esqueleto y medida de ángulos formados entre extremidades

Una vez determinado el rango óptimo de funcionamiento del sistema de monitoreo, se procedió a realizar pruebas de efectividad de medida de ángulos (véase Anexo 11), las cuales permitan establecer el error entre la medida de ángulos medidos por la cámara de profundidad y los ángulos

reales, estableciendo rangos de medida dependiendo de medida fue correcta o esta presenta algún tipo de error leve o medio.

4.2.1. Resultados obtenidos

Tabla 4.2. Variación de medida de ángulos entre medida obtenida por dispositivo y media real

Partes analizadas	**Variación de medida de ángulos**
Hombro – Antebrazo Izquierdo	± 2
Antebrazo – Brazo Izquierdo	± 2
Muslo – Tobillo Izquierdo	± 4
Hombro – Antebrazo Derecho	± 2
Antebrazo – Brazo Derecho	± 2
Muslo – Tobillo Derecho	± 2

4.2.2. Análisis de resultados

Se determinó el valor promedio de la del error entre la medida de los ángulos medidos por la cámara de profundidad y los ángulos medidos manualmente obtener un porcentaje que expresa la tasa de error de lectura.

$$P = \frac{S}{N} \tag{4.1}$$

En donde:

P = Es el promedio de la diferencia entre mediciones.

S = Valor total de la suma de la diferencia de datos.

N = Número de muestras.

$$P = \frac{265}{120} = 2.2083$$

En las pruebas realizadas se determinó una variación de ± 3°

Capítulo 5

5. Conclusiones y trabajo futuro

5.1. Conclusiones

El presente trabajo presenta un sistema de monitoreo de los movimientos del cuerpo humano a través de una cámara de profundidad, el cual permite visualizar los ángulos formados entre las extremidades empleando métodos no invasivos, logrando con esto un sistema autónomo y con gran funcionabilidad.

Para el desarrollo y la implementación de este sistema se determinó en la primera fase los requerimientos de software y hardware necesarios para el funcionamiento y la implementación del sistema, en una segunda fase se procedió a determinar las condiciones de funcionamiento del sistema, entre las que se encontraban determinar la distancia óptima para la detección del usuario, en una tercera fase se desarrolló un código de programa el cual serviría para a culminación de este proyecto, para finalmente en un cuarta fase realizar las pruebas de funcionamiento las cuales permitan conocer la fiabilidad del sistema.

Luego de la realización de las pruebas de funcionamiento se obtuvieron resultados positivos ya que se consiguió desarrollar un sistema de monitoreo capaz de determinar y mostrar los ángulos formados entre 6 diferentes puntos de interés del cuerpo humano, siendo un sistema muy fiable y con gran exactitud de medida, además se determinó el rango óptimo para el funcionamiento del

sistema que se encuentra dentro de un rango no menor a los 2.50 metros y no mayos a los 4.50 metros.

5.2. Recomendaciones

- Se aconseja aislar todos los elementos que pudiesen interferir con el óptimo funcionamiento de la cámara de profundidad; tales como: la cantidad de luz, espacio de trabajo, objetos reflejantes, obstáculos, y rango de funcionamiento.
- Se debe realizar la inspección de todas las conexiones, tanto de comunicación desde la cámara de profundidad hacia el computador, así como con las de alimentación de energía eléctrica.
- Limpiar regularmente los lentes de la cámara de profundidad, ya que impurezas u objetos extraños no permitirían el normal funcionamiento del sistema.

5.3. Trabajo futuro

El trabajo futuro se orienta al desarrollo de aplicaciones específicas para el análisis de diferentes partes del cuerpo humano e incluso de los compartimientos de la misma persona, tales como posturas, gestos y movimientos involuntarios, los cuales pueden ser utilizados para el análisis de posibles patologías o enfermedades.

Bibliografía

[1] M. Trew y T. Everett, Fundamentos del Movimiento Humano, Barcelona: Elsevier, 2006.

[2] M. Izquierdo Redín, Biomecánica y Bases Neuromuscolares de la Actividad Física y el Deporte, Madrid: Médica Panamericana, 2008.

[3] A. Tejero, F. Díaz-Pernas, M. Martínez Zarzuela y D. Gonzáles-Ortega, *Monitorización del cuerpo humano en 3D mediante tecnología Kinect*, Valladolid, 2011.

[4] XIIBI.COM, «Create your own chibi characters with manga/anime style,» 08 Octubre 2008. [En línea]. Available: http://www.xiibi.com/camaras-de-profundidad-y-kinect-hacks/.

[5] F. Telefónica, Realidad Aumentada: una nueva lente para ver el mundo, Barcelona: Ariel, 2011.

[6] I. E. Navarrete , *SISTEMA ELECTRÓNICO CON APLICACIÓN IOT DE MONITOREO DE MOVIMIENTO CORPORAL DE LAS EXTREMIDADES INFERIORES DE UN ESTUDIANTE UNIVERSITARIO QUE BRINDE ESTIMADORES DE POSICIÓN DENTRO DEL AULA EN LA UNIVERSIDAD TÉCNICA DEL NORTE A ESCALA DE LABORATORIO*, Ibarra, 2016.

[7] J. E. Carvajal Terán, *DESARROLLO DE UNA APLICACIÓN REMOTA DE INTERACCIÓN ENTRE EL PROFESOR Y EL PC PARA AGILITAR EL PROCESO ENSEÑANZA APRENDIZAJE EN LAS AULAS DE LA UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL,* Quito, 2014.

[8] L. Ilbay Llangarí, *EVALUACIÓN DE ALGORITMOS DE TRACKING 3D PARA LA SIMULACIÓN DE UN BRAZO ROBÓTICO, MEDIANTE KINECT,* Riobamba, Chimborazo, 2015.

[9] D. Ramos Gutiérrez, *ESTUDIO CINEMÁTICO DEL CUERPO,* Madrid, 2012.

[10] O. P. Osorio y F. L. Peña, *CAPTURA DE MOVIMIENTO UTILIZANDO EL KINECT PARA EL CONTROL DE UNA PLATAFORMA ROBOTICA CONTROLADA DE FORMA REMOTA POR MEDIO DE SEGUIMIENTO DE LOS PUNTOS DE ARTICULACIÓN DEL CUERPO,* Pereira, 2015.

[11] European Union, «EUROPA - European Union website, the official EU website,» 25 04 2007. [En línea]. Available: https://cordis.europa.eu/result/rcn/80842_es.html. [Último acceso: 27 05 2019].

[12] L. Wang, H. Weiming y T. Tieniu, «Recent developments in human motion analysis,» Pattern Recognition, 2003.

[13] B. Moeslund y E. Granum, «A Survey on Computer Vision-Based Human Motion Capture,» Computer Vision and Image Understanding, Boston, 2001.

[14] S. R. Kakadiaris, «Machine Vision and Applications,» Editorial Introduction, 2003.

[15] B. A. Gowitzke y M. Milner, EL CUERPO Y SUS MOVIMIENTOS. BASES CIENTÍFICAS, Barcelona: PAIDOTRIBO, 1999.

[16] N. Palastanga, D. Field y R. Soames, ANATOMÍA Y MOVIMIENTO HUMANO. ESTRUCTURA Y FUNCIONAMIENTO, Barcelona: PAIDOTRIBO, 2007.

[17] J. F. Ramírez Patiño, *DESARROLLO DE UN ALGORITMO COMPUTACIONAL PARA LA PREDICCIÓN DE FUERZAS Y MOMENTOS EN LA INTERFAZ SOCKET-EXTENSION FEMORAL,* Medellín: Universidad Nacional de Colombia, 2014.

[18] V. Tobes Pérez y R. Fernández Pérez, *Uso de Kinect para el entrenamiento de actividades físicas,* Madrid, 2017, pp. 13 - 17.

[19] J. Han, L. Shao, D. Xu y J. Shotton, *Enhanced Computer Vision with Microsoft Kinect Sensor,* New York: IEEE TRANSACTIONS ON CYBERNETICS, 2013.

[20] G. Nikolova y Y. Toshev, Estimation of male and female body segment parameters of the Bulgarian population using a 16-segmental mathematical, J. Biomech., vol. 40, 2007.

[21] J. Pantrigo, ALGORITMOS DE OPTIMIZACIÓN APLICADOS AL SEGUIMIENTO DEL MOVIMIENTO ARTICULAR Y LA DIGITALIZACIÓN AUTOMÁTICA DEL MOVIMIENTO HUMANO, 2019.

[22] R. E. Kalman, «A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME,» Journal of Basic Engineering, Houston, 1960.

[23] D. Zotkin, R. Duraiswami y L. Davis, «oint Audio-Visual Tracking Using Particle Filters.,» EURASIP Journal on Applied Signal Processing, 2001.

[24] A. Doucet, S. Godsill y C. Andrieu, «On sequential Monte Carlo sampling methods for Bayesian filtering,» *Statistics and Computing*, vol. 10, p. 197 – 208, 2000.

[25] M. S. Arulampalam, S. Maskell, . N. Gordon y T. Clapp, «A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,» *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 50, nº 2, pp. 174 - 188, 2002.

[26] C. Dal Mutto, P. Zanuttigh y M. Cortelazzo, «Time-of-Flight Cameras andMicrosoft KinectTM,» Springer, 2013.

[27] J. Sell y P. O´Connor, «THE XBOX ONE SYSTEM ON A CHIP AND KINECT SENSOR,» Published by the IEEE Computer Society, 2015.

[28] «Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications,» mdpi journal University of Twente, 2012.

[29] C. A. Ayala Cajas y R. G. Guerrero Idrovo, *Sistema de seguridad inteligente basado en reconocimiento de patrones mediante tecnología Kinect para restringir el acceso no autorizado a consolas de administración y monitoreo,* Quito, Pichincha, 2013.

[30] M. S. Magallón, *SISTEMA INTERACTIVO PARA MANEJO DE ELECTRODOMÉSTICOS EN ENTORNOS DOMÉSTICOS,* Zaragoza, 2013.

[31] WIRED, «KINECT HACKERS ARE CHANGING THE FUTURE OF ROBOTICS,» Octubre 2011. [En línea]. Available: https://www.wired.com/2011/06/mf_kinect/.

[32] A. Fernández Montoro, Python 3 al descubierto, México, D.F.: Alfaomega, 2013.

[33] JetBrains, «JetBrains,» 30 Noviembre 2017. [En línea]. Available: https://www.jetbrains.com/pycharm/.

[34] L. G. Ilbay Llangarí, *EVALUACIÓN DE ALGORITMOS DE TRACKING 3D PARA LA SIMULACIÓN DE UN BRAZO ROBÓTICO, MEDIANTE KINECT,* Riobamba, 2015, pp. 60 - 61.

[35] A. Kaehler y G. Bradski, Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, O'Reilly Media, Inc, 2016.

[36] openni.org, «www.openni.org,» 30 Noviembre 2017. [En línea]. Available: www.openni.org.

[37] Microsoft Corporation, «Human Interface Guidelines v2.0,» 2014. [En línea]. Available: http://www.kinectforwindows.com. [Último acceso: 01 05 2019].

[38] S. H. Peralta Benhumea, *Interfaz de lenguaje natural usando Kinect*, México, Distrito Federal, 2012.

[39] «Wikipedia,» 16 Marzo 2012. [En línea]. Available: https://es.wikipedia.org/wiki/Kinect.

[40] «Visión artificial e interacción sin mandos,» Diciembre 2010. [En línea]. Available: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/index.html.

[41] D. U. Silverthorn, Fisiología Humana. Un enfoque integrado, Cuarta ed., Madrir: Médica Panamericana, 2008.

[42] C. Centro Integrado Politécnico "ETI", «Célula de Fabricación Flexible ETI,» 11 Noviembre 2003. [En línea]. Available: http://www.etitudela.com/celula/downloads/visionartificial.pdf.

[43] Ledda I. Larcher, Enrique M. Biasoni, Carlos A. Cattaneo, Ana I. Ruggeri, A., *Centro de Investigación de Métodos Computacionales,* Santiago del Estero, 2016.

[44] M. Ortega Cantero, J. Bravo Rodríguez y J. Ruiz Hernández, Informatica Indistrial, Cuenca: Ediciones de la Universidad de Castilla - La Mancha, 1997.

[45] O. Trabocchi y F. Sanfilippo., «Universidad Nacional de Quilmes,» Octubre 2005. [En línea]. Available: http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Aspectos%20de%20un%20Proyecto%20de%20Visi%C3%B3n%20Artificial.pdf.

[46] G. F. VILLALBA MENESES, *SISTEMA DE VERIFICACIÓN Y CLASIFICACIÓN DE BOTELLAS CON SÓLIDOS NO DISUELTOS EN LÍQUIDOS INCOLOROS MEDIANTE VISIÓN ARTIFICIAL EN LA EMPRESA LICORES DE AMÉRICA "LICORAM S.A",* Ibarra, Imbabura, 2015.

[47] INFAIMON S.L, «Aplicaciones y sistemas de visión Artificial: cámaras de visión artificial, control de calidad, automatización industrial | Sistemas de visión Artificial | INFAIMON,» 29 Septiembre 2016. [En línea]. Available: http://www.infaimon.com/es/camaras-industria.

[48] Empresa de Telecomunicaciones de Cuba S.A., «EcuRed,» 24 Junio 2010. [En línea]. Available: https://www.ecured.cu/Lenguaje_de_programaci%C3%B3n_ABC.

[49] MARKMONITOR INC., «Blogger.com,» 24 Octubre 2002. [En línea]. Available: http://infostudio-blog.blogspot.com/2011/08/principales-caracteristicas-del_31.html.

[50] J. Raya Cabrera y L. Raya González, Implantación de Sistemas Operativos, Madrid: RA-MA, 2010.

[51] J. Orloff, Ubuntu Linux; PASO A PASo, Mexico D.F.: McGRAW-HILL/INTERAMERICANA, 2009.

[52] M. Serrat Olmos, Ubuntu Linux, Madrid: RA-MA, 2009.

[53] R. Petersen, Ubuntu 16.04 LTS Desktop: Applications and Administration, United States of America: Surfing Turtle, 2016.

[54] S. Empresa de Telecomunicaciones de Cuba, «ECured.cu,» Junio 2010. [En línea]. Available: https://www.ecured.cu/Interfaz_de_usuario.

[55] E. Ruiz y V. Sánchez, EDUCATRÓNICA: Innovación en el Aprendizaje de las Ciencias y la Tecnología, Mexico D.F.: Díaz de Santos, 2007.

[56] M. Gallego Carrillo y M. d. S. Montalvo Herranz, Interfaces gráficas en Java, Madrid: Universitaria Ramon Areces, 2005.

[57] O. S. Ltd., «Interfaz natural de usuario (NUI) en una cáscara de nuez,» Diciembre 2011. [En línea]. Available: https://www.onysus.com/interfaz-natural-de-usuario-nui-en-una-cascara-de-nuez/.

[58] Media Temple Inc., «3D print & 3D scan - I3DU,» Enero 2014. [En línea]. Available: http://www.i3du.gr/pdf/primesense.pdf.

[59] NET-CHINESE CO. LTD., «ASUS USA,» Enero 2015. [En línea]. Available: https://www.asus.com/3D-Sensor/.

[60] D. d. l. F. Garrido, «Aplicaciones de Kinect para Neurohabilitación,» 06 2012. [En línea]. Available: http://upcommons.upc.edu/bitstream/handle/2099.1/15334/memoria.pdf?sequence=1.

[61] V. Araujo Vásquez y M. Tipán Riofrío, *CONTROL DE UN ROBOT BÍPEDO HUMANIDE EN UN ESPACIO 3D MEDIANTE EL SENSOR KINECT*, Quito, 2015.

[62] Microsoft Corporation, «Natural User Interface for Kinect for Windows,» Octubre 2014. [En línea]. Available: https://msdn.microsoft.com/en-us/library/hh855352.aspx.

[63] C. Blum y A. Roli, «Metaheuristics in Combinatorial Optimization: Overview andConceptual Comparison.,» ACM Computing Surveys268-308, Oakland USA, 2003.

[64] Á. Merí Vived, FUNDAMENTOS DE FISIOLOGÍA DE LA ACTIVIDAD FÍSICA Y EL DEPORTE, Madrid: Médica Panamericana, 2005.

[65] J. L. Ayuso Gallardo, ANATOMÍA FUNCIONAL DEL APARATO LOCOMOTOR, Sevilla: WANCEULEN EDITORIAL DEPORTIVA S,L., 2008.

[66] Net-Chinese Co., Ltd. , «ASUS USA,» 22 Marzo 2019. [En línea]. Available: https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE/. [Último acceso: 16 Abril 2019].

[67] Net-Chinese Co., Ltd., «ASUS USA,» 22 Marzo 2019. [En línea]. Available: https://www.asus.com/es/3D-Sensor/Xtion_PRO/. [Último acceso: 16 Abril 2019].

[68] Net-Chinese Co., Ltd. , «ASUS USA,» 22 Marzo 2019. [En línea]. Available: https://www.asus.com/es/3D-Sensor/Xtion/. [Último acceso: 16 Abril 2019].

[69] FayerWayer, «Metro International S.A.,» CSL Computer Service Langenbach GmbH, 23 01 2019. [En línea]. Available: https://www.fayerwayer.com/2010/06/kinect-para-xbox-360-e3-2010/. [Último acceso: 15 03 2019].

[70] I. MarkMonitor, «Microsoft - Official Home Page,» Microsoft Corporation, 15 10 2014. [En línea]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx. [Último acceso: 12 02 2019].

[71] J. Prass Martins, «GitHub, Inc.,» MarkMonitor, Inc., 09 10 2007. [En línea]. Available: https://github.com/jonathanpmartins/ubuntusetup/blob/master/server/install-python-2.7.11-on-ubuntu-14.04.sh. [Último acceso: 01 05 2019].

[72] PERFECT PRIVACY, LLC, «JetBrains: Developer Tools for Professionals and Teams,» NETWORK SOLUTIONS, LLC., 09 11 2018. [En línea]. Available: https://www.jetbrains.com/pycharm/download/#section=linux. [Último acceso: 01 05 2019].

[73] GitHub, Inc., «GitHub, Inc.,» MarkMonitor, Inc. , 09 10 2007. [En línea]. Available: https://github.com/OpenNI/OpenNI. [Último acceso: 01 05 2019].

[74] GitHub, Inc., «GitHub, Inc.,» MarkMonitor, Inc., 09 10 2007. [En línea]. Available: https://github.com/OpenKinect/libfreenect. [Último acceso: 01 05 2019].

[75] GitHub, Inc., «GitHub, Inc.,» MarkMonitor, Inc., 09 10 2007. [En línea]. Available: https://github.com/opencv/opencv. [Último acceso: 01 05 2019].

6. Anexos

Anexo 1

Código de programa 1

Configuración inicial de la cámara de profundidad

```
<OpenNI>
      <Licenses>
            <License vendor="PrimeSense"
key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
      </Licenses>

      <Log writeToConsole="true" writeToFile="false">
            <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default)
-->
            <LogLevel value="3"/>
            <Masks>
                  <Mask name="ALL" on="true"/>
            </Masks>
            <Dumps></Dumps>
      </Log>

      <ProductionNodes>
            <Node type="Depth" name="Depth1">
                  <Configuration>
                        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                        <Mirror on="true"/>
                  </Configuration>
            </Node>

            <Node type="Image" name="Image1" stopOnError="false">
                  <Configuration>
                        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                        <Mirror on="true"/>
                  </Configuration>
            </Node>
      </ProductionNodes>
  </OpenNI>
```

Anexo 2

Código de programa 2

Adquisición de imágenes RGB de la cámara de profundidad

```
# import the necessary modules
import freenect
import cv2

# function to get RGB image from kinect
def get_video():
    array, _ = freenect.sync_get_video()
    array = cv2.cvtColor(array, cv2.COLOR_RGB2BGR)
    return array

if __name__ == "__main__":
    while 1:
        # get a frame from RGB camera
        frame = get_video()

        # display RGB image
        cv2.imshow('RGB image', frame)

        # quit program when 'esc' key is pressed
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break
    cv2.destroyAllWindows()
```

Anexo 3

Código de programa 3

Obtención de imágenes con profundidad

```
# import the necessary modules
import freenect
import cv2
import numpy as np


# function to get depth image from kinect
def get_depth():
    array, _ = freenect.sync_get_depth()
    array = array.astype(np.uint8)
    return array


if __name__ == "__main__":
    while 1:

        # get a frame from depth sensor
        depth = get_depth()

        # display depth image
        cv2.imshow('Depth image', depth)

        # quit program when 'esc' key is pressed
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break
    cv2.destroyAllWindows()
```

Anexo 4

Código de programa 4

Obtención de Imagen RGB y de profundidad

```
# import the necessary modules
import freenect
import cv2
import numpy as np


# function to get RGB image from kinect
def get_video():
    array, _ = freenect.sync_get_video()
    array = cv2.cvtColor(array, cv2.COLOR_RGB2BGR)
    return array


# function to get depth image from kinect
def get_depth():
    array, _ = freenect.sync_get_depth()
    array = array.astype(np.uint8)
    return array


if __name__ == "__main__":
    while 1:
        # get a frame from RGB camera
        frame = get_video()
        # get a frame from depth sensor
        depth = get_depth()
        # display RGB image
        cv2.imshow('RGB image', frame)
        # display depth image
        cv2.imshow('Depth image', depth)

        # quit program when 'esc' key is pressed
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break
    cv2.destroyAllWindows()
```

Anexo 5

Código de programa 5

Detección de usuario

```
pose_to_use = 'Psi'
from openni import *

ctx = Context()
ctx.init()

# Create the user generator
user = UserGenerator()
user.create(ctx)

# Obtain the skeleton & pose detection capabilities
skel_cap = user.skeleton_cap
pose_cap = user.pose_detection_cap

# Declare the callbacks
def new_user(src, id):
    print "1/4 User {} detected. Looking for pose..." .format(id)
    pose_cap.start_detection(pose_to_use, id)

def pose_detected(src, pose, id):
    print "2/4 Detected pose {} on user {}. Requesting calibration..."
.format(pose,id)
    pose_cap.stop_detection(id)
    skel_cap.request_calibration(id, True)

def calibration_start(src, id):
    print "3/4 Calibration started for user {}." .format(id)

def calibration_complete(src, id, status):
    if status == CALIBRATION_STATUS_OK:
        print "4/4 User {} calibrated successfully! Starting to track."
.format(id)
        skel_cap.start_tracking(id)
    else:
        print "ERROR User {} failed to calibrate. Restarting process."
.format(id)
        new_user(user, id)

def lost_user(src, id):
    print "--- User {} lost." .format(id)

# Register them
user.register_user_cb(new_user, lost_user)
pose_cap.register_pose_detected_cb(pose_detected)
skel_cap.register_c_start_cb(calibration_start)
skel_cap.register_c_complete_cb(calibration_complete)
```

```
# Set the profile
skel_cap.set_profile(SKEL_PROFILE_ALL)

# Start generating
ctx.start_generating_all()
print "0/4 Starting to detect users. Press Ctrl-C to exit."

while True:
    # Update to next frame
    ctx.wait_and_update_all()

    # Extract head position of each tracked user
    for id in user.users:
        if skel_cap.is_tracking(id):
            head = skel_cap.get_joint_position(id, SKEL_HEAD)
            print "  {}: head at ({loc[0]}, {loc[1]}, {loc[2]})
[{conf}]" .format(id, loc=head.point, conf=head.confidence)
```

Anexo 6

Código de programa 6

Detección y generación de usuario

```
# import necessary modules
import openni
import numpy as np
import cv2
import sys

# data storage
base_pose = 'Psi'

# load camera configuration
camera_configuration = 'config.xml'
maximun_depth = 10000
context = openni.Context()
context.init_from_xml_file(camera_configuration)

# generate depth
depth_generator = openni.DepthGenerator()
depth_generator.create(context)
depth_generator.set_resolution_preset(openni.RES_VGA)
depth_generator.fps=30

#generate image
image_generator = openni.ImageGenerator()
image_generator.create(context)
image_generator.set_resolution_preset(openni.RES_VGA)
image_generator.fps=30

# generate users
user = openni.UserGenerator()
user.create(context)

# captures skeleton and pose
capture_skeleton = user.skeleton_cap
capture_pose = user.pose_detection_cap

# search user and pose
def new_user(src, id):
    print "USER {} DETECTED. SEARCHING INITIAL POSE".format(id)
    capture_pose.start_detection(base_pose, id)

# detects user pose
def pose_detected(src, pose, id):
    print "DETECTING POSE {} FOR THE USER {}. WAITING CALIBRATION..."
.format(pose,id)
    capture_pose.stop_detection(id)
    capture_skeleton.request_calibration(id, True)
```

```
# calibrate user

def calibration_start(src, id):
    print "CALIBRATION STARTED FOR THE USER {}." .format(id)

# start tracking
def calibration_complete(src, id, status):
    if status == openni.CALIBRATION_STATUS_OK:
        print "USER  {} SUCCESSFULLY CALIBRATED!" .format(id)
        capture_skeleton.start_tracking(id)
    else:
        print "ERROR! {} FAILURE TO CALIBRATE. RESTORING PROCESS."
.format(id)
        new_user(user, id)

# lost user
def lost_user(src, id):
    print "USER {} LOST." .format(id)

# update catch data
def captura_rgb():
      rgb_frame = np.fromstring(image_generator.get_raw_image_map_bgr(),
dtype=np.uint8).reshape(480, 640, 3)
      return rgb_frame

# capture_rgb
### main###
# record of established parameters

user.register_user_cb(new_user,lost_user)
capture_pose.register_pose_detected_cb(pose_detected)
capture_skeleton.register_c_start_cb(calibration_start)
capture_skeleton.register_c_complete_cb(calibration_complete)
capture_skeleton.set_profile(openni.SKEL_PROFILE_ALL)
context.start_generating_all()
running = True
context.start_generating_all()

# exit the program
while running:
      context.wait_any_update_all()
      k=cv2.waitKey(10)
      if k == ord('s'):      #s
            running=False
      elif k == 27:          #ESC
            running=False
            break

# capture rgb data
      data_rgb = captura_rgb()
      cv2.imshow('Original Image', data_rgb) # returns original image
```

```
# fin while
# exit the program ends process
context.stop_generating_all()
sys.exit(0)
```

Anexo 7

Código de programa 7

Representación y visualización del esqueleto de seguimiento

```
#modulos necesarios
import openni
import numpy as np
import cv2
import sys

#almacenamiento datos
pose_base = 'Psi'
#carga congifuracion de la camara
camera_configuration = 'config.xml'
maximun_depth = 10000

context = openni.Context()
context.init_from_xml_file(camera_configuration)

#generar profundidad
depth_generator = openni.DepthGenerator()
depth_generator.create(context)

depth_generator.set_resolution_preset(openni.RES_VGA)
depth_generator.fps=30

#generar imagen
image_generator = openni.ImageGenerator()
image_generator.create(context)

image_generator.set_resolution_preset(openni.RES_VGA)
image_generator.fps=30

#generar usuarios
user = openni.UserGenerator()
user.create(context)

# capta esqueleto y pose
capture_skeleton = user.skeleton_cap
capture_pose = user.pose_detection_cap

#busca usuario y pose
def new_user(src, id):
    print "USER {} DETECTED. SEARCHING INITIAL POSE".format(id)
    capture_pose.start_detection(pose_base, id)

#detecta pose de usuarioa
def pose_detected(src, pose, id):
```

```
    print "DETECTING POSE {} FOR THE USER {}. WAITING CALIBRATION..."
.format(pose,id)
    capture_pose.stop_detection(id)
    capture_skeleton.request_calibration(id, True)

#calibra usuario
def calibration_start(src, id):
    print "CALIBRATION STARTED FOR THE USER {}." .format(id)

#inicia seguimiento
def calibration_complete(src, id, status):
    if status == openni.CALIBRATION_STATUS_OK:
        print "USER  {} SUCCESSFULLY CALIBRATED! STAR TRACKING."
.format(id)
        capture_skeleton.start_tracking(id)
    else:
        print "ERROR! {} FAILURE TO CALIBRATE. RESTORING PROCESS."
.format(id)
        new_user(user, id)

#usuario perdido
def lost_user(src, id):
    print "USER {} LOST." .format(id)

# actualiza datos de captura

def captura_rgb():
      rgb_frame = np.fromstring(image_generator.get_raw_image_map_bgr(),
dtype=np.uint8).reshape(480, 640, 3)
      return rgb_frame

# captura_rgb
### main###
#registro de parametros establecidos
user.register_user_cb(new_user,lost_user)

capture_pose.register_pose_detected_cb(pose_detected)
capture_skeleton.register_c_start_cb(calibration_start)
capture_skeleton.register_c_complete_cb(calibration_complete)

capture_skeleton.set_profile(openni.SKEL_PROFILE_ALL)
context.start_generating_all()

running = True
line_color=(0,255,0)

context.start_generating_all()
#para salir del programa
while running:

      context.wait_any_update_all()
      k=cv2.waitKey(10)

      if k == ord('s'):      #s para el programa
```

```
            running=False
      elif k == 27:          #ESC sale del programa
            running=False
            break

#captura datos rgb
      data_rgb = captura_rgb()
      img = data_rgb.copy()
#varios usuarios
      for id in user.users:
            #compara registro de usuario y calibracion esten ejecutandose
            if capture_skeleton.is_tracking(id) and
capture_skeleton.is_calibrated(id):

#adquisicion de puntos en x,y
                  head = capture_skeleton.get_joint_position(id,
openni.SKEL_HEAD)
                  neck = capture_skeleton.get_joint_position(id,
openni.SKEL_NECK)
                  tors = capture_skeleton.get_joint_position(id,
openni.SKEL_TORSO)

                  l_shoulder = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_SHOULDER)
                  l_elbow = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_ELBOW)
                  l_hand = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_HAND)
                  l_hip = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_HIP)
                  l_knee = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_KNEE)
                  l_foot = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_FOOT)

                  r_shoulder = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_SHOULDER)
                  r_elbow = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_ELBOW)
                  r_hand = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_HAND)
                  r_hip = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_HIP)
                  r_knee = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_KNEE)
                  r_foot = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_FOOT)

#adquiere profundidad
                  head = depth_generator.to_projective([head.point])[0]
                  neck = depth_generator.to_projective([neck.point])[0]
                  tors = depth_generator.to_projective([tors.point])[0]
```

```
                l_shoulder =
depth_generator.to_projective([l_shoulder.point])[0]
                l_elbow =
depth_generator.to_projective([l_elbow.point])[0]
                l_hand =
depth_generator.to_projective([l_hand.point])[0]
                l_hip = depth_generator.to_projective([l_hip.point])[0]
                l_knee =
depth_generator.to_projective([l_knee.point])[0]
                l_foot =
depth_generator.to_projective([l_foot.point])[0]

                r_shoulder =
depth_generator.to_projective([r_shoulder.point])[0]
                r_elbow =
depth_generator.to_projective([r_elbow.point])[0]
                r_hand =
depth_generator.to_projective([r_hand.point])[0]
                r_hip = depth_generator.to_projective([r_hip.point])[0]
                r_knee =
depth_generator.to_projective([r_knee.point])[0]
                r_foot =
depth_generator.to_projective([r_foot.point])[0]
#cintura division
                hip = (int((l_hip[0] + r_hip[0]) / 2), int((l_hip[1] +
r_hip[1]) / 2))

#dibuja las lineas entre los puntos
                cv2.line(img,(int(head[0]), int(head[1])),
(int(neck[0]), int(neck[1])),line_color,5)
                cv2.line(img,(int(neck[0]), int(neck[1])),
(int(tors[0]), int(tors[1])),line_color, 5)
                cv2.line(img,(int(tors[0]), int(tors[1])),
(int(hip[0]), int(hip[1])),line_color, 5)
                cv2.line(img,(int(l_shoulder[0]), int(l_shoulder[1])),
(int(r_shoulder[0]), int(r_shoulder[1])),line_color, 5)

                cv2.line(img,(int(l_shoulder[0]), int(l_shoulder[1])),
(int(l_elbow[0]), int(l_elbow[1])),line_color, 5)
                cv2.line(img,(int(l_elbow[0]), int(l_elbow[1])),
(int(l_hand[0]), int(l_hand[1])),line_color, 5)
                cv2.line(img,(int(l_hip[0]), int(l_hip[1])),
(int(l_knee[0]), int(l_knee[1])),line_color, 5)

                cv2.line(img,(int(l_knee[0]), int(l_knee[1])),
(int(l_foot[0]), int(l_foot[1])),line_color, 5)
                cv2.line(img,(int(l_hip[0]), int(l_hip[1])),
(int(r_hip[0]), int(r_hip[1])),line_color, 5)

                cv2.line(img,(int(r_shoulder[0]), int(r_shoulder[1])),
(int(r_elbow[0]), int(r_elbow[1])),line_color, 5)
                cv2.line(img,(int(r_elbow[0]), int(r_elbow[1])),
(int(r_hand[0]), int(r_hand[1])),line_color, 5)
```

```
                cv2.line(img,(int(r_hip[0]), int(r_hip[1])),
(int(r_knee[0]), int(r_knee[1])),line_color, 5)
                cv2.line(img,(int(r_knee[0]), int(r_knee[1])),
(int(r_foot[0]), int(r_foot[1])),line_color, 5)

      #muestra las lineas en pantalla
      cv2.imshow('Original Image', data_rgb) #retorna imagen original
      cv2.imshow('Skeleton Tracking', img) #imagen mas lineas y datos

# fin while
#salir del programa finaliza proceso
context.stop_generating_all()
sys.exit(0)
```

Anexo 8

Código de programa 8

Cálculo de ángulos entre articulaciones

```
##### calcula angulo con pendientes entre rectas
def calculate(x2,y2,x1,y1,x,y):

      if(x2 - x1) != 0:
            m2 = (y2 - y1) / (x2 - x1)
      else:
            m2=1.0
      if (x1 - x) != 0:
            m1 = (y1 - y) / (x1 - x)
      else:
            m1=-1.0
      if (1 + (m1*m2)) == 0:
            angle=90
      else:
            delt=(m2-m1)/(1.0+(m1*m2))
            angle=math.degrees(math.atan(delt))
      return angle
```

Anexo 9

Código de programa 9

Seguimiento de esqueleto y cálculo de ángulos entre extremidades

```
#modulos necesarios
import openni
import numpy as np
import cv2
import sys
import math

#almacenamiento datos
pose_base = 'Psi'
#carga congifuracion de la camara
camera_configuration = 'config.xml'
maximun_depth = 10000

context = openni.Context()
context.init_from_xml_file(camera_configuration)

#generar profundidad
depth_generator = openni.DepthGenerator()
depth_generator.create(context)

depth_generator.set_resolution_preset(openni.RES_VGA)
depth_generator.fps=30

#generar imagen
image_generator = openni.ImageGenerator()
image_generator.create(context)

image_generator.set_resolution_preset(openni.RES_VGA)
image_generator.fps=30

#generar usuarios
user = openni.UserGenerator()
user.create(context)

# capta esqueleto y pose
capture_skeleton = user.skeleton_cap
capture_pose = user.pose_detection_cap

#busca usuario y pose
def new_user(src, id):
    print "USER {} DETECTED. SEARCHING INITIAL POSE".format(id)
    capture_pose.start_detection(pose_base, id)

#detecta pose de usuarioa
def pose_detected(src, pose, id):
```

```
    print "DETECTING POSE {} FOR THE USER {}. WAITING CALIBRATION..."
.format(pose,id)
    capture_pose.stop_detection(id)
    capture_skeleton.request_calibration(id, True)

#calibra usuario
def calibration_start(src, id):
    print "CALIBRATION STARTED FOR THE USER {}." .format(id)

#inicia seguimiento
def calibration_complete(src, id, status):
    if status == openni.CALIBRATION_STATUS_OK:
        print "USER  {} SUCCESSFULLY CALIBRATED! STAR TRACKING."
.format(id)
        capture_skeleton.start_tracking(id)
    else:
        print "ERROR! {} FAILURE TO CALIBRATE. RESTORING PROCESS."
.format(id)
        new_user(user, id)

#usuario perdido
def lost_user(src, id):
    print "USER {} LOST." .format(id)

# actualiza datos de captura

def captura_rgb():
      rgb_frame = np.fromstring(image_generator.get_raw_image_map_bgr(),
dtype=np.uint8).reshape(480, 640, 3)
      return rgb_frame

##### calcula angulo con pendintes entre rectas
def calculate(x2,y2,x1,y1,x,y):

      if(x2 - x1) != 0:
            m2 = (y2 - y1) / (x2 - x1)
      else:
            m2=1.0
      if (x1 - x) != 0:
            m1 = (y1 - y) / (x1 - x)
      else:
            m1=-1.0
      if (1 + (m1*m2)) == 0:
            angle=90
      else:
            delt=(m2-m1)/(1.0+(m1*m2))
            angle=math.degrees(math.atan(delt))
      return angle

# captura_rgb
### main###
#registro de parametros establecidos
user.register_user_cb(new_user,lost_user)
```

```
capture_pose.register_pose_detected_cb(pose_detected)
capture_skeleton.register_c_start_cb(calibration_start)
capture_skeleton.register_c_complete_cb(calibration_complete)

capture_skeleton.set_profile(openni.SKEL_PROFILE_ALL)
context.start_generating_all()

running = True
line_color=(0,255,0)

context.start_generating_all()
#para salir del programa
while running:

      context.wait_any_update_all()
      k=cv2.waitKey(10)

      if k == ord('s'):      #s para el programa
            running=False
      elif k == 27:          #ESC sale del programa
            running=False
            break

#captura datos rgb
      data_rgb = captura_rgb()
      img = data_rgb.copy()
#varios usuarios
      for id in user.users:
            #compara registro de usuario y calibracion esten ejecutandose
            if capture_skeleton.is_tracking(id) and
capture_skeleton.is_calibrated(id):

#adquisicion de puntos en x,y
                  head = capture_skeleton.get_joint_position(id,
openni.SKEL_HEAD)
                  neck = capture_skeleton.get_joint_position(id,
openni.SKEL_NECK)
                  tors = capture_skeleton.get_joint_position(id,
openni.SKEL_TORSO)

                  l_shoulder = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_SHOULDER)
                  l_elbow = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_ELBOW)
                  l_hand = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_HAND)
                  l_hip = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_HIP)
                  l_knee = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_KNEE)
                  l_foot = capture_skeleton.get_joint_position(id,
openni.SKEL_LEFT_FOOT)
```

```
                r_shoulder = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_SHOULDER)
                r_elbow = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_ELBOW)
                r_hand = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_HAND)
                r_hip = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_HIP)
                r_knee = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_KNEE)
                r_foot = capture_skeleton.get_joint_position(id,
openni.SKEL_RIGHT_FOOT)

#adquiere profundidad
                head = depth_generator.to_projective([head.point])[0]
                neck = depth_generator.to_projective([neck.point])[0]
                tors = depth_generator.to_projective([tors.point])[0]

                l_shoulder =
depth_generator.to_projective([l_shoulder.point])[0]
                l_elbow =
depth_generator.to_projective([l_elbow.point])[0]
                l_hand =
depth_generator.to_projective([l_hand.point])[0]
                l_hip = depth_generator.to_projective([l_hip.point])[0]
                l_knee =
depth_generator.to_projective([l_knee.point])[0]
                l_foot =
depth_generator.to_projective([l_foot.point])[0]

                r_shoulder =
depth_generator.to_projective([r_shoulder.point])[0]
                r_elbow =
depth_generator.to_projective([r_elbow.point])[0]
                r_hand =
depth_generator.to_projective([r_hand.point])[0]
                r_hip = depth_generator.to_projective([r_hip.point])[0]
                r_knee =
depth_generator.to_projective([r_knee.point])[0]
                r_foot =
depth_generator.to_projective([r_foot.point])[0]
#cintura division
                hip = (int((l_hip[0] + r_hip[0]) / 2), int((l_hip[1] +
r_hip[1]) / 2))

#dibuja las lineas entre los puntos
                cv2.line(img,(int(head[0]), int(head[1])),
(int(neck[0]), int(neck[1])),line_color,5)
                cv2.line(img,(int(neck[0]), int(neck[1])),
(int(tors[0]), int(tors[1])),line_color, 5)
                cv2.line(img,(int(tors[0]), int(tors[1])),
(int(hip[0]), int(hip[1])),line_color, 5)
                cv2.line(img,(int(l_shoulder[0]), int(l_shoulder[1])),
(int(r_shoulder[0]), int(r_shoulder[1])),line_color, 5)
```

```
                cv2.line(img,(int(l_shoulder[0]), int(l_shoulder[1])),
(int(l_elbow[0]), int(l_elbow[1])),line_color, 5)
                cv2.line(img,(int(l_elbow[0]), int(l_elbow[1])),
(int(l_hand[0]), int(l_hand[1])),line_color, 5)
                cv2.line(img,(int(l_hip[0]), int(l_hip[1])),
(int(l_knee[0]), int(l_knee[1])),line_color, 5)

                cv2.line(img,(int(l_knee[0]), int(l_knee[1])),
(int(l_foot[0]), int(l_foot[1])),line_color, 5)
                cv2.line(img,(int(l_hip[0]), int(l_hip[1])),
(int(r_hip[0]), int(r_hip[1])),line_color, 5)

                cv2.line(img,(int(r_shoulder[0]), int(r_shoulder[1])),
(int(r_elbow[0]), int(r_elbow[1])),line_color, 5)
                cv2.line(img,(int(r_elbow[0]), int(r_elbow[1])),
(int(r_hand[0]), int(r_hand[1])),line_color, 5)
                cv2.line(img,(int(r_hip[0]), int(r_hip[1])),
(int(r_knee[0]), int(r_knee[1])),line_color, 5)
                cv2.line(img,(int(r_knee[0]), int(r_knee[1])),
(int(r_foot[0]), int(r_foot[1])),line_color, 5)

                ### calcula el valor de angulos entre rectas
                ag1=calculate(neck[0], neck[1], l_shoulder[0],
l_shoulder[1], l_elbow[0], l_elbow[1])

                #muestra el valor entre angulos
                cv2.putText(img, str(int(ag1)),((int(l_shoulder[0])-3),
(int(l_shoulder[1]))-3 ),fontFace=cv2.FONT_ITALIC,fontScale=1,
color=(255, 255, 255),thickness=5,lineType=cv2.LINE_8)

                ag1 = calculate(neck[0], neck[1], r_shoulder[0],
r_shoulder[1], r_elbow[0], r_elbow[1])
                cv2.putText(img, str(int(ag1)), ((int(r_shoulder[0]) +
3), (int(r_shoulder[1])) - 3),fontFace=cv2.FONT_ITALIC, fontScale=1,
color=(255, 255, 255),thickness=5,lineType=cv2.LINE_8)

                ag1 = calculate(l_shoulder[0], l_shoulder[1],
l_elbow[0], l_elbow[1], l_hand[0], l_hand[1])
                cv2.putText(img, str(int(ag1)), ((int(l_elbow[0]) - 3),
(int(l_elbow[1])) - 3),fontFace=cv2.FONT_ITALIC, fontScale=1,
color=(255, 255, 255),thickness=5,lineType=cv2.LINE_8)

                ag1 =calculate(r_shoulder[0], r_shoulder[1],
r_elbow[0], r_elbow[1], r_hand[0], r_hand[1])
                cv2.putText(img, str(int(ag1)), ((int(r_elbow[0]) + 3),
(int(r_elbow[1])) - 3), fontFace=cv2.FONT_ITALIC,fontScale=1,
color=(255, 255, 255),thickness=5, lineType=cv2.LINE_8)

                ag1 = calculate(l_hip[0], l_hip[1], l_knee[0],
l_knee[1], l_foot[0], l_foot[1])
                cv2.putText(img, str(int(ag1)), ((int(l_knee[0]) - 3),
(int(l_knee[1])) - 3), fontFace=cv2.FONT_ITALIC,fontScale=1, color=(255,
255, 255),thickness=5, lineType=cv2.LINE_8)
```

```
                ag1 = calculate(r_hip[0], r_hip[1], r_knee[0],
r_knee[1], r_foot[0], r_foot[1])
                cv2.putText(img, str(int(ag1)), ((int(r_knee[0]) + 3),
(int(r_knee[1])) - 3), fontFace=cv2.FONT_ITALIC,fontScale=1, color=(255,
255, 255),thickness=5, lineType=cv2.LINE_8)

      #muestra las lineas en pantalla
      #cv2.imshow('Original Image', data_rgb) #retorna imagen original
      cv2.imshow('Skeleton Tracking', img) #imagen mas lineas y datos

# fin while
#salir del programa finaliza proceso
context.stop_generating_all()
sys.exit(0)
```

Anexo 10

Prueba N° 1

Para la realización de la primera prueba de funcionamiento se tomó la distancia de 2 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.1.

Figura 6.1. Distancia de 2 metros entre cámara de profundidad y usuario

Tabla 6.1. Resultados obtenidos a una distancia de 2 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	NO	NO	NO	NO
2	NO	NO	NO	NO
3	SI	SI	NO	NO
4	SI	NO	NO	NO
5	NO	NO	NO	NO

Prueba N° 2

Para la realización de la segunda prueba de funcionamiento se tomó la distancia de 2.50 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.2.

Figura 6.2. Distancia de 2.50 metros entre cámara de profundidad y usuario

Tabla 6.2. Resultados obtenidos a una distancia de 2.5 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	SI	SI	SI	SI
2	NO	NO	NO	NO
3	SI	SI	SI	SI
4	SI	SI	SI	SI
5	SI	SI	SI	SI

Prueba N° 3

Para la realización de la tercera prueba de funcionamiento se tomó la distancia de 3 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.3.

Figura 6.3. Distancia de 3 metros entre cámara de profundidad y usuario

Tabla 6.3. Resultados obtenidos a una distancia de 3 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	SI	SI	SI	SI
2	SI	SI	SI	SI
3	SI	SI	SI	SI
4	SI	SI	SI	SI
5	SI	SI	SI	SI

Prueba N° 4

Para la realización de la cuarta prueba de funcionamiento se tomó la distancia de 3.50 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.4.

Figura 6.4. Distancia de 3.50 metros entre cámara de profundidad y usuario

Tabla 6.4. Resultados obtenidos a una distancia de 3.50 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	SI	SI	SI	SI
2	SI	SI	SI	SI
3	SI	SI	SI	SI
4	SI	SI	SI	SI
5	SI	SI	SI	SI

Prueba N° 5

Para la realización de la quinta prueba de funcionamiento se tomó la distancia de 4 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.5.

Figura 6.5. Distancia de 4 metros entre cámara de profundidad y usuario

Tabla 6.5. Resultados obtenidos a una distancia de 4 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	SI	SI	SI	SI
2	SI	SI	SI	SI
3	SI	SI	SI	SI
4	SI	SI	SI	SI
5	SI	SI	SI	SI

Prueba N° 6

Para la realización de la sexta prueba de funcionamiento se tomó la distancia de 4.50 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.6.

Figura 6.6. Distancia de 4.50 metros entre cámara de profundidad y usuario

Tabla 6.6. Resultados obtenidos a una distancia de 4.50 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	NO	NO	NO	NO
2	SI	NO	NO	NO
3	SI	SI	SI	SI
4	NO	NO	NO	NO
5	NO	NO	NO	NO

Prueba N° 7

Para la realización de la séptima prueba de funcionamiento se tomó la distancia de 5 metros entre cámara de profundidad y el usuario, como lo muestra la Figura 6.7.

Figura 6.7. Distancia de 5 metros entre cámara de profundidad y usuario

Tabla 6.7. Resultados obtenidos a una distancia de 5 metros

Número de muestra	Detección de usuario	Pose inicial detectada	Usuario calibrado	Inicio de esqueleto de seguimiento
1	NO	NO	NO	NO
2	NO	NO	NO	NO
3	SI	NO	NO	NO
4	SI	NO	NO	NO
5	NO	NO	NO	NO

Anexo 11

Prueba N° 8

Figura 6.8. Imagen Prueba 8

En la Tabla 6.8., se muestra la comparación de los datos obtenidos.

Tabla 6.8. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 8

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	3	0	3	ERROR LEVE
Antebrazo – Brazo Izquierdo	2	0	2	ERROR LEVE
Muslo – Tobillo Izquierdo	3	0	3	ERROR LEVE

Hombro – Antebrazo Derecho	1	0	1	ERROR LEVE
Antebrazo – Brazo Derecho	1	0	1	ERROR LEVE
Muslo – Tobillo Derecho	0	0	0	CORRECTA

Prueba N° 9

Figura 6.9. Imagen Prueba 9

En la Tabla 6.9., se muestra la comparación de los datos obtenidos.

Tabla 6.9. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 9

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	14	10	4	ERROR MEDIO
Antebrazo – Brazo Izquierdo	79	80	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	3	2	1	ERROR LEVE

Hombro – Antebrazo Derecho	15	15	0	CORRECTA
Antebrazo – Brazo Derecho	80	80	0	CORRECTA
Muslo – Tobillo Derecho	0	0	0	

Prueba N° 10

Figura 6.10. Imagen Prueba 10

En la Tabla 6.10., se muestra la comparación de los datos obtenidos.

Tabla 6.10. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 10

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	67	65	2	ERROR LEVE
Antebrazo – Brazo Izquierdo	19	20	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	4	5	-1	ERROR LEVE

Hombro – Antebrazo Derecho	72	70	2	ERROR LEVE
Antebrazo – Brazo Derecho	26	25	1	ERROR LEVE
Muslo – Tobillo Derecho	0	0	0	CORRECTA

Prueba N° 11

Figura 6.11. Imagen Prueba 11

En la Tabla 6.11., se muestra la comparación de los datos obtenidos.

Tabla 6.11. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 11

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	12	10	2	ERROR LEVE
Antebrazo – Brazo Izquierdo	5	5	1	ERROR LEVE
Muslo – Tobillo Izquierdo	7	10	-3	ERROR LEVE

Hombro – Antebrazo Derecho	10	10	1	ERROR LEVE
Antebrazo – Brazo Derecho	2	5	-3	ERROR LEVE
Muslo – Tobillo Derecho	16	15	1	ERROR LEVE

Prueba N° 12

Figura 6.12. Imagen Prueba 12

En la Tabla 6.12., se muestra la comparación de los datos obtenidos.

Tabla 6.12. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 12

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	24	25	-1	ERROR LEVE
Antebrazo – Brazo Izquierdo	14	15	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	0	0	0	CORRECTA

Hombro – Antebrazo Derecho	7	5	2	ERROR LEVE
Antebrazo – Brazo Derecho	3	5	-2	ERROR LEVE
Muslo – Tobillo Derecho	4	5	-1	ERROR LEVE

Prueba N° 13

Figura 6.13. Imagen Prueba 13

En la Tabla 6.13., se muestra la comparación de los datos obtenidos.

Tabla 6.13. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 13

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	4	0	4	ERROR MEDIO
Antebrazo – Brazo Izquierdo	1	0	1	ERROR LEVE
Muslo – Tobillo Izquierdo	5	0	5	ERROR MEDIO

Hombro – Antebrazo Derecho	1	0	1	ERROR LEVE
Antebrazo – Brazo Derecho	1	0	1	ERROR LEVE
Muslo – Tobillo Derecho	0	0	0	CORRECTA

Prueba N° 14

Figura 6.14. Imagen Prueba 14

En la Tabla 6.14., se muestra la comparación de los datos obtenidos.

Tabla 6.14. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 14

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	2	0	2	ERROR LEVE
Antebrazo – Brazo Izquierdo	85	85	0	ERROR LEVE
Muslo – Tobillo Izquierdo	5	0	5	ERROR MEDIO

Hombro – Antebrazo Derecho	7	5	2	ERROR LEVE
Antebrazo – Brazo Derecho	88	85	3	ERROR LEVE
Muslo – Tobillo Derecho	4	0	4	ERROR MEDIO

Prueba N° 15

Figura 6.15. Imagen Prueba 15

En la Tabla 6.15., se muestra la comparación de los datos obtenidos.

Tabla 6.15. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 15

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	63	60	3	ERROR LEVE
Antebrazo – Brazo Izquierdo	19	20	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	4	0	4	ERROR MEDIO

Hombro – Antebrazo Derecho	63	60	3	ERROR LEVE
Antebrazo – Brazo Derecho	24	25	-1	ERROR LEVE
Muslo – Tobillo Derecho	3	0	3	ERROR LEVE

Prueba N° 16

Figura 6.16. Imagen Prueba 16

En la Tabla 6.16., se muestra la comparación de los datos obtenidos.

Tabla 6.16. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 16

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	23	25	-2	ERROR LEVE
Antebrazo – Brazo Izquierdo	5	0	5	ERROR MEDIO
Muslo – Tobillo Izquierdo	4	0	4	ERROR MEDIO

Hombro – Antebrazo Derecho	15	15	0	CORRECTA
Antebrazo – Brazo Derecho	2	0	2	ERROR LEVE
Muslo – Tobillo Derecho	0	0	0	CORRECTA

Prueba N° 17

Figura 6.17. Imagen Prueba 17

En la Tabla 6.17., se muestra la comparación de los datos obtenidos.

Tabla 6.17. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 17

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	20	20	0	CORRECTA
Antebrazo – Brazo Izquierdo	6	5	1	ERROR LEVE
Muslo – Tobillo Izquierdo	0	0	0	CORRECTA

Hombro – Antebrazo Derecho	30	30	0	CORRECTA
Antebrazo – Brazo Derecho	10	10	0	CORRECTA
Muslo – Tobillo Derecho	3	0	3	ERROR LEVE

Prueba N° 18

Figura 6.18. Imagen Prueba 18

En la Tabla 6.18., se muestra la comparación de los datos obtenidos.

Tabla 6.18. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 18

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	9	10	-1	ERROR LEVE
Antebrazo – Brazo Izquierdo	3	5	-2	ERROR LEVE
Muslo – Tobillo Izquierdo	7	0	7	ERROR ALTO

Hombro – Antebrazo Derecho	3	0	3	ERROR LEVE
Antebrazo – Brazo Derecho	1	0	1	ERROR LEVE
Muslo – Tobillo Derecho	1	0	1	ERROR LEVE

Prueba N° 19

Figura 6.19. Imagen Prueba 19

En la Tabla 6.19., se muestra la comparación de los datos obtenidos.

Tabla 6.19. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 19

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	11	10	1	ERROR LEVE
Antebrazo – Brazo Izquierdo	87	85	-2	ERROR LEVE
Muslo – Tobillo Izquierdo	8	0	8	ERROR ALTO

Hombro – Antebrazo Derecho	13	15	-2	ERROR LEVE
Antebrazo – Brazo Derecho	89	90	1	ERROR LEVE
Muslo – Tobillo Derecho	1	0	1	ERROR LEVE

Prueba N° 20

Figura 6.20. Imagen Prueba 20

En la Tabla 6.20., se muestra la comparación de los datos obtenidos.

Tabla 6.20. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 20

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	62	60	2	ERROR LEVE
Antebrazo – Brazo Izquierdo	23	25	-2	ERROR LEVE
Muslo – Tobillo Izquierdo	8	0	8	ERROR ALTO

Hombro – Antebrazo Derecho	68	65	3	ERROR LEVE
Antebrazo – Brazo Derecho	32	35	-3	ERROR LEVE
Muslo – Tobillo Derecho	7	0	7	ERROR ALTO

Prueba N° 21

Figura 6.21. Imagen Prueba 21

En la Tabla 6.21., se muestra la comparación de los datos obtenidos.

Tabla 6.21. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 21

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	29	30	-1	ERROR LEVE
Antebrazo – Brazo Izquierdo	4	5	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	8	5	3	ERROR LEVE

Hombro – Antebrazo Derecho	12	15	-3	ERROR LEVE
Antebrazo – Brazo Derecho	3	5	-2	ERROR LEVE
Muslo – Tobillo Derecho	7	10	-3	ERROR LEVE

Prueba N° 22

Figura 6.22. Imagen Prueba 22

En la Tabla 6.22., se muestra la comparación de los datos obtenidos.

Tabla 6.22. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 22

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	28	30	-2	ERROR LEVE
Antebrazo – Brazo Izquierdo	8	0	8	ERROR MEDIO
Muslo – Tobillo Izquierdo	16	15	-1	CORRECTA

Hombro – Antebrazo Derecho	31	30	1	ERROR LEVE
Antebrazo – Brazo Derecho	4	5	-1	ERROR LEVE
Muslo – Tobillo Derecho	21	10	11	ERROR ALTO

Prueba N° 23

Figura 6.23. Imagen Prueba 23

En la Tabla 6.23., se muestra la comparación de los datos obtenidos.

Tabla 6.23. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 23

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	8	5	3	ERROR LEVE
Antebrazo – Brazo Izquierdo	2	0	2	ERROR LEVE
Muslo – Tobillo Izquierdo	4	0	4	ERROR MEDIO

Hombro – Antebrazo Derecho	4	0	4	ERROR MEDIO
Antebrazo – Brazo Derecho	8	0	8	ERROR ALTO
Muslo – Tobillo Derecho	4	0	4	ERROR MEDIO

Prueba N° 24

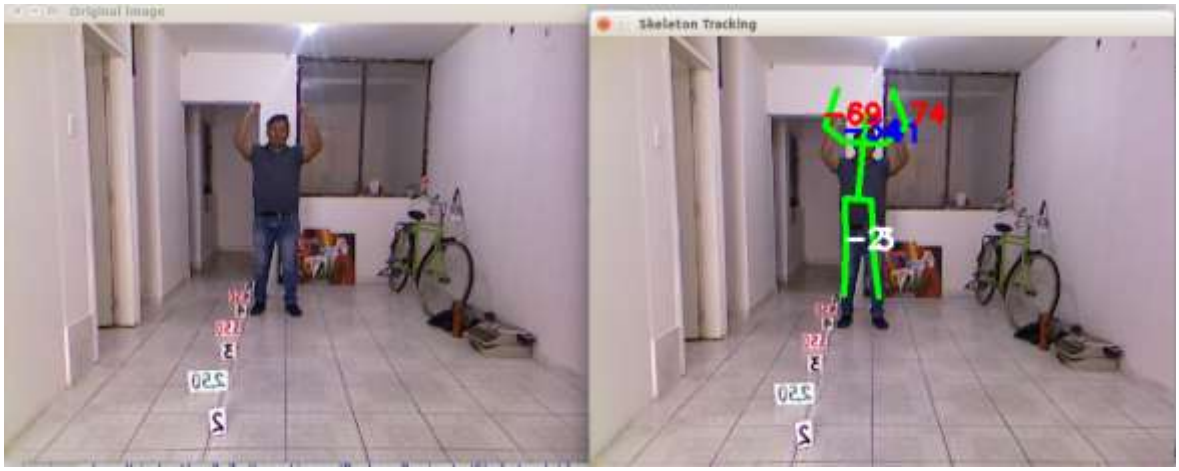

Figura 6.24. Imagen Prueba 24

En la Tabla 6.24., se muestra la comparación de los datos obtenidos.

Tabla 6.24. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 24

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	-	5	-5	ERROR MEDIO
Antebrazo – Brazo Izquierdo	74	75	-1	ERROR LEVE
Muslo – Tobillo Izquierdo	3	0	3	ERROR LEVE

Hombro – Antebrazo Derecho	-	0	0	CORRECTO
Antebrazo – Brazo Derecho	69	70	-1	ERROR LEVE
Muslo – Tobillo Derecho	2	0	2	ERROR LEVE

Prueba N° 25

Figura 6.25. Imagen Prueba 25

En la Tabla 6.25 se muestra la comparación de los datos obtenidos.

Tabla 6.25. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 25

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	69	70	-1	ERROR LEVE
Antebrazo – Brazo Izquierdo	18	20	-2	ERROR LEVE
Muslo – Tobillo Izquierdo	4	0	4	ERROR MEDIO

Hombro – Antebrazo Derecho	74	70	4	ERROR MEDIO
Antebrazo – Brazo Derecho	20	20	0	CORRECTA
Muslo – Tobillo Derecho	1	0	1	ERROR LEVE

Prueba N° 26

Figura 6.26. Imagen Prueba 26

En la Tabla 6.26., se muestra la comparación de los datos obtenidos.

Tabla 6.26. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 26

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	25	25	0	CORRECTA
Antebrazo – Brazo Izquierdo	2	5	-3	ERROR LEVE
Muslo – Tobillo Izquierdo	4	0	4	ERROR MEDIO

Hombro – Antebrazo Derecho	27	25	2	ERROR LEVE
Antebrazo – Brazo Derecho	1	0	1	ERROR LEVE
Muslo – Tobillo Derecho	1	0	1	ERROR LEVE

Prueba N° 27

Figura 6.27. Imagen Prueba 27

En la Tabla 6.27 se muestra la comparación de los datos obtenidos.

Tabla 6.27. Comparación de ángulos medidos por el dispositivo y ángulos reales – Prueba 27

Partes analizadas	**Ángulo medido por el dispositivo**	**Ángulo medido manualmente**	**Diferencia**	**Tipo de Medida**
Hombro – Antebrazo Izquierdo	22	20	2	ERROR LEVE
Antebrazo – Brazo Izquierdo	7	5	2	ERROR LEVE
Muslo – Tobillo Izquierdo	7	5	2	ERROR LEVE

Hombro – Antebrazo Derecho	22	25	-3	ERROR LEVE
Antebrazo – Brazo Derecho	18	20	-2	ERROR LEVE
Muslo – Tobillo Derecho	14	15	-1	ERROR LEVE