Universidad de las Ciencias Informáticas
Centro de Entornos Interactivos 3D (VERTEX)
Facultad 5

Reconocimiento y evaluación de la ejecución de ejercicios físicos utilizando Microsoft Kinect

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Roberto Ross Aguirre

Tutores: Ing. Andy Trujillo Rivero

Ing. Ernesto Leyva Piñeda

MSc. Marvyn Amado Márquez Rodríguez

Declaración de autoría

Declaro por este medio que yo Roberto Ross Aguirre, con carné de identidad 91072444302, soy el autor principal del trabajo final de tesis “Reconocimiento y evaluación de la ejecución de ejercicios físicos utilizando Microsoft Kinect” y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Roberto Ross Aguirre
Autor

_____
Ing. Andy Trujillo Rivero
Tutor

_____
Ing. Ernesto Leyva Piñeda
Tutor

_____
MSc. Marvyn Amado Márquez Rodríguez
Tutor

Resumen

La Inteligencia Artificial combinada con novedosos mecanismos de interacción hombre-máquina, posibilita la creación de múltiples aplicaciones al servicio de la humanidad. Una de ellas son los entrenadores virtuales, que asisten a las personas en tareas de ejercitación, rehabilitación o entrenamiento individualizado de sus habilidades motrices y de coordinación. Estos sistemas siempre están disponibles para los usuarios, generan planes de entrenamiento y demuestran paso a paso la ejecución correcta de cada movimiento. Sin embargo, muchos de ellos adolecen de la capacidad de retroalimentar a los usuarios sobre su desempeño durante la ejecución de los ejercicios. Esa es la motivación fundamental para el desarrollo de esta investigación, cuyo objetivo es reconocer y evaluar automáticamente el desempeño de personas durante la ejecución de ejercicios físicos.

Entre las principales técnicas de reconocimiento de patrones aplicables a este tipo de problemas se encuentran “Alineamiento Temporal Dinámico” y “Algoritmos Tipo Votación”. Estos fueron implementados y se demostró su capacidad de clasificación mediante una aplicación que captura datos de movimiento de una persona utilizando el dispositivo Microsoft Kinect. Los buenos resultados obtenidos son independientes de la posición en que se encuentre la persona, ya que toda la información es normalizada antes del proceso de clasificación. Sí se detectó relación entre los resultados de clasificación y los parámetros de configuración de los algoritmos. La arquitectura desarrollada es lo suficientemente extensible como incorporar un dispositivo de captura de datos de mayor precisión o implementar nuevos algoritmos que redunden en la obtención de mejores resultados de clasificación.

Palabras clave: inteligencia artificial, reconocimiento de patrones, evaluación automática de movimientos.

Índice

Índice de Tablas

Índice de Figuras

Introducción

La tecnología constituye un poderoso pilar del desarrollo cultural, social y económico de la sociedad moderna. Permite el desarrollo de productos, servicios, medios y herramientas capaces de satisfacer las necesidades humanas. Facilita el trabajo del hombre cuando se ve limitado por sus capacidades naturales. Y ya sea en las comunicaciones, la producción, la conquista del espacio, el transporte, la salud, la ciencia, el deporte, o simplemente en el hogar, ha incursionado en todas las esferas de la vida moderna.

La influencia de la tecnología es determinante en muchas áreas, pero su alcance se ve limitado cuando se necesitan virtudes humanas como el razonamiento lógico, la percepción del entorno y la reflexión. Es aquí donde juega su papel la Inteligencia Artificial (IA), ciencia que estudia y permite emular en las máquinas habilidades relacionadas con la inteligencia humana.

La IA permite programar computadoras para que lleguen a hacer aquello que la mente humana puede realizar (1). En la actualidad abarca numerosas ramas, desde aquellas de propósito general como el aprendizaje y la percepción a otras más específicas como la demostración de teoremas matemáticos y el diagnóstico de enfermedades.

La aplicación en juegos fue uno de los campos donde se desarrolló la IA. Se crearon múltiples programas para jugar ajedrez, damas y resolver el cubo de Rubik, entre otros. En relación con la educación, se han creado sistemas que actúan como tutores inteligentes, capaces de ajustar el contenido de las clases en dependencia del conocimiento y habilidad de los estudiantes. En cuanto a la robótica, ha llegado a fusionar aportes de muchas de sus ramas, mediante la creación de robots que simulan tanto las capacidades físicas del hombre como de su pensamiento lógico.

Otra de las áreas donde se han tenido avances es en el deporte. Por ejemplo: sistemas inteligentes de análisis estadístico y recomendación de estrategias de juego; los de evaluación del rendimiento deportivo; herramientas para el estudio biomecánico y entrenadores virtuales, entre otros (2).

Precisamente los entrenadores virtuales son una de las principales motivaciones de la presente investigación. Estos son sistemas para la personalización de entrenamientos, que suplen el déficit de entrenadores capacitados, abaratan sus servicios y garantizan su disponibilidad en cualquier momento. Permiten generar planes de entrenamiento individuales y demuestran detalladamente, mediante animaciones en tres dimensiones (3D) y desde diferentes vistas, cómo realizar cada ejercicio. Pero sus aplicaciones no se enfocan solamente en los deportistas de alto rendimiento, también pueden ser muy útiles a las personas comunes para realizar ejercicios físicos en casa o en el gimnasio; o para aquellas que practican artes marciales; o para las que se dedican a la danza, el ballet, la magia u otra manifestación artística que requiera perfeccionar los movimientos; o para los pacientes que se encuentren bajo tratamiento médico de rehabilitación física.

En nuestro país, donde el aumento de la calidad de vida de la población es una prioridad del Estado, se incursiona en el desarrollo y uso de las nuevas tecnologías para el mejoramiento de la salud pública. Ejemplo de ello es el entrenador virtual “Danzoterapia” que se desarrolla como parte de una colaboración entre el Centro Vertex, de la Universidad de Ciencias Informáticas (UCI) y el Centro Nacional de Rehabilitación "Julio Díaz" de La Habana.

El sistema “Danzoterapia” incluye un tutor virtual que realiza demostraciones animadas de los ejercicios que deben ejecutar los pacientes con limitaciones físico-motoras como parte de su terapia de rehabilitación. Los resultados obtenidos con la primera versión fueron positivos. Los pacientes podían ver cada detalle del ejercicio todas las veces que lo necesitaran, ya que el personaje animado las repite constantemente de forma correcta y sin cansarse. Además, posibilita la atención a un mayor número de pacientes al mismo tiempo. Sin embargo esta versión no es capaz de retroalimentar a las personas sobre el rendimiento alcanzado en la ejecución de cada movimiento. El sistema adolece de la capacidad de análisis y evaluación del desempeño de los pacientes durante la realización de los ejercicios, lo que le permitiría mostrar alertas para corregir las ejecuciones.

La capacidad de retroalimentación es un elemento significativo que complementaría en alto grado la utilidad de los entrenadores virtuales. La posibilidad de evaluar el rendimiento en la realización de ejercicios físicos,

permitirá traspasar una nueva frontera de comunicación entre el hombre y las máquinas. Para ello se requiere del concurso de las técnicas de IA, específicamente las relacionadas con el reconocimiento de patrones.

A partir de lo anterior, se impone el siguiente **problema de investigación:** ¿Cómo reconocer y evaluar de manera automática el desempeño de las personas durante la ejecución de ejercicios físicos?

Para ello se define como **objetivo de investigación:** Reconocer y evaluar automáticamente el desempeño de las personas durante la ejecución de ejercicios físicos.

El **objeto de estudio** está enfocado al área de "*reconocimiento de patrones de la IA*", y como **campo de acción**, "*el reconocimiento y evaluación de patrones de movimiento*".

Para dar cumplimiento a los objetivos planteados en este trabajo se realizan las siguientes tareas de investigación:

1. Elaboración del marco teórico de la investigación; para identificar métodos y herramientas utilizados en el desarrollo de sistemas similares.
   1.1. Caracterización del problema; para determinar qué variables y restricciones se deberán tener en cuenta en su solución.
   1.2. Caracterización de las técnicas de IA empleadas para el reconocimiento y evaluación de patrones de movimiento.
2. Diseño e implementación de un módulo de captura de datos; para obtener información sobre la ejecución de ejercicios físicos.
   2.1. Limpieza y transformación de los datos; para garantizar su calidad al eliminar errores, inconsistencias, redundancias y datos incompletos.
   2.2. Almacenamiento persistente de los datos de interés.
3. Diseño e implementación del módulo de evaluación de ejercicios físicos, con técnicas de reconocimiento de patrones de movimiento.
   3.1. Configuración de parámetros de las técnicas de reconocimiento de patrones, para su rendimiento óptimo.
4. Comprobación del desempeño y depuración del módulo de evaluación, para verificar su funcionamiento y aceptación.

Para guiar la investigación se utilizarán los siguientes métodos científicos:
Métodos teóricos:

- Histórico-lógico, para realizar un estudio de la evolución de los sistemas de captura y evaluación de movimientos relacionados con los entrenadores virtuales, determinando así las técnicas y tendencias actuales en su desarrollo.

- Analítico-sintético, para comprender los elementos esenciales del problema en cuestión, identificar en la literatura técnicas y estrategias que se han empleado anteriormente y adaptarlas para dar una solución válida.

Métodos empíricos:

- Experimentación y pruebas, para comprobar la efectividad y calidad de la solución brindada al problema.

El documento tiene la siguiente estructura:

Capítulo 1: Marco teórico de la investigación relacionado con las técnicas de IA utilizadas en sistemas de reconocimiento, captura y evaluación de patrones de movimiento. Se realiza un análisis de soluciones existentes para problemas similares. Además, se define la metodología, herramientas y tecnología que se emplearán en la solución.

Capítulo 2: Propuesta de solución. Se describe la propuesta de solución al problema de investigación. Se caracterizan los procesos involucrados. Se identifican los componentes fundamentales de la aplicación demostrativa. Se describen los algoritmos a implementar y los artefactos fundamentales que se generan en el desarrollo de la aplicación, según la metodología de desarrollo escogida.

Capítulo 3: Implementación y pruebas. Se definen los artefactos generados en la fase de implementación de la aplicación demostrativa. Además, se describen y analizan los resultados de las pruebas de software, así como del funcionamiento de los algoritmos de reconocimiento y evaluación de patrones de movimiento implementados.

Capítulo 1. Fundamentación teórica

En este capítulo se abordarán los principales elementos teóricos de las herramientas, técnicas y algoritmos utilizados en el reconocimiento y evaluación automático de patrones de movimiento de personas. Se analizan métodos para capturar, estructurar y almacenar los datos relativos al movimiento. Se analizan además varias aplicaciones que han dado solución a problemas similares.

La incorporación de capacidades para reconocer y evaluar la ejecución de rutinas físicas realizadas por las personas, permitiría a los entrenadores virtuales escalar a un nivel superior de interrelación hombre-máquina, acercándose cada vez más al objetivo de satisfacer las necesidades de los usuarios. Existen múltiples aplicaciones, desde la realización de ejercicios físicos en la casa, pasando por los deportistas y bailarines profesionales, hasta la rehabilitación física de pacientes que necesitan recuperar funciones motoras.

Para lograrlo se debe acudir a la experiencia acumulada en varias áreas del conocimiento, incluidas la “visión por ordenador” y el “reconocimiento de patrones” dentro de la IA. Ambas son áreas activas de investigación. La primera permite a las computadoras examinar y analizar imágenes en tiempo real para identificar objetos, rasgos y movimientos (3). El reconocimiento de patrones, por su parte, es la disciplina que permite la clasificación de objetos en varias categorías o clases. Estos objetos pueden ser físicos, imágenes, videos, señales en forma de onda, o cualquier otro tipo de medición que deba ser clasificada (4).

1.1 Reconocimiento y evaluación de movimientos

Para que una máquina sea capaz de reconocer y evaluar el movimiento de una persona, se identifican dos momentos fundamentales: primero debe ser capaz de identificar el movimiento, conocer cuándo inicia y cuándo termina; y luego debe clasificar la ejecución en dependencia de si fue buena o no, basándose en la experiencia.

1.1.1 Proceso de captura de movimientos

Se necesita una cámara que capte las secuencias de imágenes en movimiento. Luego deben identificarse en cada fotograma los píxeles pertenecientes a una persona, y además, seguir sus variaciones durante toda la secuencia (3). Estos

datos deben ser registrados y almacenados en la memoria de la máquina para su posterior análisis.

Debido a la gran cantidad de información que puede existir en una secuencia de video de una persona en movimiento, usualmente se transforma esta información en algo más sencillo de manejar. La idea parte de las técnicas de fotogrametría y se utiliza principalmente en el cine de animación y en los videojuegos. Se sintetiza la imagen de una persona en un esqueleto simplificado que incluye determinados puntos de interés, suficientes para describir el movimiento del cuerpo: la cabeza, el torso, la cadera, los brazos (incluyendo los codos y las muñecas) y las piernas (incluyendo las rodillas y los tobillos) (5).

Para realizar este tipo de tareas se ha popularizado en la comunidad científica internacional el uso de un dispositivo muy eficaz y relativamente barato para realizar tareas de detección y seguimiento en tiempo real de los puntos del cuerpo de una persona, el Microsoft Kinect (5). Sus características se detallan en el epígrafe 1.4.1 de este documento.

1.1.2 Evaluación de movimientos

Para evaluar cuan bien se ejecutó un movimiento, el sistema debe haber sido capaz de “aprender” previamente cuándo estos son correctos o no. Para ello es necesario valerse de técnicas de IA. El reconocimiento de patrones permite analizar los datos de los movimientos obtenidos durante el proceso de captura, y extrae de ellos características importantes que le permiten clasificar posteriormente la calidad de la ejecución de una nueva secuencia (6).

1.2 Reconocimiento de patrones

Son los medios por los cuales se puede interpretar el mundo, que permiten establecer comparaciones y desarrollar un pensamiento lógico. Es la ciencia que se ocupa de los procesos de ingeniería, computación y matemáticas, relacionados con objetos físicos y/o abstractos, con el propósito de extraer información relevante que permita determinar sus propiedades o la de sus conjuntos; así como establecer relaciones entre ellos (7).

Se definen los siguientes conceptos:

- **Objeto:** concepto con el cual representan los elementos sujetos a estudio.

- **Patrón:** es un sinónimo de objeto. En ocasiones se establece una diferencia entre un objeto a clasificar y uno ya clasificado, y en este último caso se le llama patrón.
- **Clase:** es un conjunto de objetos.
- **Rasgo:** es una propiedad o característica que debe tenerse en cuenta en el estudio de los objetos dados.
- **Reconocimiento:** se entiende por el proceso de clasificación de un objeto en un conjunto, es decir, el procedimiento por el cual se puede determinar las relaciones de pertenencia entre un objeto cualquiera y un conjunto de clase.

Un sistema de reconocimiento de patrones (Figura 1) incluye un sensor para captar información sobre los datos del problema. Dicha información se somete a un mecanismo extractor de características con el propósito de obtener lo útil y eliminar lo redundante e irrelevante. Finalmente los patrones identificados son empleados en la etapa de clasificación de nuevas instancias (8).

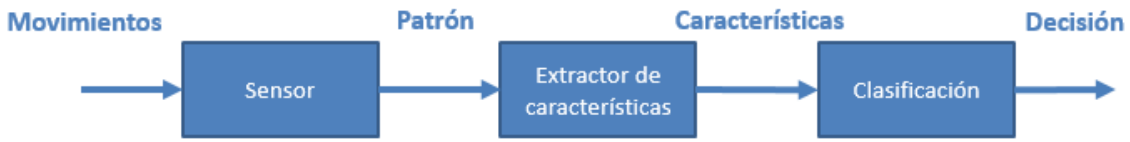

Figura 1. Sistema de reconocimiento de patrones

Sensor

Es el dispositivo encargado de la adquisición de datos que serán utilizados para la clasificación.

Extractor de características

Es el proceso de identificar características claves que puedan ser usadas en la toma de decisiones. En ocasiones viene precedido por un pre-procesado de la señal, necesario para corregir posibles deficiencias en los datos debido a errores del sensor, o bien para prepararlos con vista a posteriores procesos.

Clasificación

La clasificación automática permite asignar grupos o clases a los objetos de acuerdo a sus características descriptivas. Para ello hace uso del conocimiento adquirido durante el análisis de las características relevantes extraídas de los datos. En esta etapa se usa lo que se conoce como aprendizaje automático (del inglés *Machine Learning*). Esta rama de la IA desarrolla técnicas que permiten a las computadoras aprender basándose en la experiencia. Se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos (9). El aprendizaje puede ser:

- **Supervisado:** se basan en la disponibilidad de áreas de entrenamiento de las que se conoce a priori la clase a la que pertenecen, y que servirán para identificar modelos característicos de cada una de las clases (10).
- **No supervisado:** se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. Por lo tanto, el sistema tiene que ser capaz de reconocer relaciones válidas entre los datos, para aprender a etiquetar las nuevas entradas. Se utilizan algoritmos de agrupamiento automático en los que los individuos más próximos van formando clases (10).

En el caso de la investigación en curso, se emplearán métodos supervisados, pues se cuenta con ejemplos de ejercicios cuya ejecución ya fue clasificada. A continuación se describen algunos de los algoritmos que han sido utilizados en la literatura para reconocer patrones de movimiento.

1.2.1 Alineamiento Temporal Dinámico

El Alineamiento Temporal Dinámico (DTW, del inglés *Dynamic Time Warping*) es una técnica para encontrar el alineamiento óptimo entre 2 secuencias temporales (11). Ha sido utilizado principalmente en aplicaciones de reconocimiento de voz. También se usa en otras áreas como la medicina; la bioinformática, para alinear secuencias de proteínas; o en música, para procesar señales acústicas (12).

Su funcionamiento se basa en medir la distancia entre dos secuencias temporales deformadas y desplazadas en el eje del tiempo siendo capaz de alinearlas hasta que encuentre una coincidencia óptima entre ellas. En el contexto del reconocimiento de patrones de movimientos, esas secuencias están formadas por los puntos de las trayectorias que describen las partes del cuerpo.

El algoritmo alinea dos trayectorias estirando y encogiendo el eje temporal iterativamente hasta lograr una mínima distancia entre cada par de puntos de las trayectorias. Como resultado se obtiene un valor de distancia que indica cuánto se asemejan las dos trayectorias (12).

1.2.2 Modelo de algoritmos de votación

La idea esencial de este modelo es la de la precedencia o analogías parciales. Se trata de que un objeto puede parecerse a otro, pero no necesariamente en su totalidad. Las partes en que sí se parecen pueden dar información acerca de posibles regularidades. Por supuesto, no todas en la misma magnitud (13).

Este modelo se describe mediante 6 etapas:

1. **Sistemas de conjuntos de apoyo:** por conjunto de apoyo se entiende un subconjunto no vacío de rasgos en términos de los cuales se analizarán los objetos.
2. **Funciones de semejanza:** estas funciones definen la forma en que serán comparadas las descripciones de los objetos en cuestión, o sea la comparación entre los valores de sus rasgos.
3. **Evaluación por fila dado un conjunto de apoyo fijo:** una vez definidos el sistema de conjuntos de apoyo y la función de semejanza, se inicia un proceso de "contabilización" de votación, en cuanto a la medida de semejanza entre las diferentes partes de las descripciones de los objetos ya clasificados y los que se desean clasificar.
4. **Evaluación por clase dado un conjunto de apoyo fijo:** de lo que se trata es de totalizar las evaluaciones obtenidas para cada uno de los objetos respecto al objeto que se quiere clasificar.
5. **Evaluación por clase para todo el sistema de conjuntos de apoyo:** hasta el paso anterior todos los cálculos se habían hecho para un conjunto de apoyo fijo, ahora totalizaremos los mismos para todo el sistema seleccionado.
6. **Regla de solución: s**e trata ahora de establecer un criterio para que, sobre la base de cada una de las votaciones obtenidas en la etapa

precedente, se pueda tomar una decisión en cuanto a las relaciones. del objeto a clasificar con cada una de las clases del problema.

1.2.3 Modelos ocultos de Markov

Un modelo oculto de Markov (HMM, del inglés *Hidden Markov Model*), es un proceso estocástico que consta de un proceso de Markov no observado (oculto) $Q = \{q_t\} \forall\, t \in \mathbb{N}$ y un proceso observado $O = \{o_t\} \forall\, t \in \mathbb{N}$ cuyas observaciones son estocásticamente dependientes de los estados ocultos. Su principal objetivo es determinar los parámetros desconocidos de dicha cadena a partir de los parámetros observables (14).

Estos sistemas evolucionan en el tiempo pasando aleatoriamente de estado a estado y emitiendo en cada momento al azar algún símbolo del alfabeto P. Cuando se encuentra en el estado $q_{t-1} = i$, tiene la probabilidad a_{ij} de moverse al estado $q_t = j$ en el siguiente instante y la probabilidad $b_j(k)$ de emitir el símbolo $o_t = v_k$ en el tiempo t. Sólo los símbolos emitidos por el proceso q son observables, pero no la ruta o secuencia de estados q, de ahí el calificativo de oculto de Markov, ya que el proceso de Markov q es no observado.

Los modelos ocultos de Markov son especialmente aplicados a reconocimiento de formas temporales como reconocimiento del habla, de escritura manual y de gestos. En el reconocimiento de gestos se pueden describir los mismos como procesos de Markov donde las posiciones del cuerpo son estados, y los movimientos son transiciones a otro estado. De esta manera podemos modelar cada gesto como un conjunto de transiciones y estados (15).

1.2.4 Redes neuronales artificiales (RNA)

Son redes de elementos simples, interconectadas masivamente en paralelo y con organización jerárquica, las cuales interactúan con los objetos del mundo real simulando la forma en que lo hace el sistema nervioso biológico (16).

Una red neuronal artificial (RNA) está constituida por neuronas interconectadas y agrupadas en varias capas: una de entrada, una de salida y una o varias ocultas, donde se realiza la mayor parte del procesamiento (Figura 2).

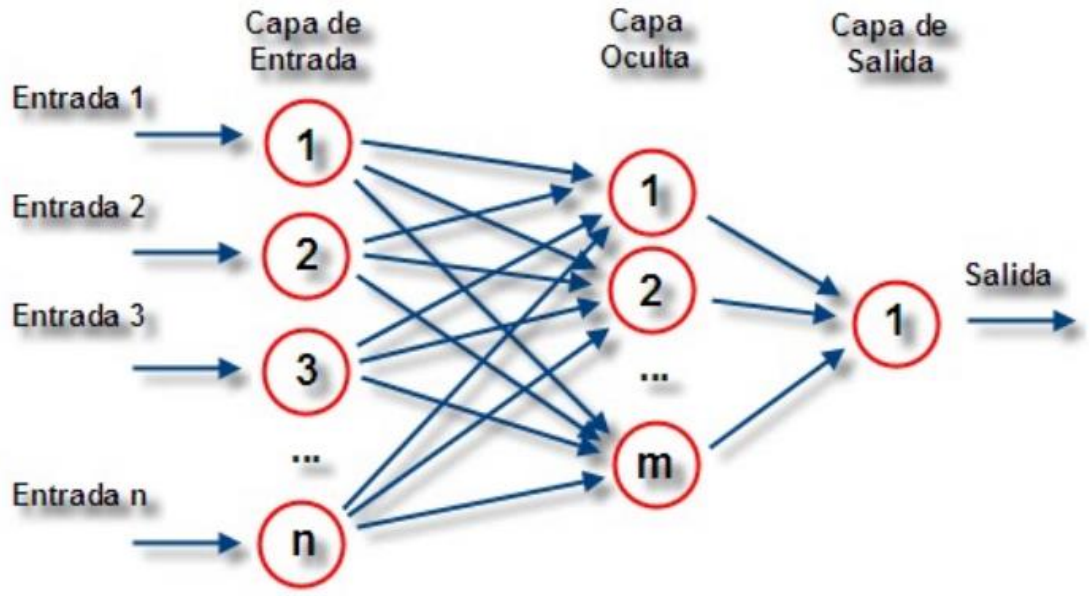

Figura 2 Esquema de una red neuronal

Las neuronas reciben estímulos provenientes de otras a través de las conexiones representadas por una matriz de pesos. El valor que trasmite cada neurona artificial a la siguiente se calcula con la función de activación, que depende de la sumatoria de las entradas pesadas a la neurona. Durante el proceso de entrenamiento de la RNA se utiliza un conjunto de entrenamiento, conformado por instancias cuya clasificación se conoce a priori. A partir del error de clasificación obtenido para cada instancia, se ajustan los pesos de las conexiones de forma que se minimice este error (17). Cuando concluye el entrenamiento, la RNA debe ser capaz de clasificar nuevas instancias.

El uso de RNA para identificar gestos no está tan extendido como otros modelos, pero se han obtenido resultados favorables, aunque queda mucho margen para la investigación (18).

1.2.5 Generalidades significativas

Para seleccionar la técnica apropiada, se debe tener en cuenta que:

- En el proceso de clasificación se hace necesario respuestas rápidas por parte del algoritmo.
- Se debe tener en cuenta qué partes del cuerpo están incidiendo sobre el resultado de la clasificación.

- La clasificación puede verse afectada por la oclusión de los puntos que describen el movimiento de la persona.

Del conjunto de técnicas analizadas que cumplen con estas condiciones, las RNA y HMM requieren de un conjunto de datos de entrenamiento. Mientras mayor sea la cantidad y calidad de los datos, mejor será el proceso de generalización que se obtenga para clasificar nuevas instancias. Esto implica la necesidad de dedicar mucho tiempo y recursos en la obtención del juego de datos. Por otra parte, las técnicas ATV y DTW no requieren de un conjunto de entrenamiento tan complejo, pues solo con unas pocas demostraciones por ejercicio, se pueden obtener resultados satisfactorios. Por tanto, estas últimas son las que se seleccionan para la evaluación de las secuencias de movimiento.

1.3 Análisis de soluciones similares

Una vez realizado el proceso de investigación en cuanto al reconocimiento y análisis de patrones de movimiento, se ha determinado que existen sistemas con particularidades afines a este estudio y que a continuación se describen.

1.3.1 Proyecto TEKI

TEKI es un proyecto de tele-asistencia de pacientes crónicos basado en la tecnología Kinect de Microsoft y resultado de la participación de las compañías Osakidetza, Accenture y Microsoft (19). En su hogar y a través del televisor el paciente recibe las indicaciones y acciones diarias a realizar por parte de su médico o enfermera de referencia como se muestra en la Figura 4. El Kinect es utilizado tanto para la interacción del paciente con la interfaz de usuario como para el análisis de los ejercicios de rehabilitación que ejecuta.

Figura 3 Interfaz de la aplicación TEKI

1.3.2 Herramienta de reconocimiento de gestos

Herramienta de reconocimiento de gestos *(Gesture Recognition Tool)* facilita la definición y reconocimiento de gestos con Kinect. Permite al desarrollador construir un conjunto de gestos entrenamiento y provee soporte para construir los clasificadores que identifiquen gestos similares a aquellos entrenados. Para la especificación del conjunto de gestos de entrenamiento, brinda una interfaz gráfica (Figura 4) que permite la grabación, edición y administración de gestos utilizando el dispositivo Kinect, permite el entrenamiento de diferentes técnicas del aprendizaje de máquinas para el reconocimiento de gestos. En particular soporta las técnicas: DTW y HMM (20).

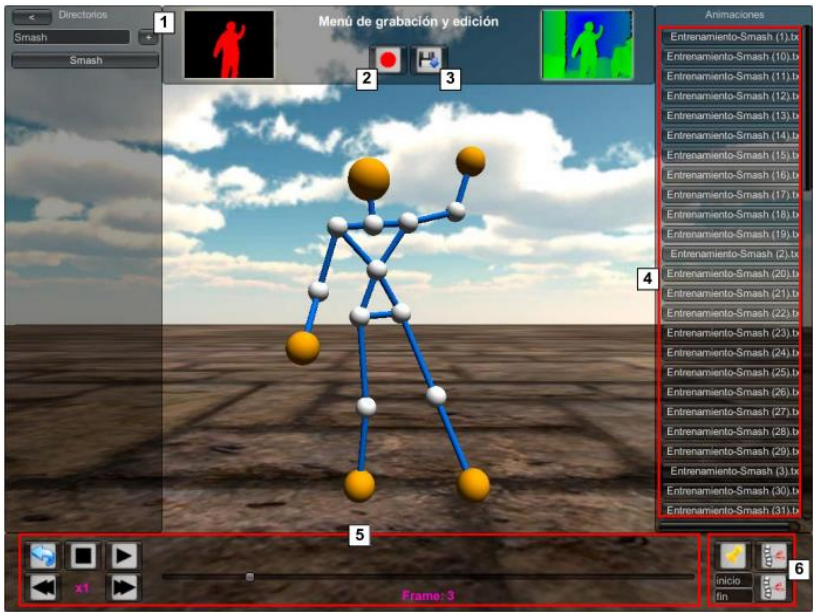

Figura 4 Interfaz de la herramienta GRTool

1.3.3 Tele operación de robots

Es el manejo de robots por un operador humano (Figura 5), en este caso mediante visión artificial llevada a cabo por la capacidad del Kinect para detectar movimientos (12). Se basa en la detección de movimientos mediante el Kinect, los datos obtenidos son procesados para finalmente ser ejecutados por el robot que se está operando. El procesamiento de los datos registrados es llevado a cabo mediante el algoritmo DTW llegando a la captura del movimiento, clasificación y evaluación del mismo.

Figura 5 Tele operación de robots

1.3.4 Generalidades significativas

Estos sistemas tienen en común la utilización del Kinect como dispositivo de captura de movimientos y se dividen en dos componentes fundamentales. El primero, orientado a la captura y tratamiento de los datos. En el procesamiento usualmente se normalizan los datos para garantizar que todas las operaciones de comparación se lleven a cabo en un mismo espacio vectorial. Y el segundo componente está relacionado con la utilización de algoritmos de reconocimiento de patrones para evaluar la ejecución de los nuevos movimientos en relación con ejecuciones correctas previamente aprendidas.

1.4 Herramientas, tecnologías y metodología a emplear

A continuación se exponen las tecnologías, herramientas y metodología seleccionadas para el desarrollo de una aplicación que incorpore la capacidad de reconocer y evaluar patrones de movimiento.

1.4.1 Microsoft Kinect

El Kinect es un controlador de juego libre y entretenimiento creado por Alex Kipman, desarrollado por Microsoft para la videoconsola Xbox 360, y desde junio del 2011 para PC a través de Windows 7 y Windows 8.3 Kinect (21). Permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional. Todo se realiza mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, objetos e imágenes. Tiene incluido cámara RGB (*RedGreenBlue*), sensor de profundidad, conjunto de micrófonos, así como emisor y receptor infrarrojos (Figura 6).

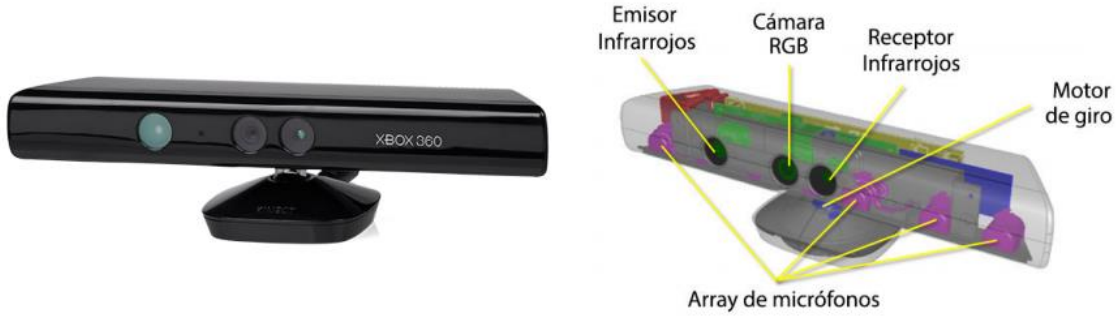

Figura 6 Composición del Kinect

Funcionamiento del escáner 3D de luz estructurada:

Kinect proyecta con el emisor de luz infrarroja (Figura 7) un patrón aleatorio de puntos, y la cámara infrarroja captura la luz reflejada. Como el patrón emitido es conocido por Kinect, la cámara puede obtener la información de profundidad a través de algoritmos de visión estéreo, como la correlación cruzada o la triangulación estéreo. Así detecta un conjunto de 20 puntos que definen el esqueleto de una persona y realiza un seguimiento del mismo en tiempo real.

Figura 7 Luz estructurada

Esqueleto detectado por el Kinect:

Kinect procesa la imagen de profundidad y obtiene la información de hasta dos los esqueletos humanos detectados en 3D en tiempo real. Las posiciones tridimensionales de cada uno de los veinte puntos de interés que definen el esqueleto (Figura 8) son almacenadas.

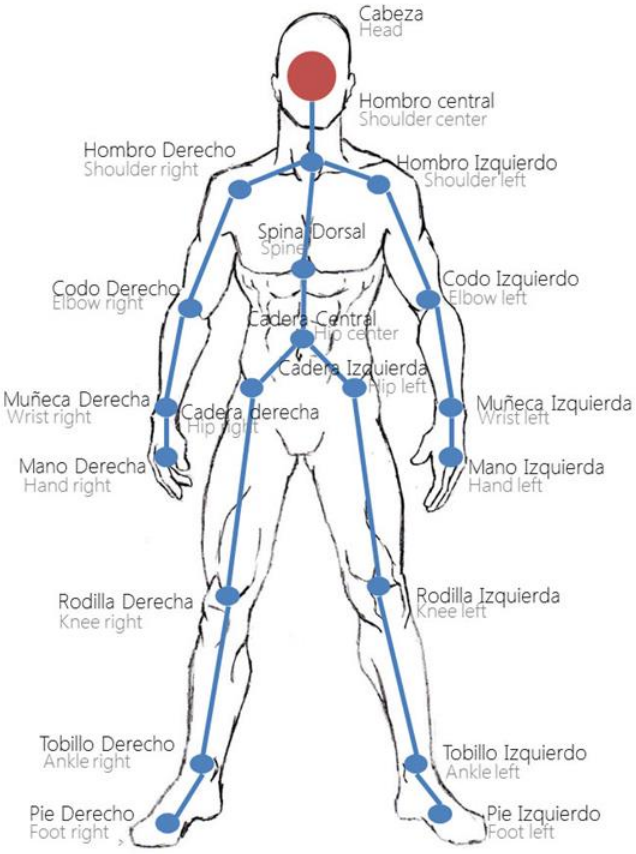

Figura 8 Ejemplo de esqueleto detectado por el Kinect

1.4.2 Unity

Unity es un motor de videojuegos multiplataforma creado por *Unity Technologies*. Puede usarse junto con *3ds Max*, *Maya*, *Blender*, *Adobe Photoshop*. Se le pueden incorporar *scripts* escritos en JavaScript, *C#* o *Boo* (cuya sintaxis está inspirada en *Python*) (22). Para el desarrollo de la aplicación es primordial lograr establecer una conexión entre el Microsoft Kinect y las interfaces de Unity 3D. Esto se logra gracias a la compatibilidad entre estas herramientas.

1.4.3 Entorno integrado de desarrollo, IDE

Un IDE (acrónimo en inglés de *Integrated Development Environment*) es un entorno de programación que integra varias herramientas con el objetivo de

facilitar el desarrollo de software sobre uno o varios lenguajes de programación. La mayoría de los IDE cuentan con herramientas tales como: editor de código, herramientas para el rastreo de código, compilador, depurador y constructor de interfaz gráfica (23).

1.4.4 MonoDevelop

MonoDevelop es un entorno de desarrollo integrado libre y gratuito, diseñado primordialmente para *C#* y otros lenguajes *.NET*. Este IDE incluye manejo de clases, ayuda incorporada, completamiento de código, soporte para proyectos y un depurador integrado. Está vinculado directamente con el motor de videojuegos Unity 3D, por lo que es recomendada su utilización en este proyecto.

1.4.5 Lenguaje de programación C#

C# es un lenguaje de programación orientado a objeto desarrollado y estandarizado por Microsoft como parte de su plataforma *.NET*, es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma *.NET*, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes. Aunque *C#* forma parte de la plataforma .*NET*, esta es una API, mientras que *C#* es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma (24). Además, la biblioteca MS-SDK utilizada para comunicarse con el Kinect fue desarrollada haciendo uso de este lenguaje.

1.5 Metodologías de desarrollo del software

Las metodologías pesadas velan por un riguroso cumplimiento de las actividades a realizar, requieren de una documentación exhaustiva para prever todo antes de que ocurra. Este tipo de metodología es eficaz y necesaria cuando se realizan proyectos complejos, que involucran una gran cantidad de recursos; y por tanto demandan un gran esfuerzo de planeación, organización y control (25). Algunas de las metodologías más conocidas de este tipo se muestran a continuación:

- **RUP**

Proceso unificado (en inglés *Rational Unified Process*) es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones,

diferentes niveles de competencia y diferentes tamaños de proyectos. Dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental (26). Se divide en cuatro fases:

- Inicio (define el alcance del proyecto)
- Elaboración (definición, análisis, diseño)
- Construcción (implementación)
- Transición (fin del proyecto y puesta en producción)

RUP define nueve disciplinas a realizar en cada fase del proyecto: Modelado del negocio, Análisis de requisitos, Análisis y diseño, Implementación, Test, Distribución, Gestión de configuración y cambios, Gestión del proyecto y Gestión del entorno. Cada fase en RUP puede descomponerse en iteraciones. Una iteración es un ciclo de desarrollo completo dando como resultado una entrega de producto ejecutable (interna o externa) (27).

- ***Microsoft Solutions Framework***

Microsoft Solutions Framework (MSF) es un enfoque personalizable para entregar correcta y más rápidamente soluciones tecnológicas, con menos personas y menos riesgo, pero con resultados de más calidad (28). MSF ayuda a los equipos a resolver directamente las causas más comunes de error en el proyecto de tecnología, lo cual mejora los índices de buenos resultados, de calidad de la solución y de impacto comerciales. MSF se centra en:

- Alinear objetivos empresariales y tecnológicos
- Establecer objetivos, roles y responsabilidades claros para el proyecto
- Implementar un proceso iterativo, basado en hitos/puntos de control
- Administrar riesgos de forma proactiva
- Respuestas efectivas a los cambios

1.5.1 Metodologías ágiles

Las metodologías ágiles emergen como una posible respuesta para todos aquellos proyectos cuyo entorno es muy cambiante y poseen tiempo limitado

para el desarrollo del sistema. Están especialmente orientadas para proyectos pequeños, por lo que constituyen una solución a medida para ese tipo de entornos. Permiten una elevada simplificación documental, pero a pesar de ello no renuncian a las prácticas esenciales para asegurar la calidad del producto (29). Actualmente existe un gran número de estas metodologías, algunas de las cuales se muestran a continuación:

- **XP**

Programación Extrema (del inglés *eXtreme Programming*) fue la primera de las metodologías ágiles. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (30). El ciclo de vida ideal de XP consiste de seis fases:

  - Exploración
  - Planificación de la Entrega (*Release*)
  - Iteraciones
  - Producción
  - Mantenimiento
  - Muerte del Proyecto

Es la metodología ágil de más renombre en la actualidad, y se diferencia de las demás metodologías que forman este grupo en un aspecto en particular: el alto nivel de disciplina de las personas que participan en el proyecto (29).

- **Scrum**

Scrum es un método iterativo e incremental que enfatiza prácticas y valores de la gestión de proyectos por sobre las demás disciplinas del desarrollo. Al principio

del proyecto se define el *Product Backlog*, que contiene todos los requerimientos funcionales y no funcionales que deberá satisfacer el sistema a construir. Los mismos estarán especificados de acuerdo a las convenciones de la organización ya sea mediante: casos de uso, diagramas de flujo de datos, incidentes, tareas, etc. La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otras – como XP – para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un marco de trabajo ágil de administración de proyectos que puede ser combinado con cualquier otra metodología (29).

1.5.2 Selección de la metodología de desarrollo

Luego de realizar un análisis de las principales metodologías de desarrollo, se identificó como la más idónea para el proyecto a desarrollar la metodología ágil XP. La aplicación a desarrollar es pequeña, con elevado riesgo técnico y pocas personas en el equipo de trabajo. Además, se dispone de un tiempo limitado y se cuenta con la participación del cliente como parte del equipo de trabajo. Todos estos elementos hacen que XP sea totalmente compatible con este proyecto.

1.6 Sumario

En este capítulo se han ofrecido los elementos teóricos que sirven de sustento científico a la investigación. Se desea resaltar los siguientes puntos:

- Para evaluar un movimiento de forma automática, primeramente se realizan los procesos de captura (detección y seguimiento, que en esta investigación estarán apoyados en las potencialidades del Kinect) y preprocesamiento de los datos (para garantizar su limpieza y que se trabaje en el mismo espacio vectorial).
- Debe entrenarse un algoritmo de reconocimiento automático de patrones para que aprenda a identificar movimientos de referencia correctos.
- Luego el modelo aprendido se utiliza para evaluar nuevos movimientos en función de su semejanza con estos.

- Entre las técnicas de reconocimiento de patrones utilizadas en el mundo para detectar movimientos de personas, las que más similitudes tienen con el problema de evaluación actual son los algoritmos DTW y ATV, por lo que se decide implementar ambos.
- Se seleccionaron las herramientas y metodologías de desarrollo para el sistema.

Capítulo 2. Características del sistema

El actual capítulo tiene como propósito presentar la propuesta de solución por medio de la metodología XP en el cual se desarrolla el problema que da origen al sistema. También se hace referencia a las cuestiones relacionadas con las fases de Exploración y Planificación. Se detallan las historias de usuario y se establece el orden en que serán implementadas atendiendo a su prioridad. En general quedan representados los pasos iniciales de la metodología de desarrollo elegida.

2.1 Sistema de evaluación de la ejecución de ejercicios físicos

El sistema de evaluación automática que se propone como solución estará diseñado de forma tal, que permita utilizar las funcionalidades del sensor Kinect en cuanto a identificar los puntos del cuerpo de una persona. Los datos obtenidos serán pre-procesados para su normalización. Además, debe permitir generar los modelos de comparación y llevar a cabo la misma. Por último, el resultado de la evaluación. El sistema estará dividido en tres partes principales orientadas a:

- Captura de datos.
- Preprocesamiento de los datos.
- Evaluación.

Y estará enfocado a dos procesos (Figura 9 y Figura 10):

- Crear modelos de comparación.

Figura 9 Proceso para crear un modelo de comparación

- Evaluar la similitud entre un nuevo movimiento y un modelo.

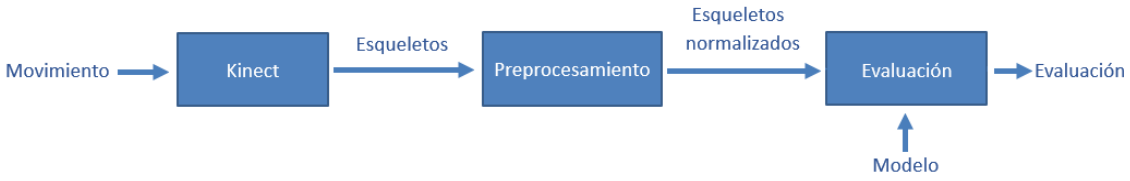

Figura 10 Proceso de evaluación

2.1.1 Preprocesamiento

Uno de los principales problemas cuando comparamos dos movimientos, aunque la persona haga exactamente el mismo movimiento, es la posición relativa de la persona que realiza el gesto respecto al Kinect. En una situación ideal, la persona estaría siempre situada de frente y a la misma distancia del sensor Kinect. Sin embargo, en una situación real, la persona no está situada siempre totalmente de frente al Kinect ni a la misma distancia. En los mejores casos, la posición variará unos centímetros y el ángulo respecto al Kinect será de unos grados; y en los peores casos, la posición variará varios metros y el ángulo puede superar los 45°.

Para solucionar esta situación se normaliza el esqueleto detectado mediante la traslación y luego la rotación del mismo. Esto garantiza comparar todos los esqueletos en una misma posición y ángulo con respecto al Kinect.

Traslación

La translación de un objeto consiste en moverlo cierta distancia, en una dirección determinada. En 3D, el sistema de referencia homogéneo tendrá 4 dimensiones, por lo que la traslación del punto $\boldsymbol{V} = (x, y, z, 1)$ quedará indicada (ecuación.1).

$$\boldsymbol{V}' = (x', y', z', 1) = \boldsymbol{V} * \boldsymbol{T} \quad (1)$$

Donde $\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$ y **matriz de traslación** en 3d. (t_x, t_y, t_z) se conoce como el vector de traslación. La expresión anterior es equivalente al sistema de ecuaciones (ecuación.2).

$$x' = x + t_x \qquad y' = y + t_y \qquad z' = z + t_z \qquad 1 = 0 + 1 \quad (2)$$

Como se comentó arriba, para trasladar un esqueleto se ha de aplicar la matriz $\boldsymbol{T}$ a todos los puntos. Es importante observar que al hacer la traslación del mismo

sus proporciones no varían, puesto que todos los **puntos significativos** (los 20 puntos detectados) se mueven la misma distancia, en la misma dirección. Para realizar la **traslación inversa** a la efectuada mediante la matriz $\boldsymbol{T}$, se ha de aplicar la **matriz inversa**, es decir, la $\boldsymbol{T}^{-1}$, que se obtiene cambiando el signo (multiplicando por −1) el **vector de traslación** (31).

En este caso se requiere trasladar el esqueleto detectado a la posición (0, 0, 0), para lograrlo se toma como punto medio del mismo el denominado Cadera Central (*Hip Center* en inglés). Por tanto, se realiza una traslación inversa y el vector de traslación que se toma es $V = HC = (x, y, z)$. Como resultado el esqueleto queda posicionado en las coordenadas (0, 0, 0) según su punto centro (Figura 11).

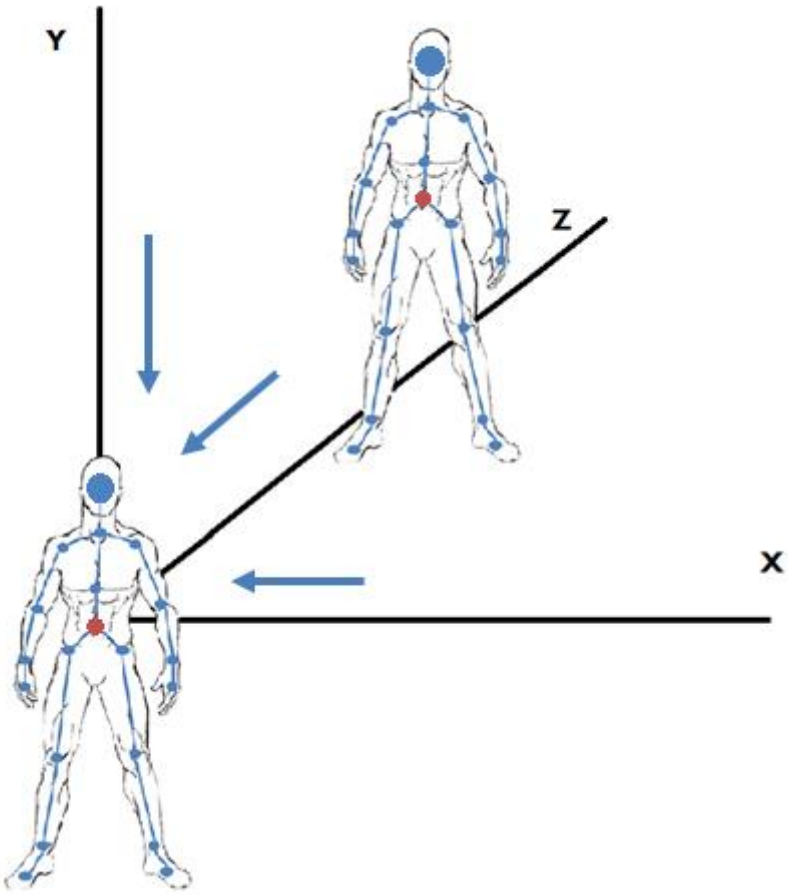

Figura 11 Traslación del esqueleto

Rotación o giro

Para realizar el giro del esqueleto, se ha de establecer un eje de rotación, así como el ángulo y el sentido de giro alrededor de dicho eje. El eje Y se toma como

el de rotación, ya que en el mismo no se presentan cambios en cuanto al ángulo con respecto al Kinect. El ángulo para girar es el mismo que se tiene con respecto al eje X, puesto que se quiere posicionar el eje natural del esqueleto paralelo a X y perpendicular al eje Z como se muestra en la Figura 12.

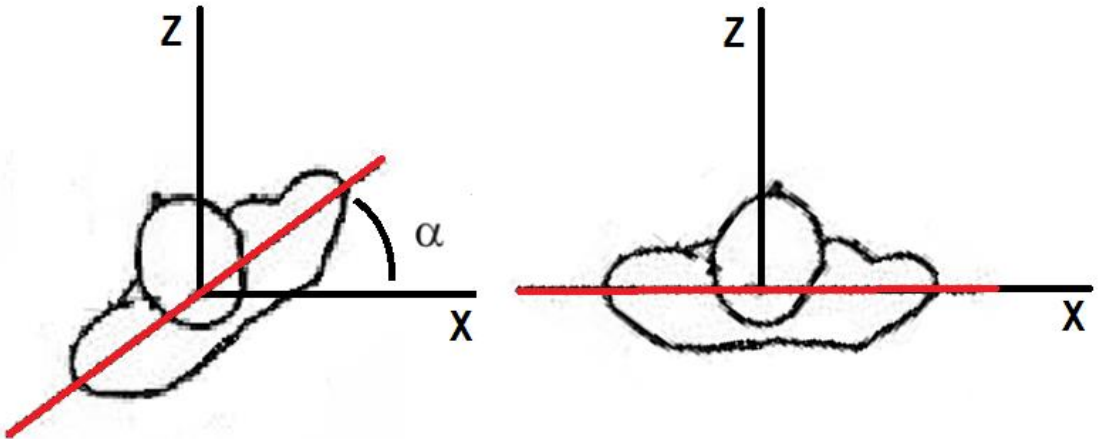

Figura 12 Rotación del esqueleto.

Las expresiones que para calcular las nuevas posiciones de los puntos se presentan (ecuación.3, 4 y 5).

$$P' = (x', y', z', 1) \quad P = (x, y, z, 1) \quad G_y = \begin{bmatrix} cos(\alpha) & 0 & sen(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sen(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$P' = P * G_Y \quad (4)$$

$$x' = x * cos(\alpha) - z * sen(\alpha) \quad z = x * sen(\alpha) + z * \cos(\alpha)$$

$$y' = y \; (Por\ ser\ el\ eje\ de\ rotación) \quad (5)$$

Al calcular cada punto del esqueleto queda el eje natural paralelo al eje X y perpendicular al Z. Con este último paso, en la normalización de los datos del esqueleto, ya se pueden comparar todos en una misma posición y ángulo respecto al Kinect.

2.2 Generar modelo de comparación

Un modelo de referencia para comparar las nuevas capturas es fundamental para evaluar la ejecución. El proceso de crear un modelo de comparación (Figura 9) consiste en realizar una captura con el Kinect, los datos son pre-procesados

y por último se crea el modelo con la secuencia de esqueletos ya normalizados. Esta etapa es realizada por un experto y se asegura de que la secuencia cumple con los requerimientos para ser utilizada como modelo de comparación. Además, debe cerciorarse que el ejercicio capturado se realiza de la forma correcta para no provocar errores posteriores. Ya cumplidas las condiciones para la obtención de un modelo correcto, las secuencias de esqueletos son guardadas en un fichero y se consulta en el momento de comparar con un nuevo movimiento. Este proceso define un modelo para cada ejercicio y la evaluación de un ejercicio debe realizarse con el modelo que le corresponde al mismo.

2.3 Evaluación de la ejecución del ejercicio físico

En esta etapa es donde se llega a un criterio de evaluación que clasifica la ejecución del ejercicio. Para esto se debe tener el modelo correspondiente al ejercicio que se desea evaluar, luego se realiza la captura de la ejecución, son pre-procesados los datos de la misma y se comparan con los pertenecientes al modelo.

Tanto la captura como el modelo se definen como una secuencia de esqueletos (Figura 13), cada uno compuesto por los 20 puntos y representados en el sistema de coordenadas tridimensionales (ecuación.6).

$$\begin{cases} E = (e_1, e_2, e_3, \dots, e_n)\ \forall n \in N \\ e_i = (p_1, p_2, p_3, \dots, p_{20})\ \forall i \in [1, n] \\ p_j = (x_j, y_j, z_j) \forall j \in [1{,}20] \end{cases} \quad (6)$$

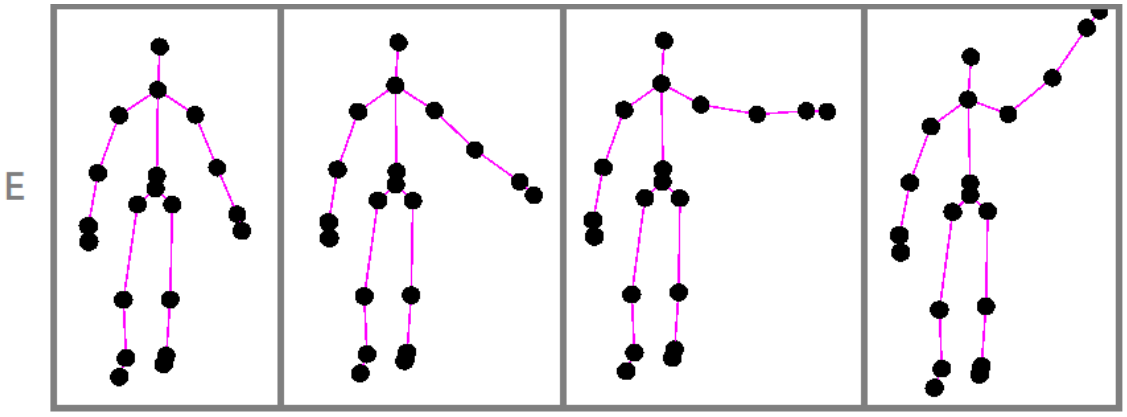

Figura 13 Secuencia de esqueletos

Ya obtenidas las secuencias $E = (e_1, e_2, \dots, e_N)$ (perteneciente al modelo) de longitud $N \in \mathbb{N}$ y $S = (s_1, s_2, \dots, s_M)$ (perteneciente a la captura) de longitud $M \in$

$\mathbb{N}$. Mediante los algoritmos ATV y DTW se calcula el grado de similitud entre el modelo y la captura.

- **Algoritmo tipo votación (ATV)**

Este algoritmo está motivado por la mecánica de los algoritmos de votación, principalmente a la evaluación por rasgos de los objetos, pero no está orientado a la clasificación de los mismos sino a encontrar cuan parecidos son o se puede decir el grado de similitud entre los mismos. En este caso los objetos son las secuencias de esqueletos definidas anteriormente.

Primeramente se calcula la distancia punto a punto de cada uno de los que componen los esqueletos pertenecientes a las secuencias E y S mediante la distancia euclidiana al cuadrado (ecuación.7).

$$d(p_{ej}, p_{sj})^2 = (x_{ej} - x_{sj})^2 + (y_{ej} - y_{sj})^2 + (z_{ej} - z_{sj})^2 \quad (7)$$

Al obtener los valores de distancias se ajustan primero a una función cuadrática, donde los resultados se encuentran de 0 al valor de ajuste K (ecuación.8). El valor de ajuste se determina como dato experto, o sea un especialista define el valor de K, luego se calcula el grado de similitud entre cada uno de los esqueletos (ecuación.9).

$$f(p_{ej}, p_{sj}) = K - 100 * d(p_{ej}, p_{sj})^2 \quad f(p_{ej}, p_{sj}) \in [0, K] \quad (8)$$

$$G(e_i, s_i) = \frac{\sum_{j=1}^{J} f(p_{ej}, p_{sj})}{K * J} * 100 \quad (9)$$

Estos cálculos se realizan para conocer cuan distinto es un esqueleto de otro según la posición en que se encuentra dentro de la secuencia (Figura 14).

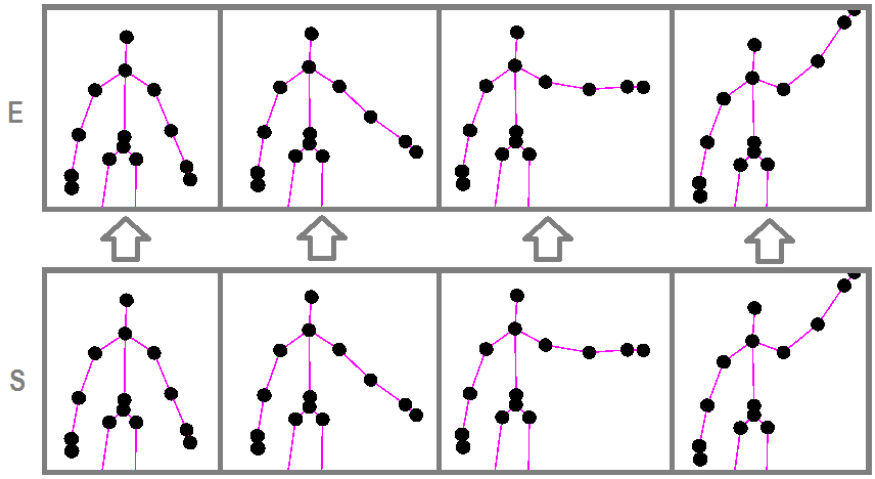

Figura 14 Comparación de los esqueletos de las secuencias E y S.

El valor calculado $G(e_i, s_i)$ se toma como el grado de similitud, de cada par de esqueletos (e_i, s_i) y es inversamente proporcional a la distancia entre los mismos. Por tanto, si la diferencia de posición entre los puntos de un esqueleto con respecto a otro aumenta, la distancia entre los mismos crece haciendo disminuir desde un valor K a 0 el grado de similitud.

Para el resultado general $ATV(E, S)$ se encuentra el grado de similitud entre las secuencias mediante el promedio en cuanto al grado de similitud de los esqueletos que componen a la secuencias (ecuación.10).

$$ATV(X, Y) = \frac{\sum_{l=1}^{N} G_l(e_i, s_i)}{W} \quad (10)$$

Donde W es la cantidad de evaluaciones realizadas, que para secuencias de igual longitud toma esta como valor $W = N \; \forall \; N \; es \; la \; longitul \; de \; E$.

Para la comparación de secuencias de distintas longitudes se tiene el valor W que en este caso toma la menor de las longitudes entre las secuencias E y S (ecuación.11). Además, este valor representa la cantidad de evaluaciones, por par de esqueletos, que aportan al resultado final, lo que no significa que se dejen de evaluar partes de la secuencia.

$$W = \begin{cases} N \; si \; N \leq M \\ M \; en \; otro \; caso \end{cases} \quad (11)$$

En cuanto a la evaluación de los pares de esqueletos tenemos que llegar a considerar W pares sin dejar de evaluar ninguno de los componentes de E y S, para esto se crea una ventana de comparación de longitud D, siendo esta la

diferencia de longitudes de E y S $D = |N - M|$. Los pares se evalúan de forma tal que cada esqueleto de la secuencia de menor longitud es comparado con la ventana definida y se toma el mayor valor (ecuación.12, Figura 15).

$$S_i = \begin{cases} \max\{G(e_i, s_{i+0}), G(e_s, s_{i+1}), \dots, G(e_i, s_{i+d})\} \forall\, d \in [0, D]\ si\ N < M \\ \max\{G(e_{i+0}, s_i), G(e_{i+1}, s_i), \dots, G(e_{i+d}, s_i)\} \forall\, d \in [0, D]\ en\ otro\ caso \end{cases} \quad (12)$$

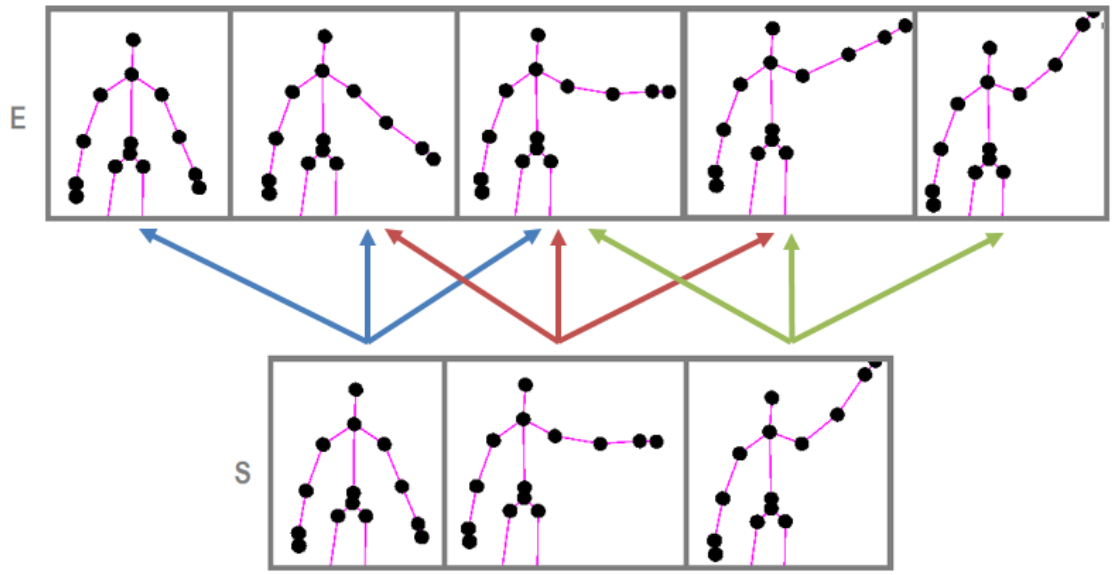

Figura 15 Comparación entre dos secuencias de distinto tamaño

De esta forma se garantiza que se evalúen todos los esqueletos de ambas secuencias y se queden para promediar solo los de mejor resultado en cuanto al grado de similitud (ecuación.13).

$$ATV(E, S) = \frac{\sum_{i=1}^{W} S_i}{W} \quad (13)$$

- **Alineamiento temporal dinámico (DTW)**

El objetivo del algoritmo DTW es encontrar el alineamiento entre las secuencias de esqueletos E y S con el mínimo coste. Un camino de alineamiento (Figura 16) es una secuencia $p = (p_1, p_2, p_3, \dots, p_n)/p_i = (n_i, m_i) \in [1, N] \times [1, M] \forall i \in [1, L]$ que cumple las siguientes condiciones:

1. De frontera: $p_0 = (0,0) \wedge\ p_L = (n, m)$
2. De monotonicidad: $n_1 \leq n_2 \leq \dots \leq n_L\ \wedge\ m_1 \leq m_2 \leq \dots \leq m_L$
3. De desplazamiento: $p_{l+1} - p_l \in \{(1,0), (0,1), (1,1)\}\ \forall\ l \in [1, L-1]$

La primera condición impone que el camino inicie en la primera posición de ambas secuencias y finalice en la última (Figura 17). En términos de alineamiento esto se traduce a que los primeros elementos deben ser alineados como el primer punto del camino, y los últimos elementos de las 2 secuencias deben ser alineados como el último punto del camino.

La condición de monotonicidad realmente es una consecuencia de la condición de desplazamiento, ya que al avanzar de n_1 a n_2 o de m_1 a m_2 sólo puede variar 0 o 1 (ecuación.14).

$$p_2 = (n_2, m_2) = \begin{cases} (n_1 + 1, m_1) \\ (n_1, m_1 + 1) \\ (n_1 + 1, m_1 + 1) \end{cases} \Rightarrow \begin{cases} n_1 \leq n_2 \leq \cdots \leq n_L \\ m_1 \leq m_2 \leq \cdots \leq m_L \end{cases} \quad (14)$$

La condición de desplazamiento establece que ningún elemento de las 2 secuencias puede ser omitido o repetido (Figura 18).

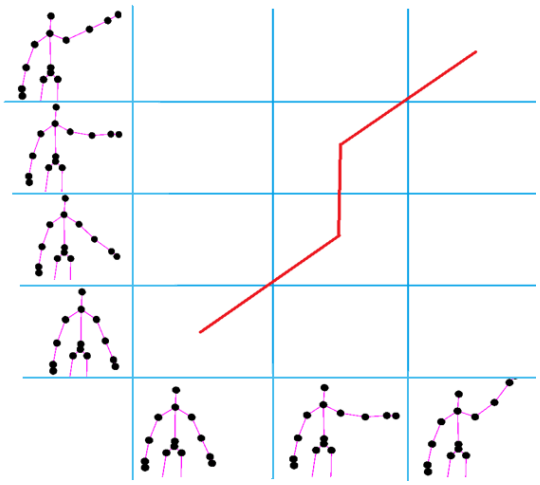

Figura 16 Camino de alineamiento

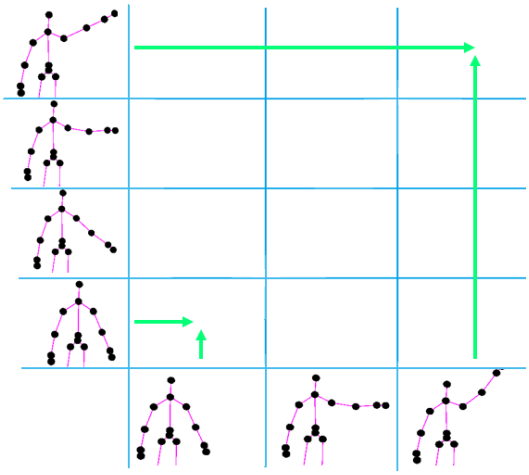

Figura 17 Condición de frontera

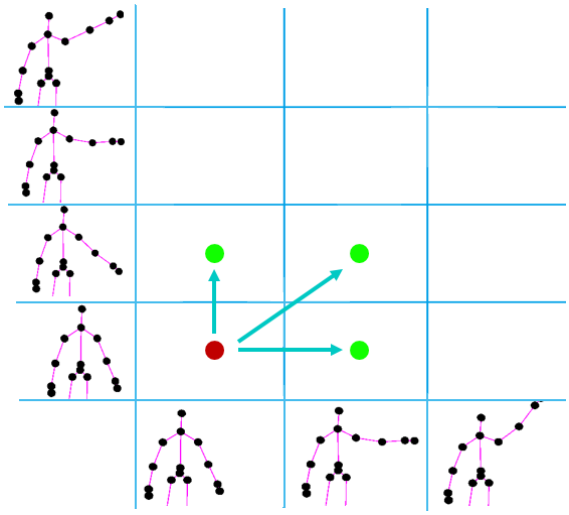

Figura 18 Condición de desplazamiento

Para obtener el camino de alineamiento, como primer paso, se calcula la matriz de distancias entre esqueletos o de costos $C(n,m) = d(x_n, y_m)\ \forall\ C \in R^{N\times M}, n \in [0,N],\ m \in [0,M]$. A partir de aquí se calcula la matriz de distancia acumulada D cumpliendo con sus restricciones (ecuación.15).

$$\begin{cases} D(n,1) = \sum_{k=1}^{n} C(e_k, s_1)\ \forall n \in [1{:}N] \\ D(1,m) = \sum_{k=1}^{m} C(e_1, s_k)\ \forall m \in [1{:}M] \\ D(n,m) = min\{D(n-1,m), D(n,m-1), D(n-1,m-1)\} + C(e_n, s_m) \\ \forall 1 < n < N \wedge 1 < m < M \end{cases} \quad (15)$$

El algoritmo para calcular el camino de alineamiento óptimo sigue el procedimiento:

Entrada: Matriz de distancias acumuladas.

Salida: Camino de alineamiento óptimo $p*$.

Procedimiento: El camino óptimo $p*(p_1,\dots,p_L)$es calculado en orden inverso empezando por $p_L=(N,M)$ (ecuación.16).

$$p_{L-1}=\begin{cases}(n-1,m)sim=1\\(n,m-1)sin=1\\argmin\{D(n-1,m-1),D(n,m-1),D(n-1,m)\}enelresto\end{cases} \quad (16)$$

Se define el camino de alineamiento óptimo entre dos secuencias de esqueletos E y S como el camino de alineamiento $p*$ que recorre la mínima distancia de entre todos los caminos de alineamiento posibles (ecuación.17).

$$d_{p*}(E,S)=min\{d_p(E,S)\}\ p\ es\ un\ punto\ (n,m)\ del\ camino\ de\ alineamiento$$

Luego de encontrar $p*$, dado el procedimiento del algoritmo, se tiene entonces: $\mathrm{DTW(E,S)}=C_{p*}(E,\mathrm{S})=p*(0{,}0)$, siendo este el valor del costo de alineamiento entre las secuencias de esqueletos E y S , este valor se toma como grado de similitud entre las mismas ya que el costo de alinearlas decrece mientas más parecidas sean y aumenta en caso contrario, cualidad que lo convierte en un excelente exponente de la similitud entre las secuencias de esqueletos E y S.

2.3.1 Criterio de evaluación

Ya comparadas las secuencias de esqueletos se tiene como resultado el grado de similitud de las mismas y se tiene que transformar a un criterio de evaluación (bien, regular, mal). Mediante reglas duras que establecen un rango se convierte el grado de similitud a una calificación. Las reglas deben de ser definidas como dato experto, o sea es un especialista el encargado de generarlas.

De esta manera queda conformada la propuesta de solución y para llegar al desarrollo de la misma corresponde continuar con las restantes fases de la metodología XP.

2.4 Exploración

En la metodología XP, se define como la etapa en la que el cliente plantea sus necesidades a través de historias de usuario. Simultáneamente, el equipo de

desarrollo se familiariza con el problema y estima su desarrollo a partir de dichas historias, además de estudiar las herramientas, tecnologías o prácticas que son candidatas para su utilización en el proyecto. Se prueban las tecnologías y se aprueban las bases arquitectónicas del proyecto. Es una fase corta, aunque una posible dilatación depende del alcance del proyecto y las capacidades del equipo de desarrollo (32).

2.4.1 Historia de usuario

Una historia de usuario (HU) es una escritura medianamente informal, en el lenguaje del cliente, que establece un requisito que debe satisfacer el sistema, una historia tiene la particularidad de que debe ser lo suficientemente sencilla como para que el cliente pueda comprenderla en su totalidad, y estar lo suficientemente completa para que el desarrollador sepa lo que tiene que implementar y poder llevar un seguimiento sobre el desarrollo de esta funcionalidad. Al escribir una historia de usuario, el equipo de desarrollo puede determinar que no es lo suficiente sencilla como para implementarla como funcionalidad atómica; en este caso, puede dividirse en dos o más historias, siempre con la aprobación del cliente (33).

En el presente trabajo se obtuvieron un total de 21 HU, las cuales se realizarán en 4 iteraciones. Teniendo en cuenta la dimensión del sistema propuesto, se realiza una selección de las HU de mayor importancia. A continuación, se describen 3 de estas.

Tabla 1. Conexión con el Kinect.

Historia de usuario	
Número: 1	**Nombre de historia de usuario:** establecer la conexión con el sensor Kinect.
Usuario: usuario	**Iteración asignada:** 1
Prioridad en negocio: alta	**Puntos estimados:** 2
Riesgo en desarrollo: alto	**Puntos reales:** 2

Descripción: El sistema establece una conexión con el sensor Kinect para el uso de sus funcionalidades en cuanto a la detección y seguimiento de los puntos que representan el cuerpo de una persona.
Observaciones: ninguna.

Tabla 2. Pre-procesamiento de los datos.

Historia de usuario	
Número: 10	**Nombre de historia de usuario:** pre procesamiento de los datos.
Usuario: usuario	**Iteración asignada:** 2
Prioridad en negocio: alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: el sistema debe obtener y transformar los datos de la captura del movimiento en la información necesaria para la posterior evaluación de los ejercicios.	
Observaciones: realizar operaciones como la normalización.	

Tabla 3. Aplicar un algoritmo evaluador.

Historia de usuario	
Número: 19	**Nombre de historia de usuario:** aplicar un algoritmo evaluador.
Usuario: usuario	**Iteración asignada:** 3
Prioridad en negocio: alta	**Puntos estimados:** 3
Riesgo en desarrollo: alto	**Puntos reales:** 3

Descripción: el sistema debe utilizar un algoritmo capaz de comparar los datos obtenidos en la correcta ejecución de un ejercicio con los obtenidos en la captura.
Observaciones: ninguna.

2.5 Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. La fase de planificación toma un par de días. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución (34).

La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales se ejecutan al final de cada iteración. Al final de la última iteración el sistema está listo para producción.

2.5.1 Requisitos funcionales (RF)

Definen funcionalidades o características con las que el sistema deberá contar. A continuación, se muestra un listado con los requisitos funcionales identificados para el sistema a realizar.

De conexión:

1- Establecer la comunicación con el sensor Kinect.

2- Detectar los puntos en el cuerpo de una persona.

3- Seguir los puntos del cuerpo de una persona en tiempo real.

4- Obtención la secuencia de esqueletos del movimiento.

5- Generar la matriz perteneciente a los datos del movimiento.

De captura de datos:

1- Grabar sesiones de video, asociando puntos del cuerpo a cada fotograma del video.

2- Mostrar puntos capturados del cuerpo.

3- Identificar inicio y final del movimiento.

4- Seleccionar el conjunto de puntos clave del esqueleto.

5- Preprocesamiento.

6- Clasificar el movimiento capturado de forma manual.

7- Guardar modelo de comparación generado.

De evaluación:

1- Cargar fichero del modelo perteneciente al movimiento a evaluar.

2- Configurar parámetros de actualización del Kinect. (Mediante la conexión)

3- Establecer la comunicación con el sensor Kinect. (Mediante la conexión)

4- Realizar la captura y seguimiento del movimiento. (Mediante la captura de datos)

5- Pre procesamiento de los datos. (Mediante la captura de datos)

6- Configurar parámetros necesarios para el algoritmo evaluador.

7- Aplicar el algoritmo evaluador.

8- Transformación del resultado obtenido en un criterio de evaluación (bien, regular, mal).

9- Guardar los resultados de la evaluación.

2.5.2 Requerimientos no funcionales (RNF)

Los requerimientos no funcionales son propiedades o cualidades que el producto debe cumplir. Se deben asumir como propiedades o argumentos que hacen que el producto sea atractivo y aceptado. Aunque no definen el éxito general del producto, influyen en la evaluación del cliente. Teniendo en cuenta la plataforma

donde operará el módulo de tendencias, se establecen los siguientes requerimientos no funcionales:

Seguridad:

El sistema no requiere de altos niveles de seguridad ya que no se trabaja con información acerca de los usuarios.

Interfaces externas:

La debe permitir fácil acceso a todas las funciones elementales y de configuración de las técnicas empleadas, manteniendo un ambiente que permitan al usuario centrarse en la ejecución del ejercicio.

Rendimiento

El sistema utilizará eficientemente los recursos que lo constituyen, tanto de hardware como de software; asegurándose pequeños tiempos de respuesta a las peticiones realizadas por los usuarios.

Hardware:

Propiedades de la PC: procesador Intel/AMD a 2.6 GHz o superior, memoria RAM: 2GB o superior.

Sensor Kinect.

2.5.3 Plan de iteraciones

El plan de iteraciones describe las fases de implementación en las que se divide y las historias de usuario que las componen. Para garantizar un desarrollo iterativo e incremental y asegurar una organización de las tareas de desarrollo, se realiza este plan rector que se asumen como documento de planificación del proyecto.

En este caso, el desarrollo se dividió en tres fases, dado el grado de prioridad de las tareas, su complejidad y tiempo necesario para realizarlas.

Tabla 4. Plan de iteraciones.

Iteración	Descripción	Orden de la HU a implementar	Duración total
1	Se desarrollarán las historias que dan entrada al proceso de	1 – 5	7.0

	establecer la conexión con el sensor Kinect, así como la detección y seguimiento de los puntos en el cuerpo de una persona.		
2	Contiene las historias con relación a las funcionalidades de captura, entrada, representación y almacenamiento de datos.	6 – 12	11.0
3	Serán objetivo las historias que se centran en la parte de evaluación de la ejecución de los ejercicios.	13 – 21	12.0

2.5.4 Plan de entrega

El plan de entrega es el calendario de salida de terminación de cada fase, por lo que debe generar artefactos o segmentos del sistema entregables al sistema. En el caso del módulo de tendencias abordado en la presente investigación, se tiene el siguiente plan de entrega:

Tabla 5. Plan de entrega.

Iteración	1	2	3
Cantidad de HU	5	7	9
Fecha de Entrega	25-01-2016	30-03-2016	15-04-2016

2.5.5 Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. Al final de la última iteración el sistema estará listo para entrar en producción. (35)

Diseño del sistema:

La metodología XP no requiere de la representación del sistema mediante diagramas de clases utilizando UML para diseñar las aplicaciones, en este caso utiliza otras técnicas como las tarjetas CRC (Cargo o clase, Responsabilidad, Colaboración).

2.5.6 Tarjetas CRC

Son una herramienta de reflexión en el diseño de software orientado a objetos, son caracterizadas por ser sencillas, fáciles de entender y permiten al equipo de trabajo en su totalidad participar en la tarea del diseño.

Se compone de tal forma que el nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. Una clase describe cualquier objeto o evento, mediante los atributos y los métodos, las responsabilidades son las tareas que realizan o los métodos correspondientes a la clase y los colaboradores son las demás clases con las que trabaja conjuntamente para cumplir con sus responsabilidades. A continuación, se describen 3 de estas.

Tabla 6. Tarjeta CRC Conexión.

Conexión	
Responsabilidades	Colaboraciones
Conectar	Capturar
Identificar	Evaluar
Generar Matriz	

Tabla 7. Tarjeta CRC Captura.

Capturar	
Responsabilidades	Colaboraciones
Grabar	Conexión
Mostrar esqueleto	Evaluar
Pre procesar	
Guardar	

Tabla 8. Tarjeta CRC Evaluar.

Evaluar	
Responsabilidades	Colaboraciones
Cargar	Conexión
Recapturar	Capturar
Evaluar	

Guardar	

2.6 Arquitectura del software

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Es donde se define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos.

2.6.1 Modelo de capas

Es el modelo de una arquitectura que organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios. Esta arquitectura también soporta bien los cambios, en la medida en que su interfaz no se modifique, una capa puede remplazarse por otra equivalente. Además, cuando las interfaces de la capa cambian o se añaden nuevas facilidades a una de las capas, solamente se ve afectada la capa subyacente.

La arquitectura de la solución (Figura 19) está compuesta por 3 capas dentro de las que se encuentran: la capa de interfaz que contiene las diferentes vistas del sistema, así como los métodos para mostrar la información de las demás capas y el proceso que se realiza. Le sigue la capa de procesos en la cual se llevan a cabo las principales funciones del sistema como son la evaluación y captura de los datos, además contiene las clases encargadas de las mismas. Por último, en la capa de datos se generan los archivos donde se almacenan los datos que posteriormente dicha capa se encarga de acceder.

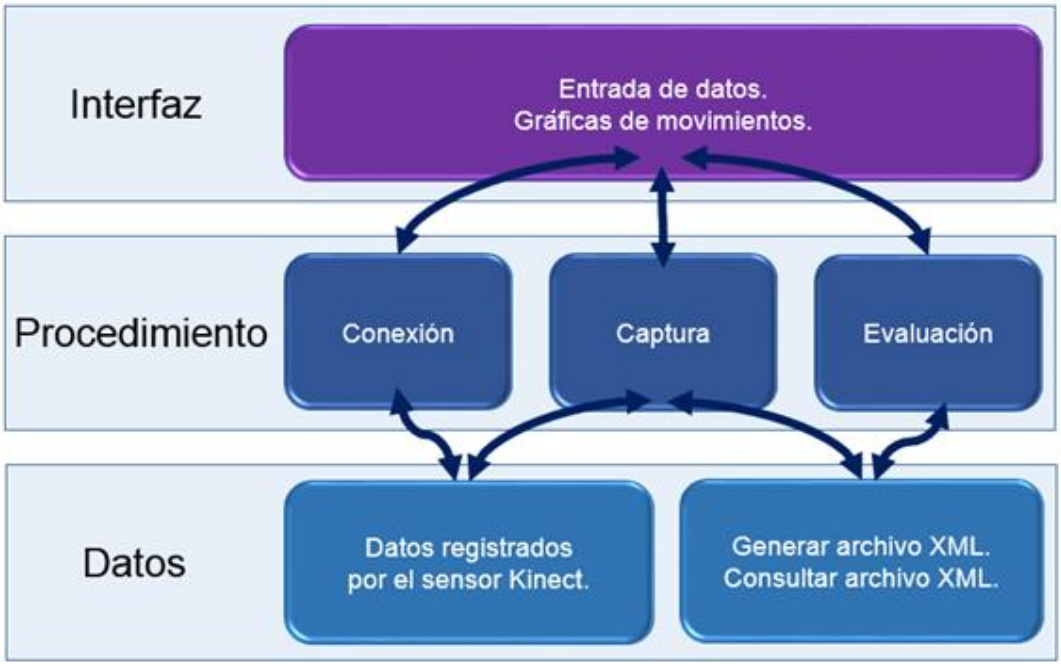

Figura 19 Arquitectura de capas del sistema

2.6.2 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Estas soluciones no son un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema, la cual puede ser utilizada en diversas situaciones. Los mismos pretenden: proporcionar catálogos de elementos reusables en el diseño de sistemas software, evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente y facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Experto

Este patrón es el principio básico de asignación de responsabilidades. Nos indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Ejemplo son las tres principales clases: Capturar, Evaluar y Conexión, en las cuales se implementan sus métodos en relación a las tienen distintas responsabilidades que cumplen por separado.

Patrón de instancia única (*singleton*)

El *singleton* o patrón de instancia única es una solución básica de arquitectura de software. Como patrón de creación a nivel de objeto proporciona una forma de agrupar el código de una unidad lógica, a la que se puede acceder haciendo uso de una única variable, garantizando la unicidad de esta instancia y ofreciendo un punto de acceso global a esta variable.

Esto se ve reflejado a la hora de obtener los datos de un ejercicio, pues solo existe una instancia de los mismos accesible por las distintas clases en cada parte del proceso.

Alta Cohesión

Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Además, las tareas corresponden a una secuencia de pasos propia del proceso y todas afectan a los mismos datos. Esto es en sí como trabaja el sistema puesto que se realiza primeramente el paso de conexión, luego la captura de datos y por último la evaluación, de tal forma que cada clase encargada de una etapa modifica o consulta los mismos datos.

Bajo Acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Desde el inicio del proceso se tiene en cuenta esto pues se crean tres clases principales y su relación está dada en el aporte de datos de una a otra según la etapa del proceso.

2.7 Sumario

En este capítulo se desarrollan las primeras fases de la metodología XP. Como resultados se obtienen, para la implementación de la solución, las bases necesarias:

- Queda conformada la propuesta de solución.
- Se especifican las historias de usuario que representan las funcionalidades del sistema.
- Es planificado el transcurso del desarrollo del sistema.

- Es definida una arquitectura basada en el modelo de capas, los componentes, las tareas que llevan a cabo y la comunicación entre ellos.

Capítulo 3. Implementación y pruebas

En este capítulo se abordan los temas de la implementación del sistema, basados en todo el trabajo acumulado a lo largo de los capítulos anteriores. Además, se hará un análisis de los resultados obtenidos en cuanto a aplicación de los algoritmos ATV y DTW, las funcionalidades del sistema y cumplimiento de los objetivos propuestos.

3.1 Estándar de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la eficacia del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener (36).

A continuación, se enumeran las normas que se siguieron en la codificación, junto con ejemplos, así como las justificaciones de los mismos.

Indentación:

Todo el código desarrollado tendrá una indentación de 8 espacios (Figura 20).

```
public double[] StepOne(KMSkeletonData _a, KMSkeletonData _b)
{
    double[] result = new double[cantJoints];

    for(int i=0; i<cantJoints; i++)
        result[i] = EuclideanDistance(_a.joints[i], _b.joints[i]);

    return result;
}
```

Figura 20. Ejemplo de indentación.

La razón principal de tener una buena indentación es que solo necesitas un primer vistazo para ver la estructura general del código. Además, si el código comienza a extenderse a la derecha más de lo debido probablemente significa que debe rescribirse el código de una manera más legible.

Nombrado de los símbolos:

Es interesante seguir un buen nombrado para las clases, variables y funciones para así evitar usar nombres establecidos en otras partes, además de facilitar la lectura del código.

Clases (Figura 21)

```
public class Evaluation
```

Figura 21. Ejemplo de la declaración de una clase.

Las clases definidas van escritas con la primera inicial en mayúscula.

Variables (Figura 22)

```
public Distances distance;
public int cantFrames;
public int cantJoints;
public double evaluation;
public double[,] distanceXNodes;
public double[] percentXFrame;
```

Figura 22. Ejemplo dela declaración de variables.

Las variables definidas van escritas con la primera inicial de cada palabra que la conforma en mayúscula, excepto la primera y se juntan todas las palabras. Los mismos tienen que ser descriptivos y tener una relación con la función que desempeñan.

Funciones (Figura 23)

```
public double Evaluate(KMMotionData _model, KMMotionData _capture)
{
    switch(typeEvaluation)
    {
        case EvaluationType.ATV:
            return EvaluateWithATV(_model, _capture);

        case EvaluationType.DTW:
            return EvaluateWithDTW(_model, _capture);

        default:
            return 0;
    }

}
```

Figura 23. Ejemplo de la declaración de una función.

Las funciones definidas van escritas con la primera inicial de cada palabra que la conforma en mayúscula y el resto en minúscula.

3.2 Desarrollo de las iteraciones

En la fase de planificación se detallaron las historias de usuarios correspondientes a cada una de las iteraciones para desarrollar el sistema, teniendo presente las necesidades requeridas por el cliente. Durante el transcurso de cada iteración se realizó una revisión del plan de iteraciones. Como parte de este plan se desglosaron las historias definidas en tareas de programación o ingeniería, asignándole a un equipo de desarrollo o a una persona la responsabilidad de su implementación. Estas tareas son para el uso estricto del programador.

A continuación, se perfila con mayor detalle las tareas de desarrollo que se realizaron en cada una de las iteraciones:

3.2.1 Iteración 1

Esta iteración presenta como objetivo dar cumplimiento a las HU que dan entrada al proceso de establecer la conexión con el Kinect, así como la detección y seguimiento de los puntos en el cuerpo de una persona. Al concluir dicha iteración se cuenta con todas las funcionalidades descritas en las HU 1 a la 5.

Tabla 9. Tiempo de implementación de la primera iteración.

Historia de usuario	Tiempo de implementación (semanas)	
	Estimación	Real
Conexión con el Kinect	2	2
Detectar los puntos en el cuerpo	1	1
Seguir los puntos del cuerpo	1	1
Obtención la secuencia de esqueletos	2	2
Generar los datos del movimiento	1	1
Total	7	7

Tabla 10. Tarea de la historia de usuario 1.

Tarea de ingeniería	
No. de la tarea: 1	**No. de la HU:** 1
Nombre de la tarea: Creación de una clase prototipo para la conexión.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/01/2016	**Fecha fin:** 15/01/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, crear un prototipo de la clase Conexión que permita el acceso a las propiedades del Kinect.	

Tabla 11. Tarea de la historia de usuario 4.

Tarea de ingeniería	
No. de la tarea: 1	**No. de la HU:** 4
Nombre de la tarea: Implementar el medio para la obtención de la secuencia de esqueletos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/01/2016	**Fecha fin:** 22/01/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, generar la secuencia de esqueletos del movimiento que se está capturando.	

3.2.2 Iteración 2

Esta iteración tuvo como finalidad dar cumplimiento a las HU 6-10, las cuales aluden a la captura de los datos del movimiento, así como generar un modelo que comprenda la secuencia de esqueletos, puntos claves del movimiento, id del mismo, etc., o sea lo principal para la posterior comparación.

Tabla 12. Tiempo de implementación de la segunda iteración.

Historia de usuario	Tiempo de implementación (semanas)	
	Estimación	Real
Grabar sesiones de video	2	2
Mostrar puntos capturados del cuerpo	2	2
Identificar inicio y final del movimiento	3	3
Seleccionar el conjunto de puntos clave del esqueleto	1	1
Pre procesamiento de los datos	1	1
Guardar el movimiento capturado	2	2
Total	11	11

Tabla 13. Tarea de la historia de usuario 7.

Tarea de ingeniería	
No. de la tarea: 1	**No. de la HU:** 7

Nombre de la tarea: Diseñar una interfaz que muestre los datos capturados.	
Tipo de tarea: Diseño	**Puntos estimados:** 0.5
Fecha inicio: 25/02/2016	**Fecha fin:** 27/02/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, diseñar un prototipo de la interfaz que visualiza los puntos del esqueleto capturados por el Kinect en tiempo real.	

Tabla 14. Tarea de la historia de usuario 9.

Tarea de ingeniería	
No. de la tarea: 2	**No. de la HU:** 8
Nombre de la tarea: Implementación del método para identificar inicio y final del movimiento.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/03/2016	**Fecha fin:** 25/03/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, dotar a la clases Evaluación y Captura de identificar inicio y final del movimiento.	

3.2.3 Iteración 3

Esta es la última iteración del mecanismo propuesto, en la cual se da cumplimiento a las HU 11-17. Estas cumplen con las funcionalidades de cargar los modelos generados y compararlos con los movimientos. Para ello se aplica un algoritmo que determina cuán similares son las secuencias.

Tabla 15. Tiempo de implementación de la tercera iteración.

Historia de usuario	Tiempo de implementación (semanas)	
	Estimación	Real
Cargar los datos del modelo a evaluar	2	2
Configurar parámetros necesarios para el algoritmo evaluador	1	1
Realizar la captura del movimiento a evaluar	2	2
Aplicar el algoritmo evaluador	3	3
Transformación del resultado obtenido en una evaluación comprensible	2	2
Guardar los resultados de la evaluación	2	2
Total	12	12

Tabla 16. Tarea de la historia de usuario 14.

Tarea de ingeniería	
No. de la tarea: 1	**No. de la HU:** 14
Nombre de la tarea: Implementar la forma de captura del movimiento a evaluar.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 15/03/2016	**Fecha fin:** 18/03/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, obtener los datos mediante la clase captura del movimiento a evaluar.	

Tabla 17. Tarea de la historia de usuario 15.

Tarea de ingeniería	
No. de la tarea: 3	**No. de la HU: 15**
Nombre de la tarea: Aplicar los algoritmos ATV y DTW a un conjunto de datos generados.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20/03/2016	**Fecha fin:** 27/03/2016
Programador responsable: Roberto Ross Aguirre	
Descripción: Tiene como fin, comprobar el funcionamiento de los algoritmo con datos estáticos pasando un conjunto de entradas definidas y comparando las salidas con las que se definieron.	

3.3 Pruebas

Como define la metodología XP, uno de sus pilares es el proceso de pruebas. XP anima a probar tanto como sea posible. Esto permitió aumentar la calidad del sistema reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permitió aumentar la seguridad al evitar efectos colaterales no deseados realizando modificaciones. Esta metodología divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de cada iteración se consiguió la funcionalidad requerida por el cliente (37).

3.3.1 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de

requisitos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades deben poner el mayor esfuerzo y atención (37).

A partir de las historias de usuario definidas en la etapa de exploración y planeación, se realizaron los casos de prueba. A continuación, se muestran tres de estos:

Tabla 18. Caso de prueba de aceptación 7.

Caso de prueba de aceptación	
Número: 7	**Historia de usuario:** 7
Nombre: mostrar puntos capturados del cuerpo.	
Descripción: esta prueba se encarga de validar la HU “Mostrar puntos capturados del cuerpo”, se debe graficar la secuencia de esqueletos generada por un movimiento.	
Condiciones de ejecución: Se tiene que establecer conexión con el sensor Kinect o cargar un movimiento grabado.	
Entradas/Pasos de ejecución: 1. Conectar Kinect o cargar un movimiento grabado.	
Resultado esperado: el sistema debe graficar en pantalla los puntos del cuerpo en la correcta posición conformando el esqueleto y las transiciones de los mismos.	
Resultado obtenido: se grafica correctamente.	
Evaluación de la prueba: satisfactoria.	

Tabla 19. Caso de prueba de aceptación 12.

Caso de prueba de aceptación	
Número: 12	**Historia de usuario:** 12
Nombre: generar modelo de comparación.	

Descripción: esta prueba se encarga de validar la HU “Guardar la información del movimiento capturado”, se debe generar un archivo XML con los datos del movimiento capturado para su utilización como modelo.
Condiciones de ejecución: tiene que grabarse el movimiento con antelación.
Entradas/Pasos de ejecución: 1. Grabar el movimiento deseado. 2. Escribir el nombre que lo identifica. 3. Hacer clic en el botón Guardar.
Resultado esperado: el sistema debe crear un archivo XML con la información referente al movimiento que fue capturado (nombre, esqueleto inicial, secuencia de esqueletos).
Resultado obtenido: los datos del movimiento capturado se almacenaron correctamente en el archivo.
Evaluación de la prueba: satisfactoria.

Tabla 20. Caso de prueba de aceptación 17

Caso de prueba de aceptación	
Número: 17	**Historia de usuario:** 19
Nombre: evaluar.	
Descripción: esta prueba se encarga de validar la HU “Aplicar algoritmo evaluador”, establece la comparación entre una nueva captura y el modelo previamente cargado.	
Condiciones de ejecución: cargar un modelo para comparar y el sensor Kinect debe estar conectado.	
Entradas/Pasos de ejecución: 1. Conectar el sensor Kinect. 2. Cargar un modelo. 3. Elegir el algoritmo que se desea emplear en la evaluación.	

4. Configurar el algoritmo. 5. Hacer clic en el botón Evaluar. 6. Realizar una nueva captura. 7. Hacer clic en el botón Resultados.
Resultado esperado: el sistema debe mostrar los resultados detallados de la comparación realizada.
Resultado obtenido: los resultados se muestran correctamente.
Evaluación de la prueba: satisfactoria.

Resultados

De un total de 20 casos de pruebas de aceptación, 3 de ellos resultaron no satisfactorios, lo cual representa el 15% del total de casos de pruebas realizadas. Mientras que los 26 restantes resultaron satisfactorios para un 85%. Los errores detectados por los casos de pruebas no satisfactorios fueron mitigados después de 3 iteraciones de prueba.

3.3.2 Pruebas unitarias

El objetivo de las pruebas unitarias está enfocado en probar los elementos más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca (37).

Pruebas de caja blanca

Se centran en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Este tipo de prueba será aplicada a la estructura procedimental (código fuente) de las funcionalidades que implementa cada historia de usuario, a través de la técnica del camino básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye

el Grafo de Flujo asociado y se calcula su complejidad cíclica. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad cíclica del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Al sistema desarrollado se le aplicó un grupo de 5 pruebas de caja blanca, a continuación, se analizan y enumeran las sentencias de código del método (Figura 24).

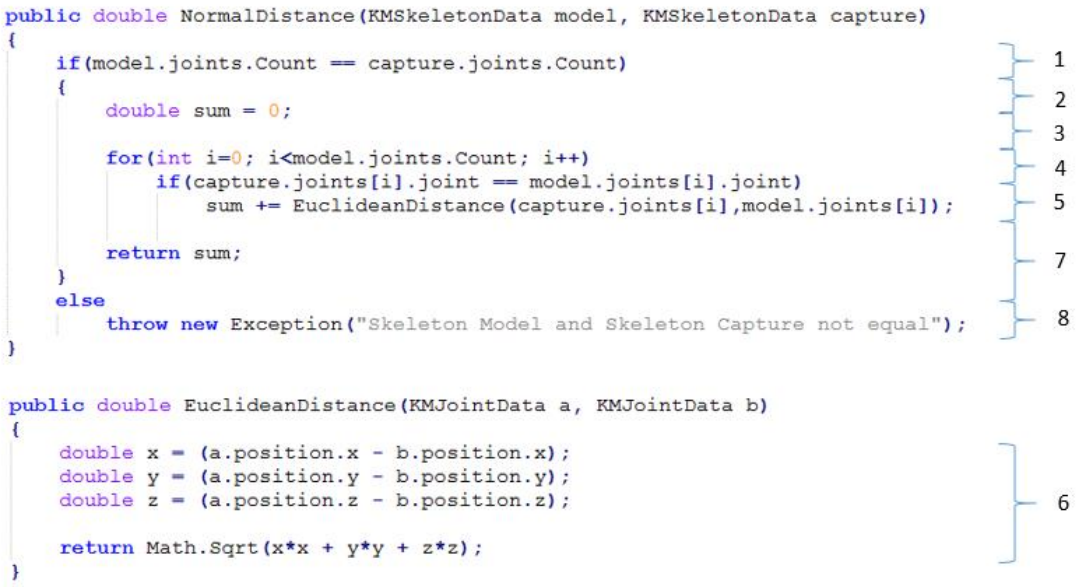

Figura 24. Numeración del código.

Luego, es necesario representar el grafo de flujo (Figura 25) asociado al código antes presentado a través de nodos, aristas y regiones.

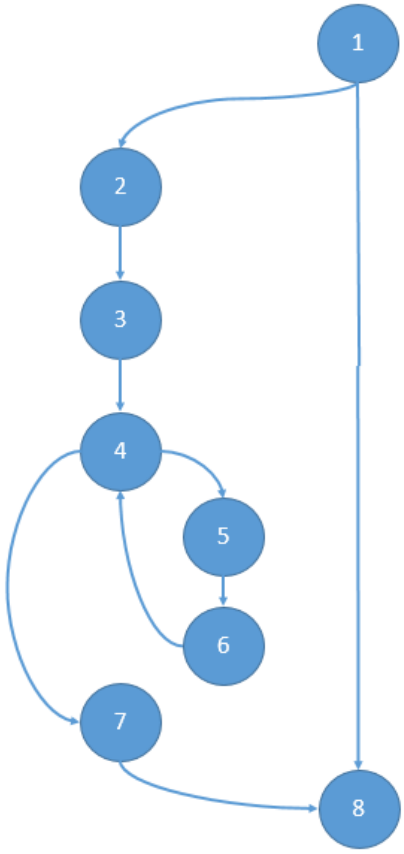

Figura 25. Grafo de flujo perteneciente al método "NormalDistance".

Luego de construido el grafo de flujo asociado al procedimiento anterior, se determina la complejidad cíclica (ecuaciones. I, II, III), la cual es una métrica de software muy útil, pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad cíclica define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar.

Complejidad Cíclica

$$V(g) = (a - n) + 2(I) \qquad V(g) = (p + 1)(II) \qquad V(g) = r(III)$$

$V(g)$ - Valor de la complejidad cíclica.

a - Cantidad total de aristas.

n - Cantidad total de nodos

p - Cantidad total de nodos predicados (de los cuales parten dos o más aristas)

r - Cantidad total de regiones, se incluye el área exterior del grafo como una región más.

$$V(g) = (9 - 8) + 2 = 3(I) \qquad V(g) = (2 + 1) = 3(II) \qquad V(g) = 3(III)$$

Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado. Seguidamente, es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

- Camino básico 1: 1-8.
- Camino básico 2: 1-2-3-4-5-6-4-7-8.
- Camino básico 3: 1-2-3-4-7-8.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. Para definir los casos de prueba es necesario tener en cuenta:

- **Descripción:** se describe el caso de prueba y se especifican los aspectos fundamentales de los datos de entrada.
- **Condición de ejecución:** se verifica que cada parámetro cumpla las condiciones de ejecución.
- **Entrada:** se muestran los parámetros de entrada del procedimiento.
- **Resultados esperados:** Se especifica el resultado que debe arrojar el procedimiento.

Tabla 21. Caso de prueba camino básico 1.

Caso de prueba camino básico 1
Descripción: el método Distancia-Normal calcula la distancia entre dos esqueletos.
Condiciones: definidas las entradas. En este caso los esqueletos modelo y captura.
Entrada: esqueleto {cantidad de puntos, arreglo de puntos} modelo {20, *array* {*joints*}} captura {17, *array* {*joints*}}
Resultado Esperado: devolver la excepción modelo y captura distintos.
Resultado: satisfactorio.

Tabla 22. Caso de prueba camino básico 2.

Caso de prueba camino básico 2
Descripción: El método Distancia-Normal calcula la distancia entre dos esqueletos.
Condiciones: definidas las entradas. En este caso los esqueletos modelo y captura.
Entrada: esqueleto {cantidad de puntos, arreglo de puntos} modelo {20, *array* {*joints*}} captura {20, *array* {*joints*}}
Resultado esperado: devolver la distancia entre modelo y captura.
Resultado: satisfactorio.

Resultados

Se probaron, mediante casos de pruebas de caja blanca o Pruebas Unitarias (PU) como también se le conocen, las funcionalidades más críticas del sistema. Los casos de PU aplicados obtuvieron resultados satisfactorios, representando el 100% de los realizados.

3.3.3 Análisis y resultado de los algoritmos

En el sistema existen parámetros que influyen en la evaluación de la ejecución de los ejercicios. Con el objetivo de analizar con detalle esta influencia, se han realizado una serie de evaluaciones con varios perfiles de configuración. A continuación, analizaremos los resultados de dos tipos de evaluaciones diferentes:

1. Evaluación de 8 ejercicios con el cuerpo de frente al Kinect.
2. Evaluación de 8 ejercicios con el cuerpo a distintas distancias del Kinect.

Cada evaluación ha sido configurada con tres perfiles:

1. Sin pre-procesamiento de datos.
2. Con pre-procesamiento de datos.
3. Personas de distintas estaturas.

Los movimientos que se evalúan comprenden solo la parte superior del cuerpo pues los modelos de comparación que se generaron sólo incluyen movimiento de brazos y manos.

Las reglas para el otorgamiento del criterio de evaluación que se utilizan en estas pruebas son definidas, en este caso, como reglas de pruebas ya que solo se usan para medir como responde el sistema. Para resultados más fiables estas reglas deben ser creadas por un experto. Para el resultado del algoritmo ATV se tienen las reglas:

$$\begin{cases} Bien & ATV(E,S) > 80 \\ Regular & ATV(E,S) > 50 \\ Mal & en\ otro\ caso \end{cases}$$

Para el resultado del algoritmo DTW se tienen las reglas:

$$\begin{cases} Bien & DTW(E,S) < 30 \\ Regular & DTW(E,S) < 50 \\ Mal & en\ otro\ caso \end{cases}$$

A continuación, se presentan algunos resultados obtenidos y las conclusiones a las que se arriban a partir de los mismos.

Tabla 23. Evaluación de 8 ejercicios a distintas distancias de Kinect sin preprocesamiento.

Movimiento	Resultado ATV	Calificación ATV	Fallo	Resultado DTW	Calificación DTW	Fallo
Ambas manos subir y bajar.	5.85	Mal	si	123.91	Mal	si
Ambas manos subir y bajar de frente.	10.23	Mal	si	113.95	Mal	si
Mano derecha subir y bajar.	10.99	Mal	si	117.1	Mal	si
Mano derecha subir y bajar de frente.	20.75	Mal	si	71.94	Mal	si
Mano izquierda subir y bajar.	52.47	Regular	si	48.81	Regular	si
Mano izquierda subir y bajar de frente.	39.80	Mal	si	87.38	Mal	si

Mano derecha subir de lado y bajar de frente.	9.03	Mal	si	112.05	Mal	si
Mano derecha subir de lado y bajar de frente.	14.33	Mal	si	104.25	Mal	si

Como se pudo apreciar tanto el algoritmo DTW como el ATV comparan cada ejercicio de la evaluación con su modelo tal cual se grabaron y detectan todas las ejecuciones como incorrectas en el perfil sin pre-procesado. Esto se debe a que las distancias de los movimientos detectados se ven afectadas por la ausencia del preprocesamiento y conlleva al error en los resultados. Si nos situamos en la misma posición exactamente en el modelo y en la evaluación, la distancia entre el gesto entrenado y el gesto de la evaluación será muy pequeña. Y cuanto más nos alejemos, más aumentará esa distancia.

Tabla 24. Evaluación de 8 ejercicios a distintas distancias de Kinect con pre-procesamiento.

Movimiento	Resultado ATV	Calificación ATV	Fallo	Resultado DTW	Calificación DTW	Fallo
Ambas manos subir y bajar.	98.85	Bien	no	2.91	Bien	no
Ambas manos subir y bajar de frente.	99.5	Bien	no	0	Bien	no
Mano derecha subir y bajar.	95.26	Bien	no	7.1	Bien	no
Mano derecha subir y bajar de frente.	98.47	Bien	no	3.94	Bien	no
Mano izquierda subir y bajar.	98.24	Bien	no	3.81	Bien	no
Mano izquierda subir y bajar de frente.	93.84	Bien	no	5.38	Bien	no
Mano derecha subir de lado y bajar de frente.	92.03	Bien	no	5.05	Bien	no
Mano derecha subir de lado y bajar de frente.	96.63	Bien	no	4.25	Bien	si

Respecto a la anterior evaluación los algoritmos aciertan en un 100% demostrando la importancia y el efecto de realizar la misma sin el preprocesamiento de los datos. Por tanto, se propone utilizar dicha característica a la hora de evaluar para solamente enfocarse en la ejecución del movimiento sin importar la distancia y el ángulo que se tenga respecto al Kinect.

Tabla 25. Evaluación de 8 ejercicios con personas de distintas estaturas.

Movimiento	Resultado ATV	Calificación ATV	Fallo	Resultado DTW	Calificación DTW	Fallo
Ambas manos subir y bajar.	93.85	Bien	no	28.91	Bien	no
Ambas manos subir y bajar de frente.	75.5	Regular	si	85.45	Mal	si
Mano derecha subir y bajar.	95.83	Bien	no	7.1	Bien	no
Mano derecha subir y bajar de frente.	98.47	Mal	si	93.94	Mal	si
Mano izquierda subir y bajar.	28.24	Mal	si	43.81	Regular	si
Mano izquierda subir y bajar de frente.	93.84	Bien	no	5.38	Bien	no
Mano derecha subir de lado y bajar de frente.	85.03	Bien	no	5.05	Bien	no
Mano derecha subir de lado y bajar de frente.	96.63	Bien	no	4.25	Bien	no

En esta última muestra se aprecian fallos por parte del sistema. Esto se debe a la diferencia de estatura entre la persona que se evalúa y la que se capturó en la realización del modelo. Para una diferencia de más de 15 cm se notan alteraciones en los resultados, aumenta la distancia entre esqueletos y se llega a valorar como una ejecución incorrecta ejercicios bien ejecutados. Una forma de mitigar este error es generar los modelos de comparación con personas de estatura promedio para asegurar la menor diferencia de estatura posible.

3.4 Sumario

Este capítulo es fundamental para corroborar el buen funcionamiento del sistema mediante el desarrollo las pruebas:

- De aceptación, con las cuales se detectaron errores funcionales del sistema.
- Unitarias, para la inspección minuciosa del código del sistema.
- De análisis de los algoritmos ATV y DTW, para saber sus respuestas ante un entorno cambiante, como son la estatura del usuario y la posición que se adopte respecto al Kinect.

Conclusiones generales

A partir de los resultados de la investigación, se puede concluir que:

- El sensor Kinect ofrece amplias funcionalidades para la detección y seguimiento de los puntos del cuerpo. Sin embargo, para garantizar una mayor fiabilidad en el resultado de la comparación de las secuencias de movimiento de personas, estas deben ser transformadas al mismo espacio vectorial.

- Los parámetros de configuración utilizados por los algoritmos ATV y DTW para comparar secuencias de movimiento de personas requieren de un proceso de ajuste previo. Este depende del tipo, duración y umbral requerido para decidir cuándo un movimiento es o no correcto.

- La arquitectura del sistema desarrollado es lo suficientemente extensible como para tolerar la incorporación de un dispositivo de captura de datos de mayor precisión que redunde en la obtención de mejores resultados de clasificación. Para ello solo se requiere que los datos capturados estén representados por secuencias de coordenadas tridimensionales que representen el movimiento de una persona.

Recomendaciones

Teniendo en cuenta la experiencia adquirida y los resultados obtenidos en el proceso de investigación se recomienda:

- Implementar un método de escalado para la normalización de los datos y evitar que la diferencia de estaturas entre las personas afecte la evaluación.
- Investigar con relación a otras técnicas y las ventajas que podrían ofrecer.
- Implementar nuevas versiones con otras de las técnicas que facilitan la evaluación de movimientos.

Referencias bibliográficas

1. **Russell, Stuart J y Norvig, Peter.** *Inteligencia Artificial, un enfoque moderno. Segunda edición.* Madrid : PEARSON. Prentice Hall, 2004. 84-205-4003-X.

2. **Bello Perez, Rafael Esteban, y otros.** *Aplicaciones de la inteligencia artificial.* Guadalajara : Centro Universitario de Ciencias Econ6mico Administrativas, 2002. 970-27-0177-5.

3. **Universidad del país Vasco.** www.ehu.eus. *www.ehu.eus.* [En línea] 25 de 11 de 2009. [Citado el: 5 de 2 de 2016.] www.ehu.eus/ccwintco/uploads/d/d4/PresentacionMundoVirtual.pdf.

4. **Sergios Theodoridis, Konstantinos Koutroumbas.** *Pattern Recognition.* 2003.

5. **Xsens 3D motion tracking.** www.xsens.com. *www.xsens.com.* [En línea] Xsens 3D motion tracking, 27 de 6 de 2014. [Citado el: 20 de 2 de 2016.] https://www.xsens.com/tags/motion-capture/.

6. **Aggarwal, Jake K. y Cai, Quin.** *Human motion analysis: A review. En Nonrigid and Articulated Motion Workshop.* s.l. : Proceedings, 1997.

7. **Shulcloper, José R., Gusmán, Adolfo y Ma, J. Francisco .** *Enfoque Lógico Combinatorio al Reconocimiento de Patrones.* Mexico : Centro de Investigación en Computación, 1999.

8. **O. Duda, Richard, E. Hart, Peter y G. Stork, David.** *Pattern classification (2ª edición).* New York : Wiley, 2001. ISBN 0-471-05669-3.

9. **Bishop, Christopher M.** *Pattern Recognition and Machine Learning.* New York : Springer Science +Business Media, 2008.

10. **Ibañez, Rodrigo, y otros.** *Evaluación de técnicas de Machine Learning para el reconocimiento de gestos corporales.* Buenos Aires : s.n., 2014.

11. **Bellman, Ricard y Kalaba, Robert.** *On adaptive control processes. Automatic Control.* 1959.

12. **Gonzaléz De Dios, Alberto.** *Desarrollo y evaluación de un sistema de teleoperación de robots basado en reconocimiento de gestos usando el sensor*

Kinect. Madrid : Escuela Tecnica Superior De Ingenieros De Telecomunicación., 2014.

13. **Shulcloper, José R., Gusmán, Adolfo y Ma, J. Francisco .** *Enfoque Lógico Combinatorio al Reconocimiento de Patrones.* 1999.

14. **Rabiner, Lawrence R.** *A tutorial on hidden Markov models and selected applications in speech recognition.* 1989. 257-286.

15. **Sezgin, Tevfik Metin y Davis, Randall.** *HMM based eficient sketch recognition.* 2005.

16. **Martín Del Brío, Bonifacio y Sanz Molina, Alfredo.** *Redes Neuronales y Sistemas Difusos.* Zaragoza. : Alfaomega Ra-Ma., 2001.

17. *Artificial neural networks: approximation and learning theory.* **White, H.** 1992.

18. **Murakami, Kouichi y Taguchi, Hitomi.** *Gesture recognition using recurrent neural networks.* 2007.

19. **Osakidetza; Accenture; Microsoft.** *Proyecto TEKI. Tele-asistencia de pacientes Crónicos.* Madrid : Eusko JaurlariTza, 2012.

20. **Ibañez, Rodrigo y Fanaro, Damián.** *Herramienta para facilitar el desarrollo de aplicaciones basadas en Kinect.* Argentina : Instituto de Investigación ISISTAN, 2014.

21. **KeebaliMedia.** www.kinecthacks.com. *www.kinecthacks.com.* [En línea] KeebaliMedia, 2015. [Citado el: 25 de 1 de 2016.] http://www.kinecthacks.com/kinect-custom-gesture-recognition-program/.

22. **Unity Technologies.** www.unity.com. *www.unity.com.* [En línea] Unity Technologies, 2012. [Citado el: 25 de 1 de 2016.] http://www.unity.com/unity-for-developer.

23. **Nourie, Dana.** *Getting Started with an Integrated Development Environment (IDE).* 2005.

24. **Hejlsberg, Anders, Wiltamuth, Scott y Golde, Peter.** *C# language specification.* s.l. : Addison-Wesley Longman Publishing Co, 2003.

25. *Cuatro enfoques metodológicos para el desarrollo de Software.* **Pérez, Oiver Andrés.** 10, s.l. : Revista Inventum, 2011.

26. **KROLL, Per y KRUCHTEN, Philippe.** *The rational unified process made easy: a practitioner's guide to the RUP.* New York : Addison-Wesley Professional, 2003.

27. **Pressman, Roger S.** *Conceptos y principios del anàlisis. Ingenieria del Software. Un enfoque práctico.* Madrid : McGraw-Hill, 2001.

28. **Keeton, Marlys.** *Microsoft Solutions Framework (MSF): A Pocket Guide.* New York : Addison-Wesley Professional, 2005.

29. **Calderón, Amaro , Sarah , Dámaris y Carlos y Valverde Rebaza , Jorge.** *Metodologías Ágiles.* Trujillo : Universidad Nacional de Trujillo, 2007.

30. **Canós, José H., Letelier, Patricio y Penadés, Carmen.** *Metodologías ágiles en el desarrollo de software.* Valencia : Universidad Politécnica de Valencia, 2003.

31. **Ramos, Ricardo.** *Transformaciones lineales en 3d .* Oviedo : Universidad de Oviedo, 2010.

32. **Sommerville, Ian y Galipienso, María Isabel Alfonso.** *Ingeniería del Software. Septima edición.* Madrid : Pearson Educación, 2005. 84-7829-074-5.

33. **Cohn, Mike.** *User stories applied: For agile software development.* s.l. : Addison-Wesley Professional, 2004.

34. **Fernández Escribano, Gerardo.** *Introducción a Extreme Programming.* 2002.

35. **Calderón, Amaro , Sarah , Dámaris y Carlos y Valverde Rebaza , Jorge.** *Metodologías Ágiles.* 2007.

36. **Microsoft.** msdn.microsoft.com. *msdn.microsoft.com.* [En línea] Microsoft, 2015. [Citado el: 26 de 1 de 2016.] https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

37. **Gutiérrez, J. J. , y otros.** *Pruebas del Sistema en Programación Extrema.* Sevilla : Departamento de Lenguajes y Sistemas Informáticos, 2006.