Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Sistema de seguimiento de humanos mediante la utilización del robot Pioneer 3DX y Kinect

Autor: Alejandro Vegas García

Tutor: Moisés Martínez Muñoz

Resumen

En este trabajo se presenta el diseño de un sistema de seguimiento de humanos utilizando el robot Pioneer 3DX y el dispositivo Kinect de Microsoft. El objetivo del trabajo consiste en la implementación de un sistema que permita el reconocimiento de un usuario específico y sea capaz de seguirle en su desplazamiento. Además, durante el proceso de seguimiento, el robot será capaz de evitar obstáculos que se interpongan entre él y el usuario al que está siguiendo. Con el fin de realizar un reconocimiento del usuario más exhaustivo y fiable, se han propuesto tres alternativas basadas en visión artificial: reconocimiento por color, reconocimiento por forma y reconocimiento mixto.

Abstract

This document shows the design of a human tracking system using the Pioneer 3DX robot and the Microsoft Kinect device. The objective of this work consist of implement a system that allows an user especific recognition and be able to follow him in his displacement. Besides, during this tracking process, the robot will be able to avoid obstacles that stand in the way of the tracked user and the robot. In order to perform a more thorough and reliable user tracking, three alternatives have been proposed based on computer vision: recognition by color, recognition by shape and joint recognition.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1.1 Descripción del problema

La Federación Internacional de Robótica [27] define un robot de servicio como:

Un robot que opera de manera automática para realizar servicios útiles al bienestar de los humanos o a su equipamiento, excluyendo las operaciones de fabricación.

Desde los primeros desarrollos en la década de los 50, la robótica se centró en las aplicaciones industriales, experimentando una extraordinaria expansión en este ámbito. El objetivo de estos robots era sustituir o ayudar a las personas en tareas repetitivas, peligrosas o que requieran de una elevada precisión. Pero en la última década ha aparecido una necesidad de extender estas funcionalidades a otros ámbitos, tratando así que los robots realicen tareas diferentes a las que realizan los robots industriales. Para atender a esta demanda apareció lo que se denomina **Robótica de Servicio** [2].

Actualmente, se realizan cada vez más investigaciones y desarrollos de robots que entran en la vida diaria para desempeñar una función de ayuda al hombre, incluyendo comportamientos como el seguimiento de personas. Este trabajo se va a orientar en el seguimiento de personas mediante visión artificial, determinando así la posición del sujeto a seguir. Un ejemplo de este tipo de aplicaciones, en el marco de la robótica de servicio, es el que se presenta en [3], donde se utiliza un robot móvil para mantener un tanque de oxígeno cerca del paciente que lo requiere, evitando el esfuerzo que le supone arrastrarlo. Otro ejemplo, se describe en [4], donde se plantea el uso de una plataforma robótica móvil de seguimiento para el transporte de maletas de viajeros en un aeropuerto.

Para conseguir de forma efectiva el seguimiento de una persona por parte de un robot, éste debe identificar en primer lugar a la persona que va a seguir. Existen diferentes aproximaciones para resolver este problema, dependiendo del método que se utilice para identificar al individuo. Si es posible introducir identificadores artificiales en el individuo, estos pueden ser utilizados para facilitar el proceso de detección, pero si esto no es posible, se debe utilizar únicamente la información aportada por el individuo y la posibilidad de ser detectado por los sensores

disponibles [5]. Un problema que puede aparecer en el ámbito de los robots de seguimiento, es la localización e identificación de un individuo concreto en escenarios donde pueden aparecer otros individuos u obstáculos que impidan su seguimiento. En el presente documento, se va a abordar este problema creando un sistema de seguimiento mediante el uso de la visión artificial y el posicionamiento del robot en el entorno.

En este trabajo, se ha desarrollado un sistema de seguimiento de humanos para el robot Pioneer 3DX, mediante la utilización del dispositivo Kinect de Microsoft. A través de la información visual obtenida por la Kinect, se identifica al usuario único a seguir. Mientras que el robot P3DX se encarga de utilizar la información del usuario para realizar el seguimiento.

1.2 Motivación

En un principio, escogí realizar este trabajo porque me llamó la atención el uso combinado que se proponía entre la Kinect y el P3DX. Durante el resto del Grado en Ingeniería Informática no hemos trabajado en el área de la robótica, de hecho prácticamente no hemos trabajado con hardware. De ahí que me llamara la atención el campo de la robótica, en particular me atrajo sobretodo ver las posibilidades que tenía Kinect en un ámbito ajeno al de los videojuegos.

La oportunidad de que los robots de servicio tengan un impacto real en la vida doméstica todavía está a años de consumarse, pero el desarrollo de este trabajo me ha servido para darme cuenta de que ese momento no está tan lejos como me esperaba.

1.3 Objetivos del trabajo

El objetivo principal que se persigue con el desarrollo de este trabajo, consiste en el diseño e implementación de un sistema de seguimiento de humanos. Para realizar este objetivo se diseñará un sistema de control que permita al robot Pioneer 3DX [18] reconocer a un único usuario y seguir sus movimientos. Para realizar el reconocimiento y posicionamiento del usuario, se utilizará el dispositivo Kinect de Microsoft [19]. El objetivo principal de este trabajo ha sido dividido en pequeños objetivos. A continuación cada uno de ellos es descrito de forma detallada:

1. Familiarización del entorno ROS

Se deberá realizar un estudio detallado de este *framework*, con el fin de familiarizarse con sus características y aprovecharlas para realizar un mejor diseño. Además deberá realizarse un estudio de los paquetes ya existentes en ROS que puedan ser utilizados para el desarrollo de este trabajo.

2. Familiarización con el robot P3DX

   Se realizará un estudio del funcionamiento del robot P3DX, así como de los distintos paquetes de software que ROS ofrece para controlarlo.

3. Familiarización con el dispositivo Kinect

   Se realizará un estudio del funcionamiento del sensor Kinect, así como de los distintos paquetes de software que ofrece de ROS para controlarlo.

4. Desarrollo del reconocimiento de humanos

   Se deberá investigar cómo se obtendrá la información de los usuarios a través de la Kinect, y de qué manera integrarla en ROS para realizar las modificaciones necesarias para el desarrollo del sistema. Una vez obtenida dicha información, se desarrollará un sistema que sea capaz de emparejar al robot con un humano. También deberá ser capaz de identificar a un usuario marcado entre varios candidatos. Así como en caso de perdida del usuario, volver a reconocerlo al reaparecer en la escena.

5. Desarrollo del seguimiento de humanos

   Se desarrollará un sistema que utilizando la información obtenida del usuario, sea capaz de transmitir al robot P3DX órdenes para que realice movimientos de seguimiento del usuario a través del entorno. También será capaz de evitar posibles obstáculos que se encuentre en su camino.

6. Experimentación

Se diseñará una batería de pruebas en los que participen distintos usuarios, para comprobar el correcto funcionamiento del sistema desarrollado en un entorno real con el fin de evaluar el sistema.

7. Desarrollo de la documentación

   Se escribirá un documento describiendo detalladamente el sistema que se ha implementado y el proceso seguido para realizarlo.

1.4 Estructura del documento

A continuación se realiza una descripción detallada de la estructura del documento, presentando un resumen de los contenidos de cada uno de los capítulos:

- Introducción

  En este capítulo se realiza una breve introducción del problema que se quiere resolver y de la solución propuesta. También se presentan los objetivos establecidos para el trabajo.

- Estado del arte

  En este capítulo se presenta el estado de la cuestión en la que se ubica el trabajo. En primer lugar se presentan los diferentes diseños que han llevado a la aparición de la robótica y su evolución en sus últimos 50 años. A continuación, se realiza una descripción del concepto de visión por computador y de cómo puede ser utilizada en robótica. Por último, se presentan el sensor Kinect y el robot P3DX; los elementos básicos que han sido utilizados para el desarrollo de este trabajo.

- Descripción del sistema

  En este capítulo se realiza una descripción del sistema que ha sido desarrollado. Inicialmente se presentará el análisis del trabajo. A continuación se presenta el diseño arquitectónico del sistema y finalmente se realizará una descripción detallada del sistema desarrollado.

- Experimentación

En este capítulo se realiza una descripción de las pruebas que se han realizado sobre el sistema para comprobar su correcto funcionamiento en diferentes entornos, así como el análisis de los resultados obtenidos.

- Gestión del proyecto

En este capítulo se aborda el proceso de planificación del desarrollo del trabajo y el presupuesto deducido de dicha planificación y de su desarrollo.

- Conclusiones y trabajos futuros

En este capítulo se recogen las conclusiones extraídas durante todo el trabajo, junto con las posibles mejoras aplicables al sistema.

- Anexos

En este capítulo se incluyen los anexos considerados para añadir información adicional que puede resultar útil para la comprensión de algunos elementos de este trabajo.

Capítulo 2: Estado del Arte

En este capítulo se describe el contexto teórico en el que se enmarca el trabajo a desarrollar. En un principio se realiza una introducción a la robótica a través de su historia, seguido por la introducción al concepto de visón por computador, aplicado a la robótica. Por último, se describen los principales elementos necesarios para la realización del trabajo.

2.1 Robótica

A la hora de establecer un punto inicial en la historia de la robótica, lo primero que deberíamos decidir es lo que consideramos robótica o a qué elementos podemos llamar robots.

> En el ámbito de la informática, la robótica se conoce como la ciencia que estudia el diseño y programación de agentes físicos, también conocidos como robots. Por lo tanto, podemos definir a los robots como agentes físicos capaces de sustituir a un ser humano en la realización de alguna tarea.

Existen muchas historias acerca de los primeros agentes físicos o autómatas en la antigua Grecia , aunque suelen ser extraídos de los mitos. También se habla de un caballero mecánico creado por Leonardo da Vinci, pero no hay documentos que garanticen que alguna vez existió. El autómata más antiguo que se conserva es el Gallo de Estrasburgo, presentado en la Figura 1, que funcionó de 1352 hasta 1789, y que movía las alas y el pico cuando el reloj de la catedral daba las horas. Dado que estos autómatas antiguos no tenían una tarea que realizar no podemos considerarlos robots, pero sí un boceto en lo que basarse a la hora de establecer la necesidad de crear agentes físicos que funcionen de manera automática.

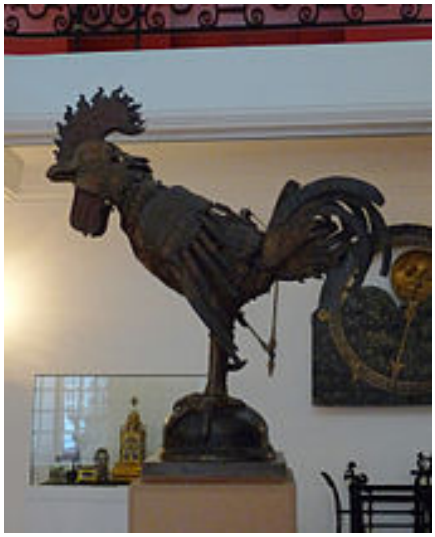

Figura 1 - Gallo de Estrasburgo

Donde se puede establecer que se fijaron los pilares de la robótica fue en el siglo XVIII, con la llegada de la revolución industrial que impulsó el desarrollo de autómatas mecánicos. En esta época se empezaron a desarrollar los primeros autómatas que sustituyeron a los humanos en la realización de tareas complejas o que requerían gran precisión. Algunos ejemplos que se pueden destacar de época son:

- El pato mecánico presentado en la Figura 2. Fue desarrollado por Jacques de Vaucanson en 1738.
- La hiladora giratoria construida en 1770 por el carpintero y tejedor James Hargreaves.
- El telar mecánico creado por Edmund Cartwright en 1785.
- Máquina textil programable mediante tarjetas perforadas, Joseph Jacquard (1801).
- Henri Maillardert creó en 1805, una muñeca mecánica, presentada en la Figura 3, que era capaz de escribir.

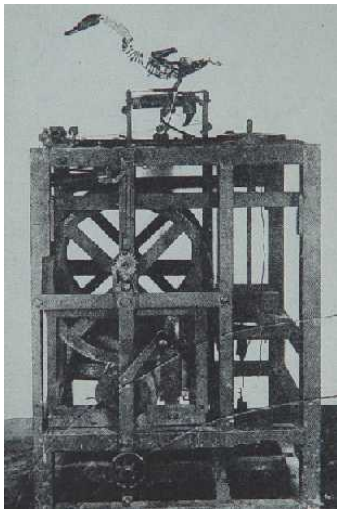

Figura 2 - Pato de Vaucanson

Figura 3 - Muñenca de Maillardert

Fue en 1920, donde se utilizó por primera vez la palabra *Robot* en la obra Rossum's Universal Robots del escritor checo Karel Capek [1]. En esta obra, la palabra robot se utiliza para designar a máquinas orgánicas de forma humanoide que trabajan para los seres humanos. En cuanto a su etimología, la palabra robot proviene de la palabra checa *robota*, que significa labor forzada o servidumbre.

Por otro lado, el término *Robótica* fue utilizado por primera vez por Isaac Asimov en su obra Yo Robot (1942). Fue también Asimov en su relato *Runaround*, donde estableció las famosas tres leyes de la Robótica [6]:

- Un robot no puede hacer daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
- Un robot debe obedecer las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley.
- Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley.

Pero es a partir de la década de los 50 cuando se produce una verdadera revolución en el diseño y desarrollo de autómatas, dando paso a lo que podríamos llamar el inicio de la robótica moderna. Uno de los grandes inventos que impulsaron este gran avance en la robótica, fue la creación del transistor (1951). Inventado por William Shockley, revolucionó lo que hasta entonces se venía dando en circuitos eléctricos, logrando así un gran crecimiento en el desarrollo computacional, al

igual que optimizar los procesos de automatización ya que ayudó a la miniaturización de los componentes. A partir de este momento, la robótica avanza rápidamente y se expande a diferentes ámbitos tecnológicos. Como por ejemplo, la creación en 1961 del primer brazo robótico industrial por parte de la empresa automovilística General Motors. Este robot llamado *Unimate* (Figura 4), cumplía uno de los objetivos básicos de la robótica, realizar tareas repetitivas o peligrosas, sustituyendo así a los trabajadores en las cadenas de montaje. Podemos asegurar que este robot estableció las bases de la robótica industrial.

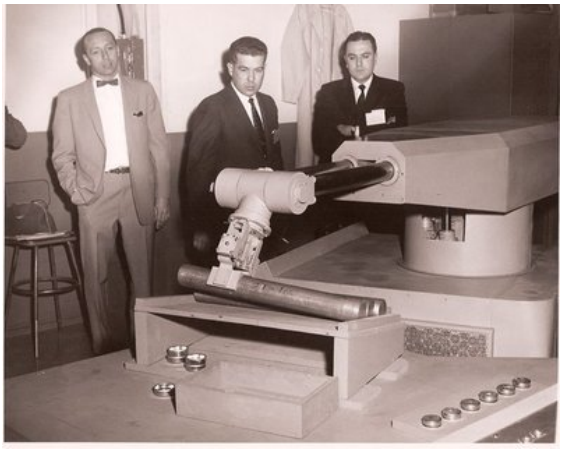

Figura 4 - Primer modelo de Unimate

A partir de la década de los 70, las investigaciones en el campo de la robótica aumentan exponencialmente, dando pie a la aparición de una gran variedad de robots. A continuación se enumeran algunos de los robots más destacables a lo largo de 40 años:

- En 1975 se creó PUMA (*Programmable Universal Machine for Assembly*), un brazo robot industrial dedicado a tareas de montaje, desarrollado por la empresa *Unimation.*
- En 1989, Rodney A. Brooks creó en el MIT (Massachussetts Institute of Technology) el robot Genghis, un prototipo de robot pequeño y ligero para reconocer la superficie marciana. En la Figura 5 se aprecia el pequeño tamaño del robot.
- En 1993 ingenieros de la universidad de Carnegie Mellon desarrollaron el robot Dante. Este robot estaba destinado a la exploración futura de Marte. Estaba diseñado para escalar y recoger muestras de gases. En 1994 se desarrolla el Dante II, modelo que triunfó donde su antecesor fracasó, consiguiendo entrar en el volcán Spurr en Alaska y recoger muestras gaseosas. En la Figura 6 se observa al robot Dante II durante la expedición a dicho volcán.

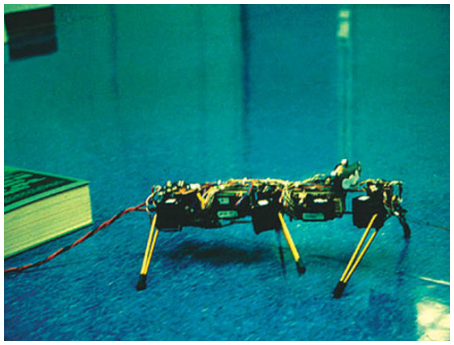

Figura 5 - Robot Genghis

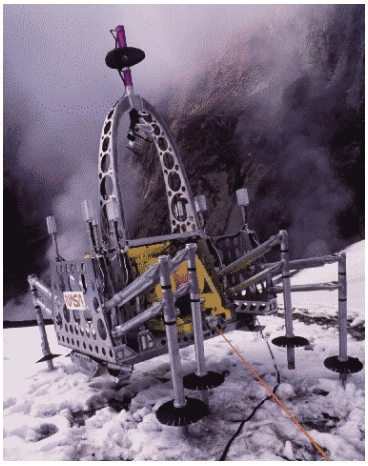

Figura 6 - Robot Dante II

- Entre los años 1993 y 2005, la empresa nipona Honda desarrolló una serie de robots humanoides teleoperados (P-series). Cabe destacar el último modelo llamado ASIMO, que era capaz de andar, correr, comunicarse con humanos, reconocimiento facial e interactuar con el entorno. En las figuras 7 y 8 se puede apreciar la evolución entre el primer modelo de las P-series y el último.

Figura 7 - Robot Honda P1

Figura 8 - Robot ASIMO de Honda

- En 2003, la NASA envía a la superficie de Marte a los *rovers* autónomos Spirit y Opportunity.
- En 2004, Aldebaran Robotics desarrolla el robot humanoide NAO, que a partir de 2007, será utilizado en la competición internacional de robótica Robocup.
- En 2011, la NASA envió a la Estación Espacial Internacional al robot humanoide Robonaut, con el objetivo de ayudar a los astronautas que se encuentran allí (Figura 9).

Figura 9 - Robonaut2 NASA

- En 2012, el rover Cusiosity fue enviado por la NASA a Marte con el objetivo de analizar la superficie del planeta e investigar la capacidad del planeta para albergar vida. Éste fue el último de los distintos tipos de rovers que ha enviado la NASA a la exploración del planeta rojo. En la Figura 10 se muestran los distintos modelos enviados.

Figura 10 - Rovers enviados a Marte por la NASA

2.2 Visión por computador

La visión por computador o visión artificial, es una técnica que consiste en extraer información del entorno mediante imágenes previamente captadas a través de una cámara y que son procesadas en un computador. De forma más coloquial, el objetivo de la visión artificial es imitar el comportamiento del ojo humano.

Inicialmente, la información visual consiste en energía luminosa procedente del entorno. Para poder utilizar esta información es preciso transformarla a un formato susceptible de ser procesado. En la visión humana esta tarea la realiza el ojo humano, en concreto la retina. En cambio, en visión artificial esta tarea la realiza el dispositivo de vídeo que estemos utilizando, que convierte la energía luminosa en corriente eléctrica, que puede ser entonces muestreada y digitalizada para su procesamiento en un ordenador. Las etapas a considerar en un proceso de visión artificial dependen del objetivo perseguido: reconocer, localizar, etc. En la Figura 11 se observan las diferentes etapas en las que se descompone de forma general el proceso de visión artificial. Cada una de las etapas es descrita a continuación.

- Adquisición de la imagen: Plasmar en una imagen digital el mundo real tridimensional, mediante el uso de algún dispositivo de captura.
- Preprocesamiento o filtrado: En esta fase se pretende conseguir una versión de la imagen en un formato que sea más fácilmente entendible para su manipulación. En esta etapa se solucionan problemas derivados del ruido de la imagen, deficiencias en la iluminación, bajo contraste, etc. Dentro de esta etapa, existen diferentes técnicas, como la conversión de la imagen a niveles de gris (para aumentar la calidad de la imagen), las transformaciones geométricas (para corregir la perspectiva), transformación del histograma (para la modificación del contraste), y filtrado espacial y en frecuencias (para pequeñas transformaciones de la imagen original).
- Segmentación: La segmentación consiste en aislar aquellos objetos que nos interesa estudiar, del resto de la imagen. Existen tres técnicas principales de segmentación: aplicación de umbrales de niveles de gris, agrupación por rasgos comunes y extracción de bordes.

- Representación y descripción: En esta fase se extraen las características más importantes de la imagen. Se obtiene una representación matemática de los objetos previamente segmentados.
- Reconocimiento de formas e interpretación: Se clasifican los objetos como pertenecientes a aquella clase cuyas características más se asemejen a la del objeto. Como en todo problema de clasificación, será necesaria una fase de selección de características y una de aprendizaje o entrenamiento. En concreto, en el reconocimiento de formas aplicado a la visión artificial, se utilizan técnicas como clasificación no supervisada o redes neuronales.

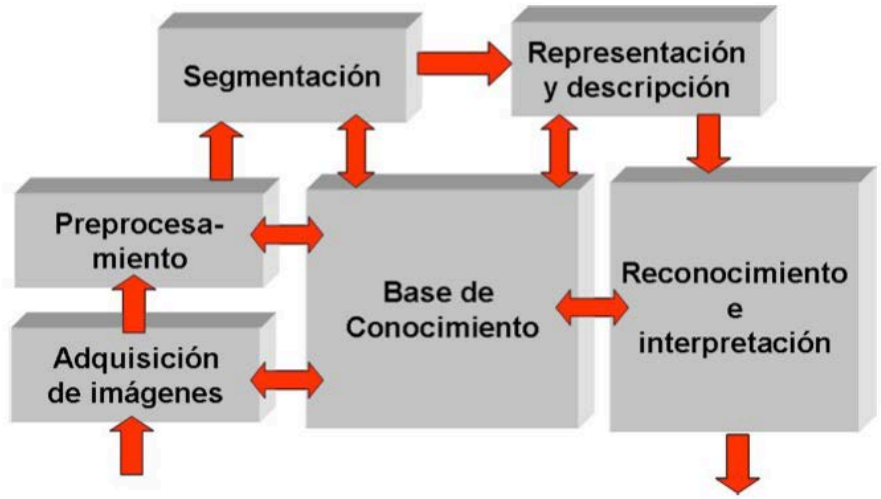

Figura 11 - Etapas de la visión artificial

Todo este proceso depende del dispositivo a través del cual se adquiere la imagen a procesar, la cámara digital. Actualmente el tipo de cámaras más utilizadas en el campo de la visión artificial son las CCD/CMOS.

- Las cámaras CCD (*charge-coupled device*) o dispositivo de carga acoplada, constan de un chip de silicio cuya superficie se divide en diminutas células fotoeléctricas o píxeles sensibles a la luz. Cuando un fotón alcanza un píxel, se contabiliza una pequeña carga eléctrica que permite registrar la imagen. Posteriormente, esta imagen es procesada por la cámara y almacenada en la tarjeta de memoria.
- Las cámaras CMOS (*complementary metal-oxide-semiconductor*) se denominan así porque emplean tecnología CMOS para controlar los fotodiodos que captan la luz, es decir, al igual

que en las cámaras CCD, su sensor está basado en el efecto fotoeléctrico, pero utilizando componentes electrónicos CMOS (transistores MOSFET).

La industria relacionada con la visión por computador está actualmente en fase de rápido crecimiento, debido tanto a la incursión en nuevas áreas de aplicación como a la mayor importancia que está tomando en las ya consolidadas. Entre las principales aplicaciones prácticas de la visión por computador destacan:

- Industria
  - Control de calidad e inspección
  - Identificación de piezas
  - Ensamblaje
  - Medición de objetos
  - Guiado de robots
- Medicina
  - Pruebas de laboratorio automáticas
  - Diagnóstico por computador
- Tele-medición
  - Exploración geofísica
  - Meteorología
- Defensa Militar
  - Vigilancia por satélite
  - Guiado de larga distancia
  - Armas inteligentes
- Robótica
  - Reconocimiento de objetos, personas , etc
  - Reconocimiento facial
  - Mapeado del entorno

Entre las dificultades que encontramos a la hora de crear un sistema de visión artificial reside en la propia naturaleza de las imágenes manejadas por el computador. En primer lugar, éstas imágenes son digitales, lo que sin duda añade una dificultad adicional y , además, están afectadas por ruido diverso. En segundo lugar, y más importante aún, en los niveles de intensidad de los píxeles intervienen un gran número de factores como son, fundamentalmente:

- La iluminación de la escena.
- La geometría del objeto.
- El color y la textura de las superficies.
- Los parámetros y distorsiones de la cámara.

Lo realmente complicado de esto es que la contribución individual de estos factores es imposible de precisar, con lo cual se complica el proceso de extracción de información de los objetos de la escena.

2.2.1 Modelos de color

Como se ha comentado anteriormente, una de las funciones de la visión artificial es el reconocimiento de patrones. Específicamente en este trabajo, se ha utilizado el reconocimiento de patrones de color para identificar usuarios. Para determinar qué color se está captando en una escena determinada, se utilizan los modelos de color, ya que permiten representar los colores de forma numérica y crear filtros para discriminar colores. Existen muchos modelos de color, que se diferencian en la representación numérica que realizan del color. A continuación se describen los modelos utilizados en el desarrollo del trabajo, el RGB y el YUV.

2.2.1.1 Modelo RGB

Este modelo se basa en lo que se conoce como síntesis aditiva, con lo que es posible representar un color por la mezcla por adición de los tres colores primarios con los que se forma: rojo (*R*), verde (*G*) y azul (*B*). Le asignamos un valor a cada uno de los colores primarios para indicar con qué proporción se mezcla cada color. Así, por ejemplo, el valor 0 significa que no interviene en la mezcla, y en la medida que ese valor aumenta, aportará más intensidad a la mezcla, hasta llegar al valor 255 o 1 si se normaliza. Cada color primario es representado por un byte, de allí que el valor máximo sea 255.

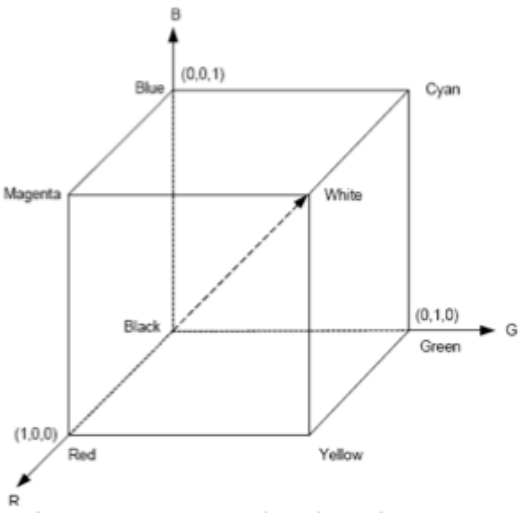

Figura 12 - Tetraedro modelo RGB

Este modelo está basado en el sistema de coordenadas cartesianas. El sub-espacio de color de interés es el tetraedro mostrado en la Figura 12, en el cual los valores *RGB* están en tres vértices; cyan, magenta y amarillo se sitúan en otros tres vértices, el negro (0, 0, 0) corresponde al origen y el blanco (255, 255, 255)en el vértice más alejado del origen.

2.2.1.2 Modelo YUV

El modelo *YUV* define un espacio de color en términos de una componente de luminancia (*Y*) y dos componentes de crominancia (*U*,*V*). Codifica una imagen en color teniendo en cuenta la percepción humana, de este modo hace que sea más robusto ante cambios de iluminación.

La luminancia se define como la densidad angular y superficial de flujo luminoso que incide, atraviesa o emerge de una superficie siguiendo una dirección determinada. Por su parte, la crominancia contiene información del color, no de la luz o brillo como la luminancia.

Es posible obtener los valores *YUV* de un píxel partiendo de sus valores *RGB* mediante la siguiente ecuación:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Se asume que *R*, *G* y *B* están normalizados. *Y* está en el rango 0 a 1, *U* está en el rango -0.436 a 0.436 y *V* está en el rango -0.615 a 0.615.

Este modelo es el utilizado en los sistemas de difusión de televisión *PAL* y *NTSC*, los cuales son los estándares en la mayoría del mundo.

2.2.2 Mapas de profundidad

Una aplicación muy utilizada de la visión artificial es la reconstrucción tridimensional de una escena o la extracción de información de una escena utilizando mapas de profundidad. En este trabajo, se va a utilizar la Kinect para el reconocimiento de personas en la escena. Dicho dispositivo usa los mapas de profundidad para extraer la información necesaria de la escena para conseguir su objetivo.

Un mapa de profundidad es una matriz que contiene valores de distancias, es decir, cada punto representa la distancia entre el emisor del sensor y el punto detectado. En todos los aspectos, un mapa de profundidad es equivalente a una imagen, sólo que en lugar de almacenar valores de color, almacena distancias.

Para poder crear un mapa de profundidad, la Kinect utiliza un emisor IR, cuyo haz pasa por un filtro y es dispersado en un patrón semi-aleatorio, pero constante de pequeños puntos infrarrojos. El patrón reflejado es capturado por el receptor y analizado. El funcionamiento de dicho sensor es similar al del sónar de un barco, dado que los rayos de luz infrarroja se desplazan por el aire hasta encontrar un obstáculo donde rebotar, teniendo en cuenta que irán perdiendo intensidad a medida que recorren cierta distancia. De este modo, las variaciones de intensidad en el patrón proyectado permiten obtener la estructura 3D de la escena. En la Figura 13 se muestra la emisión del patrón de puntos infrarrojos de la Kinect.

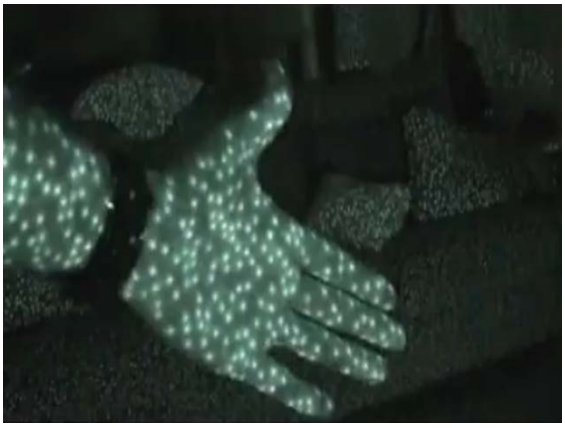

Figura 13 - Emisión IR de la Kinect

De esta forma, como se observa en la Figura 14, a través del dispositivo Kinect se pueden detectar personas en una escena analizando el patrón de puntos y comparándolas a unas plantillas pre-establecidas de posibles formas de personas. Como veremos más adelante, mediante el mapa de profundidad generado, también seremos capaces de extraer un esqueleto virtual de los usuarios detectados, para así conocer su posición en la escena.

Figura 14 - Mapa profundidad de la Kinect

2.2.2.1 Cámaras RGB

El uso de cámaras RGB como sensor es una de las técnicas más básicas utilizada en robots de seguimiento para el proceso de adquisición de imágenes y después utilizar un algoritmo para identificar a las personas. En [8], utilizan un robot que usa una cámara RGB para percibir el ambiente y posteriormente analizan la imagen mediante histogramas de color y modelos de contorno. En [9], utilizan la información de color que provee una cámara para detectar la piel de

un humano y poder localizar la cabeza, de modo que el robot debe moverse para lograr que la parte superior de la persona quede en el centro de la imagen captada por la cámara. El problema de este tipo de métodos son las variaciones de color del *background* y de la iluminación de la escena causadas por el movimiento del robot.

2.2.2.2 Láser

Muchos sistemas de seguimiento utilizan como sensor uno o varios LRF (*Laser Range Finder*) y después utilizan un algoritmo de detección de piernas para detectar las extremidades inferiores de la persona. Su funcionamiento consiste en la emisión de un haz láser para determinar distancias, midiendo el tiempo que tarda el haz en ser reflejado, pudiendo así generar un mapa de distancias de un escenario. En [10], construyeron un robot seguidor combinando el uso de láser, el algoritmo de detección de piernas y un algoritmo de inferencia basado en reglas para compensar el ruido y los falsos negativos que ocurren al detectar las piernas del guía.

Otra técnica, consiste en la detección de personas utilizando mapas de diferencia. El robot debe aprender el mapa del lugar antes de que haya personas. Posteriormente, compara el mapa actual (personas incluidas) con el original, y calcula la diferencia para verificar si hay algo nuevo [11].

Otra opción es combinar el uso de láser y visión. En [12], se propone un robot de servicio que utiliza LRFs para detectar candidatos a personas, localizarlas y rastrearlas. Utilizando las imágenes de la cámara, se hace una verificación detectando rostros y la parte superior del cuerpo. Concepto similar al que utilizan en [13], combinando el láser para detectar piernas y la cámara para detectar cara.

2.2.2.2 Cámaras estereoscópicas

Otra forma de reconocer personas es utilizar la información 3D que se puede obtener de este tipo de cámaras. En [14], proponen un robot seguidor que busca la parte inferior de una persona analizando la escena 3D generada mediante una cámara estereográfica.

En [15], proponen la combinación de cámaras estereoscópica con modelos probabilísticos de piel. Lo primero que hace el sistema es obtener una imagen de disparidad utilizando una cámara estereoscópica, permitiendo calcular a la distancia que se encuentran los objetos. En la imagen de

disparidad se hace una segmentación de objetos, y a cada uno de ellos se les aplica un modelo de contorno semi-elíptico para verificar si la forma corresponde a la de un humano. Junto con esto, se emplea un modelo probabilístico para la detección de piel que fue obtenido mediante una máquina de aprendizaje, tomando como muestras regiones de imágenes correspondientes a piel de humanos de diferentes razas y edades.

2.2.2.4 Cámaras térmicas

El uso de cámaras térmicas es otra opción para reconocer a una persona, aprovechando el hecho de que los seres humanos tenemos niveles de temperatura distintos al de otros objetos de la posible escena. Una de las ventajas de este método es que los datos obtenidos del sensor no se ven afectados por los cambios en la luminosidad y se pueden detectar personas incluso en la oscuridad.

En [16], proponen un sistema de reconocimiento de personas combinando información térmica y de color. En su sistema, se obtiene una imagen térmica a la cual se le aplican técnicas de análisis de imagen para segmentar a las personas. Esta información es usada para segmentar las regiones correspondientes en la imagen de color usando una transformación afín para resolver la correspondencia entre las dos imágenes. Después de la segmentación, la región de la imagen que contiene a la persona es dividida en secciones que corresponde a la cabeza, el torso y los pies.

2.3 Herramientas

A continuación se realiza una descripción de los elementos básicos utilizados en el desarrollo del trabajo, entre los que se incluyen el dispositivo Kinect y el robot P3DX.

2.3.1 Kinect

En el E3 (Electronic Entertainment Expo) de 2009, Microsoft anunció un nuevo proyecto llamado 'Project Natal', que consistía en un nuevo controlador para la videoconsola Xbox360. Fue un año más tarde, una vez más en el E3, cuando se dió a conocer el verdadero nombre del dispositivo, Kinect (kinetic + connect). Kinect salió a la venta el 4 de Noviembre de 2010 en Norteamérica y el 10 de Noviembre en Europa.

Este dispositivo permite al usuario interactuar directamente mediante sus movimientos con la consola, sin necesidad de tener contacto físico con un controlador de videojuegos tradicional como puede ser un mando. En la Figura 15 se observa como dos usuarios utilizan sus movimientos para jugar a la Xbox360. Esta forma de interacción usuario-máquina también es conocida como interfaz natural de usuario (NUI), ya que permite al usuario una interacción con la consola a través de gestos , reconocimiento facial y comandos de voz.

Figura 15 - Kinect en Xbox 360

En Febrero de 2012, Microsoft dio un paso más para aumentar las posibilidades del dispositivo, sacando al mercado la versión de Kinect para Windows. Gracias a esto, Microsoft amplió el desarrollo de aplicaciones que utilizan Kinect, más allá del mudo de los videojuegos.

En cuanto a la tecnología utilizada por la Kinect, se basa en un software desarrollado por la compañía de videojuegos Rare, que pertenecía a Microsoft. Junto con este software se utilizó una cámara capaz de medir distancias desarrollada por la compañía israelí PrimeSense, que permitía desarrollar un sistema de interfaz natural de usuario.

2.3.1.1 Desripción

A simple vista, el sensor Kinect es una barra horizontal de unos 23 centímetros conectada a una base circular con un eje de articulación que permite modificar el ángulo vertical para dirigir la vista del sensor, pero como se aprecia en la Figura 16, el dispositivo está compuesto por una serie de

elementos. A continuación se describen estos elementos que unidos, permiten crear un sistema de interfaz natural de usuario.

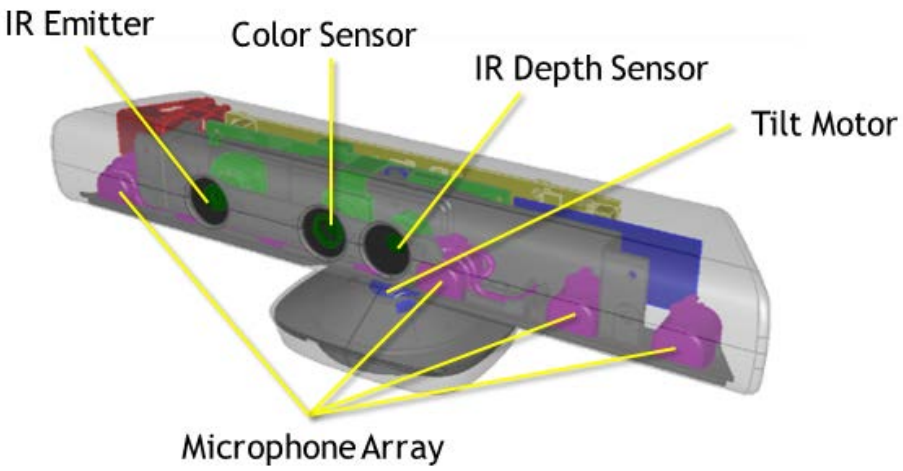

Figura 16 - Elementos que componen la Kinect

Cámara RGB

Una cámara RGB que almacena tres canales de datos con una resolución de 640x480 con hasta 30 fps, aunque en su versión para Windows puede llegar a una resolución de 1280x960. Esta parte del sensor se encarga de la captura de vídeo en diferentes formatos de color.

El funcionamiento de la cámara RGB es como el de una cámara digital estándar. La luz atraviesa una lente que la dirige a un filtro encargado de separarlo en los colores primarios, los cuales son proyectados sobre un sensor fotosensible.

Sensor de profundidad

Está formado por dos componentes: un proyector de luz infrarroja (IR) y un sensor CMOS monocromo estándar. El proyector emite rayos infrarrojos, imperceptibles por el ojo humano, y el sensor de profundidad lee estos rayos infrarrojos una vez vuelven al emisor al ser reflejados por algún objeto. Los rayos reflejados son convertidos en información de profundidad midiendo la distancia entre un objeto y el sensor, partiendo de la base de que cuanto más alejado esta el objeto en el que rebota el rayo, con menos intensidad volverá el rayo. Esto es lo que hace posible capturar la imagen de profundidad.

Micrófono multiarray

El dispositivo también contiene cuatro micrófonos para capturar sonido. La inclusión de cuatro micrófonos permite , a parte de grabar mejor el audio, encontrar el origen del sonido así como la dirección de la onda sónica. Esto se consigue gracias a que cada uno de los cuatro micrófonos recibe el sonido en un instante de tiempo distinto y el desfase entre esos sonidos es lo que permite calcular el origen del sonido. Otra característica que tiene este tipo de configuración es que permite a la Kinect aislar el sonido ambiente y centrarse en el sonido que proviene del usuario.

Base motorizada

El último elemento de la Kinect, es la encargada de ajustar la inclinación en caso de que la cámara no detecte bien lo que tenga delante

Estos elementos facilitan las información necesaria para que el software desarrollado para la Kinect pueda proporcionar una captura de movimiento de todo el cuerpo en 3D, diferenciando a los usuarios del resto de objetos, reconocimiento facial del usuario y reconocimiento de voz procedente del usuario.

2.3.1.2 Especificaciones

En la siguiente tabla se pueden observar las especificaciones técnicas del dispositivo Kinect de Microsoft.

Ángulo de visión horizontal	57grados
Ángulo de visón vertical	43 grados
Ángulo de inclinación vertical	+/- 27 grados
Cámara de color	640 x 480 32-bit de color 30fps
Sensor de profundidad	320 x 240 16-bits de profundidad 30 fps
Formato de audio	16-bit a 16 kHz
Entrada de audio	4 micrófonos con 24-bit conversión analógica-digital (ADC)
Rango de profundidad del sensor	1,2-3,5 metros
Área mínima requerida	6 m^2

Tabla 1 . Especificaciones técnicas de la Kinect

2.3.2 Pioneer 3DX

En robot en el que se desarrolla el trabajo es el Pioneer 3DX, comercializado por la empresa *ActivMedia*. El 3DX está dotado de tres elementos básicos, unos sensores que le permiten medir el entorno y evitar obstáculos, unos procesadores que le dan la capacidad de cálculo y unos motores encargados del movimiento del robot. Respecto al procesamiento, el robot dispone de un único procesador de 18 MHz Hitachi H8S/2357 de 32 Kb de RAM y 128 Kb de memoria flash.

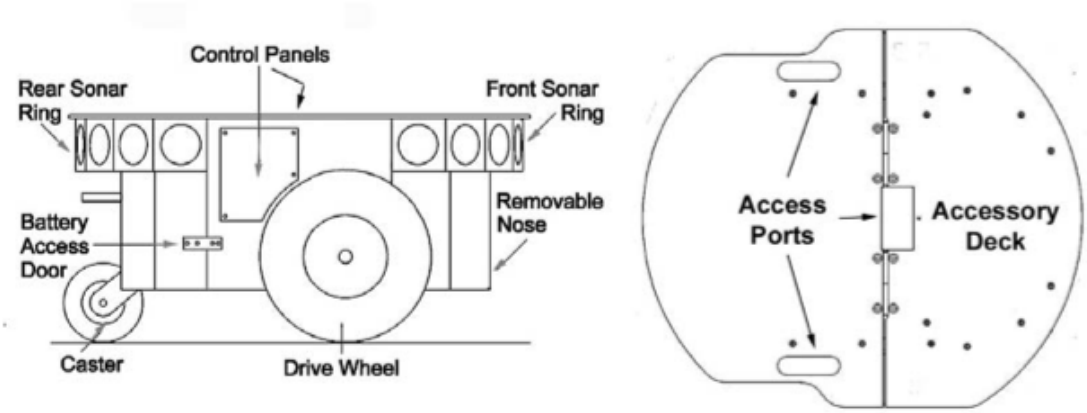

Figura 17 - Componentes del robot P3DX

En cuanto al aparatado sensorial, dispone de una corona de 16 sensores de ultrasonido que rodean al robot y llevan incorporados unos odómetros que cuentan las vueltas de cada rueda.

En el apartado motriz, el robot está compuesto de tres ruedas, dos ruedas fijas impulsadas por un motor de corriente continua cada una y una tercera rueda "loca" sin tracción, que facilita el giro del robot. El P3DX tiene una autonomía de 5 horas, es capaz de adquirir una velocidad lineal de 1'8 *m/s* y una velocidad angular de 360 *grados/s*. Puede llegar a soportar una carga de 23 Kg y tiene una tamaño de 40x30 cm.

Capítulo 3: Descripción del sistema

En este capítulo se realiza una descripción detallada del sistema desarrollado, a través de la definición previa de los casos de uso y requisitos que describen las características y el alcance del sistema. A continuación se presenta una descripción detallada de la arquitectura del sistema, junto con la explicación de su funcionamiento.

3.1 Introducción

El objetivo de este trabajo es diseñar e implementar un sistema que permita que el robot Pioneer 3DX sea capaz de seguir a una persona mediante la utilización del dispositivo Kinect de Microsoft. Creando así un sistema que permita al usuario comunicarse con el robot mediante movimientos naturales. Para realizar la comunicación entre estos dos elementos, el sistema utiliza el *framework* ROS, un sistema operativo para robots. El sistema deberá ser capaz de reconocer a los humanos presentes en las escena, pero únicamente deberá identificar al usuario deseado. El usuario a identificar mostrará un elemento identificativo que el sistema utilizará para reconocerlo de entre el resto de posibles personas. Una vez identificado al usuario, el sistema deberá ser capaz de seguir al usuario durante su recorrido y evitar posibles obstáculos que puedan aparecer en el camino.

3.2 Análisis del sistema

A continuación se describe el proceso de análisis realizado para establecer el alcance del trabajo. En primer lugar se definirán unos casos de uso que describen las funcionalidades del sistema. A continuación se establecerán las restricciones y el entorno operacional del sistema. Por último, se definirán una serie de requisitos funcionales y no funcionales obtenidos a partir de la información anterior.

3.2.1 Descripción de las características funcionales

A continuación se describen las funcionalidades básicas de las que se compondrá el sistema:

- El proceso de visión artificial deberá ser capaz de reconocer al usuario mediante la utilización de información del propio usuario. Para realizar este tipo de reconocimiento se han definido tres modelos de identificación:
  1. Reconocimiento por color: El sistema identificará al usuario mediante el reconocimiento del color de su camiseta, previamente calibrado.
  2. Reconocimiento por forma: El sistema identificará al usuario mediante el reconocimiento de una plantilla con un círculo dibujado.
  3. Reconocimiento mixto: Se identificará al usuario combinando las dos alternativas anteriores.
- En caso de que el sistema de reconocimiento pierda al usuario, el robot será capaz de volver a reconocerlo y seguirle, siempre y cuando sea el usuario el que vuelva a la escena y no sea el robot el que se encargue de buscarle.
- El sistema será capaz de reconocer a un usuario de entre un conjunto de éstos y seguirle.
- El usuario simplemente deberá moverse hacia una dirección, y el robot le seguirá, incluyendo tanto desplazamientos lineares como angulares.
- En ningún momento deberá cambiar de usuario a seguir y no podrá seguir a más de un usuario a la vez
- El sistema también deberá ser capaz de mostrar al usuario perseguido lo que está observando el robot a través de la Kinect. Así como información textual sobre el estado del seguimiento.
- El robot no deberá ir a más de una cierta velocidad para no poner en peligro su integridad.

3.2.2 Restricciones del sistema

En este apartado se indican las restricciones que imponen tanto el software como el hardware al desarrollo de nuestra sistema.

3.2.2.1 Restricciones Hardware

Dado que el sistema a desarrollar está compuesto por dos elementos de hardware, a continuación se exponen las restricciones a las que nos someten.

- El dispositivo Kinect necesita un espacio mínimo de 6 metros cuadrados para poder realizar correctamente las calibraciones.
- El dispositivo Kinect tiene un rango de visión máximo que ronda los tres metros y medio, por lo que la distancia entre el usuario y el dispositivo no puede ser mayor, para así garantizar el correcto funcionamiento.
- El dispositivo Kinect tiene un rango de visión mínimo que ronda el metro, por lo que el usuario no debe acercarse a menos de esa distancia si quiere ser reconocido correctamente.
- El número máximo de usuario que puede reconocer la Kinect en una misma escena es de 6.
- La Kinect al no tener batería, debe estar conectada en todo momento a la corriente, por lo que el sistema deberá permanecer cerca de una toma de corriente
- El robot P3DX tiene una autonomía máxima de 5 horas.
- El P3DX alcanza un velocidad máxima de 1,8 m/s. Aunque el sistema está diseñado para no ir a más de 1 m/s para garantizar su seguridad.
- El P3DX puede soportar una carga máxima de 23 Kg. Dado que el robot ya lleva encima todo el sistema de cableado, un soporte de madera, la Kinect y un ordenador, no es recomendable añadir más peso sobre él.

3.2.2.2 Restricciones Software

Estas restricciones hacen referencia a aquellas limitaciones impuestas por el software que utiliza nuestro sistema.

- El sistema debe implementarse mediante el *framework* ROS (*Robot Operating System*).

- ROS debe ser ejecutado sobre una distribución de Ubuntu. Puesto que es el sistema operativo en el que ROS ofrece una mayor estabilidad. Esta distribución dependerá de la versión de ROS que se utilice.
- Para la correcta identificación del usuario, éste se debe posicionar a una distancia mayor de 0,5 metros.
- El proceso de calibración del color de la camiseta del individuo a seguir se deberá realizar previamente a ejecutar el sistema.
- Para realizar el proceso de calibración, se requerirá que un usuario que seleccione el color de la camiseta del usuario a seguir, lo marque en la imagen y guarde los datos en un fichero.
- El lenguaje de programación que se va a emplear para el desarrollo del trabajo será C++. Dado que entre todos los lenguajes de programación que ofrecen soporte en ROS es con el que estoy más familiarizado.

3.2.3 Entorno operacional

En este apartado se describe el entorno operacional, tanto hardware como software, necesario para el correcto funcionamiento del sistema que se ha desarrollado.

- En cuanto al software, serán necesarias las siguientes herramientas:
  - Sistema operativo Ubuntu 11.10.
  - Distribución Electric del *framework* ROS.
  - El *stack* openni_kinect de ROS, encargado de la integración del dispositivo Kinect con ROS.
  - El *stack* p2os de ROS, encargado de la integración de los drivers del robot 3DX con ROS.
  - El *stack* cmvision de ROS, encargado del procesamiento para el reconocimiento de colores.
  - El *stack* vision_opencv de ROS, encargado del procesamiento de imágenes.

- Las herramientas hardware necesarias para el desarrollo del sistema son las siguientes:
  - Un robot Pioneer 3DX.

- Un ordenador portátil que contenga el entorno operacional software descrito.
- Un dispositivo Kinect de Microsoft.

3.2.4 Especificación de casos de uso

En este apartado se presenta el diagrama de casos de uso y la descripción detallada de cada uno de ellos.

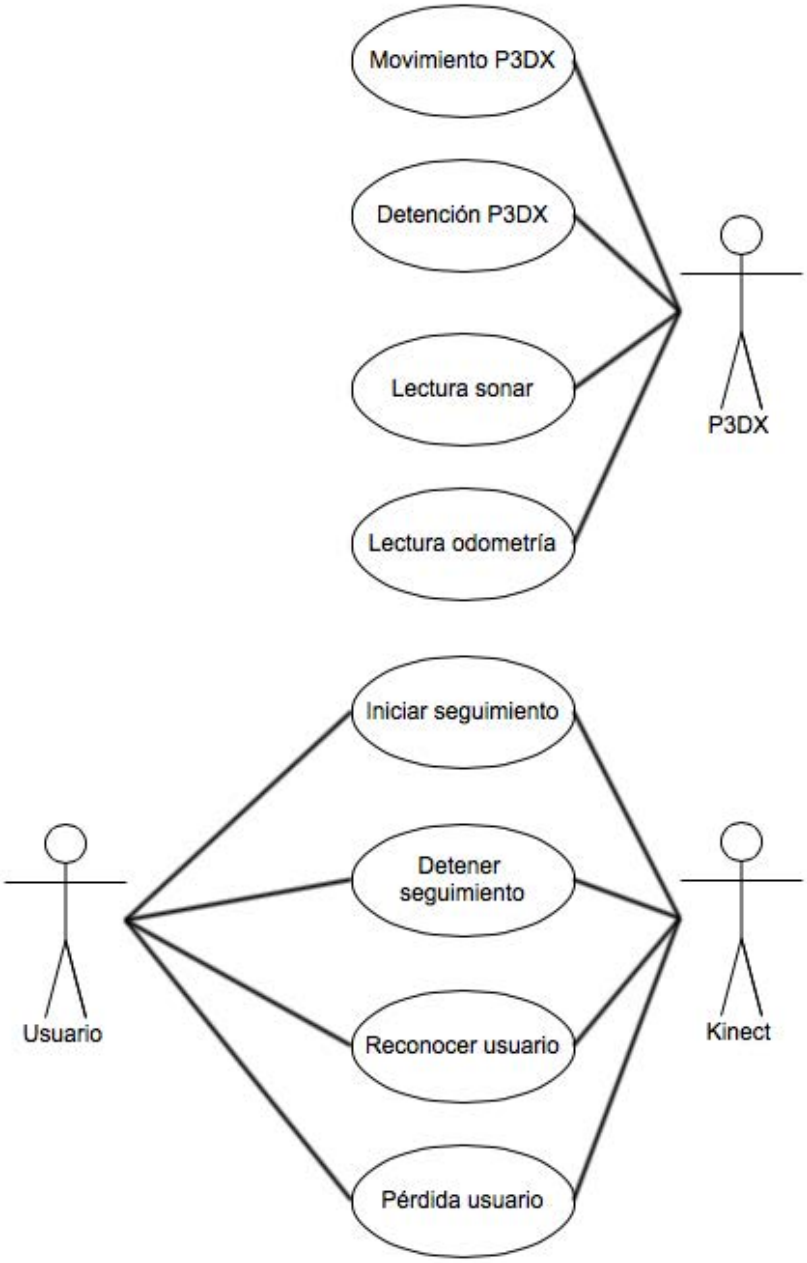

Figura 18 - Diagrama de casos de uso

3.2.4.1 Descripción de los actores

Como se puede observar en la Figura 18, el sistema está compuesto por tres actores. La función de este sistema es establecer una relación entre el movimiento del usuario y el movimiento del robot, mediante un actor intermedio que sería la Kinect. A continuación se realiza una breve descripción de cada uno de los actores que intervienen en el funcionamiento del sistema:

- Usuario: Este actor es el encargado de proporcionar la información necesaria para realizar su reconocimiento y su posterior seguimiento.
- Kinect: Este actor es el encargado de recoger la información proporcionada por el usuario y procesarla para establecer su reconocimiento.
- P3DX: Este actor es el encargado de utilizar la información procesada por la Kinect y la obtenida del entorno para realizar el seguimiento del usuario.

3.2.4.2 Descripción de los atributos de los casos de uso

Para la realización de la descripción textual de los distintos casos de uso, se han seleccionado una serie de atributos que describen cada uno de los casos de uso. A continuación se realiza una descripción del significado de cada uno de los atributos utilizados para la descripción de los casos de uso.

- Código: Identificación unívoca abreviada del caso de uso, se construye mediante CU seguido de un - y de tres dígitos. Por ejemplo CU-001.
- Nombre: Identificación extendida del caso de uso.
- Actores: Conjunto de entidades que interactúan con el caso de uso. El caso de uso representa una funcionalidad demandada por un actor.
- Descripción: Se realiza una descripción básica de la funcionalidad o funcionalidades del caso de uso.
- Precondiciones y poscondiciones: Se realiza una descripción de las condiciones que deben cumplirse para poder realizar una operación, y el estado en el que queda el sistema tras realizar una operación.
- Escenario: Se realiza una descripción básica de las acciones que se ejecutaran paso a paso en el caso de uso.

3.2.4.3 Descripción textual de los casos de uso

Caso de uso	
Código	CU-001
Nombre	Reconocer usuario
Actores	Usuario, Kinect
Descripción	Permite a la Kinect obtener la información del usuario.
Precondiciones	• La Kinect debe estar encendida y conectada al ordenador. • El usuario debe aparecer en la escena.
Poscondiciones	• El usuario ha sido reconocido
Escenario	1. Se calibra el color por el que será reconocido el usuario. 2. Se detecta al nuevo usuario que aparece en la escena. 3. Se localiza al usuario, asignándole un esqueleto a partir de sus articulaciones. 4. Se identifica al usuario a través de su color de camiseta. 5. Se publican los mensajes correspondientes a la posición del usuario.

Tabla 2 - Casos de uso CU-001

Caso de uso	
Código	CU-002
Nombre	Pérdida usuario
Actores	Usuario, Kinect
Descripción	Se detiene el proceso de reconocimiento de usuario.
Precondiciones	• La Kinect debe estar encendida y conectada al ordenador. • El usuario desaparece de la escena durante más de 10 segundos.
Poscondiciones	• El usuario es perdido.
Escenario	1. Se pierde al usuario reconocido. 2. Se detiene la publicación de los mensajes correspondientes a la posición del usuario.

Tabla 3 - Caso de uso CU-002

Caso de uso	
Código	CU-003
Nombre	Iniciar seguimiento
Actores	Usuario, Kinect, P3DX
Descripción	Permite al P3DX iniciar el proceso del seguimiento del usuario.
Precondiciones	• El usuario debe estar reconocido por la Kinect. • El P3DX debe estar encendido y conectado al ordenador. • Seguimiento detenido
Poscondiciones	• Seguimiento iniciado
Escenario	1. Se reciben los mensajes correspondientes al la posición del usuario. 2. Se extrae la posición del torso del usuario. 3. Si dicha posición es distinta a la anterior: 4. Se transforma esa posición en componentes de velocidad. 5. Se publican las velocidades a seguir.

Tabla 4 - Caso de uso CU-003

Caso de uso	
Código	CU-004
Nombre	Detener seguimiento
Actores	Usuario, Kinect, P3DX
Descripción	Se detiene el proceso de seguimiento del usuario.
Precondiciones	• El P3DX debe estar encendido y conectado al ordenador. • Seguimiento en proceso. • Deja de reconocerse al usuario.
Poscondiciones	• Seguimiento detenido.
Escenario	1. Se reciben los mensajes correspondientes al la posición del usuario. 2. Se extrae la posición del torso del usuario. 3. Si dicha posición es igual que la anterior: 4. Se establece la velocidad a 0.

Tabla 5 - Caso de uso CU-004

Caso de uso	
Código	CU-005
Nombre	Movimiento P3DX
Actores	P3DX
Descripción	El robot se desplaza al usuario reconocido.
Precondiciones	• El usuario debe estar reconocido por la Kinect • Seguimiento iniciado.
Poscondiciones	• Desplazamiento P3DX.
Escenario	1. Se recibe la velocidad a la que debe moverse el robot. 2. Si la velocidad es positiva se mueve hacia delante. 3. Si la velocidad es negativa se mueve hacia atrás. 4. Si la velocidad tiene componente angular positiva gira a la izquierda. 5. Si la velocidad tiene componente angular negativa gira a la derecha.

Tabla 6 - Caso de uso CU-005

Caso de uso	
Código	CU-006
Nombre	Detención P3DX
Actores	P3DX
Descripción	El robot se detiene.
Precondiciones	• El usuario debe estar reconocido por la Kinect • Seguimiento en proceso. • El usuario se detiene, desaparece de la escena o realiza un gesto de parada.
Poscondiciones	• P3DX parado.
Escenario	1. Se recibe la velocidad a la que debe moverse el robot. 2. La velocidad recibida es 0.

Tabla 7 - Caso de uso CU-006

Caso de uso	
Código	CU-007
Nombre	Lectura sonar
Actores	P3DX
Descripción	El robot realizará lecturas del sonar que tiene incorporado para detectar posibles obstáculos.
Precondiciones	• Seguimiento en proceso. • Se detecta un obstáculo .
Poscondiciones	• P3DX realiza un movimiento de evasión del obstáculo.
Escenario	1. Se reciben los rangos de distancia cada uno de los sonares. 2. En caso de que la distancia al obstáculo detectado sea menor a la del umbral establecido, se realiza un movimiento de evasión.

Tabla 8 - Caso de uso CU-007

Caso de uso	
Código	CU-008
Nombre	Lectura odometría
Actores	P3DX
Descripción	El robot realizará lecturas de la odometría que tiene incorporada para establecer su posición en el espacio.
Precondiciones	• P3DX empieza un movimiento de evasión del obstáculo.
Poscondiciones	• P3DX termina un movimiento de evasión del obstáculo.
Escenario	1. El sistema ha detectado un obstáculo. 2. El robot lee la posición actual en la que se encuentra. 3. En caso de que el robot deba rotar, se utiliza la orientación dada por la odometría para girar unos grados dados. 4. En caso de que el robot deba avanzar, se utiliza la posición dada por la odometría para avanzar una distancia dada.

Tabla 9 - Caso de uso CU-008

3.2.4.2 Descripción de los atributos de los requisitos

Para la realización de la descripción textual de los distintos requisitos que han sido identificados, se han seleccionado una serie de atributos que describen cada uno de los requisitos. A continuación se realiza una descripción del significado de cada uno de los atributos utilizados para su descripción:

- Código: Identificación unívoca abreviada del requisito, se construye mediante el código del requisito seguido de un - y de tres dígitos. Los requisitos serán divididos en funciones y no funcionales y sus códigos son RF para los requisitos funciones y RNF para los requisitos no funcionales. Por ejemplo RF-001.
- Nombre: Identificación extendida del requisito.
- Descripción: Se realiza una descripción básica del requisito que ha sido identificado.
- Fuente: Indica a través de que fuente ha sido identificado el requisitos. Normalmente este valor se corresponderá con uno o varios códigos de los casos de uso.

- Necesidad: Determina el grado de implementación del requisito. Los valores que puede tomar este atributo son los siguientes:
    - Esencial: El requisito tiene que ser implementado.
    - Deseable: Es preferible implementar el requisito, pero no es obligatorio.
    - Opcional: El requisito se podrá implementar, pero no es importante ni obligatorio.
- Prioridad: Define la importancia del requisito, de forma que permita definir el orden en el cual serán incluido en el proceso de diseño y el orden de implementación. Los valores que puede tomar este atributo son los siguientes:
    - Alta: El requisito debe ser implementado en las fases iniciales del desarrollo.
    - Media: El requisito debe ser implementado una vez que hayan sido implementados los requisitos de prioridad alta.
    - Baja: El requisito debe ser implementados en las fases finales del desarrollo. Estos requisitos no influirán en el correcto funcionamiento del sistema.
- Estabilidad: Define la estabilidad del requisitos durante la vida útil del software. Esto implica si el requisito podrá ser o no modificado durante el ciclo del vida. Los valores que puede tomar este atributo son los siguientes:
    - Estable: El requisito no puede variar durante el ciclo de vida del sistema.
    - Inestable: El requisito puede variar a lo largo del ciclo de vida del sistema.
- Verificabilidad: Define el grado de verificabilidad de un requisito, es decir indica en qué grado es posible comprobar que el requisito se ha incorporado en el sistema desarrollado. Los valores que puede tomar este atributo son los siguientes:
    - Alta: Se puede verificar que el requisito ha sido implementado en el sistema. Este tipo de requisitos se corresponden con las funcionalidades básicas del sistema.
    - Media: Se puede verificar que el requisito ha sido implementado en el sistema. Pero requiere de una comprobación compleja o del código fuente del sistema.
    - Baja: Es difícil verificar si el requisito ha sido implementado en el sistema o en algunos casos no es posible.

3.2.5 Especificación de requisitos

Requisito del sistema			
Código	**RF-001**	**Fuente**	CU-001
Nombre	Detectar usuario		
Descripción	El sistema deberá ser capaz de procesar la imagen de cualquier usuario que aparezca en la escena y así establecer su posición.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 10 - Requisito RF-001

Requisito del sistema			
Código	**RF-002**	**Fuente**	CU-001
Nombre	Reconocer usuario		
Descripción	El sistema deberá ser capaz de analizar al usuario detectado, para determinar si es el usuario buscado.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 11 - Requisito RF-002

Requisito del sistema			
Código	**RF-003**	**Fuente**	CU-003, CU-005
Nombre	Desplazamiento hacia delante		
Descripción	Cuando el usuario se aleje del robot, éste deberá ser capaz de desplazarse hacia delante.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 12 - Requisito RF-003

Requisito del sistema			
Código	**RF-004**	**Fuente**	CU-003, CU-005
Nombre	Desplazamiento hacia atrás		
Descripción	Cuando el usuario se acerque al robot, éste deberá ser capaz de desplazarse hacia atrás.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 13 – Requisito RF-004

Requisito del sistema			
Código	**RF-005**	**Fuente**	CU-003, CU-005
Nombre	Giro a la derecha		
Descripción	Cuando el usuario se mueva hacia la izquierda, el robot deberá rotar a la derecha.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 14 - Requisito RF-005

Requisito del sistema			
Código	**RF-006**	**Fuente**	CU-003, CU-005
Nombre	Giro a la izquierda		
Descripción	Cuando el usuario se mueva hacia la derecha, el robot deberá rotar a la izquierda.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 15 - Requisito RF-006

Requisito del sistema			
Código	**RF-007**	**Fuente**	CU-002, CU-004, CU-006
Nombre	Parada		
Descripción	El sistema deberá parar sin intervención del usuario cuando este desaparezca de la escena.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 16 - Requisito RF-007

Requisito del sistema			
Código	**RF-008**	**Fuente**	CU-001, CU-003
Nombre	Reencontrar usuario		
Descripción	El sistema deberá parar sin intervención del usuario cuando este desaparezca de la escena.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 17 - Requisito - RF-008

Requisito del sistema			
Código	**RF-009**	**Fuente**	CU-001
Nombre	Identificación de color		
Descripción	El sistema identificará al usuario al que debe seguir mediante el reconocimiento del color de la camiseta, previamente establecido.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inestable	**Verificabilidad**	Alta

Tabla 18 - Requisito RF-009

Requisito del sistema			
Código	**RF-010**	**Fuente**	CU-001
Nombre	Calibrar color		
Descripción	El sistema deberá permitir al usuario calibrar el color con el que se identificará el usuario.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Inestable	**Verificabilidad**	Alta

Tabla 19 - Requisito RF-010

Requisito del sistema			
Código	**RF-011**	**Fuente**	CU-007. CU-008
Nombre	Esquivar obstáculo		
Descripción	El robot deberá ser capaz de esquivar simples obstáculos, mediante el uso del sonar del robot.		
Necesidad	Esencial	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 20 - Requisito RF-011

Requisito del sistema			
Código	**RF-012**	**Fuente**	CU-003, CU-004
Nombre	Mostrar información textual		
Descripción	El sistema deberá mostrar por consola texto que indique el estado del seguimiento.		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 21 - Requisito RF-012

Requisito del sistema			
Código	**RF-013**	**Fuente**	CU-001, CU-003
Nombre	Mostrar esqueleto		
Descripción	El sistema deberá mostrar por pantalla el esqueleto de los diferentes usuarios presentes en la escena.		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 22 - Requisito RF-013

Requisito del sistema			
Código	**RF-014**	**Fuente**	CU-001
Nombre	Mostrar patrón de color		
Descripción	El sistema deberá mostrar por pantalla el patrón de identificación de color que se está realizando.		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 23 - Requisito RF-014

Requisito del sistema			
Código	**RNF-001**	**Fuente**	Entorno operacional
Nombre	Utilizar el robot P3DX		
Descripción	El robot utilizado en el desarrollo del trabajo será el Pioneer 3DX.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 24 - Requisito RNF-001

Requisito del sistema			
Código	**RNF-002**	**Fuente**	Entorno operacional
Nombre	Utilizar la Kinect		
Descripción	El robot utilizado en el desarrollo del trabajo será el Pioneer 3DX.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 25 - Requisito RNF-002

Requisito del sistema			
Código	**RNF-003**	**Fuente**	Restricciones del sistema
Nombre	Utilizar ROS		
Descripción	El sistema deberá utilizar el *framework* ROS		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 26 - Requisito RNF-003

Requisito del sistema			
Código	**RNF-004**	**Fuente**	Restricciones del sistema
Nombre	Restricción de velocidad		
Descripción	El sistema deberá limitar la velocidad alcanzada por el robot al seguir al usuario, para garantizar la integridad del robot.		
Necesidad	Opcional	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 27 - Requisito RNF-004

Requisito del sistema			
Código	**RNF-005**	**Fuente**	Restricciones del sistema
Nombre	Restricción distancia		
Descripción	El sistema deberá establecer una distancia mínima entre el robot y el usuario, para que el proceso de detección se realice correctamente.		
Necesidad	Opcional	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 28 - Requisito RNF-005

Requisito del sistema			
Código	**RNF-006**	**Fuente**	Restricciones del sistema
Nombre	Configuración sencilla		
Descripción	La configuración del sistema debe poder realizarse de forma rápida y sencilla para facilitar su uso a usuarios no experimentados con él.		
Necesidad	Opcional	**Prioridad**	Baja
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 29 - Requisito RNF-006

3.3 Diseño del sistema

En este apartado se presenta el diseño arquitectónico del sistema a desarrollar, a partir del análisis presentado en el apartado anterior. En primer lugar se describirá la arquitectura del sistema, seguidamente se presentará la descripción general del sistema, realizando una descripción de las etapas por la que pasa el sistema. Por último, se presenta el diseño detallado del sistema, describiendo todos aquellos elementos que intervienen en él.

3.3.1 Arquitectura del sistema

El sistema se ha diseñado de forma que existen cuatro módulos:

- Módulo Kinect: Se encarga de adaptar el dispositivo Kinect a ROS.
- Módulo de reconocimiento: Este módulo se corresponde con la función de reconocer personas e identificarlas.
- Módulo de seguimiento: A través de este módulo el sistema será capaz de seguir a personas una vez hayan sido identificadas.
- Módulo P3DX: Se encarga de controlar el robot a través de ROS.

Cada uno de estos módulos se divide en una serie de nodos que intercambian información entre ellos mediante mensajes. En la Figura 19 se muestra la arquitectura general diseñada.

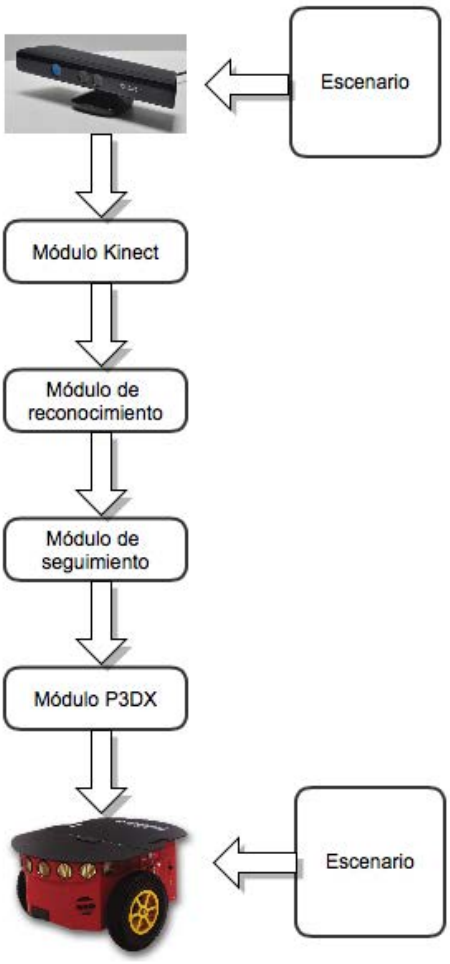

Figura 19 - Arquitectura del sistema

La entrada del sistema se realiza a través de la Kinect que capta la información de la escena. El módulo de la Kinect se encarga de controlar la Kinect y procesar la información obtenida a través del dispositivo. El módulo de reconocimiento se encarga de recoger la información obtenida por el procesamiento de la Kinect y realizar el reconocimiento del usuario. Este reconocimiento se dividirá en tres fases; detección, localización e identificación. Una vez se ha reconocido al usuario, el módulo de reconocimiento se comunica con el módulo de seguimiento. El módulo de seguimiento, obtiene la información necesaria para ubicar al usuario a través del módulo de reconocimiento. Una vez ubicado al usuario, se moverá hacia él a través del escenario, evitando posibles obstáculos que puedan aparecer. El módulo del robot P3DX, se encarga de procesar las peticiones de movimiento del módulo de seguimiento de manera que sea entendible por el robot y facilitar la información obtenida a través del sónar y la odometría del robot. La comunicación

entre los seis elementos que componen la arquitectura del sistema se realiza a través de una serie de mensajes. Estos mensajes tienen diferentes estructuras de datos dependiendo de la información que transportan.

A continuación se muestran los esquemas de la descomposición de cada uno de los módulos de la arquitectura en los diferentes nodos que intervienen en ella. Las comunicación entre los nodos aparece en forma de línea continua, mientras que las dependencias entre nodos aparecen en línea discontinua. En primer lugar, se muestra el diagrama del modulo de Kinect, Figura 20.

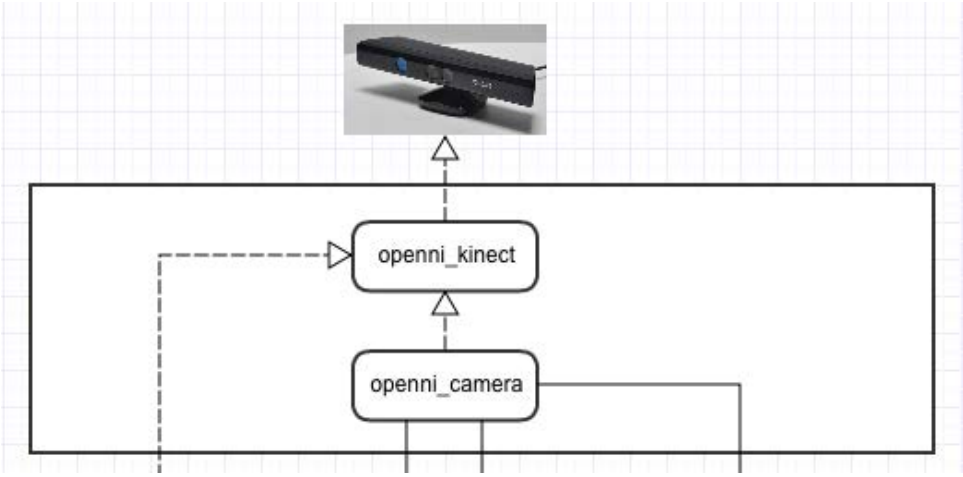

Figura 20 - Arquitectura del módulo Kinect

Los nodos que intervienen en este módulo son los siguientes:

- *Openni_kinect*: nodo correspondiente a *Openni,* un *framework* multiplataforma que define *APIs* para desarrollar aplicaciones utilizando Interacción Natural. En el anexo 4 se hace una descripción de dicho *framework*.
- *openni_camera*: Este nodo realiza la función de driver de la Kinect para ROS, de modo que publica los diferentes formatos de imagen generados por este dispositivo.

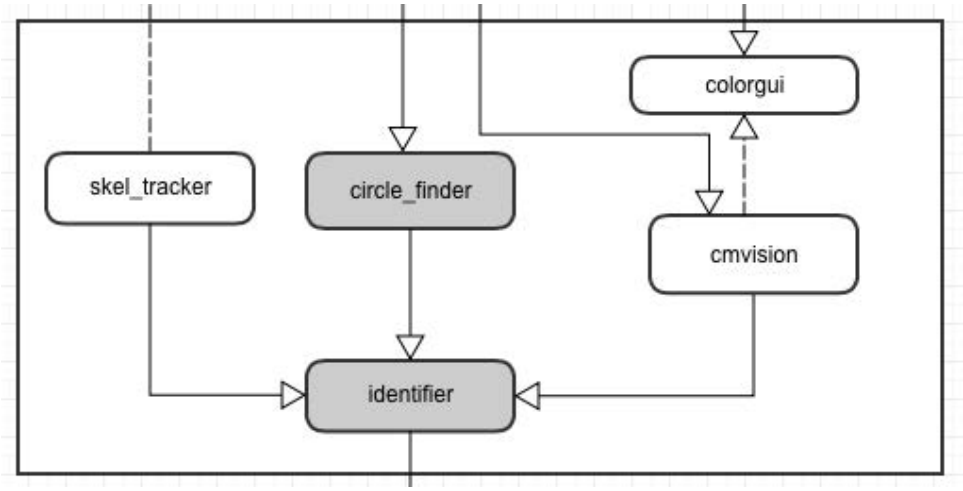

Figura 21 - Arquitectura del módulo de reconocimiento

A continuación se describen brevemente los nodos que componen el módulo de reconocimiento (Figura 21):

- *skel_tracker:* se encarga de captar la información obtenida a través de la Kinect y realizar una calibración de los usuarios que aparecen en la escena para así obtener su posición en la escena. La información sobre los esqueletos de los usuarios es publicada en una serie de mensajes para que pueda ser accedida desde otros nodos.
- *cmvision*: Este nodo obtiene la información recogida del nodo anterior para detectar los *blobs* correspondientes al color que se quiere identificar. Los *blobs* obtenidos son publicados en un mensaje para su uso en otros nodos.
- *circle_finder*: Este nodo se encarga de reconocer la marca con el círculo dibujado que llevará el usuario, para así identificarlo de entre los demás usuarios de la escena. La posición del círculo reconocido será publicada en un mensaje.
- *identifier*: Este nodo recibirá tanto la información de los usuarios detectados por el nodo *skel_tracker* como los *blobs* detectados por el nodo *cmvision,* así como el círculo reconocido por el nodo *circle_finder*. De forma que, tendrá la función de determinar qué usuario es el que tiene que seguir.

Por otra parte, la arquitectura detallada del módulo de seguimiento aparece en la siguiente figura.

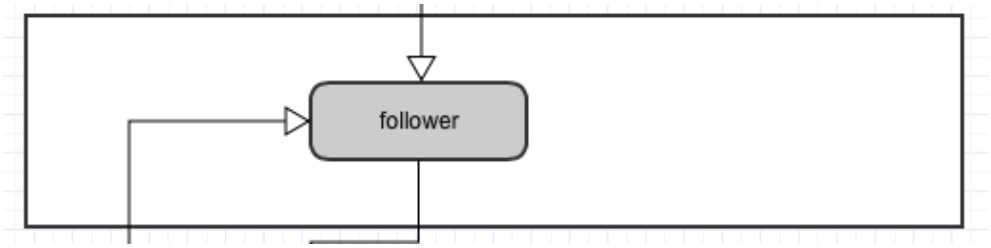

Figura 22 - Arquitectura del módulo de seguimiento

El único nodo que interviene en este módulo es el siguiente:

- *follower*: Este nodo recibe la información del usuario a través del nodo *indentifier* para determinar la variación en la posición del usuario y a partir de allí, establecer la velocidad a la que se deberá mover el P3DX. También recibirá información proveniente del sónar del robot para determinar si existe algún obstáculo que le impida seguir al usuario y de la odometría, para determinar la ubicación del robot en el espacio. Una vez calculada la velocidad a seguir, será publicada en un mensaje destinado al siguiente nodo.

Por último, en la Figura 23 se plasma la arquitectura del módulo del robot:

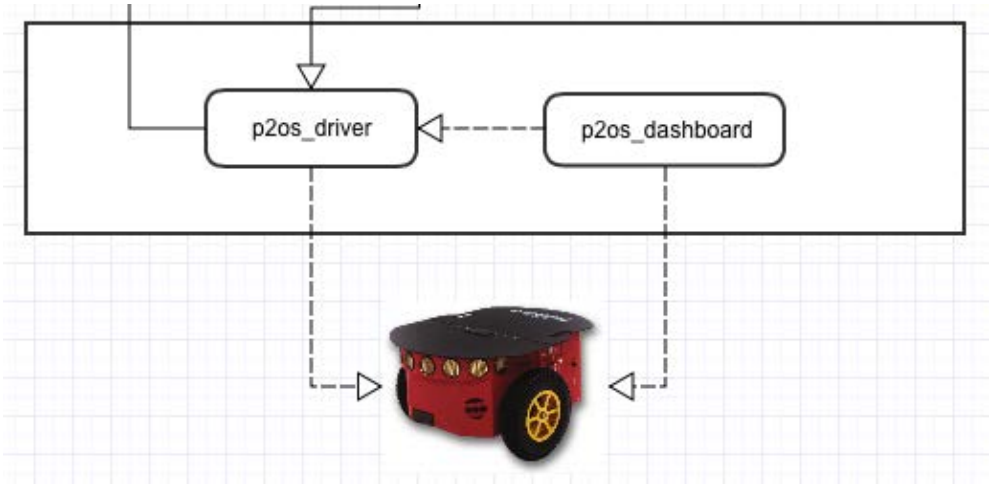

Figura 23 - Arquitectura del módulo P3DX

- *p2os_driver*: Como su nombre indica, este nodo es el driver del P3DX para ROS. La funciones principales de este nodo son recibir la velocidad a la que debe moverse el robot para que ejecute el desplazamiento y publicar las información recogida mediante el sónar y la odometría.

- *p2os_dashboard:* Este nodo muestra una interfaz gráfica para controlar y comprobar el estado del P3DX. Será necesario ejecutarlo para activar el motor del robot.

3.3.2 Descripción general del sistema

Una vez identificados los diferentes elementos de la arquitectura del sistema se va a proceder a describir de forma general las diferentes etapas por las que debe pasar el sistema, a partir del diagrama de flujo mostrado en la Figura 24. Este diagrama se corresponde con el proceso de ejecución del sistema para el modelo de reconocimiento mixto, ya que es el más completo. En caso del modelo de reconocimiento de color, se eliminaría el paso número 4, mientras que para el modelo de reconocimiento de forma, se eliminarían los pasos número 1 y 3.

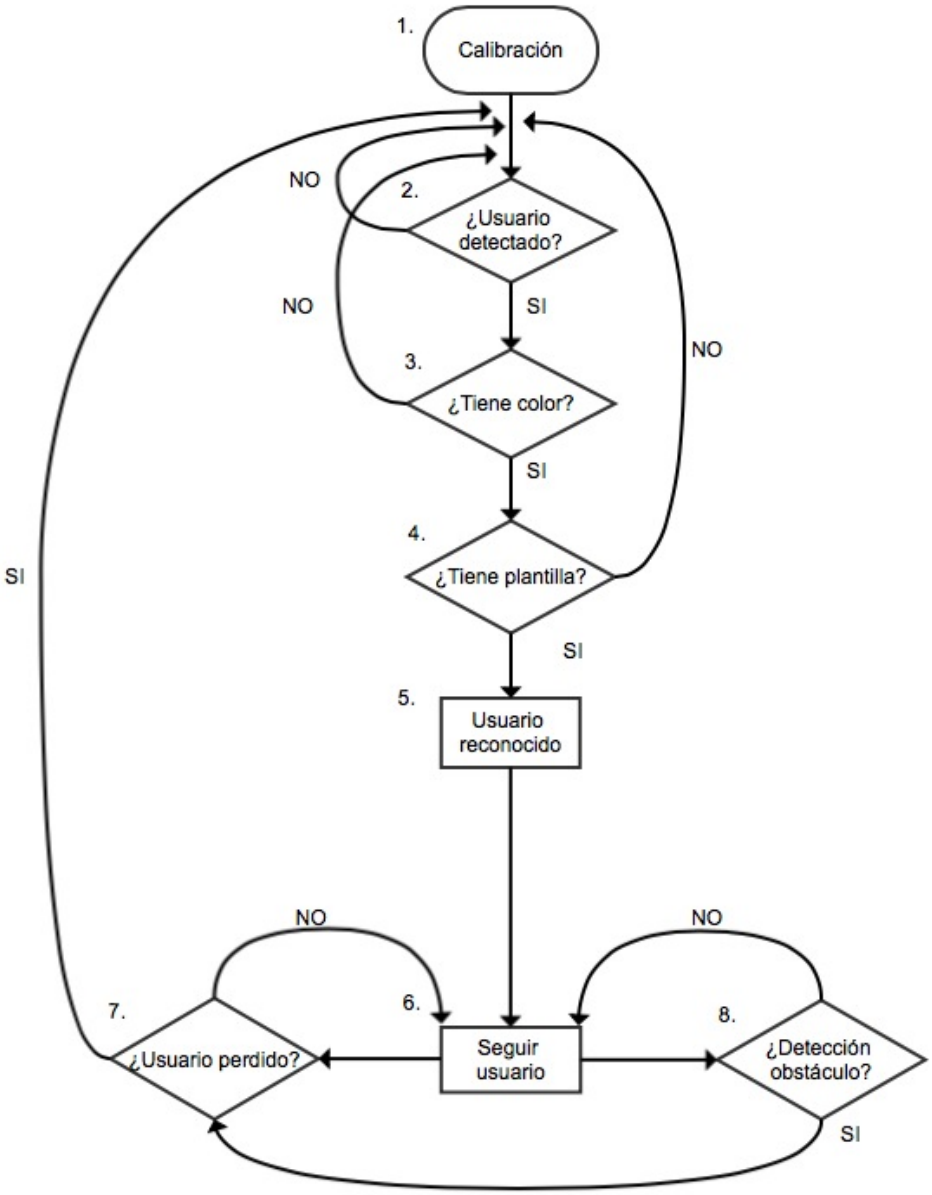

Figura 24 - Diagrama de flujo del sistema

1. En primer lugar, se realiza una calibración del color de la camiseta del individuo a seguir. De esta forma, dicho color será el único reconocido por el sistema. Esta calibración se realiza de manera manual.
2. Una vez se ha realizado la calibración se inicia el sistema. Tras esto, se realiza una búsqueda de los posibles humanos que se encuentren en su rango de visión. En este momento pueden darse dos situaciones:

   a. No se detecta ningún individuo en la escena. El sistema continúa esperando a que aparezca algún usuario sobre el que realizar el proceso de reconocimiento.
   b. Si uno o más usuarios son detectados en la escena, el sistema comienza la fase de reconocimiento del usuario al que debe seguir.

3. En este momento, el sistema tiene que determinar si alguno de los individuos reconocidos cuadra con el color calibrado inicialmente. Las dos alternativas serán:
   a. Ninguno de los usuarios de la escena es identificado según las características definidas en el proceso de calibrado. Se vuelve a la fase de detección de usuarios para comprobar si aparece algún usuario nuevo.
   b. Si un usuario es identificado, el sistema pasará a la siguiente fase de reconocimiento.
4. En esta fase se determina si el usuario seleccionado en la fase anterior posee la marca de control.
   a. En caso negativo, el sistema vuelve a la fase de detección de nuevos usuarios en la escena.
   b. En caso afirmativo, el usuario habrá sido identificado por el sistema y estará listo para ser seguido.
5. En este momento el usuario está reconocido por el sistema, que está preparado para seguirle.
6. Esta es la fase de control dinámico mediante la cual el sistema persigue al usuario variando de forma reactiva su velocidad y trayectoria según las condiciones del entorno y la velocidad del usuario que tiene asignado. Esta fase se interrumpirá en caso de que se produzcan alguna de las siguientes situaciones.
7. Mientras el sistema está siguiendo al usuario, puede ocurrir que se pierda al usuario a causa de varios factores como que el usuario vaya demasiado rápido, que haya cambios en la iluminación, … En ese caso, el sistema volverá a la fase inicial de detección de individuos para intentar volver a reconocer al usuario.
8. El otro acontecimiento que puede ocurrir mientras el sistema está persiguiendo al usuario es que detecte un obstáculo que se interponga en su trayectoria. En este caso, se iniciará el proceso de esquive del obstáculo, por lo que el sistema perderá de vista momentáneamente al usuario.

3.3.3 Descripción de componentes

En este apartado se describe de forma detalla la arquitectura presentada en el apartado anterior, explicando de forma detallada cada uno de los nodos y mensajes que intervienen en dicha arquitectura.

3.3.3.1 Módulo de Kinect

A continuación se detalla el funcionamiento y las características de los nodos que forman el módulo de la Kinect de la arquitectura diseñada. La Figura 25 muestra los nodos que intervienen en el funcionamiento de este módulo.

3.3.3.1.1 Openni_camera

Este nodo, contenido en el paquete *openni_kinect* de ROS, realiza la función de controlador para los sensores de profundidad (+ RGB) disponibles en OpenNI, entre los que se encuentra el sensor Microsoft Kinect. De esta forma se crea la conexión entre el sensor y ROS, permitiendo su adaptación al sistema de comunicaciones (mensajes, servicios, etc.) de ROS.

Como se observa en la Figura 25, la funcionalidad de este nodo es captar la información a través de la Kinect y publicarla mediante una serie de mensajes. En concreto, publica la información relacionada con las imágenes de profundidad, RGB e IR captadas a través del sensor.

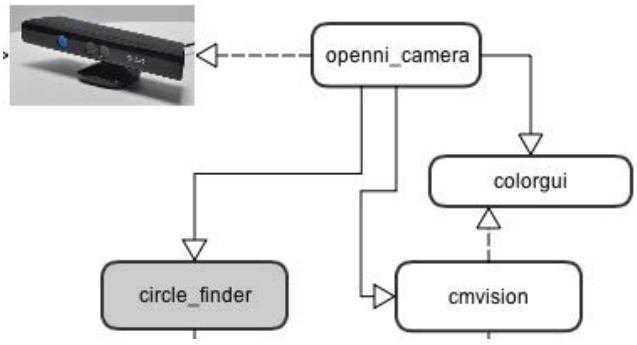

Figura 25 - Esquema openni_camera

Para el desarrollo del trabajo, sólo se utiliza la imagen RGB, por lo que la función de este nodo es extraer la imagen RGB para su posterior tratamiento en otro nodo. Para ello, dicho nodo deberá suscribirse al mensaje *camera/rgb/image_color* publicado por *openni_camera*. Este mensaje transporta la información sobre la imagen RGB captada por la Kinect. En la Figura 26 se puede observar la imagen RGB conseguida.

Figura 26 - Imagen RGB de la Kinect

El mensaje *camera/rgb/image_color* es del tipo *sensor_msgs/Image* cuyo contenido se describe en la siguiente tabla.

M-1	
Mensaje	camera/rgb/image_color
Tipo de mensaje	sensor_msgs/Image
Contenido del mensaje	
Header header	Encabezamiento con información temporal.
uint32 height	Altura de la imagen (número de filas).
uint32 width	Anchura de la imagen (número de columnas).
string encoding	Codificación de los píxeles.
uint8 is_bigendian	Determina si los datos están o no en big-endian.
uint32 step	Tamaño de una fila en bytes
uint8[] data	Matriz de datos actual

Tabla 30 - Mensaje M-1

3.3.3.2 Módulo de reconocimiento

A continuación se describen detalladamente los nodos que intervienen en el módulo de reconocimiento tal y como se observa en la Figura 27.

3.3.3.2.1 Skel_tracker

Este nodo pertenece a un conjunto de paquetes desarrollados por el MIT para el uso de la Kinect en ROS. Se basa en la utilización de la API de OpenNI para realizar el reconocimiento de usuarios mediante la Kinect.

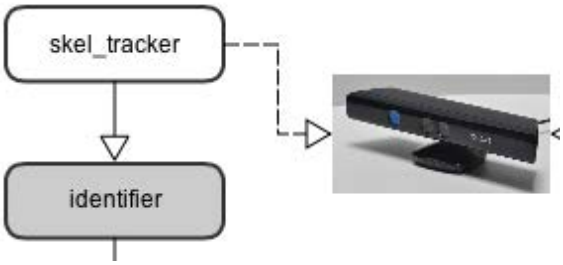

Figura 27 - Esquema skel_tracker

El objetivo de este nodo es reconocer a los usuarios que entren en la escena. Para esto, el nodo realiza las siguiente funciones:

- Detectar cuando un individuo aparece en la escena captada, iniciar su calibración y asignarle un identificador de usuario. En versiones anteriores a la utilizada en este trabajo, era necesario realizar la pose Psi para realizar la calibración del usuario. En la Figura 28 se observa como al detectar un individuo en la escena, lo identifica con un número y un color específico.

Figura 28. Calibrando usuario

- Asignar al usuario un esqueleto formado por la unión de diferentes segmentos (Figura 29), para ubicarle dentro de la escena. Estos segmentos son la unión de los puntos de referencia (*joints*) que se toman del usuario. Estos puntos serán publicados mediante los mensajes que se describirán a continuación.

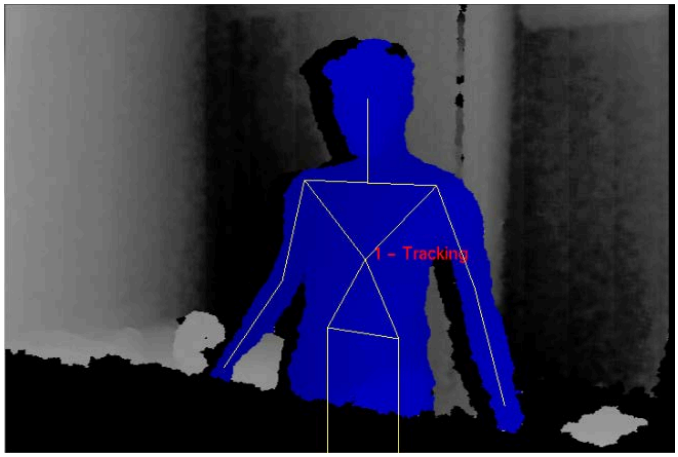

Figura 29 - Esqueleto del usuario reconocido

Debido a la incompatibilidad de la versión de ROS utilizada con el paquete de mensajes *body_msgs* que utiliza *skel_tracker* para publicar los esqueletos, se ha realizado una modificación en *skel_tracker*. Dicha modificación consiste en la eliminación de la dependencia de este nodo con *body_msgs*, de modo que se han tenido que crear el tipo de mensajes de *body_msgs* en el propio nodo *skel_tracker*. De esta forma, dichos mensajes pasarán a ser de tipo *skel_tracker*.

Para la publicación del esqueleto del usuario, se utiliza una jerarquía de tres mensajes. Estos mensajes son los siguientes:

- skel_tracker/skeletonJoint: Este es el mensaje de más bajo nivel de la jerarquía, se encarga de almacenar un punto correspondiente a una articulación (*joint*) de un individuo.
- skel_tracker/skeleton: Este mensaje, contiene 15 mensajes del tipo anterior, de modo que almacena cada uno de los puntos que permiten formar un esqueleto.
- skel_tracker/skeletons: Este es el mensaje de más alto nivel de la jerarquía, contiene tantos mensajes del tipo anterior como usuarios se hayan detectado en la escena. Es decir, contiene los esqueletos de todos los usuarios visibles.

El mensaje principal, que es el que se publicará, será skel_tracker/skeletons que contiene los esqueletos de todos los usuarios detectados en la escena. Su contenido se detalla en la siguiente tabla:

M-2	
Mensaje	/skeletons
Tipo de mensaje	skel_tracker/skeletons
Contenido del mensaje	
Header header	Encabezamiento con información temporal.
skel_tracker/skeleton[] skeletons	Contiene cada uno de los esqueletos de los usuarios detectados en la escena.

Tabla 31- Mensaje M-2

Como se observa, los datos relevantes del mensaje pertenecen a otro tipo de mensaje que captura el esqueleto a un nivel más bajo que el anterior. A continuación se muestra el contenido del mensaje *skel_tracker/skeleton*:

M-3	
Mensaje	/skeleton
Tipo de mensaje	skel_tracker/skeleton
Contenido del mensaje	
Int32 playerid	Identificador del usuario al que pertenece el esqueleto.
skel_tracker/skeletonJoint head	Información de la cabeza.
skel_tracker/skeletonJoint neck	Información del cuello.
skel_tracker/skeletonJoint right_hand	Información de la mano derecha.
skel_tracker/skeletonJoint left_hand	Información de la mano izquierda.
skel_tracker/skeletonJoint right_shoulder	Información del hombro derecho.

M-3	
skel_tracker/skeletonJoint left_shoulder	Información del hombro izquierdo.
skel_tracker/skeletonJoint right_elbow	Información del codo derecho.
skel_tracker/skeletonJoint left_elbow	Información del codo izquierdo.
skel_tracker/skeletonJoint torso	Información del torso.
skel_tracker/skeletonJoint left_hip	Información de la cadera izquierda.
skel_tracker/skeletonJoint right_hip	Información de la cadera derecha.
skel_tracker/skeletonJoint left_knee	Información de la rodilla izquierda.
skel_tracker/skeletonJoint right_knee	Información de la rodilla derecha.
skel_tracker/skeletonJoint left_foot	Información del pie izquierdo.
skel_tracker/skeletonJoint right_foot	Información del pie derecho.
skel_tracker/pixel torso_pixel	Coordenadas del torso en píxeles.

Tabla 32 – Mensaje M-3

En la Figura 30 se observa el esqueleto representado por cada uno de los puntos que componen el mensaje M-3.

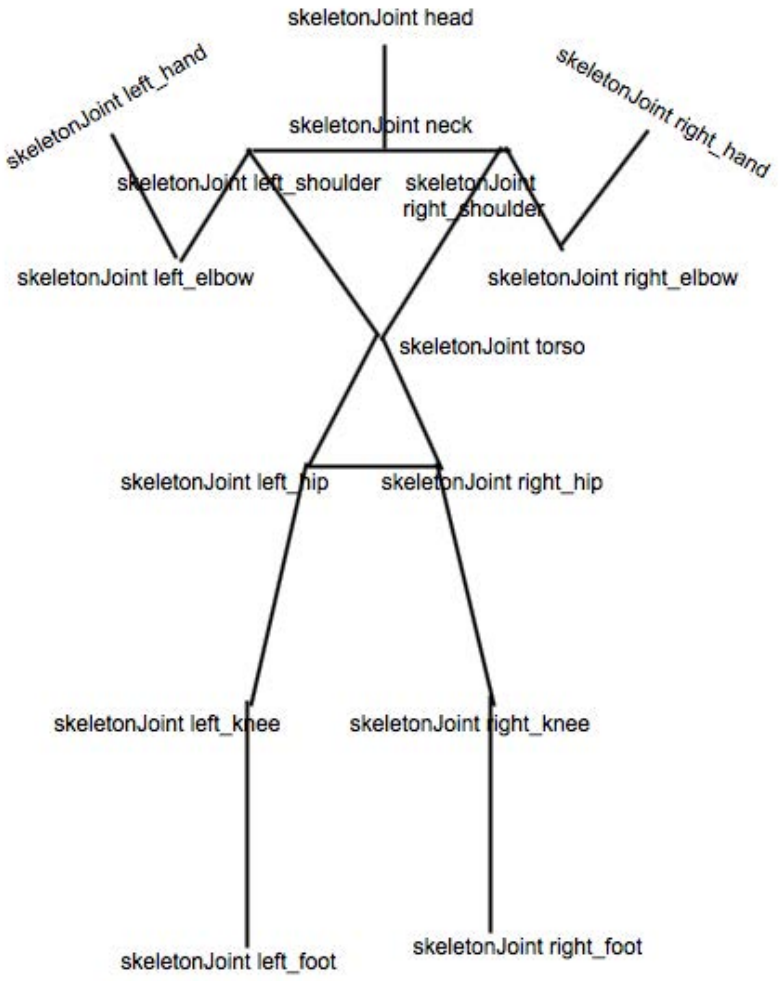

Figura 30 - Representación gráfica del mensaje M-3

Observamos que este mensaje está compuesto por cada una de las articulaciones que componen el esqueleto de un usuario, definidas en otro tipo de mensaje de la jerarquía de esqueletos llamado skel_tracker/skeletonJoint, el cual se describe en la siguiente tabla:

M-4	
Mensaje	/skeletonJoint
Tipo de mensaje	skel_tracker/skeletonJoint
Contenido del mensaje	
geometry_msgs/Point position	Punto del espacio tridimensional donde se encuentra la articulación
Float32 confidence	Indica el grado de confianza con el que se ha detectado la articulación. Su valor oscilará entres 0 en caso de que no se detecte, y 1 si se detecta perfectamente

Tabla 33 – Mensaje M-4

Este último mensaje contiene la posición de la articulación que representa mediante un mensaje tipo *Point*, que esta compuesto por tres componente, X, Y y Z.

A parte de la modificación de los mensajes mencionada anteriormente, también se han realizado dos pequeñas modificaciones funcionales descritas a continuación.

- Como se ha comentado, la posición de las articulaciones viene definida como un punto en el espacio tridimensional. Dado que, como veremos más adelante, también es necesario obtener la posición

Otra modificación que se ha realizado sobre este nodo es la inclusión de un función para calcular la posición del torso del usuario en píxeles. Esto se realiza para establecer un sistema de referencia en común con los nodos *cmvision* y *circle_finder* que transmiten sus resultados en píxeles. En la descripción del nodo *identifier* se detallará el proceso. A parte de incluir la función, también se ha creado un nuevo mensaje llamado pixel, compuesto de dos atributos; la coordenada *x* del píxel y la coordenada *y*. Para no tenerlo que publicar en otro *topic* diferente, el mensaje se ha añadido como atributo del mensaje *skel_tracker/skeleton*, de este modo cuando el nodo *skel_tracker* publique la información de los esqueletos de los usuarios detectados, también incluirá las coordenadas del píxel correspondiente a la articulación *torso* del esqueleto.

En resumen, la información que nos interesa extraer de este nodo es la ubicación del usuario publicada en los mensajes *skel_tracker* creados. Como se ha descrito, la jerarquía de los tres mensajes de este nodo establece que cada esqueleto está compuesto por 15 articulaciones, que están representadas por un vector tridimensional. El mensaje que se publicará será *skel_tracker/skeletons*, que contiene cada uno de los esqueletos de los usuarios que han sido reconocidos en la escena.

3.3.3.2.2 Colorgui

Este nodo pertenece al paquete *cmvision* de ROS. Este paquete añade a ROS la posibilidad de utilizar la librería CMVision (Color Machine Vision). Dicha librería permite la detección de *blobs* o regiones de una imagen que poseen características diferentes al resto de la escena. En nuestro caso se utiliza esta librería para detectar cierto color en la escena.

En cuanto al nodo *colorgui*, para ejecutarlo será necesario publicar un mensaje del tipo *sensor_msgs/Image*, ya que el nodo está suscrito a este tipo de mensajes. En nuestro caso el mensaje se lo proporcionaremos mediante el nodo *openni_camera* a través de la publicación del mensaje *camera/rgb/image_color*. Para realizar esta comunicación con éxito, deberemos remapear el nombre del *topic* al que esta suscrito para sustituirlo por el *topic camera/rgb/image_color*.

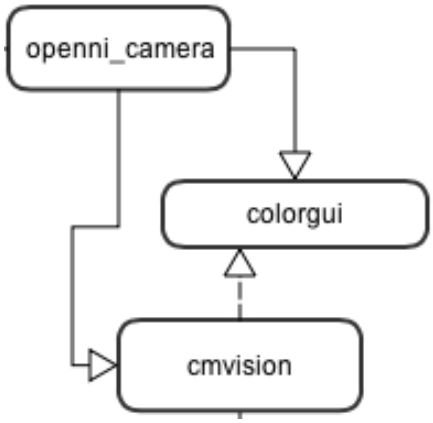

Figura 31 - Esquema colorgui

Una vez ejecutamos el nodo, se presenta una interfaz gráfica donde se muestra el flujo de imágenes obtenido de la cámara RGB de la Kinect. Esta interfaz permite al usuario seleccionar el

color que se quiere detectar en la imagen. Para ello, deberemos hacer *click* izquierdo sobre el color deseado y la interfaz mostrará el valor del color en el espacio de color RGB y los umbrales en el espacio de color YUV. El siguiente paso será guardar los valores RGB y YUV resultantes en un fichero de texto con el siguiente formato:

[colors]

(159, 62, 71) 0.000000 7 Identificador

[thresholds]

(56:99, 108:117, 173:193)

Cabe destacar que será necesario realizar la selección de color en un ambiente en el que la luminosidad sea constante, ya que sino, se pueden tener problemas con el reconocimiento posterior del color seleccionado.

3.3.3.2.3 Cmvision

Este nodo, al igual que el descrito anteriormente, también pertenece al paquete *cmvision* de ROS. Su función es detectar unos blobs determinados en un flujo de imágenes. Para ello, leerá los datos contenidos en el fichero de texto creado mediante el nodo *colorgui* y se buscará en el flujo de imágenes dicho color mediante la asignación de *blobs*.

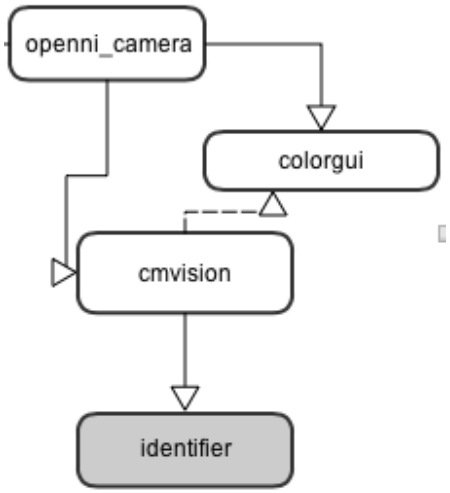

Figura 32 - Esquema cmvision

Para realizar la identificación del color deseado, el nodo deberá recibir a través de un mensaje tipo *sensor_msgs/Image* el flujo de imágenes donde deberá encontrarlo. Por lo que el nodo dependerá de que el nodo *openni_camera* esté publicando mensajes. Al igual que el nodo anterior también será necesario remapear el nombre del *topic* al que está suscrito.

Figura 33 - Identificación de blobs

Una vez reconocido el color, el nodo irá publicando un mensaje de tipo *cmvision/Blobs* con los diferentes *blobs* de dicho color que está encontrando (Figura 33). Este mensaje será publicado en el *topic blobs* y tendrá el siguiente contenido:

M-5	
Mensaje	/Blobs
Tipo de mensaje	cmvision/Blobs
Contenido del mensaje	
Header header	Encabezamiento con información temporal.
uint32 image_width	Anchura de la imagen (número de columnas).
uint32 image_height	Altura de la imagen (número de filas).
uint32 blob_count	Indica el número de blobs identificados.
Blob[] blobs	Información de cada uno de los blobs identificados.

Tabla 34 – Mensaje M-5

Dado que la información que nos interesará en nodos siguientes está en el atributo "blobs" que es de tipo de mensaje *cmvision/Blob*, en la Tabla 35 se descompone dicho mensaje.

M-6	
Mensaje	/Blob
Tipo de mensaje	cmvision/Blob
Contenido del mensaje	
uint32 red	Indica el valor de la componente roja del espacio de color RGB.
uint32 green	Indica el valor de la componente verde del espacio de color RGB.
uint32 blue	Indica el valor de la componente azul del espacio de color RGB.

M-6	
uint32 area	Indica el área comprendida por el blob.
uint32 x	Indica la posición horizontal del blob en la imagen, tomando su centroide como referencia.
uint32 y	Indica la posición vertical del blob en la imagen, tomando su centroide como referencia.
uint32 left	Indica la posición en píxeles del límite del blob por la izquierda.
uint32 right	Indica la posición en píxeles del límite del blob por la derecha.
uint32 top	Indica la posición en píxeles del límite del blob por arriba.
uint32 bottom	Indica la posición en píxeles del límite del blob por abajo.

Tabla 35 – Mensaje M-6

Un detalle a tener en cuenta es que en el mensaje *cmvision/Blobs* los *blobs* son ordenados dependiendo de su área, por lo que el *blob* almacenado en el atributo Blob[0] será el mayor de los reconocidos.

3.3.3.2.4 Circle_finder

La función de este nodo es reconocer los círculos presentes en la escena. Para realizar dicha función, se ha utilizado la librería de procesamiento de imágenes OpenCV (*Open Source Computer Vision*). Esta librería ofrece una combinación entre algoritmos de visión artificial y aprendizaje automático. Algunos ejemplos de lo que permiten hacer estos algoritmos son el reconocimiento de caras, la identificación de objetos, seguimiento de objetos, etc.

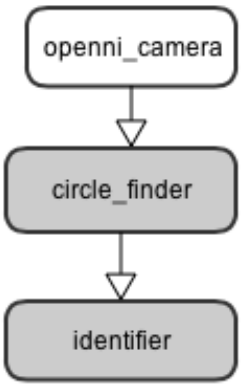

Figura 34 - Esquema circle_finder

OpenCV está disponible en ROS a través del *stack vision_opencv*. Este *stack* permite la transformación del tipo de imágenes procesadas por ROS al tipo de imágenes utilizadas en OpenCV. En concreto, lo hace a través del paquete *cv_bridge*, que convierte la imágenes de tipo *sensor_msgs/Image* (formato ROS) en imágenes de tipo *cv::Mat* (formato OpenCV), para a partir de allí utilizar las funcionalidades que ofrece OpenCV sobre la imagen. En la Figura 35 se muestra dicha conversión.

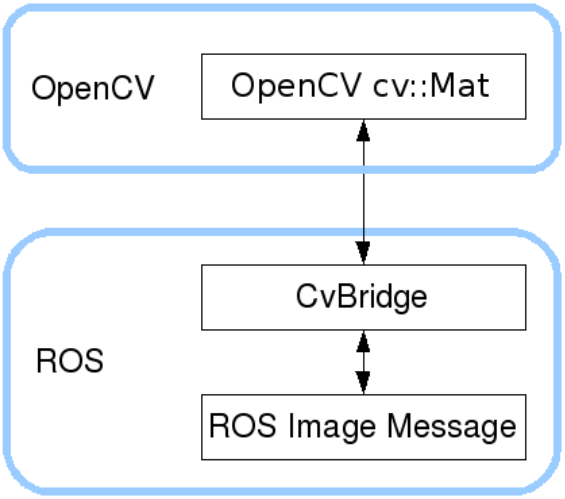

Figura 35 - Conversión imágenes

El funcionamiento del nodo *circle_finder* se divide en las siguientes etapas:

1. En primer lugar, el nodo está suscrito al *topic /camera/rgb/image_color* donde la Kinect publica la imagen captada por su cámara RGB.

2. Una vez obtenida la imagen de tipo *sensor_msgs/Image*, se convierte a formato *cv::Mat* mediante el uso de la funcionalidad que ofrece *cv_bridge*. Con esto, ya tendremos la imagen recibida en formato OpenCV y podremos utilizar las funciones que ofrece para el procesamiento de imágenes.
3. Para trabajar con reconocimiento de formas es preferible hacerlo mediante la imagen en escala de grises, por lo que se utiliza la función *cvtColor()* para transformar nuestra imagen en formato RAW.
4. Una vez tenemos la imagen en escala de grises, aplicamos la función cv::HoughCircles() para encontrar los círculos presentes en la escena y guardarlos en un vector de tres posiciones donde se guardarán las coordenadas de su punto medio en píxeles y su radio.
5. Tras esto, se muestra la ventana con la imagen recibida, donde se dibujan los círculos encontrados como muestra la Figura 36.
6. El último paso, será publicar el centro del mayor círculo detectado en un mensaje de tipo skel_tracker/pixel.

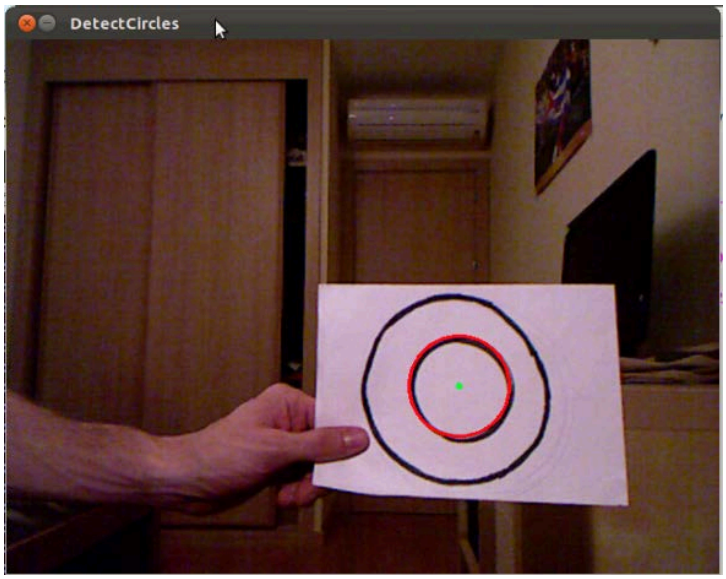

Figura 36 - Detección de círculos mediante OpenCV

3.3.3.2.5 Identifier

Este nodo realiza la función de identificar al usuario que debe seguir de entre todos los posibles individuos que aparezcan en la escena. Se han implementado tres maneras distintas de identificar al usuario. Esto se ha hecho para garantizar que el usuario sea identificado correctamente, ya que al realizarse los experimentos en un entorno en el que la luz no está controlada, puede aparecer ruido sobretodo a la hora de reconocer el color. Posteriormente, en la evaluación del sistema se analizarán los resultados obtenidos al realizar los experimentos con cada uno de los sistemas. A continuación se describen cada una de las tres implementaciones.

- Sistema de identificación de color: Este primer sistema realiza la identificación del usuario a partir del color de la camiseta, que debe haber sido calibrado previamente. Por lo que un usuario únicamente será identificado si está reconocido en la escena y se ha reconocido el patrón de color .
- Sistema de identificación de forma: Este sistema identifica al usuario mediante el reconocimiento de una plantilla con un círculo dibujado, que deberá llevar el usuario en todo momento. Un individuo será identificado como el usuario a seguir si está reconocido en la escena y se reconoce la plantilla con el círculo. En un principio se diseñó para reconocer triángulos, pero al ser una figura más compleja el reconocimiento no se realizaba correctamente.
- Sistema de identificación mixto: Este último sistema es una combinación de los dos anteriores, por tanto el usuario será identificado únicamente si su camiseta es del color calibrado y lleva la plantilla; si el sistema no reconoce que lleve ambos elementos no le seguirá.

Para realizar la descripción del nodo *identifier,* se va a tener en cuenta el tercer sistema (Figura 37), ya que los otros dos están contenidos en él.

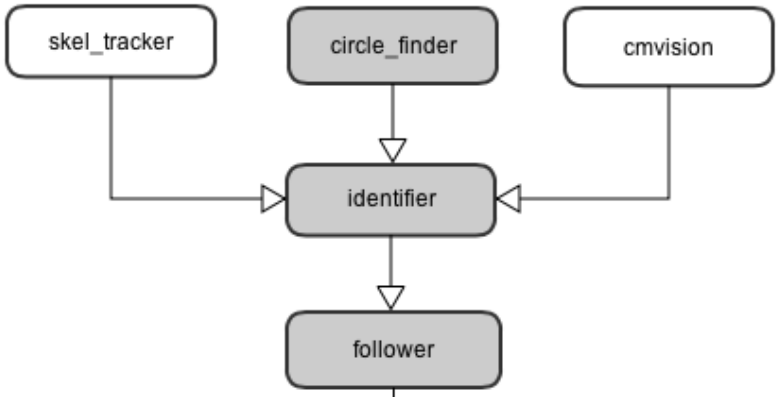

Figura 37 - Esquema identifier

Para realizar su función, el nodo deberá obtener la información de los nodos *skel_tracker*, *cmvision* y *circle_detection* como muestra la Figura 37. Por lo que recibirá los esqueletos de los diferentes usuarios reconocidos , los *blobs* que se estén detectando y el círculo que se esté reconociendo. A continuación se describe el proceso seguido por este nodo para realizar la identificación del usuario.

1. Con la información obtenida, el nodo deberá determinar cuál de los usuarios presentes en la escena es el que lleva la camiseta y la plantilla. Para ello deberá obtener la articulación correspondiente al torso de cada uno de los usuarios, procesando el mensaje recibido del nodo *skel_tracker*. Esto se hará obteniendo la posición en píxeles del torso de los usuarios. Se tomará la posición en píxeles y no la correspondiente a la posición respecto a la Kinect, ya que tanto el centro de los *blobs* como el del círculo vienen dados en píxeles, de esta forma los tres elementos estarán en la misma unidad de medida.
2. Una vez se ha obtenido, deberá comprobar si el mensaje */blobs* contiene algún *blob*, cosa que indicará que se ha detectado el color de la camiseta previamente calibrado. Si los hay, deberá obtener el *blob* más grande, que será el que está en la primera posición del *array* de *blobs*, ya que recordemos que dicho *array* se ordena por el área del *blob*.
3. Una vez tenemos el punto del torso de todos los usuarios y el punto medio del área del mayor *blob*, calculamos la distancia entre los dos puntos y obtenemos la menor de entre todos los candidatos.
4. Se repetirá el mismo proceso del cálculo de distancias entre los usuarios y el círculo obtenido, calculando la distancia entre los puntos referentes a los torsos de los individuos en la escena y el centro del círculo y tomando la menor de esas distancias.

5. Por último, una vez sabemos cuál es el usuario más cercano al mayor blob y cuál es el usuario más cercano al círculo, comparamos si esos usuarios son el mismo. En ese caso, publicamos el torso (con los valores de distancia respecto de la Kinect y no en píxeles) del usuario identificado a través de un mensaje de tipo *geometry_msgs/Point*, que como hemos visto antes se corresponde con un punto en el espacio 3D. En caso de que no coincidan no se publicará nada, por lo que el robot no se moverá. Únicamente se publicará el torso y no todo el esqueleto porque es el punto de referencia que se toma para seguir al usuario.

En resumen, el nodo está diseñado para que el robot únicamente siga a un usuario, siempre y cuando ese usuario tenga la plantilla y el color establecido. Si tiene sólo uno de los dos elementos, el mensaje no se publicará, por lo que el robot no se moverá.

3.3.3.3 Módulo de seguimiento

A continuación se detalla el funcionamiento y las características de los nodos que forman el módulo de seguimiento de la arquitectura diseñada. La Figura 22 muestra los nodos que intervienen en el funcionamiento del módulo de seguimiento.

3.3.3.3.1 Follower

Este nodo tiene la función de establecer la velocidad a la que se debe mover el robot P3DX, así como la de evadir de obstáculos. Para ello deberá publicar la velocidad en el mensaje *geometry_msgs/Twist*, al que está suscrito el nodo *p2os_driver* a través del *topic /cmd_vel*. Este mensaje está compuesto por los siguientes atributos:

M-7	
Mensaje	/Twist
Tipo de mensaje	geometry_msgs/Twist
Contenido del mensaje	
Vector3 linear	Componente lineal de la velocidad.
Vector3 angular	Componente angular de la velocidad.

Tabla 36 – Mensaje M-7

Como se ve en la tabla anterior, la velocidad se divide en dos componentes, que a su vez se dividen en las tres componentes que lo representan como un vector en el espacio (*x, y, z*). El objetivo del nodo *follower* es rellenar este vector de forma que el robot realice los mismos movimientos que el usuario. A continuación se describirán detalladamente las dos funciones de este nodo.

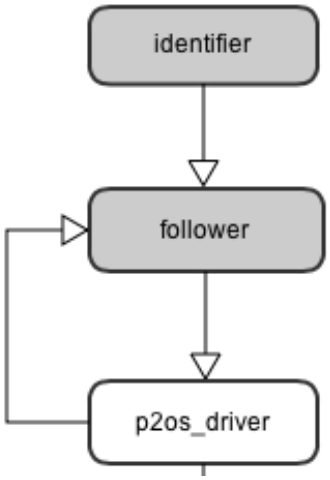

Figura 38 - Esquema follower

Establecer velocidad

La primera función es establecer la velocidad a la que se moverá el robot para seguir al usuario. Para ello se debe considerar la información útil que recibe el nodo y la forma en que comunicará la velocidad al driver del robot.

Como se observa en la Figura 38, el nodo recibe información tanto del nodo *identifier* como del *p2os_driver*. Para establecer la velocidad de movimiento no será necesario atender a la información proporcionada por el driver, por lo que únicamente será útil la información proporcionada por el nodo *identifier*. Como se ha explicado anteriormente, el nodo *identifier* publica un punto en el espacio tridimensional correspondiente a la posición del torso del usuario respecto de la Kinect.

Por otro lado, como muestra la Figura 38, el nodo *follower* se comunica con el nodo *p2os_driver* publicando el mensaje */Twist* descrito en la Tabla 36. Este mensaje establece la velocidad a la que se moverá el robot, mediante de la especificación de la componente lineal y la angular del vector velocidad. Para que el robot realice movimientos lineales, deberemos dar valor a la componente *x* del vector *linear*. Mientras que para que rote, deberemos dar valor a la componente z del vector *angular*.

Dado que la componente *z* del punto del torso equivale a la profundidad o distancia entre el humano y el robot, asignaremos dicho valor a la componente *x* del vector *linear.* Este valor se deberá escalar para que no sobrepase la velocidad de seguridad establecida para el robot. En cambio para establecer la velocidad angular, asignaremos a la componente *z* (eje sobre el que gira el robot) del vector *angular* la componente *x* del punto del torso, ya que ésta equivale al desplazamiento lateral del usuario. Por ejemplo, si el usuario se encuentra a 2 metros del robot y desplazado 0,5 metros sobre el eje x, la componente *z* de la velocidad angular será 0,5 y la componente *x* de la velocidad lineal será (2*0,6)-0,6, donde 0,6 es una valor establecido para escalar. En la siguiente figura se observa la orientación de los ejes, teniendo en cuenta que el usuario está de cara al robot.

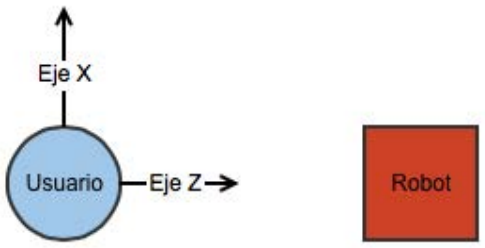

Figura 39 - Orientación de los ejes

Esquivar obstáculos

La segunda función de este nodo es establecer un modo de esquivar obstáculos. Para ello, se utiliza la información recibida a partir de los sensores del robot, en este caso, la información que proporciona el sónar que lleva incorporado el robot y la información sobre la odometría.

La información referente al sónar se obtiene mediante la suscripción al *topic /sonar*, que transporta un mensaje de tipo *p2os_driver/sonarArray*. Este mensaje contiene la información sobre la distancia a la que e encuentra un objeto en la dirección hacia la que están orientados los emisores y receptores del sónar. Los sónares están agrupados de la forma que indica la Figura 40, donde se observan seis ubicados en la parte frontal del robot y dos ubicados en su parte lateral. Esta información se utilizará para determinar una distancia mínima a la que el robot se aproximará a un objeto.

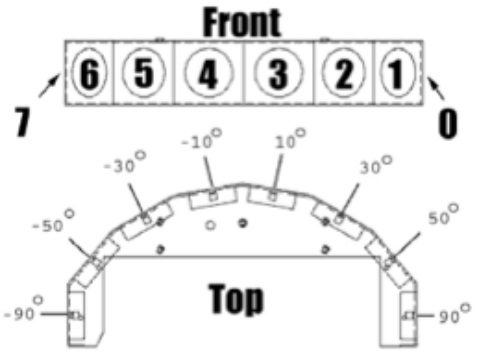

Figura 40 - Sónares del robot P3DX

El nodo también está suscrito al *topic /pose*, que contiene un mensaje de tipo *nav_msgs/Odometry* . Este mensaje contiene información sobre la posición actual del robot respecto a su origen (posición inicial del robot) y sobre su orientación (formato de cuaternión). El cuaternión es un tipo de notación matemática que se utiliza para representar orientaciones y rotaciones de objetos en 3 dimensiones. La odometría se utiliza en dos casos:

- Para que el robot avance cierta distancia. Esto se realiza estableciendo una velocidad lineal fija y comparando la posición objetivo del robot con su posición actual. Una vez llegado al punto, se cambia la velocidad a 0.
- Para que el robot rote cierta distancia. Este proceso requiere un paso previo que consiste en transformar la rotación en cuaterniones a rotación en ángulos de Euler, en concreto en la rotación sobre el eje Z (Yaw) mediante la siguiente fórmula:

$$\psi = \text{atan2}(2 * ((q_0 * q_3) + (q_1 * q_2)), 1 - 2 * (q_2^2 + q_3^2))$$

Donde, q_0, q_1, q_2 y q_3 son los elementos de cierta rotación representada en cuaterniones y ψ el ángulo de dicha rotación. Una vez calculado, se establece una velocidad angular fija y se compara el ángulo destino con el actual.

El proceso de esquivar un obstáculo será un proceso lineal que constará de los siguientes pasos:

1. Los sónares delanteros (3 y 4 de la Figura 40) detectan un valor de proximidad inferior al umbral configurable.
2. El robot rota 90 grados hacia la izquierda.
3. El robot avanza medio metro hacia delante.
4. El robot gira 90 grados hacia la derecha.
5. El sistema empieza el proceso de reconocimiento de los individuos de la escena.

En resumen, este nodo se encarga de que el sistema siga al usuario, a no ser que se encuentre un obstáculo que se interponga en su trayectoria.

3.3.3.4 Módulo del P3DX

A continuación se describen los nodos que aparecen en el módulo del robot 3DX vistos en la Figura 23.

3.3.3.4.1 P2os_driver

Este nodo, contenido en el paquete *p2os* de ROS, realiza la función de controlador para cualquier robot de *Adept Mobile Robots* que utilicen los *firmwares* P2OS, AROS o ARCOS. Por lo que el P3DX

es soportado, ya que utiliza el *firmware* ARCOS. Permitiendo así la comunicación entre ROS y el P3DX.

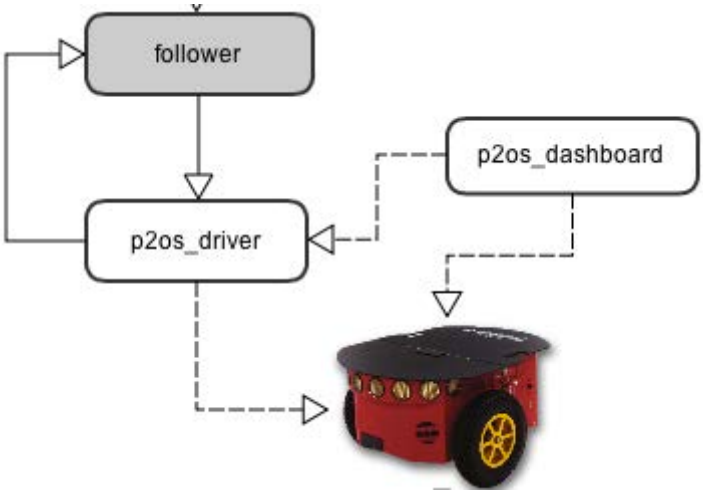

Figura 41 - Esquema p2os_driver

La funcionalidad de este nodo es recoger la información necesaria para el movimiento del robot, el uso de *gripper* o el control del estado del motor, así como publicar información como la lectura del sónar o de la odometría. Para realizar el movimiento del robot, el nodo está suscrito al *topic /cmd_vel*. Este *topic* transporta un mensaje de tipo *geometry_msgs/Twist*. Por lo que cuando reciba algún mensaje de ese tipo a través de ese *topic* el robot se moverá o no de acuerdo a los datos obtenidos. Como hemos visto en la Tabla 36, este tipo de mensajes tiene dos componentes, el vector de la velocidad lineal y el de la velocidad angular.

En el sistema diseñado, este nodo recibirá por parte del nodo *follower* la velocidad a la que debe moverse el robot. Una vez recibida, el nodo *p2os_driver* se encarga de procesar esta información para realizar el movimiento de los motores de cada una de las ruedas.

Capítulo 4: Experimentación

En este capítulo se describe de forma detallada el proceso de evaluación al que ha sido sometido el sistema para comprobar que cumple los objetivos descritos inicialmente. Para realizar este proceso de evaluación, se han definido 3 experimentos que serán ejecutados por cuatro personas ajenas al desarrollo del trabajo. Una vez estas personas hayan realizado los distintos experimentos, serán sometidas a un cuestionario para evaluar el funcionamiento del sistema e indicar aquellos problemas que hayan detectado en el sistema.

Cada uno de los experimentos que han sido diseñados para evaluar el sistema, serán analizados en tres etapas. Cada una de estas etapas se corresponde con los diferentes modos de detección de usuarios que han sido diseñados con el fin de realizar un comparativa y analizar cual de ellos es más eficiente a la hora de detectar a los usuarios y tener en cuenta las posibles contingencias del entorno. Cada uno de los usuarios seleccionados para realizar el proceso de evaluación realizaran los experimentos en cada uno de los modos. Una vez recogida la información aportada por los usuarios durante la evaluación ésta será analizada.

4.1 Descripción de los experimentos

A continuación se realiza una descripción detallada de cada uno de los experimentos y seguidamente, una descripción de los objetivos que se persiguen con la obtención de los resultados.

4.1.1 Experimento 1

En este primer experimento, se realizará un prueba de seguimiento básico del robot a lo largo de un pasillo. El usuario deberá entrar en la escena para que el sistema le reconozca. Una vez reconocido, el robot iniciará el seguimiento y la tarea del usuario será desplazarse diez metros. Una vez llegado al punto de destino, el usuario deberá girar 180 grados y volver al punto de inicio, de manera que el robot realice también el giro y continúe siguiéndole. En la Figura 42 se muestra el diseño del experimento.

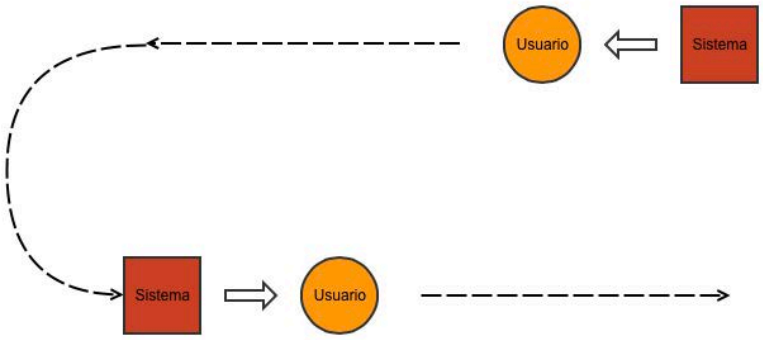

Figura 42 - Esquema del experimento 1

El objetivo de este experimento es comprobar el correcto funcionamiento del sistema en un entorno controlado, en el que no entran en la escena posibles obstáculos ni otros usuarios que puedan alterar el proceso de seguimiento. El sistema se limita a detectar a un único usuario que hay en escena, al cuál deberá seguir o perseguir.

4.1.2 Experimento 2

Este segundo experimento consistirá en la realización de un circuito, a través del cual el usuario deberá evitar dos obstáculos haciendo desplazamientos en cualquier dirección de manera que el robot sea capaz de esquivar dichos obstáculos para continuar con el seguimiento del usuario. Como en el experimento anterior, el usuario deberá ubicarse de espaldas al robot para ser reconocido y así comenzar el seguimiento. En la Figura 43 se muestra el diseño del experimento.

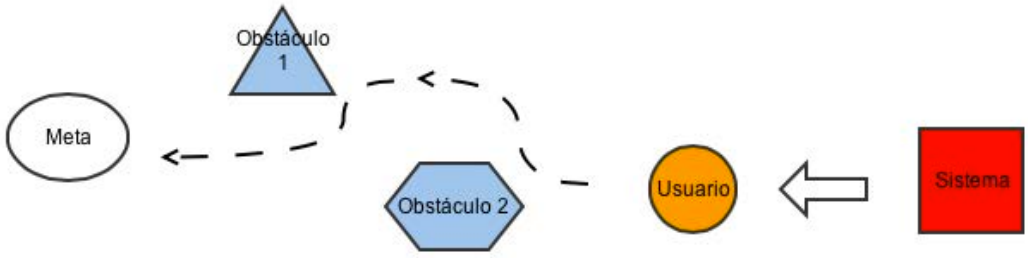

Figura 43 - Esquema del experimento 2

El objetivo de este experimento es introducir movimientos avanzados en el sistema así como el uso del sónar y la odometría para esquivar obstáculos. De esta forma, se comprobará la posibilidad de utilizar nuestro sistema de seguimiento en localizaciones menos controladas.

4.1.3 Experimento 3

En este tercer experimento, se introducen nuevas dificultades para comprobar el correcto funcionamiento del reconocimiento de usuarios. El escenario será similar al del segundo experimento. El usuario deberá ubicarse delante del robot y desplazarse diez metros a los largo del pasillo. Durante este desplazamiento, aparecerán dos usuarios que se cruzarán entre el robot y el usuario impidiendo así la identificación del usuario seguido. El sistema deberá pararse hasta que vuelva a reconocer al usuario, omitiendo la detección de los otros dos usuarios. En la Figura 44 se muestra el diseño del experimento.

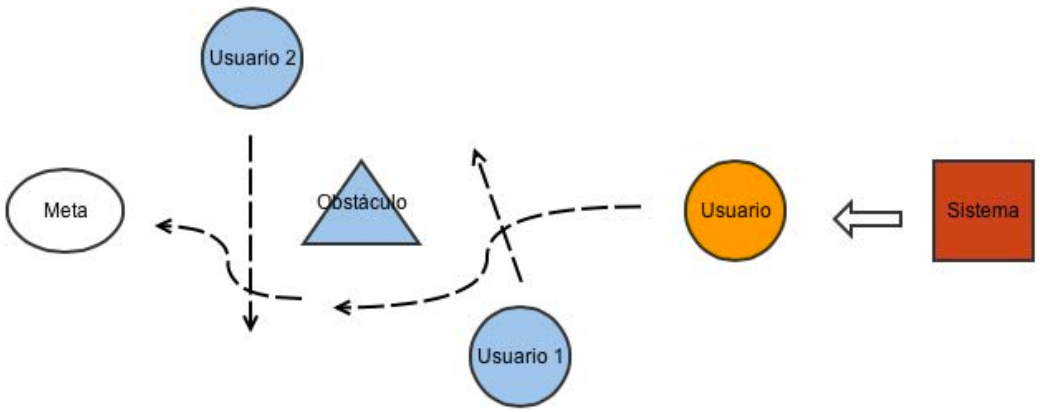

Figura 44 - Esquema del experimento 3

El objetivo de este experimento es plantear una situación de confusión para el robot, que no solo pierde de vista al usuario inicial, sino que también debe esquivar a otros usuarios y obstáculos de la escena. Con este experimento se simulará un escenario real en el que podría utilizarse nuestro sistema, como podría ser una aeropuerto. En el que el robot se encarga de llevar las maletas del usuario, al que debe seguir en un entorno con mucho movimiento y obstáculos.

4.2 Análisis de los resultados

Para realizar el proceso de evaluación del sistema se seleccionarán varios usuarios para que realizaran cada uno de los experimentos descritos en el apartado anterior. Una vez hayan realizado cada uno de los experimentos, se solicitará a estos usuario que realicen un cuestionario para así conocer sus sensaciones acerca del funcionamiento del sistema. De esta manera se comprobará el alcance del trabajo así como la posibilidad de trabajos futuros a partir de él. Los usuarios deberán realizar tres veces los cuestionarios de cada uno de los experimentos, para así realizar la evaluación de las tres alternativas de reconocimiento propuestas.

A continuación se muestran los resultados de los cuestionarios. Estos cuestionarios se pueden ver en el anexo 5.

MODELO DE RECONOCIMIENTO DE FORMA				
Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Experimento 1				
1.	Sí	Sí	Sí	Sí
2.	-	-	-	-
3.	1	1	1	1
4.	4	4	4	5
5.	5	4	3	3
6.	3	4	4	4
Experimento 2				
7.	Sí	Sí	Sí	Sí
8.	-	-	-	-
9.	1	1	2	2
10.	1	1	2	1
11.	4	5	4	4
12.	3	3	3	4
Experimento 3				
13.	Sí	Sí	Sí	Sí
14.	-	-	-	-
15.	1	1	1	1
16.	No	No	No	No
17.	1	2	1	1
18.	Sí	Sí	Sí	Sí

Tabla 37 - Respuestas al primer cuestionario

MODELO DE RECONOCIMIENTO DE COLOR				
Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Experimento 1				
1.	Sí	Sí	Sí	Sí
2.	-	-	-	-
3.	1	1	1	1
4.	4	5	5	5
5.	1	2	1	1
6.	3	3	3	3
Experimento 2				
7.	Sí	Sí	Sí	Sí
8.	-	-	-	-
9.	1	2	1	1
10.	2	1	2	1
11.	5	5	4	4
12.	3	3	4	4
Experimento 3				
13.	Sí	Sí	Sí	Sí
14.	-	-	-	-
15.	1	1	1	1
16.	No	No	No	No
17.	1	1	1	1
18.	Sí	Sí	Sí	Sí

Tabla 38 - Respuestas al segundo cuestionario

MODELO DE RECONOCIMIENTO MIXTO				
Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Experimento 1				
1.	Sí	Sí	Sí	Sí
2.	-	-	-	-
3.	1	2	2	1
4.	3	4	3	3
5.	4	3	3	4
6.	3	2	4	3
Experimento 2				
7.	Sí	Sí	Sí	Sí
8.	-	-	-	-
9.	1	2	1	1
10.	2	3	3	2
11.	2	3	3	3
12.	2	2	3	3
Experimento 3				
13.	Sí	Sí	Sí	Sí
14.	-	-	-	-
15.	1	1	1	2
16.	No	No	Sí	No
17.	3	Más de 3	Más de 3	Más de 3
18.	Sí	Sí	No	Sí

Tabla 39 - Respuestas al tercer cuestionario

PREGUNTAS GENERALES				
Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4
19.	Reconocimiento mediante color	Reconocimiento mediante color	Reconocimiento mediante color	Reconocimiento mediante color
20.	Reconocimiento mediante color	Reconocimiento mediante color	Reconocimiento mediante forma	Reconocimiento mediante color
21.	Reconocimiento mediante forma	Reconocimiento mediante color	Reconocimiento mediante color	Reconocimiento mediante forma
22.	Experimento 3	Experimento 3	Experimento 3	Experimento 3
23.	Sí	Sí	Sí	Sí
24.	Seguimiento personas mayores como carrito de la compra	Control paternal dentro de un hogar.	Tareas de vigilancia	Juguete para niños pequeños.
25.	-	-	-	-
26.	3	4	4	4
27.	5	5	5	4
28.	3	2	2	2
29.	Mejor cámara	Rapidez del tercer sistema	-	Velocidad del tercer sistema y rapidez de evasión de obstáculos
30.	-	-	-	-

Tabla 40 - Respuestas al cuestionario general

Tras obtener las respuestas de los usuarios, se pueden sacar las siguientes conclusiones:

- El modelo que usa reconocimiento por forma ha sido bien valorado ya que no requiere una calibración previa y el movimiento del robot es bastante fluido. En muy pocas ocasiones se pierde al usuario y la mayoría son causadas porque el individuo se aleja demasiado del robot en un corto espacio de tiempo. Una de las principales desventajas

que han visto los usuarios en este modelo es que es difícil que el robot no pierda de vista al usuario en giros de 180 grados, ya que la plantilla con el círculo únicamente se encuentra en la espalda del usuario, provocando así que se deba realizar el giro lentamente.

- En cuanto al modelo con reconocimiento de color, se ha concluido que es el más eficiente para los usuarios, ya que en muy pocas ocasiones el robot deja de seguir al usuario, incluso cuando parecen otros individuos en la escena con un color de camiseta similar. La única desventaja encontrada de este sistema es que requiere una calibración previa del color de la camiseta del usuario.
- Por último, el sistema que usa un modelo de reconocimiento mixto, aunque es el más complejo, ha sido el peor valorado por los usuarios. Esto se debe principalmente a que requiere una mayor cantidad de procesamiento y en ocasiones el robot realiza el movimiento lentamente e incluso puede no moverse aunque el usuario se esté moviendo.
- En cuanto al sistema en general, la valoración ha sido muy positiva, algunos detalles que los usuarios han propuesto para mejorar el sistema se podrían tomar como trabajos futuros. Por ejemplo, se ha propuesto que el robot realice una evasión de obstáculos más rápida rodeando el obstáculo en lugar de realizar una evasión predeterminada. Otro aspecto a mejorar sería el de ampliar el rango de visión de la cámara, ya que en ocasiones el robot perdía de vista al usuario porque éste se había alejado demasiado. Esto ocurría sobretodo tras la evasión de los obstáculos.
- Por otra parte, hablando del tema de posibles aplicaciones del sistema, la mayoría de usuarios están de acuerdo en que sus funciones estarían orientadas a ayudar o servir a los usuarios, que es la idea que motivó el trabajo. También tareas como la vigilancia o la ayuda a personas mayores.

Capítulo 5: Gestión del proyecto

En este capítulo se muestran las diferentes fases de desarrollo del trabajo, indicando qué trabajo se ha realizado en cada una de ellas y qué tiempo se ha invertido en realizarlas. También se realiza un desglose del presupuesto estimado para el desarrollo del trabajo.

5.1 Descripción de las fases del proyecto

Para determinar las fases que componen el desarrollo del trabajo, se va a establecer el ciclo de vida que seguirá el trabajo, mediante el uso del modelo de desarrollo en espiral (Figura 45). Este modelo definido por Barry Boehmen 1986 [26], se basa en el desarrollo iterativo de las diferentes actividades que componen el trabajo. Una de las razones por la que se ha escogido este modelo ha sido la posibilidad que ofrece a la hora de incluir cambios y nuevos requerimientos sin romper la metodología, a favor de otros modelos como el de casada que ofrece más rigidez en ese aspecto.

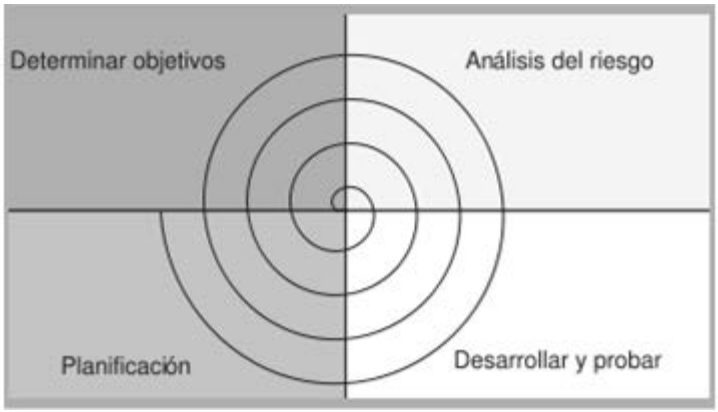

Figura 45 - Modelo de desarrollo en espiral

El modelo de desarrollo en espiral establece cuatro tareas predeterminadas que se deben realizar en cada iteración del desarrollo. Estas tareas son:

- Determinar objetivos
- Análisis del riesgo
- Planificación de la próxima fase
- Desarrollo y pruebas

Dado que estas tareas están diseñadas para proyectos de gran calibre, más adelante se describirán las diferentes tareas en las que se ha divido este trabajo. Estas tareas deberán realizarse en cada una de las iteraciones que se establezcan para el trabajo. Dichas iteraciones se corresponden con las funcionalidades que se desarrollarán para el sistema para cumplir sus objetivos. Estas iteraciones serán las siguientes:

- En la primera iteración se desarrollará el sistema de reconocimiento de humanos.
- En la segunda iteración se llevará acabo el desarrollo del sistema de seguimiento del robot.
- En la tercera iteración se realizará la experimentación y evaluación del sistema junto con la redacción de la documentación.
- En la cuarta iteración se añadirá una nueva funcionalidad en el módulo de seguimiento y las pruebas y documentación que conlleva.

Como se ha mencionado anteriormente, cada una de las iteraciones estará compuesta por las fases correspondientes al diseño establecido para el modelo en espiral. En este caso, dichas fases son las siguientes:

- **Análisis:** En esta fase se estudian las diferentes herramientas y tecnologías necesarias para el desarrollo de la iteración y se detallan las especificaciones y requisitos que deberá tener la funcionalidad correspondiente a la iteración
- **Diseño:** En esta fase se presenta la arquitectura del sistema después de la inclusión del módulo correspondiente a la iteración.
- **Implementación:** En esta fase se desarrolla la funcionalidad de cada iteración, teniendo en cuenta las dos fases anteriores.
- **Pruebas:** En esta fase se realizan las pruebas básicas para comprobar el funcionamiento de los módulos desarrollados durante la iteración, para así establecer si se puede avanzar a la siguiente iteración.
- **Documentación:** En esta fase se recopila toda la información y referencias utilizadas durante la iteración.

Estas fases, intervendrán en cada una de las cuatro iteraciones de la espiral, pero dependiendo de la iteración tendrán más o menos complejidad, o serán más o menos importantes. En cuanto a las dos primeras iteraciones, no habrá mucha variación en el objetivo de la fase, ya que corresponden

al desarrollo de los dos módulos del sistema. En el caso de la tercera iteración, las fases de análisis, diseño e implementación tendrán menos peso que en las dos iteraciones anteriores, mientras que las fases de pruebas y documentación cobrarán especial importancia. En cambio, la cuarta iteración será equilibrada, ya que se basa en la inclusión de una nueva funcionalidad del sistema y la finalización del trabajo.

5.2 Planificación

En la Tabla 41, se muestra la duración de las diferentes iteraciones y sus correspondientes fases, estableciéndose la fecha de inicio y fin, así como su duración en horas.

Tarea	Fecha Inicio	Fecha Fin	Duración
1ª Iteración	17/12/2012	02/04/2013	77
Análisis	17/12/2012	25/01/2013	30
Diseño	28/01/2013	06/02/2013	8
Implementación	11/02/2013	15/03/2013	25
Pruebas	18/03/2013	29/03/2013	10
Documentación	02/04/2013	02/04/2013	1
2ª Iteración	05/04/2013	21/05/2013	33
Análisis	05/04/2013	11/04/2013	5
Diseño	15/04/2013	22/04/2013	6
Implementación	23/04/2013	13/05/2013	15
Pruebas	14/05/2013	20/05/2013	5
Documentación	21/05/2013	21/05/2013	1
3ª Iteración	22/05/2013	22/06/2013	23
Análisis	22/05/2013	22/05/2013	1
Diseño	23/05/2013	23/05/2013	1
Implementación	24/05/2013	24/05/2013	1
Pruebas	10/06/2013	18/06/2013	7
Documentación	27/05/2013	21/06/2013	20
4ª Iteración	03/10/2013	06/02/2014	40
Análisis	03/10/2013	10/10/2013	6
Diseño	11/10/2013	16/10/2013	4
Implementación	17/10/2013	01/11/2013	12
Pruebas	02/01/2014	13/01/2014	8
Documentación	24/01/2014	06/02/2014	10

Tabla 41 - Duración del trabajo

En las figuras 46 y 47 se presenta el diagrama de Gantt con las distintas iteraciones y sus respectivas tareas.

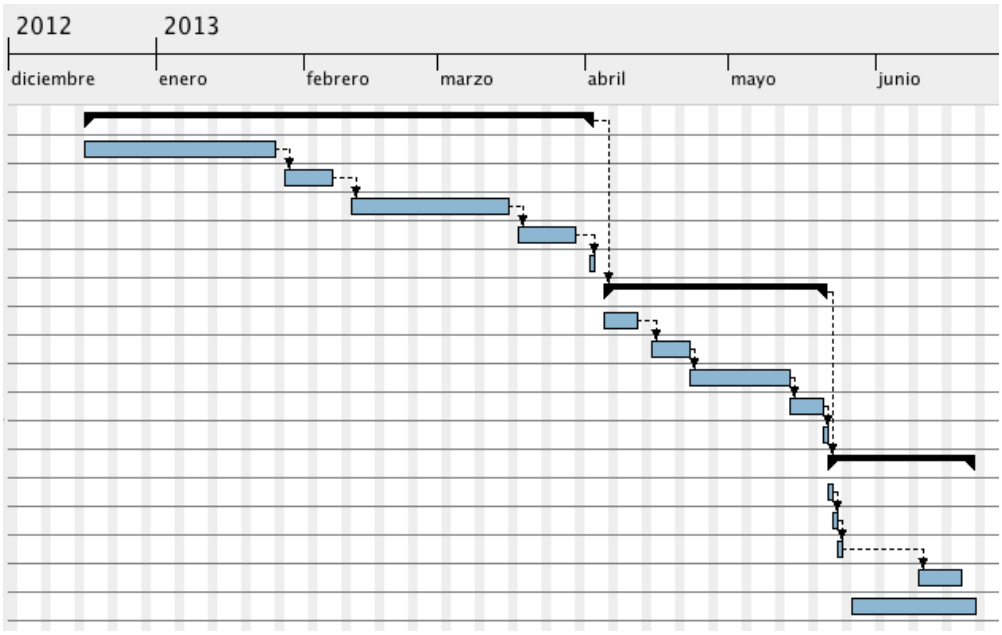

Figura 46 - Diagrama de Gantt (1)

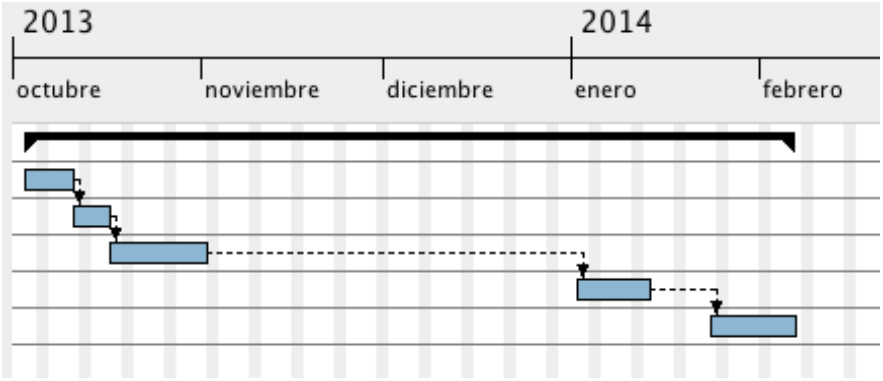

Figura 47 - Diagrama de Gantt (2)

5.3 Presupuesto

A continuación se detalla el desglose del presupuesto total establecido para el desarrollo del trabajo.

5.3.1 Costes personales

Dado que el desarrollo del trabajo se ha dividido en fases bien diferenciadas, se van a establecer una serie de roles a los que se asignará cada una de las fases de acuerdo a sus competencias. El jefe de proyecto será el encargado de establecer las bases del trabajo, así como de la revisión del trabajo realizado; el analista se encargará de la investigación de las tecnologías necesarias para el desarrollo de las tareas; el programador se encargará de realizar las taras de diseño del sistema, implementación del mismo y realización de las pruebas junto al jefe de proyecto. La fase de documentación será realizada conjuntamente por el programador y el analista.

De acuerdo con estos perfiles y con las tablas de salarios proporcionadas por el BOE, en la Tabla 42 se puede observar el tiempo que ha dedicado al desarrollo del trabajo cada uno de los perfiles establecidos. Se van a tener en cuenta jornadas laborales de 4 horas.

Perfil	Horas trabajadas	Coste
Jefe de proyecto	60	2.100,00€
Analista	228	6.840,00€
Programador	460	11.500,00€
Total		20.440,00€

Tabla 42 - Resumen de costes personales

5.3.2 Costes materiales

Los costes materiales para el trabajo incluyen los dispositivos utilizados, así como las licencias de software necesarias para su realización. En la siguiente tabla se muestra el coste de cada uno de estos recursos.

Artículo	Coste	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable
Portátil Acer	700€	7	48	102,08€
Pioneer 3DX	1000€	6	48	125,00€
Microsoft Kinect	70€	6	36	11,67€
Adaptador corriente Kinect	10€	5	0	10,00€
Ubuntu 11.10	0€	7	0	0,00€
ROS	0€	0	0	0,00€
Fungibles	35€	0	0	35,00€
TOTAL				283,75€

Tabla 43 - Resumen de costes materiales

Para el cálculo de la amortización de los recursos se utiliza la siguiente fórmula:

$$(A/B)\ x\ C$$

Donde,

A = número de meses que el equipo ha sido utilizado.

B = periodo de depreciación.

C = coste del equipo.

5.3.3 Costes totales

Para realizar el cómputo final, a los gastos totales de personal y materiales hay que sumar un 8 % de gastos indirectos (internet, luz, transporte, …).

Resumen de costes	
Costes Personales	20.440,00€
Costes Materiales	283,75€
Costes Indirectos	680,30€
TOTAL	21.404,05€

Tabla 44 - Resumen de los costes totales

Teniendo en cuenta el desglose de costes establecidos, como muestra la tabla anterior, el coste total del trabajo ascendería a:

PRECIO TOTAL: 21.404,05 Euros (IVA no incluido).

Veintiún mil cuatrocientos cuatro con cinco céntimos (IVA no incluido).

Capítulo 6: Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones obtenidas tras la realización del trabajo. En primer lugar se presentan las conclusiones generales que han sido obtenidas tras la realización de este trabajo. Seguidamente, se exponen las conclusiones para cada uno de los objetivos del trabajo, así como los diferentes problemas que han sido encontrados a lo largo del proceso de realización de este trabajo. En último lugar se presentan las posibles mejoras aplicables al sistema.

6.1 Conclusiones generales

El desarrollo del trabajo me ha permitido fomentar las labores de investigación a la hora de realizar un trabajo de esta escala. Descubriendo así un campo que me ha parecido muy interesante como es la visión artificial. Mediante esta investigación he tenido la posibilidad de descubrir aplicaciones muy interesantes creadas a partir de este campo y especialmente las posibilidades que ofrece el sensor Kinect, de la que sólo conocía sus aplicaciones en el ámbito del entretenimiento.

La realización del trabajo también me ha servido para aprender el funcionamiento de un *framework* como ROS, tan útil en el campo de la robótica. Este software permite la creación de infinidad de aplicaciones utilizando muchos de los robots y sensores más utilizados en el campo de la investigación.

Otro aspecto que destacaría es la posibilidad de haber trabajado con hardware tan poco habitual como es el robot P3DX. Pudiendo apreciar así las dificultades que aparecen al trabajar con este tipo de dispositivos, pero aumentando la satisfacción de ver el sistema diseñado funcionando más allá de una pantalla de ordenador.

6.2 Conclusiones referentes a los objetivos

Para cada uno de los objetivos fijados para este trabajo, se han obtenido las siguientes conclusiones:

Familiarización del entorno ROS.

Tras comenzar la implementación y realizar un estudio más exhaustivo de ROS, se ha llegado a apreciar las grandes posibilidades que ofrece. En especial, un sistema de comunicaciones entre nodos, que permite implementaciones a bajo nivel que facilitan la comunicación entre los robots. Otra ventaja que tiene ROS es la gran variedad de paquetes que ofrece destinados a ayudar en la implementación de tu propia aplicación para robots. La única pega que le pondría a ROS es la falta de documentación en algunos paquetes importantes.

Familiarización con el robot P3DX

En este caso, se ha realizado un estudio de las funcionalidades que ofrece este robot, y de cómo pueden ser utilizadas en el sistema, en concreto en su capacidad motriz y el uso de los sónares incorporados.

Familiarización con el dispositivo Kinect

El estudio de este sensor ha servido para examinar la cantidad de posibilidades que ofrece en el ámbito de la visión artificial.

Desarrollo del reconocimiento de humanos

Esta ha sido la parte más compleja del trabajo, aunque también ha sido en la que más se ha aprendido. Se ha estudiado el funcionamiento del *framework* OpenNI para utilizarlo en las fases de detección y localización de usuarios del módulo de reconocimiento del trabajo, así como los distintos tipos de reconocimiento existentes mediante la visión artificial. La única crítica que se hace a OpenNI es la desactualización y el mal funcionamiento de algunas funciones, aunque también es verdad que no se ha podido utilizar su última versión (OpenNI 2.0), ya que al ser reciente no tenía soporte en ROS. En cuanto a la identificación de usuarios, se han podido comparar varias alternativas y comprobar así cuál es más precisa.

Desarrollo del seguimiento de humanos

En un principio se tuvieron dificultades con el desarrollo de esta tarea debido a la falta de documentación sobre el robot y la resolución del problema de extrapolar la información obtenida del usuario a través de la Kinect al movimiento del robot.

Experimentación

La experimentación me ha permitido observar que opinión generaba en diferentes individuos el trabajo realizado. Pudiendo así evaluar el sistema de una forma más objetiva y realista, así como conocer las propuestas de mejora del sistema.

Desarrollo de la documentación

La elaboración de la documentación ha servido para aprender a elaborar una descripción detallada de un sistema, así como de la investigación y el estudio previo a su realización.

6.3 Trabajos futuros

A continuación se enumeran algunas mejoras que podrían ampliar las funcionalidades del sistema desarrollado:

- Automatizar el proceso de calibración del color de la camiseta del usuario.
- Incorporar la funcionalidad de reconocimiento de gestos por parte del sistema.
- Utilizar otra forma de identificación del usuario asignado que no dependa tanto de las variaciones en el entorno como es el reconocimiento de colores.
- Añadir cierta autonomía al robot para que se capaz de encontrar al usuario perdido sin la necesidad de que éste vuelva a la escena.
- Añadir autonomía total al robot e intercambiar el rol con el usuario de modo que el robot haga de guía y así poder realizar otras funciones vistas en la literatura como puede ser la guía de invidentes.
- Actualizar el módulo de reconocimiento para que utilice la nueva versión de OpenNI.

Capítulo 7: Anexos

7.1 Manual de instalación

Para realizar los pasos enumerados en este manual, será necesario tener instalada una versión de Ubuntu posterior a la 10.04 y anterior a la 12.04. En concreto, este manual describe la instalación de ROS sobre la distribución 11.10 de Ubuntu (Oneiric Ocelot).

1. Configurar el archivo source.list:

```
sudo sh –c ‘echo “deb http://packages.ros.org/ros/ubuntu oneiric main” > /etc/apt/sources.list.d/ros-latest.list’
```

2. Configurar las claves:

```
wget http://packages.ros.org/ros.key –O - | sudo apt-key add -
```

3. Instalación:

```
sudo apt-get update

sudo apt-get install ros-electric-desktop-full
```

4. Configuración del entorno:

```
echo “source /opt/ros/electric/setup.bash” >> ~/.bashrc

. ~/.bashrc
```

Una vez instalado ROS en nuestro sistema, será necesario añadir la paquetes necesarios para el funcionamiento del sistema.

5. Dar permisos de escritura al directorio de ROS:

```
cd /opt/ros/electric

sudo chmod –R 777 ./stacks
```

```
cd /opt/ros/electric/stacks
```

6. Descargar el stack desarrollado para el trabajo en el directorio actual de forma que se cree el siguiente directorio:

/opt/ros/electric/stacks/tfg_tracker

7. Configurar el entorno:

```
echo ". /opt/ros/electric/stacks/tfg_tracker/setup.bash" >> ~/.bashrc

. ~/.bashrc
```

8. Descarga e instalación del paquete openni_kinect:

```
sudo apt-get install ros-electric-openni-kinect
```

9. Descarga y compilación del stack cmvision:

```
svn –co https://code.ros.org/svn/wg-ros-pkg/branches/trunk_cturtle/vision/cmvision

rosmake cmvision
```

10. Descarga y compilación del stack p2os:

```
svn –co https://usc-ros-pkg.svn.sourceforge.net/svnroot/usc-ros-pkg/trunk/p2os

rosmake p2os
```

11. Una vez instalados los paquetes necesarios, compilamos los paquetes desarrollados:

```
rosmake skel_tracker

rosmake circle_finder

rosmake identifier

rosmake follower
```

7.2 Manual de usuario

En este anexo se describen los pasos necesarios para la ejecución del sistema. Para ejecutar el sistema se recomienda que haya dos usuario, uno que se encargue de ejecutar el sistema y comprobar su estado durante el seguimiento; y otro que sea el usuario al que seguirá el robot, que deberá ponerse el peto con la marca circular .

También es recomendable que el robot tenga la batería cargada para evitar el uso de cables que puedan dificultar el movimiento del robot, aunque el movimiento del robot estará restringido a la distancia entre el robot y la toma de corriente más cercana, ya que la Kinect debe permanecer conectada en todo momento. Dicho esto, colocaremos el portátil sobre el robot y conectaremos a él el P3DX y la Kinect. A continuación se deberán seguir los siguientes dos pasos:

Abriremos un nuevo terminal en el portátil y ejecutaremos el siguiente comando:

```
roslaunch skel_tracker colorgui.launch
```

Tras esto se abrirá una ventana con la imagen RGB proporcionada por la Kinect, donde deberemos hacer click izquierdo en el ratón sobre la indumentaria del usuario al que el robot debe seguir. Una vez comprobamos que aparecen distintos *blobs* sobre el color, copiamos el valor RGB y el YUV que aparecen en la parte inferior de la ventana y los pegaremos en el fichero colors.txt ubicado en el paquete *skel_tracker*. Una vez hecho esto, habremos calibrado el color de la camiseta del usuario y podremos terminamo la ejecución de este nodo.

En segundo lugar, ejecutaremos el siguiente comando:

```
roslaunch  follower tracker.launch
```

La función de este lanzador es ejecutar cada uno de los nodos del sistema en orden. En la Figura 48 se muestra el contenido del fichero.

```
<launch>
    <!-- launch p2os_driver -->
    <node pkg="p2os_driver" type="p2os_driver" name="p2os_driver" respawn="true">
        <rosparam file="$(find p2os_driver)/launch/p2os_driver.yaml" command="load" />
    </node>

    <!-- launch p2os_dashboard -->
    <node pkg="p2os_dashboard" type="p2os_dashboard" name="p2os_dashboard" respawn="true"/>

    <!-- launch kinect sensor -->
        <arg name="debug" default="false"/>
        <arg if="$(arg debug)" name="launch_prefix" value="xterm -rv -e gdb -ex run -args"/>
        <arg unless="$(arg debug)" name="launch_prefix" value=""/>
    <node pkg="openni_camera" type="openni_node" name="openni_node1" output="screen" launch-prefix="$(arg launch_prefix)">
            <param name="device_id" value="#1"/>
            <rosparam command="load" file="$(find openni_camera)/info/openni_params.yaml" />
            <param name="rgb_frame_id" value="/openni_rgb_optical_frame" />
            <param name="depth_frame_id" value="/openni_depth_optical_frame" />
            <param name="use_indices" value="false" />
            <param name="depth_registration" value="true" />
            <param name="image_mode" value="2" />
            <param name="depth_mode" value="2" />
            <param name="debayering" value="2" />
            <param name="depth_time_offset" value="0" />
            <param name="image_time_offset" value="0" />
    </node>

    <!-- launch skel_tracker -->
        <param name="config_file_path" value="$(find skel_tracker)/SamplesConfig.xml" />
    <node pkg="skel_tracker" type="skel_tracker" name="skel_tracker" respawn="true" >
    </node>

    <!-- launch cmvision -->
    <node name="cmvision" pkg="cmvision" type="cmvision" respawn="false">
            <remap from="image" to="/camera/rgb/image_color"/>
            <param name="color_file" value="/home/avg/electric_workspace/sandbox/tfg_tracker/skel_tracker/colors.txt" />
            <param name="mean_shift_on" value="false" />
            <param name="color_radius_pix" value="0" />
            <param name="spatial_radius_pix" value="0" />
            <param name="debug_on" value="true" />
    </node>

    <!-- launch circle_finder -->
    <node pkg="circle_finder" type="circle_finder"  name="circle_finder"/>

    <!-- launch identifier -->
    <node pkg="identifier" type="identifier"  name="identifier"/>

    <!-- launch follower -->
    <node pkg="follower" type="follower"  name="follower"/>

</launch>
```

Figura 48 - Contenido de tracker.launch

7.3 ROS

ROS (Robot Operating System) es un meta-sistema operativo de código abierto para robots. Proporciona los servicios que se esperan de un sistema operativo, incluida la abstracción de hardware, control de dispositivos a bajo nivel, implementación de las funcionalidades más utilizadas, paso de mensajes entre procesos y gestión de paquetes. También proporciona herramientas y librerías para escribir y ejecutar código a través de varios ordenadores.

ROS funciona como una red P2P de procesos que se comunican entre sí utilizando una estructura de comunicación interna. Implementa distintos sistemas de comunicación, entre los que se incluyen las comunicaciones síncronas RPC como "Servicios", el paso de mensajes asíncronos como "Topics" y un sistema de almacenamiento de datos llamado "Parameter Server".

El objetivo principal de ROS es permitir la reutilización de código aportado por la comunidad científica, para así facilitar las labores de investigación y desarrollo de nuevo software. Para la consecución de dicho objetivo, este *framework* agrupa los diferentes procesos existentes en paquetes, que pueden ser fácilmente compartidos y accesibles para el resto de la comunidad.

A continuación se enumeran algunas de las características que diferencian a ROS de otras plataformas de software para robots:

- **Ligero**: ROS está diseñado para ser lo más ligero posible, por lo que el código escrito para ROS puede ser usado con otros frameworks para robots.
- **Librerías agnósticas**: El modelo de desarrollo preferido es escribir librerías agnósticas con interfaces funcionales limpias.
- **Independencia del lenguaje**: El framework de ROS es fácil de implementar en cualquier lenguaje de programación moderno. Actualmente están implementadas librerías en Python, C++ y Lisp.
- **Escalable**: ROS es apropiado por grandes sistemas de ejecución y por grandes procesos de desarrollo.

7.3.1 Elementos básicos

Para entender mejor el funcionamiento de este meta-sistema operativo, a continuación se describen los principales elementos que intervienen en su proceso de comunicación:

- **Nodos**: Son procesos que se ejecutan y combinan entre ellos para ofrecer una funcionalidad. Estos procesos se comunican entre ellos mediante los topics, RPC services y el Parameter Server. Un sistema de control de un robot estará compuesto por varios nodos, donde cada nodo realiza una función. El uso de nodos aporta muchos beneficios al sistema creado, como es la modularidad, que permite la delimitación de cada nodo a su función, de forma que aísla al nodo de posibles fallos en otro nodo perteneciente al sistema. Un ejemplo de la distribución de las funciones de un sistema en nodos, podría ser el de un sistema de navegación para un robot. En este sistema, un nodo se encargaría de controlar el sónar, otro el movimiento del robot, otro se encargaría de la localización y otro realizaría el *path planning*. De este modo, el sistema funcionaría mediante comunicación de cada uno de estos nodos a través de mensajes.
- **Master**: Proporciona un registro de nombres y la búsqueda del resto de los nodos. Sin el nodo Master, el resto de los nodos no serían capaces de encontrarse unos con otros, intercambiar mensajes o invocar servicios. En el sistema descrito en el ejemplo, la función del nodo Master sería permitir que los nodos se encuentren y comuniquen entre ellos.
- **Parameter Server**: Ofrece almacenamiento para la información de ROS. Los nodos usan este servidor para almacenar y recuperar parámetros en tiempo de ejecución.
- **Mensajes**: Es la unidad básica de comunicación a través de la cual se comunican los nodos. Un mensaje es, simplemente, un conjunto de datos organizados en diferentes campos. El intercambio de mensajes entre los nodos se realiza mediante la publicación de éstos en un canal llamado *topic*. La comunicación mediante mensajes es unidireccional, por tanto, asíncrona. Para realizar comunicaciones síncronas, deberemos utilizar servicios (petición/respuesta). En el ejemplo planteado, los nodos se comunicarían mediante el paso de mensajes. Por ejemplo, el nodo encargado del sónar, se comunicaría con el nodo encargado del movimiento del robot utilizando un mensaje con los datos obtenidos de su lectura para que dicho nodo pueda procesar la información y detectar si hay obstáculos en su camino.

- **Topics**: El sistema de comunicación entre nodos se basa en publicadores y subscriptores, es decir, un *topic* es el canal a través del cual se transporta un mensaje . Un nodo envía un mensaje publicándolo en un *topic*. El *topic* es el nombre que se utiliza para identificar el contenido del mensaje. Entonces, cuando un nodo quiere comunicarse con otro, deberá subscribirse o publicar en el *topic* que tienen en común, de esta manera intercambiarán la información contenida en el mensaje. En el ejemplo anterior, el nodo del sónar publicaría un mensaje en el *topic /sonar* y el nodo de movimiento se suscribiría a ese *topic* para recibir las lecturas del sónar.

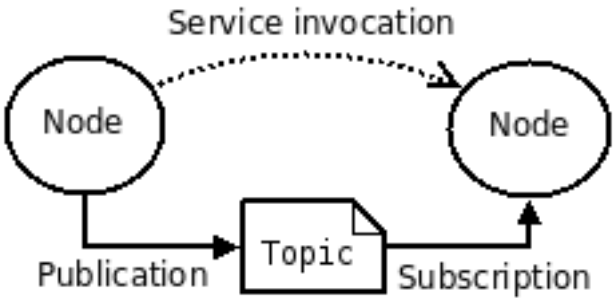

Figura 49 - Sistema de comunicaciones de ROS

A continuación se describen los elementos que componen el sistema de archivos de ROS. Este sistema de archivos tiene el objetivo de promover la reutilización de código y la encapsulación de funcionalidades:

- **Packages**: Son la principal unidad de organización de software en ROS. Contienen procesos de ejecución (nodos), librerías, ficheros de configuración y demás elementos que se almacenan de forma conjunta.
- **Manifests**: Proporcionan información acerca de los metadatos del paquete, incluyendo información sobre la licencia, dependencias o el lenguaje en el que esta programado.
- **Stacks**: Son colecciones de paquetes a los que agregan nuevas funcionalidades.
- **Stack Manifests**: Proporcionan metadatos sobre un determinado stack.

7.4 Openni

Figura 50 - Logo Openni

OpenNI (Open Natural Interaction) [23] es un *framework* multiplataforma que define *APIs* para desarrollar aplicaciones utilizando Interacción Natural. La interacción de OpenNI con otros elementos del sistema se ilustra en la Figura 51, donde se pueden observar 3 capas. La capa inferior corresponde a los dispositivos de hardware que funcionan como sensores, pudiendo ser sensores 3D, cámaras RGB, cámaras IR (infrarrojas) o dispositivos de audio. La capa intermedia corresponde a OpenNI, y provee las interfaces de comunicación que interactúan con los sensores y los componentes *middleware* . La capa superior corresponde a software que implementa aplicaciones haciendo uso de OpenNI.

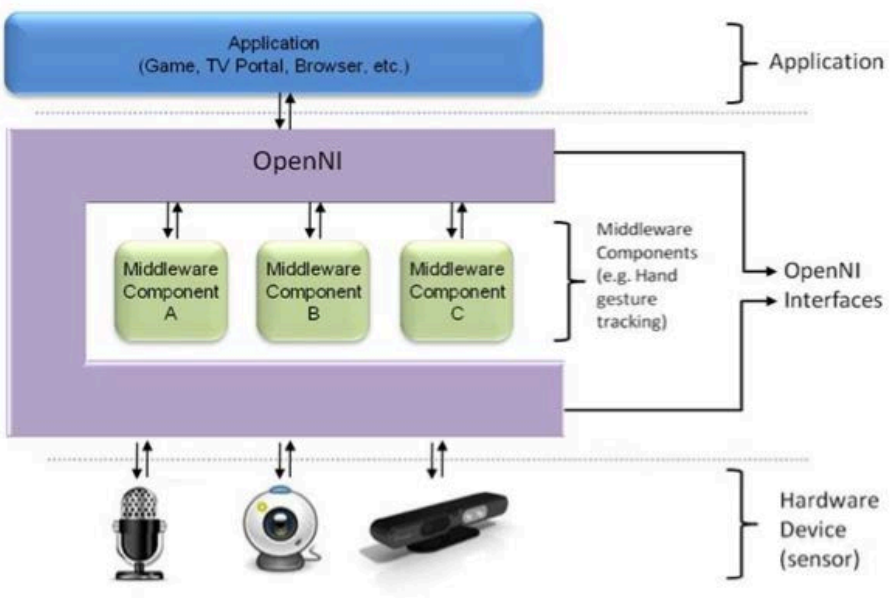

Figura 51 - Esquema de funcionamiento de Openni

El funcionamiento de OpenNI da inicio obteniendo datos de la escena mediante sus dispositivos hardware. Dependiendo de la aplicación que se le vaya a dar, esta información se puede almacenar en los siguientes formatos:

- Mapa de color: Es una imagen digital en donde cada píxel está representado por un valor RGB.
- Mapa de profundidad: Es una imagen digital en donde cada píxel está representado por la distancia al sensor.
- Mapa infrarrojo: Es una imagen digital en donde cada píxel representa el brillo de ese píxel en escala de grises.

Después de esto, el *middleware*, se utiliza para procesar los datos obtenidos de los sensores en alguno de los formatos anteriores y obtener información de nivel más alto, que puede ser entendida y utilizada por la aplicación.

OpenNI trabaja utilizando nodos de producción, que son conjuntos de componentes que tienen un rol productivo en el proceso de creación de datos. Cada nodo de producción encapsula la funcionalidad que se relaciona con la generación de un tipo específico de dato, y puede proveer este dato a la aplicación o a otro nodo de producción.

Para comprobar el funcionamiento de estos nodos de producción, tomaremos como ejemplo la aplicación *openni_tracker* del paquete *openni_kinect* de ROS, que integra OpenNI en ROS. El objetivo de la aplicación es realizar el *tracking* de una figura humana en 3D. El primer paso consiste la creación de un nodo de producción de OpenNI llamado *depth generator* , que se encarga de leer periódicamente los datos de un sensor de profundidad, que en nuestro caso sería la Kinect, y generar un mapa con la información obtenida.

Un segundo nodo de producción de OpenNI llamado *user generator* lee constantemente los mapas de profundidad que el nodo *depth generator* le proporciona, y los analiza para generar como resultado datos acerca de cuerpos humanos en la escena. Este nodo de producción está en espera de detectar objetos en el mapa de profundidad cuya forma corresponda a la de un cuerpo humano. Cuando se detecta a un posible ser humano, el nodo de producción le asigna un identificador numérico único y lo considerará a partir de ese momento como un usuario.

Una vez que un usuario ha sido detectado, el nodo de producción realiza el *tracking* de esa persona. Si en el mapa de profundidad M_t el nodo *user generator* tiene detectados *n* usuarios, al recibir el siguiente mapa de profundidad M_{t+1}, se buscará la posición de cada uno de esos *n* usuarios. Para cada usuario U_i presente en M_t, se realiza una búsqueda en M_{t+1} para encontrar cuál de los objetos presentes en el nuevo mapa corresponde a U_i. Es posible que del instante *t* al *t+1* el usuario U_i haya realizado algún movimiento, lo que implica que M_{t+1} se debe buscar al objeto que se parezca más a la descripción de U_i en M_t. Si la forma de U_i en M_t es muy diferente a la que toma en M_{t+1} es posible que ocurran errores de identificación.

El *tracking* que se realiza del usuario se obtiene a partir de puntos clave e su cuerpo a los que OpenNI denomina *joints* (articulaciones). En la Figura 52 podemos observar cuáles son esos puntos. Cada uno de estos puntos tiene asociada una coordenada en el espacio tridimensional, correspondiente a su ubicación en la escena 3D. En el sistema de coordenadas rectangulares utilizado por la aplicación, el origen corresponde O corresponde al lugar donde se encuentra el sensor de profundidad, el eje horizontal es X, el eje vertical es Y, y la profundidad corresponde al eje Z.

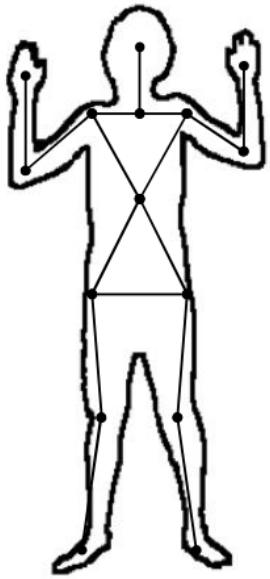

Figura 52 - Esqueleto determinado por Openni

7.5 Cuestionarios de evaluación

Evaluación P3DX_tracker

* Required

Experimento 1

Las siguientes preguntas hacen referencia al primer experimento, en el que una vez reconocido por el sistema, el usuario debía desplazarse hacia atrás y hacia delante, para comprobar que el robot realiza el seguimiento correctamente

1.- ¿Se ha realizado el experimento con éxito? *

- [ ] Sí
- [ ] No

2.- En caso negativo, indique las razones *

3.- ¿Cuántos intentos se han realizado? *

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] No lo he terminado

4.- ¿Cómo valora el tiempo que ha tardado el sistema en empezar a seguirle? *

	1	2	3	4	5	
Muy lento	◯	◯	◯	◯	◯	Muy rápido

5.- ¿En cuantas ocasiones el robot ha dejado de seguirle?

- ☐ Ninguna
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ Más de 3

6.- Califique el proceso de calibración, teniendo en cuenta su rapidez y fiabilidad

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

Experimento 2

Las preguntas propuestas a continuación, hacen referencia al segundo experimento, donde se le pidió que una vez identificado por el robot, debía completar un circuito esquivando obstáculos.

7.- ¿Se ha realizado el experimento con éxito? *

- [ ] Sí
- [ ] No

8.- En caso negativo, indique las razones *

9.- ¿Cuántos intentos se han realizado? *

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] No lo he terminado

10.- ¿Cuántas veces se ha perdido el robot durante el experiemento? *

- [ ] Ninguna
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] Más de 3

11.- ¿Cómo valora la capacidad de reacción del robot antes sus movimientos? *
Es decir, el tiempo entre que usted realiza un movimiento y el del robot

	1	2	3	4	5	
Muy lento	◯	◯	◯	◯	◯	Muy rápido

12.- ¿Cómo valora la capacidad para evitar obstáculos que tiene el robot? *
En cuanto al tiempo tardado en realizarlo

	1	2	3	4	5	
Muy lento	◯	◯	◯	◯	◯	Muy rápido

Experimento 3

Las siguientes preguntas hacen referencia al tercer experimento, donde se ha presentado un escenario en el que dos usuarios se han interpuesto en usted y el robot durante un periodo de tiempo.

13.- ¿Se ha realizado el experimento con éxito? *

- [ ] Sí
- [ ] No

14.- En caso negativo, indique las razones *

15.- ¿Cuántos intentos se han realizado? *

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] No lo he terminado

16.- En el momento que han aparecido los usuarios, ¿el robot ha dejado de seguirle a usted para seguir a otro usuario?

- [ ] Sí
- [ ] No

17.- ¿Cuántas veces ha dejado de seguirle el robot?

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ Más de 3

18.- ¿El robot ha esquivado correctamente a los usuarios que se han interpuesto en su trayectoria?

- ☐ Sí
- ☐ No

Preguntas generales

19.- ¿Cuál de las tres alternativas considera que ha sido más eficiente para el experimento 1?

- [ ] Reconocimiento mediante forma
- [ ] Reconocimiento mediante color
- [ ] Reconocimiento mixto

20.- ¿Cuál de las tres alternativas considera que ha sido más eficiente para el experimento 2?

- [ ] Reconocimiento mediante forma
- [ ] Reconocimiento mediante color
- [ ] Reconocimiento mixto

21.- ¿Cuál de las tres alternativas considera que ha sido más eficiente para el experimento 3?

- [ ] Reconocimiento mediante forma
- [ ] Reconocimiento mediante color
- [ ] Reconocimiento mixto

22.- ¿Cuál considera que es el experimento más difícil de realizar? *

- [ ] Experimento 1
- [ ] Experimento 2
- [ ] Experimento 3

23.- ¿Cree que el sistema sería de utilidad para ayudar a las personas en ciertas tareas? *

- [ ] Sí
- [ ] No

24.- En caso afirmativo, ¿podría nombrar alguna aplicación práctica? *

25.- En caso negativo, ¿qué problemas observa en el sistema? *

26.- ¿Qué valoración final le daría al sistema 1? *
Reconocimiento por color

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Muy bueno

27.- ¿Qué valoración final le daría al sistema 2? *
Reconocimiento por forma

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Muy bueno

28.- ¿Qué valoración final le daría al sistema 3? *
Reconocimiento mixto

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Muy bueno

29.- ¿Qué mejoras incluiría en el sistema?

30.- Observaciones

BIBLIOGRAFÍA

1] Karel Capek. *Rossum's Universal Robots*. Science-fiction novel, 1921.

[2] Rafael Aracil, Carlos Balaguer, Manuel Armada. *Robots de servicio*. Revista Iberoamericana de Automática e Informática industrial (RIAI).

[3] Gen Endo, Masatsugu Iribe, Atsushi Tani, Toshio Takubo, Edwardo F. Fukushima, and Shigeo Hirose. *Study on a practical robotic follower to support daily life -development of a mobile robot with 'hyper-tether' for home oxygen therapy patients*. Vol. 23, pages 316 – 323. Fuji Technology Press, 2009.

[4] M. Kristou, A. Ohya, and S. Yuta. *Panoramic vision and lrf sensor fusion based human identification and tracking for autonomous luggage cart*. In Robot and Human Interactive Communication, 2009. RO-MAN 2009. *The 18th IEEE International Symposium on*, volume 27, pages 711 –716, October 2009.

[5] Òscar Franco, Xavier Perez-Sala, Cecilio Angulo. *Identificación y seguimiento de personas usando Kinect por parte de un robot seguidor.*

[6] Isaac Asimov. *Runaround*. Science-fiction novel, 1942.

[7] Grzegorz Cielniak y Tom Duckett. *People Recognition by Mobile Robots*. En *Journal of Intelligent and Fuzzy Systems*, 15, pages 21-27, 2004.

[8] C. Schlegel, J. Illman, K. Jaberg, M. Shuster y R. Wörz. *Vision based person tracking with a mobile robot*. En *Proceedings of the Ninth British Machine Vision Conference*, Reino Unido, pages 418-427, 1998.

[9] H. Sidenbladh, D. Kragić y H. I. Christensen. *A Person Following Behaviour for a Mobile Robot*. En *IEEE International Conference on Robotics and Automation*, Vol. 1, pages 670-675, 1999.

[10] Samir, Shacker, Jean J. Saade y Daniel Asmar. *Fuzzy Inference-Based Person-Following Robot.* En *International Journal of Systems Applications, Engineering & Development.* Issue 1, Vol 2, pages. 29.34, 2008

[11] Michael Montemerlo, Joelle Pineau, Nicholas Roy, Sebastian Thrun y Vandi Verma.

Experiences with a mobile robotic guide for the elderly. En *Proceedings of the National Conference of Artificial Intelligence*. pages. 587-592, 2002.

[12] Jörg Stückler, David Dröschel, Kathrin Gräve, Dirk Holz, Michael Schreiber y Sven Behnke. *NimbRo@Home 2011 Team Description*. En http://www.nimbro.net/, 2008

[13] Nicola Belloto y Huosheng Hu. *People tracking with a mobile robot: a comparison of Kalman and Particle Filters*. En *Proceedings of the 13th IASTED International Conference Robotics and Applications*. Alemania, pages. 388-393, 2007

[14] David Beymer y Kurt Konolige. *Tracking People from a Mobile Platform*. En *IJCAI-2001 Workshop on Reasoning with Uncertainty in Robotics*, Estados Unidos, 2001

[15] José Alberto Méndez-Polanco, Angélica Muñoz-Meléndez y Eduardo F. Morales-Manzanares. *Detection of Multiple People by a Movile Robot in Dynamic Indoor Environments*. En *Advances in Artificial Intelligence – Iberamia 2010.* Springer, Alemania, pages. 522-531, 2010

[16] Arturo de la Escalera Hueso. *Visión por computador: fundamentos y métodos*. Ed. Prentice Hall.

[17] PrimeSense. www.primesense.com

[18] Especificaciones de Kinect. http://msdn.microsoft.com/en-us/library/jj131033.aspx

[19] Pioneer 3 Operations Manual.

[20] Web oficial de ROS. www.ros.org

[21] Especificaciones de ROS en Willow Garege http://www.willowgarage.com/pages/software/ros-platform

[22] IEEE recommended practice for software requeriments especifications.

[23] OpenNI User Guide.

[24] Jon Aristondo Etxeberria. *Algoritmo de reconocimiento de forma y color para una plataforma robótica*. Tesis de Máster, 2010.

[25] Arturo Rodríguez García. *Implementación de la prueba Follow Me del concurso RoboCup at Home utilizando modelos de diálogo y arquitectura cognitiva*. Tesis, 2012.

[26] Barry Boehm, *A Spiral Model of Software Development and Enhancement*, ACM SIGSOFT Software Engineering Notes, ACM, 1986.

[27] Web oficial de la Federación Internacional de Robótica, http://www.ifr.org