uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Electrónica Industrial y Automática

Trabajo Fin de Grado

"Aplicación de Algoritmos Bioinspirados Basados en Visión por Computador Para la Detección de Equilibrio de Robots Humanoides"

Miguel García Domínguez

Tutor

Juan Miguel García Haro

3 de Julio de 2019, Leganés

RESUMEN

Existen gran cantidad de estudios que señalan la importancia de la visión en seres humanos para el control del equilibrio. El objeto de este proyecto es sentar las bases y analizar la viabilidad de la utilización de sistemas bioinspirados basados en visión por computador como herramienta para el control del equilibrio de robots humanoides.

El presente proyecto realiza una revisión del estado del arte y analiza qué algoritmos existentes y disponibles utilizados para la obtención de la posición en tiempo real mediante visión por computador resultan interesantes de cara a ser implementados para control del equilibrio. Optándose por implementar el algoritmo Fovis de odometría visual y el método iterativo de resolución del problema "Perspective-n-Point".

Se han realizado una serie de pruebas con el robot humanoide TEO de la Universidad Carlos III de Madrid. Los resultados obtenidos por ambos algoritmos fueron comparados entre ellos y con los obtenidos por el sensor inercial actualmente utilizado por TEO para tareas de control de equilibrio.

Los resultados obtenidos por Fovis en las pruebas llevan a concluir que la odometría visual representa una opción a tener en cuenta para el control del equilibrio de robots humanoides, abriendo una vía a futuros trabajos.

Este trabajo forma parte de un proyecto más amplio que busca utilizar el robot humanoide TEO para imitar el comportamiento de un camarero, para lo cual el control de la postura corporal y el equilibrio son factores de gran relevancia.

Palabras clave: Bioinspirado, Robots humanoides, Control de equilibrio, Estimación de posición, Visión por computador, Odometría visual, "Perspective-n-Point".

ABSTRACT

There exist many studies that evidence the important role that vision plays on balance control in human beings. The aim of this project is to lay the groundwork and analyze the viability of using bioinspired systems based on computer vision in order to control balance on humanoid robots.

This Project makes an state of art revission and analyzes wich of the existant and available real time pose estimation algorithms fit for being used on balance control tasks on humanoid robots. Odometry visual algorithm Fovis and iterative "Perspective-n-Point" problem solver were chosen to be implemented.

Several tests were done in humanoid robot TEO of University Carlos III of Madrid. A comparission between both method's results was done and also with current inertial sensor used by TEO for balance control tasks.

The results obtained by Fovis lead to conclude that visual odometry represents a good option for balance control tasks on humanoid robots, laying the ground for further work.

This work is carried out within the framework of a bigger project that looks for using humanoid robot TEO to imitate a waiter's behavior, for that objective pose and balance control are quite relevant issues.

Palabras clave: Bioinspired, Humanoid Robots, Balance control, Pose estimation, Computer vision, Visual Odometry, "Perspective-n-Point".

AGRADECIMIENTOS

Quiero agradecer a mi madre por el inmenso apoyo que ha sido para mí, no solo durante la realización de este proyecto, sino durante toda la carrera. Muchas gracias por apoyarme y por creer siempre en mí, no hubiera sido capaz sin ti.

También quiero agradecer a mi pareja, Romina, que me ha acompañado durante toda la carrera, y sobre todo durante la realización de este proyecto, ayudándome y apoyándome en todo momento.

Finalmente quiero agradecer a mi tutor, Juan Miguel, por toda la ayuda y la guía durante la realización del proyecto.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

La robótica humanoide es un campo de investigación en gran crecimiento. La realización de robots humanoides que imiten a los humanos en todos los aspectos posibles abre muchas posibilidades de cara al futuro. Se pueden desarrollar robots para realizar tareas en colaboración con humanos [1], ayudar en el tratamiento de niños con autismo [2] o incluso ser utilizados en misiones espaciales [3], entre muchas otras posibilidades. Por tanto, el desarrollo de la robótica humanoide resulta de gran interés e importancia.

Los robots humanoides son concebidos para llevar a cabo tareas que son realizadas por personas, lo cual puede involucrar manejo de herramientas, vehículos o equipamientos diseñados para humanos. Por tanto, es imprescindible que los robots humanoides imiten a los humanos, no solo en la apariencia física, sino también en todos los aspectos de nuestro comportamiento posibles. Se podría decir que el ejemplo a seguir para la creación de robots humanoides somos los propios humanos.

Para llegar a desarrollar robots humanoides realmente cercanos a nosotros, es necesario implementar en ellos sistemas bioinspirados que imiten la forma en que nuestro cerebro y cuerpo funcionan en todos los aspectos. Así mismo, a la hora de plantear posibles soluciones a problemas que surjan en el desarrollo de dichos robots, inspirarse en el modelo humano puede ser una buena solución.

Uno de los problemas planteados en el desarrollo de robots humanoides es el mantenimiento del equilibrio y control postural ante estímulos externos. Conociendo la influencia de la visión en el control del equilibrio en seres humanos, resulta interesante analizar la posibilidad de incorporar sistemas de visión por computador para ayudar en dicha tarea en robos humanoides.

1.1 Objetivos

El objeto principal de este proyecto es dar los primeros pasos de cara a mejorar la precisión del sistema de equilibrio del robot humanoide TEO (Task Enviroment Operator) de la universidad Carlos III mediante la utilización de visión por computador. Al ser un primer paso, este proyecto dará una gran relevancia a la investigación. Se podría separar el objetivo principal en cuatro objetivos menores:

- Revisión del estado del arte relativo a la obtención de la pose mediante visión por computador: Existe un estado del arte muy amplio para la solución del problema de la obtención de la pose mediante visión por computador. Se buscará profundizar en el estado del arte existente y clasificar las opciones disponibles. De esta revisión del estado del arte se seleccionan aquellos métodos que se ajusten de mejor manera a los objetivos y condiciones del presente proyecto.
- Implementación de los algoritmos seleccionados en el robot humanoide TEO: Una vez seleccionados los algoritmos a utilizar, estos serán implementados en TEO. Dichos algoritmos serán modificados para que se adecúen a las necesidades del proyecto y que la información que aporten sean los ángulos de rotación de la cabeza de TEO respecto a los ejes XYZ.
- Realizar un control en lazo abierto para comparar el resultado obtenido por los algoritmos de visión con la información aportada por el IMU (sensor de referencia inercial). También se realizará un programa para poder comparar la eficiencia de los diversos algoritmos.
- Análisis y comparación de resultados: Una vez las diferentes pruebas hayan sido realizadas, se pondrán en contraste los resultados obtenidos por cada algoritmo y así se podrán extraer conclusiones del trabajo realizado.

1.2 Marco socioeconómico

Para analizar el impacto tanto social como económico de la robótica humanoide hay que hablar en futuro. Si bien es cierto que actualmente la robótica industrial ya juega desde hace años un papel muy importante a nivel económico, gracias al proceso de actualización industrial conocido como cuarta revolución industrial [4], el desarrollo de la robótica humanoide aún tiene un largo camino que recorrer.

Pese a que aún queda mucho por hacer en este campo, el ritmo al que evoluciona la robótica humanoide es realmente vertiginoso. Si hace relativamente pocos años no era posible imaginarse la existencia de robots humanoides, en vista de la calidad de robots humanoides como Asimo (desarrollado por Honda) [5], Toro (desarrollado por el centro aerospacial alemán) [6] o Atlas [7], el conocido robot desarrollado por Boston Dynamics Fig 1.1, es claro que la robótica humanoide va a jugar un papel muy relevante en un futuro no demasiado lejano. De hecho este tipo de robots empiezan a adquirir gran popularidad a nivel social, hasta el punto de que en la actualidad los talleres de robótica infantiles son muy comunes.

Fig. 1.1 El robot Atlas, capaz de correr, saltar y esquivar obstáculos, hacer volteretas...todo ello con una gran precisión.

Este proyecto está centrado en la investigación, por lo cual es ciertamente difícil analizar cuál es su impacto social y económico, lo que queda claro es que todo el campo de investigación en robótica humanoide acabará resultando fundamental para el futuro.

Al ser este un trabajo con un gran peso de investigación en visión por computador, se debe analizar también el impacto que este área produce a nivel social y económico. Y es que la visión por computador sí tiene una gran influencia en el presente. Se trata, por ejemplo, de una herramienta fundamental para el desarrollo del coche autónomo [8]. Importantes marcas como Tesla utilizan la visión por computador en el desarrollo de sus vehículos autónomos. Sin salir del sector automovilístico, la visión por computador es también muy utilizada como herramienta de seguridad activa en vehículos modernos [9].

Otra implementación muy importante de la visión por computador en la actualidad es la seguridad, pudiéndose utilizar por ejemplo para detectar rostros de personas, caso conocido es el desbloqueo por rostro de teléfonos móviles.

1.3 Marco regulador

Este trabajo está centrado en la investigación, encontrándose en un ámbito académico, por lo tanto, no hay un marco regulador claro que lo afecte. En cualquier caso, siendo el robot humanoide TEO un robot asistencial, cabe mencionar la existencia de la norma internacional ISO 13482:2014 la cual afecta a robots no industriales y a robots asistenciales no médicos.

Esta normativa señala los requisitos de seguridad a cumplir por los robots asistenciales. En concreto, en relación al trabajo realizado en este proyecto, cabe mencionar los capítulos 5.16 “Peligros debidos a errores de posicionamiento y navegación” y 6.6 “ control de estabilidad”.

En el capítulo 5.16 de la normativa, se expone la incertidumbre en la localización del robot no debe en ningún caso conducir a riesgos inaceptables. La capacidad de navegación de un robot asistencial debe ser lo suficientemente precisa como para planificar un movimiento hacia cualquier meta. Se deben aplicar medidas que aseguren lo anterior, así, si se utilizan puntos de referencia para obtener la localización, estos deben ser suficientes e inequívocos para poder ser detectados en todo momento del desplazamiento.

En el capítulo 6.6 de la normativa se especifica que el robot asistencial debe ser estable en todas las situaciones de uso previstas o razonablemente previstas. Al ser TEO un robot de grandes dimensiones y peso corporal, se exigen unos altos requisitos (nivel de rendimiento "d").

Cabe mencionar, que en un marco legislativo, referente a la influencia que la robótica puede llegar a tener en la sociedad en los próximos años, el parlamento europeo en 2017 un informe [10] en el que se exponía un paquete de legislaciones a implementar en los próximos años respectivos a la robótica y la inteligencia artificial. Este tipo de regulaciones sin duda afectarán de algún modo a la robótica humanoide pues su impacto en el mercado laboral también será muy importante.

1.4 Estructura del trabajo

Este proyecto ha sido estructurado de la siguiente manera:

- En el primer capítulo se expondrán las motivaciones que han movido a la realización de este proyecto, enmarcando el trabajo dentro de un proyecto mayor. En primer lugar se describe la labor que la visión juega en los seres humanos, citando artículos que respalden esa labor. A continuación se expone como se ha estado enfocado hasta ahora el control del equilibrio en robots humanoides y qué lugar ocupa el presente proyecto. Finalmente se realiza una introducción al problema de la obtención de la pose mediante visión por computador, ya que la gran cantidad de algoritmos diferentes motiva a su implementación para la obtención de la pose y control del equilibrio en robots bípedos. En este apartado se lleva a cabo también una primera clasificación a partir de la cual se partirá el proyecto, separando entre algoritmos basados en solucionar el problema conocido como "Perspective-n-Point" y métodos de odometría visual.
- En el tercer capítulo se detalla el hardware y software con el cual se va a llevar a cabo este proyecto, dándole especial relevancia al robot Humanoide TEO. También se expone el presupuesto para la realización de este proyecto.
- En el cuarto capítulo se muestra y desarrolla brevemente un primer algoritmo que se desarrolló para realizar la obtención de la pose. Explicando cual era el objetivo del algoritmo y porque se decidió desecharlo y orientar el proyecto de otra manera.

- En el quinto capítulo se aborda todo lo relativo a los algoritmos basados en resolver el mencionado problema "Perspective-n-Point". Este capítulo se divide en tres partes. En la primera parte se realiza la revisión del estado del arte, explicando en que consiste el problema a resolver, clasificando los enfoques existentes y analizando cuál se ajusta mejor a las condiciones del proyecto. En la segunda parte se detalla la implementación del método seleccionado, este proceso de implementación se separa a su vez en dos partes, la primera parte la calibración de la cámara y la segunda la obtención de los ángulos de Euler que definen la orientación de TEO. Finalmente, la última parte de este capítulo está centrada en detallar en qué consisten las pruebas que se realizaron para analizar la precisión y capacidad de reacción del algoritmo así mismo se muestran y analizan los resultados obtenidos en las diversas pruebas por este algoritmo.
- En el sexto capítulo se aborda todo lo relativo a odometría visual. La estructura de este capítulo semejante a la vista en el capítulo anterior. El primer apartado de este capítulo se centra en explicar en qué consisten los algoritmos de odometría visual, que objetivos buscan, cual ha sido la evolución y como pueden ser clasificados. A continuación, sin salir del estado del arte, se clasifican detalladamente los métodos de odometría visual disponibles para ser utilizados con cámaras RGB-D (como la cámara Kinect que se utiliza en este proyecto) con el objetivo de decidir cual se ajusta mejor a las condiciones existentes. Tras ello se muestran y comentan los resultados obtenidos. Finalmente el último apartado de este capítulo está dedicado a mostrar los resultados que se obtuvieron con el algoritmo FastICP, el cual se quedó fuera del proyecto.
- En el séptimo capítulo se comparan los resultados implementados decidiendo cual es el algoritmo más apropiado de cara a realizar un control del equilibrio mediante visión. Finalmente en el capítulo de conclusiones, comparación se exponen las conclusiones e ideas obtenidas con este proyecto. De igual manera, se ahonda en cuales son los siguientes pasos a llevar a cabo y que futuros proyectos pueden nacer de este.

2 MOTIVACIONES

Este proyecto se realiza dentro de un proyecto más amplio, el cual busca utilizar al robot humanoide TEO para desempeñar la labor de un camarero [11]. Para realizar esta tarea el sistema de equilibrio de TEO tiene que tener una gran precisión, pues debe llevar una botella sobre una bandeja como se ve en la Fig 2.1. La motivación principal de este proyecto es analizar la viabilidad de utilizar algoritmos de visión para mejorar el control del equilibrio de TEO y en robots humanoides en general.

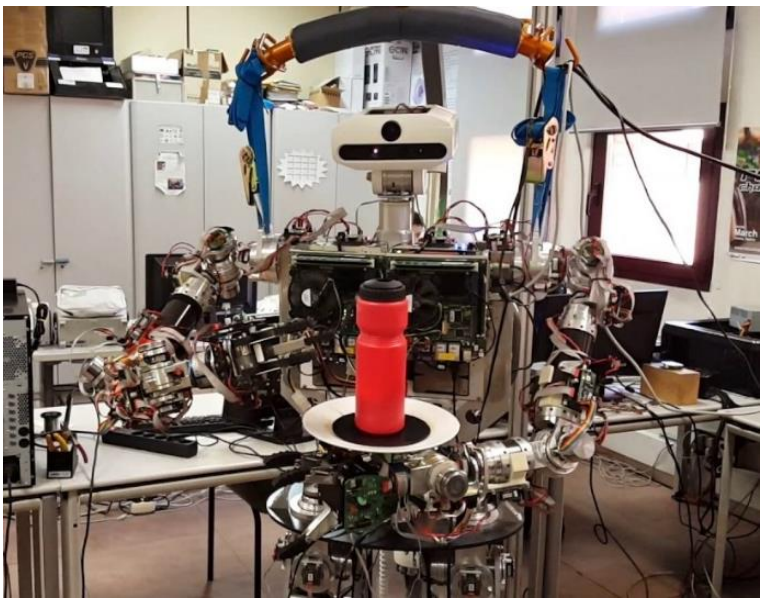

Fig. 2.1 TEO camarero

2.1 Papel de la visión en el equilibrio en seres humanos.

El sistema vestibular, formado por órganos internos del oído, es el principal encargado de ayudar a mantener el equilibrio y conocer la orientación y posición de nuestra cabeza. Sin embargo, el sistema vestibular no es el único sistema involucrado en el control del equilibrio, la visión es también una herramienta importante para el control del mismo.

Se han realizado estudios que demuestran en base a personas mayores, que una pérdida de visión afecta significativamente a la capacidad de mantener el equilibrio aumentando el riesgo de caída [12][13][14]. En concreto, todo parece indicar que la visión es la herramienta principal que los humanos utilizamos para mantenernos erguidos en situaciones estáticas [15].

Especialmente se menciona la importancia para esta labor que tiene la visión periférica, que es la parte de la visión que abarca aquello a lo que no se está prestando atención, al detectar el flujo óptico [16] (movimiento aparente de superficies y bordes de una escena causado por el movimiento relativo entre espectador y escena). La detección de este movimiento sería la que ayuda a poder corregir la postura buscando posiciones de equilibrio.

2.2 Equilibrio en robótica humanoide

Existe un amplio estado del arte respectivo al control del equilibrio y estabilidad en robots bípedos humanoides. Se pueden encontrar un gran número de enfoques distintos. Los enfoques principales suelen estar basados en la utilización de indicadores tales como FRI ("Foot Rotation indicator") [17], ZMP ("Zero Moment Point")[18] [19], la velocidad de variación del momento angular [20] o en la utilización de redes neuronales [21].

Estos enfoques han demostrado buenos resultados, sin embargo, se basan principalmente en el control del equilibrio dinámico, es decir, equilibrio durante el movimiento del robot. Sin embargo en situaciones estáticas la herramienta que se suele utilizar para conocer la pose en robots humanoides es el sensor inercial (IMU), como es el caso de TEO [22]. Este tipo de sensores, sin embargo, suelen presentar problemas relacionados con la acumulación de error con el tiempo [23], siendo necesario, por tanto, buscar diversas alternativas.

Como se mencionó previamente, la visión juega un papel fundamental en el mantenimiento del equilibrio en situaciones estáticas, ayudando a mantenernos erguidos. Conociendo la gran influencia de la visión en el control del equilibrio en seres humanos y existiendo en el estado del arte de la visión por computador diversas maneras de obtener la posición, resulta muy interesante analizar la viabilidad de utilizar la visión en sustitución del IMU para llevar a cabo un control más óptimo del equilibrio en situaciones estáticas.

Este enfoque bioinspirado busca no depender de ningún modo del sensor inercial, es decir, solo se utilizará aquella información proveniente de la cámara, dejando enfoques mixtos para futuros trabajos.

2.3 Estimación de la orientación de la cámara en tiempo real mediante visión por computador.

La estimación de la posición consiste principalmente en obtener la matriz de transformación entre el sistema de referencia del mundo o de un objeto y el sistema de referencia de la cámara. Se trata de un problema muy recurrente en el campo de la visión por computador. Normalmente el objetivo que se busca en las aplicaciones del estado del arte es el de localizar la orientación de un objeto determinado y conocido respecto a la cámara. Por ejemplo en el campo de la robótica es muy útil la utilización de aplicaciones de visión para detectar la orientación de objetos para así poder ser manipulados por el robot. Un ejemplo concreto podemos encontrarlo en [24], donde utilizando un patrón ArUco (Fig 2.2) se detecta la posición, no solo del objeto, sino también de la propia mano, para así poder aproximarse al objeto de la manera correcta, tal y como se puede ver en las Fig. 2.3 y 2.4.

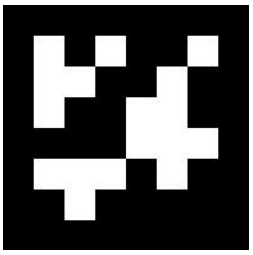

Fig. 2.2 Ejemplo de patrón ArUco

Fig. 2.3 Posición inicial, el robot humanoide gracias al patrón ArUco conoce la posición de su mano y también del objeto a coger.

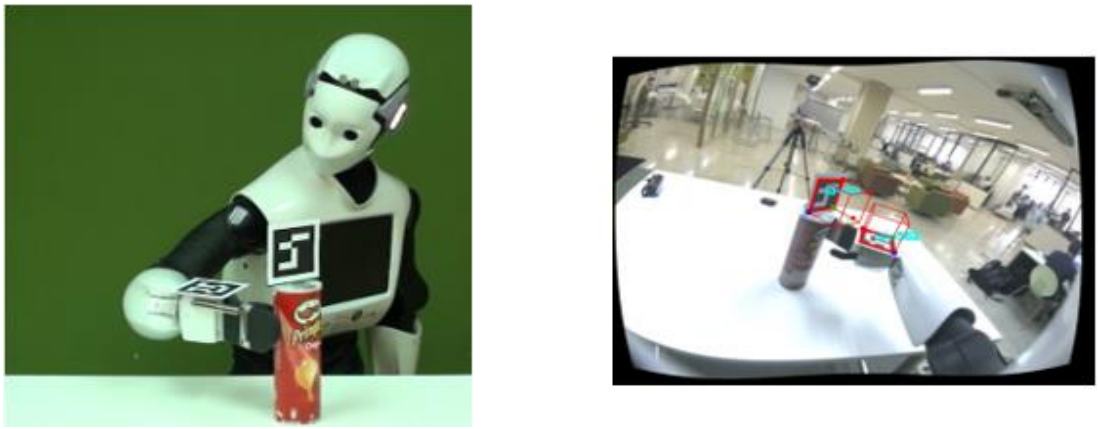

Fig. 2.4 Posición final, una vez se ha acercado la mano al objeto con la orientación adecuada para cogerlo

*E*ntre los algoritmos clásicos de obtención de la posición mediante visión por computador podemos encontrar algunos basados en algoritmos genéticos [25], redes neuronales [26] o conocer la proyección de una serie de puntos 3D en el plano de la imagen (el ya mencionado problema "Perspective-n-Point").

Los algoritmos genéticos presentan el problema de que no se pueden utilizar en tiempo real, por lo que no resultan útiles para el objetivo del presente proyecto. Por otro lado, los algoritmos basados en redes neuronales presentan gran complejidad computacional y requieren de escenarios con buenas condiciones para identificar los objetos de referencia, por lo que no resultan de gran interés de cara a la aplicación que se busca desarrollar, aunque en futuros proyectos pueden ser probados.

Los métodos basados en "Perspective-n-point" son una opción interesante, ya que no tienen un gran coste computacional y son el enfoque más estudiado y utilizado en el estado del arte.

Por otro lado, el avance en los campos de vehículos y robots autónomos ha impulsado en gran medida el estudio de SLAM [27], es decir, localización y mapeado simultáneo (Simultaneous Localization and Mapping). Los algoritmos SLAM buscan principalmente mapear un entorno desconocido, y al mismo tiempo encontrar la localización de un robot o vehículo en dicho entorno sin disponer de referencias concretas en el mismo. Originalmente los algoritmos de SLAM no utilizaban la cámara como sensor principal,

debido a que exigía un gran coste computacional. Sin embargo, con el avance tanto en las cámaras, especialmente cámaras RGB-D, como en el resto de hardware involucrado, se están desarrollando más algoritmos de V-SLAM (Visual SLAM) basados en visión [28][29] o que incorporan la cámara con otros sensores [30].

Separando la parte de localización en dichos algoritmos de la parte de mapeado surgen los algoritmos de Odometría visual [31][32]. Estos se basan principalmente en comparar una nube de puntos o frames con otro anterior, para así estimar la transformación entre ambos y actualizar la posición estimada.

Por tanto la utilización de odometría visual resulta de gran interés para este proyecto, pues es la única forma de obtener la orientación de la cabeza de TEO en tiempo real y sin necesidad de referencias externas, lo cual resulta la forma más aproximada de imitar el papel de la visión humana.

Existen multitud de algoritmos de odometría visual diferentes. En este caso, por razones que se exponen en el capítulo 6 se ha decido implementar el algoritmo Fovis.

Fovis [23] está basado en imágenes RGB y utiliza un enfoque centrado en puntos característicos de la imagen. Es un algoritmo accesible, eficiente computacionalmente y que obtiene muy buenos resultados, como se puede ver en la evaluación realizada en teléfonos móviles en [33].

3 HARDWARE Y SOFTWARE UTILIZADO

En este capítulo se exponen todas las herramientas que se han utilizado en el desarrollo del proyecto, tanto hardware como software.

3.1 Hardware

En cuanto a Hardware, se especifica la información referente al robot humanoide TEO, la cámara Kinect y IMU (unidad de medida inercial).

3.1.1 Robot Humanoide TEO

Como ya se ha mencionado anteriormente, el presente proyecto se enmarca dentro de un proyecto más amplio que busca utilizar al robot TEO para simular el comportamiento de un camarero.

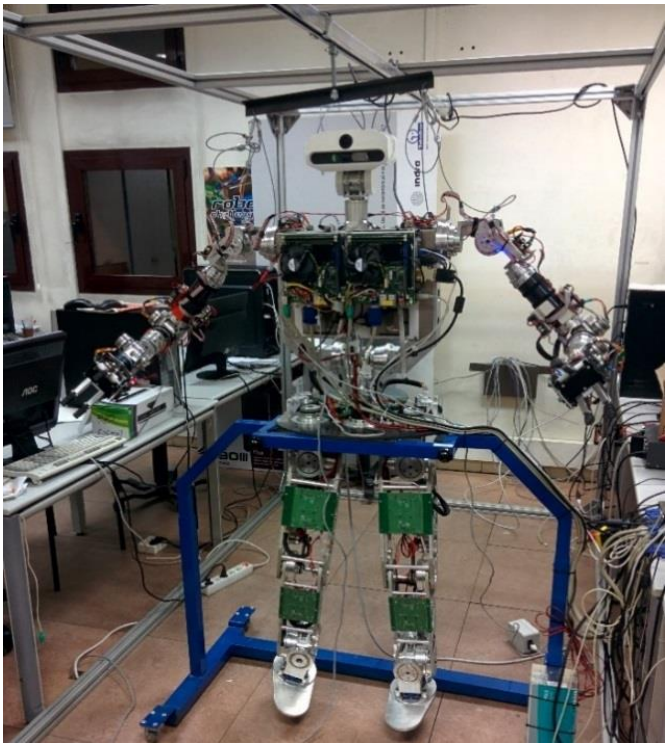

Fig. 3.1 Robot humanoide TEO

TEO es un robot humanoide completo creado por el grupo de investigadores Robotics Lab, evolución del robot Rh-0. Sus características físicas son similares a las de una persona adulta, midiendo 1.65 metros de altura y con un peso de 60 kg.

Una de las principales características de TEO es que tiene 28 grados de libertad, distribuidos de la siguiente forma:

- Cada pierna tiene 6 grados de libertad; La articulación del tobillo puede rotar tanto en el plano frontal como en el sagital; La rodilla por su parte presenta un único grado de libertad en el plano frontal; La cadera puede rotar en los planos frontal, sagital y axial.
- El tronco tiene dos grados de libertad, uno en el plano sagital y otro en el plano axial.
- Los brazos disponen, al igual que las piernas, de 6 grados de libertad: tres grados de libertad en los hombros en los planos axial, sagital y frontal. El codo presenta un único grado de libertad en el plano frontal y la muñeca dos grados de libertad, en planos frontal y axial.
- El cuello puede rotar en los planos axial y frontal.

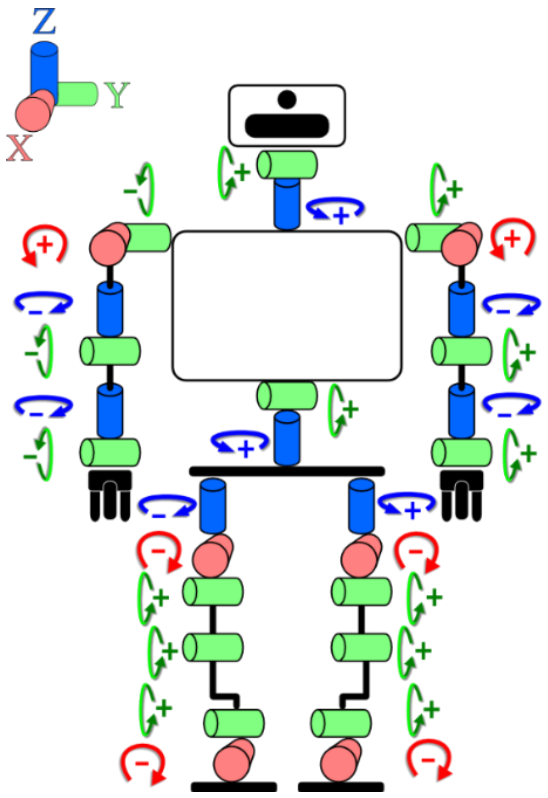

Fig. 3.2 Diagrama de las rotaciones posibles en las diferentes articulaciones de TEO

TEO está compuesto por tres ordenadores: el ordenador de la cabeza, el de manipulación y el de locomoción. El pc de la cabeza cuenta con un procesador Intel i5-4570S. Su principal función es controlar la visión. En este ordenador se han implementado todos los algoritmos de visión. El ordenador de locomoción, con un procesador Intel Core 2 Duo CPU E7500, es el encargado de controlar los actuadores del tronco y de las piernas. Ha sido utilizado para llevar a cabo las pruebas de los algoritmos. El ordenador de manipulación se encarga de controlar los actuadores del cuello y los brazos, por lo que no ha sido utilizado en este proyecto.

Además de los ordenadores de manipulación y de la cabeza, se utiliza un tercer ordenador conectado remotamente desde el cual se lanzan las aplicaciones.

3.1.2 Kinect

TEO tiene montada la cámara Kinect (Fig. 3.3) en el ordenador de la cabeza y es la cámara utilizada en este proyecto.

Fig. 3.3 Cámara Kinect

Kinect es una cámara RGB-D lanzada al mercado en 2010 y desarrollada por Microsoft. En un origen se lanzó como controlador de videojuegos para la consola Xbox 360. Sin embargo, fue una de las primeras cámaras RGB-D que se lanzaron al mercado y como ya se verá más adelante el lanzamiento de estas cámaras aportó mucho a la investigación de SLAM y odometría visual.

El sensor Kinect está compuesto por un proyector laser (izquierda en Fig. 3.3), una cámara infrarroja con resolución 320x240 funcionando a 30fps (centro de Fig. 3.3), y una cámara RGB con resolución 640x480 pixeles (derecha de Fig 3.3).

El láser de Kinect proyecta un patrón (Fig 3.4) el cuál es captado por la cámara infrarroja. En un primer momento, el patrón se proyecta sobre un plano cuya distancia de la cámara es conocida. Este patrón se almacena en la cámara para servir como patrón de referencia.

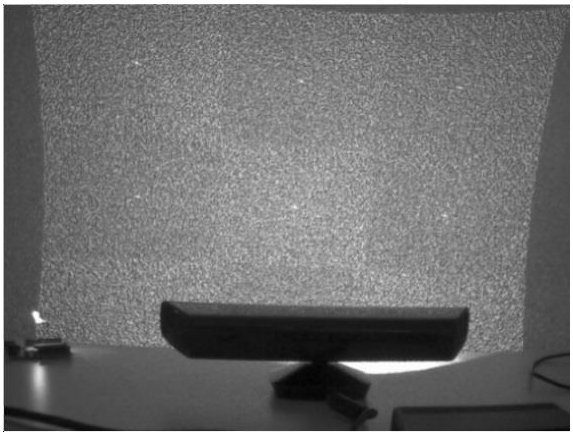

Fig. 3.4 Patrón proyectado por kinect en una pared cuya distancia a la cámara es conocida [36]

Cuando se proyecte el patrón a un objeto cuya distancia sea diferente a la utilizada para almacenar el patrón, dicho patrón se encontrará distorsionado con respecto al patrón de referencia. Para captar la distancia Kinect utiliza correlación, relacionando ambos patrones y encontrando correspondencias entre ellos para, a continuación, encontrar la distancia a la que se encuentra el nuevo plano.

En la Fig 3.6 se muestra el esquema de la triangulación que se realiza entre el sensor infrarrojo y el proyector laser, siendo o un punto cualquiera del patrón y k dicho punto en el nuevo plano proyectado. Conociendo la distancia entre el láser y el sensor infrarrojo (b), así como la distancia Z_o de la cámara al patrón de referencia y la disparidad (d), se puede obtener mediante triángulos semejantes la distancia de la cámara al nuevo objeto en que se ha proyectado el patrón. De esta manera Kinect consigue obtener la profundidad de los objetos.

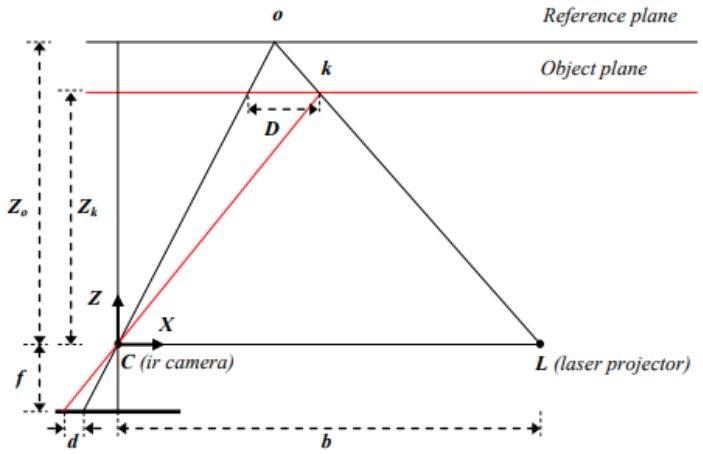

Fig. 3.6 Esquema de la triangulación empleado para obtener la profundidad conociendo la distancia al patrón de referencia [37]

Para poder utilizar el sensor Kinect es necesario utilizar el driver OpenNI.

Fig. 3.5 Logo de OpenNI

3.1.3 IMU

El sensor inercial o IMU es un sensor electrónico que aporta información acerca de la velocidad, orientación y fuerzas gravitatorias de un aparato (de un robot en este caso) gracias a una combinación de acelerómetros y giróscopos. En concreto el giróscopo permite conocer la orientación del robot en los ejes XYZ.

TEO tiene un sensor inercial colocado en su cadera, el cual permite conocer la orientación del torso en tiempo real. El sensor inercial que incorpora TEO es el sensor Xsens MTi-28A53G35, el cual es una unidad de medida adecuada para ser utilizado en robots, vehículos, cámaras… Se caracteriza por un diseño robusto y compacto y fácil de integraren diversos sistemas y aplicaciones.

Como se mencionó previamente, este tipo de sensores acumula error con el tiempo, lo cual hace que su implementación para el control de equilibrio sea problemática.

Fig. 3.7 Xsens MTi-28A53G35

3.1.4 Software

En lo relativo al software, se debe mencionar la plataforma Yarp, el entorno de programación y las librerías de visión por computador utilizadas.

3.1.5 Yarp

Yarp (Yet Another Robot Platform) es la plataforma que permite el control de TEO. Se trata de un software gratuito y abierto diseñado por y para investigadores en robótica humanoide, el cual permite crear un sistema de control a partir de una colección de programas conectados mediante P2P, esto es, una serie de ordenadores conectados como iguales sin existir un cliente o servidor.

Fig. 3.8 Logo de Yarp

El laboratorio de robótica humanoide de la universidad Carlos III está formado por varios ordenadores, desde los cuales se pueden controlar los ordenadores de TEO remotamente y lanzar programas desde ellos, o lanzar programas directamente desde cada ordenador, todo ello se puede llevar a cabo gracias a Yarp. Yarp no es un sistema operativo, sino que es una serie de protocolos y librerías que permiten comunicar los diferentes sensores, procesadores y actuadores que forman a TEO, manteniéndolos completamente separados e independientes, facilitando la evolución del sistema [34].

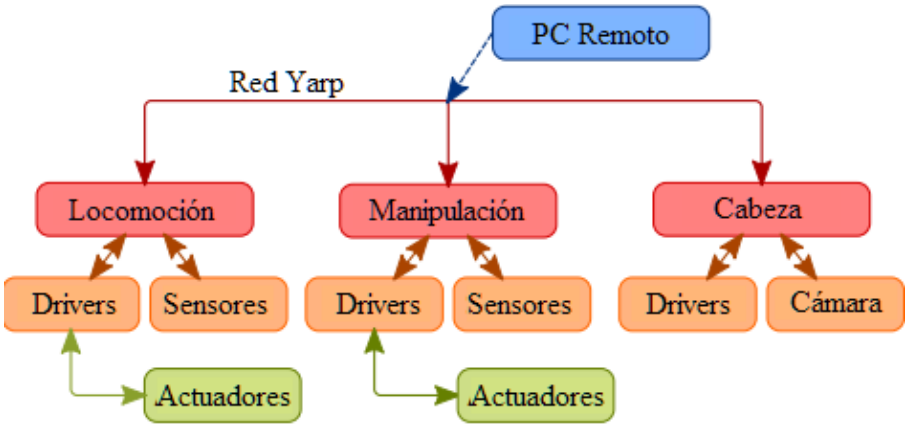

Fig. 3.9 Diagrama de la red Yarp implementada en TEO

Como se observa en la Fig 3.9, tanto desde los tres ordenadores principales (Cabeza, Manipulación y Locomoción), como desde un ordenador conectado a la red remotamente se pueden controlar todos los sensores y actuadores disponibles en TEO.

En este proyecto se utilizan tres ordenadores, tal y como se mencionó en el apartado de hardware. Todos los programas son lanzados desde el ordenador de la cabeza, el cual está controlado con un ordenador remoto. El ordenador de la cabeza se conecta con el ordenador de locomoción y de este modo recibe la información proveniente del IMU y actúa sobre los actuadores controlados por dicho ordenador.

3.1.6 Entorno de programación

La implementación de todos los algoritmos de visión por computador, así como el programa diseñado para comparar la información de los algoritmos de visión está llevado a cabo mediante el lenguaje de programación C++.

La interfaz utilizada para el desarrollo y compilación de los programas fue QtCreator. El presente proyecto se realiza en el sistema operativo Ubuntu 16.04 LTS y se utilizó la herramienta multiplataforma CMake para la compilación de las diversas librerías utilizadas en visión por computador.

Fig. 3.10 Izquierda logo de C++, Centro logo de Qtcreator, derecha logo CMake

3.1.7 Visión por computador.

En cuanto a visión por computador se utilizaron tres librerías fundamentalmente: libFovis, Eigen y OpenCV.

OpenCV

Para implementar el algoritmo basado en "Perspective-n-Point" se utilizó OpenCV. Se trata de una librería de visión artificial desarrollada por Intel, con interfaces en C++, Python y Java. OpenCV incorpora una gran cantidad de funciones diferentes, pudiéndose utilizar para desarrollar una amplia variedad de aplicaciones basadas en visión artificial, como detección de movimiento, reconocimiento facial, calibración de cámaras, realidad aumentada… En concreto se utilizó la versión 3.0.0.

libFovis

Fovis no tiene una implementación en OpenCV. Sin embargo es posible implementarlo utilizando la librería de LibFovis. Esta librería contiene todas las funciones necesarias para implementar Fovis así como una demo que obtiene la posición de la cámara en tiempo real, será esta demo la que se ha utilizado en el presente proyecto.

Eigen

Esta librería se utiliza como complemento imprescindible para poder trabajar con la librería libFovis. Se trata de una librería enfocada principalmente en el desarrollo de operaciones de algebra lineal, algunas de las cuales son utilizadas por libFovis.

.

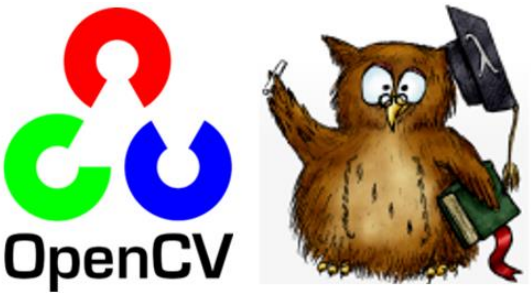

Fig. 3.11 Izquierda logo de OpenCV, derecha logo de Eigen

3.2 Presupuesto

Todo el software utilizado en la realización de este proyecto es de carácter abierto y gratuito, por tanto el coste del proyecto reside únicamente en el hardware y en la realización del propio trabajo.

Concepto	Multiplicador	Unidades	Importe
Hardware			
Pc cabeza (Intel i5-4570S)	N/A	1	450,12 €
Pc Locomoción (Intel Core 2 Duo)	N/A	1	149,99 €
Pc Remoto (Intel Core i3-4170)	N/A	1	256,36 €
IMU Xsens MTi-28A53G35	N/A	1	742,31 €
Kinect V1	N/A	1	39,99 €
Total Hardware	N/A	**1**	1.638,78 €
Mano de Obra			
Búsqueda de información	50 €/h	300	15.000,00 €
Implementación	30 €/h	50	1.500,00 €
Redacción de memoria	50 €/h	150	7.500,00 €
Supervisión del tutor	65 €/h	50	3.250,00 €
Total Mano de Obra	N/A	1	27.250,00 €
Total Proyecto	N/A	1	28.888,78 €

Tabla 1 Presupuesto para la realización del proyecto

4 APROXIMACIÓN PERSONAL AL PROBLEMA DE LA ORIENTACIÓN EN TIEMPO REAL EN VISIÓN POR COMPUTADOR .

En un primer momento el proyecto se enfocó de manera diferente, ya que no se buscaba probar diferentes algoritmos para obtener la posición de la cámara, sino que se intentaba realizar un algoritmo desde cero utilizando las funciones disponibles en OpenCV.

Se buscaba imitar de alguna manera el sistema llamado PHVD ("Peripheral vision horizon display") con el que los pilotos de avión pueden detectar la línea de horizonte para que así el piloto tenga siempre la referencia de su altitud y rotación.

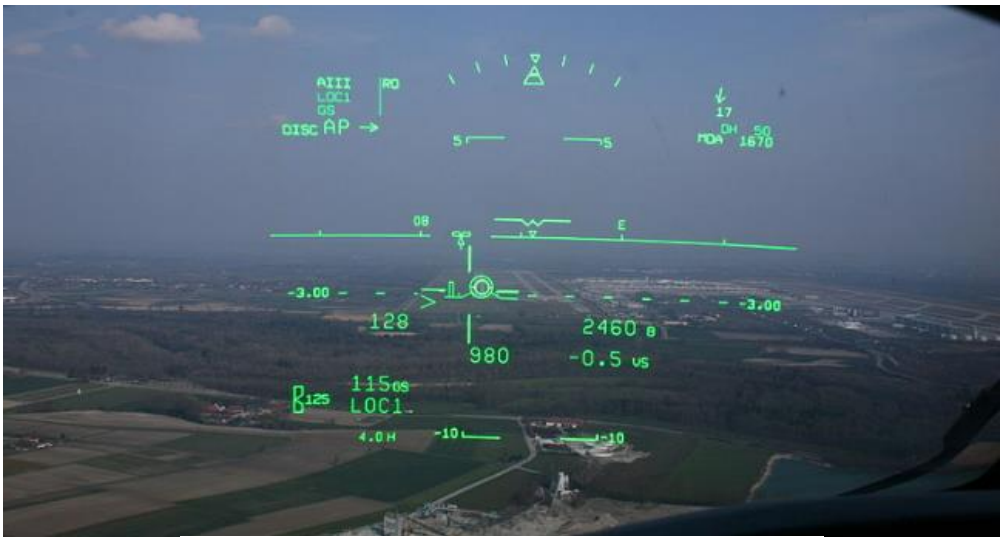

Fig. 4.1 Ejemplo del PHVD, cuya idea se buscó imitar

La implementación de sistemas PHVD en la cabina del piloto responde precisamente a la gran importancia que la visión periférica juega en el control y conocimiento de la orientación en seres humanos y su implementación disminuyó en gran medida la cantidad de accidentes debido a desorientación. Su función principal es mostrar un falso horizonte al piloto, de esta manera mantener una referencia visual cuando las condiciones no permitan ver el horizonte real. El brillo del falso horizonte está regulado de manera que el piloto no pueda verlo cuando se fija en él, sino que lo ve mediante la visión periférica, manteniendo un control inconsciente de la orientación y altitud.

Este tipo de sistemas buscan solucionar las limitaciones del sistema vestibular humano a la hora de detectar desplazamientos verticales, ya que los seres humanos no nos despegamos de la superficie de la tierra al andar, por lo cual el sistema vestibular presenta problemas para detectar desplazamientos que no sean horizontales. Así mismo, dicho sistema vestibular no es capaz de detectar correctamente desplazamientos demasiado largos en el tiempo, por lo que sin referencias visuales resulta muy difícil mantener el sentido de la orientación [35].

El PHVD resultó ser una revolución en la aviación, una herramienta fundamental hoy en día para cualquier piloto. En vista de esto, se consideró que imitar el funcionamiento de los PHVD en robots humanoides podía ser interesante.

La idea que se trató de llevar a cabo consistía en detectar líneas del entorno, seleccionar aquella que se considerase más adecuada para utilizarla como referencia (sería como él horizonte del PHVD) y tomar las coordenadas de inicio y fin de la línea para definirla. A continuación se llevaba a cabo un seguimiento del movimiento de dicha línea, comparando la posición en cada instante con su posición inicial. Mas adelante se buscaría la forma en la que se compararían ambas líneas para obtener la matriz de rotación que exprese la rotación realizada entre imágenes.

4.1 Detección de Línea

Para poder detectar las líneas del entorno se utiliza la transformada “Hough Line Transform”. Esta transformada, a partir de la detección de bordes en la imagen, permite obtener todas las líneas que se encuentran en la imagen detectada por la cámara.

El paso previo y necesario para poder aplicar la transformada “Hough Lines” es aplicar un detector de bordes de la imagen. Este detector mediante el estudio del gradiente de la intensidad en la imagen permite aislar del fondo los objetos presentes en la imagen, lo cual permite obtener los diferentes bordes presentes en la imagen, tal y como se puede ver en la Fig 4.2.

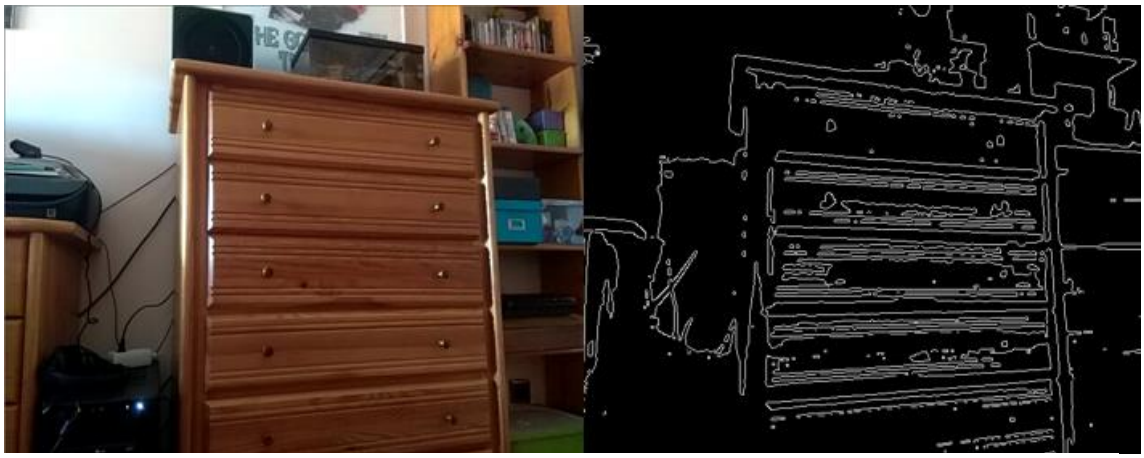

Fig. 4.2 A la izquierda imagen original, a la derecha imagen con el filtro Canny para detección de bordes

Sobre la imagen filtrada con el detector Canny se puede implementar la ya mencionada transformada de Hough. Esta transformada permite detectar todas aquellas líneas rectas presentes en la imagen. La detección de líneas solo se realizaría cada vez que se arranque el programa o que se cambie la línea de referencia.

En Fig 4.3 se pueden ver los resultados de la detección de líneas sobre la imagen 4.2. Cabe mencionar que la precisión de dicha detección no es muy elevada y varía bastante según las condiciones de la imagen.

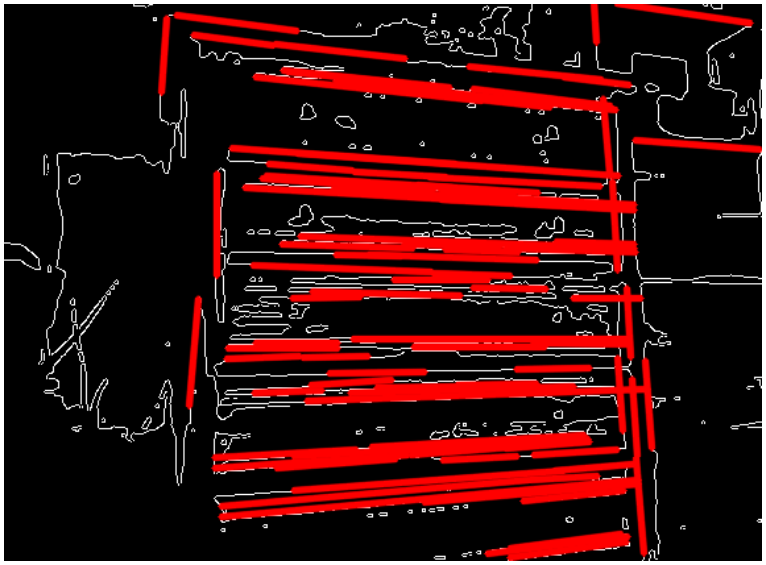

Fig. 4.3 Imagen una vez detectadas las líneas rectas presentes en ella.

4.2 Seguimiento de línea

A continuación el algoritmo selecciona una línea del entorno. La línea de referencia a seleccionar debía ser una línea que se encontrase en la zona central de la imagen, de este modo no se perdería el rastro de la línea a no ser que se realizase un gran desplazamiento.

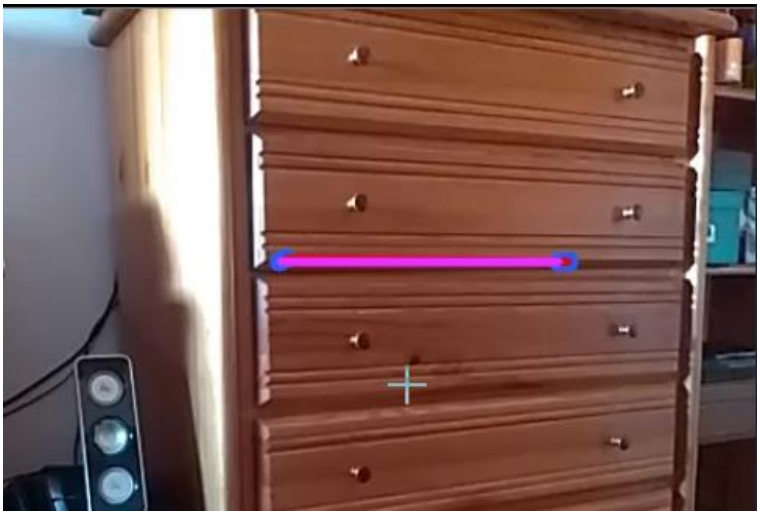

Fig. 4.4 En esta imagen se puede ver una línea seleccionada como línea de referencia.

De este modo la línea debía encontrarse dentro de unos márgenes predefinidos, una vez que se perdiese el rastro de dicha línea se buscaría otra con condiciones semejantes y se acumularía la rotación estimada. Tras seleccionar la línea se marcan sus puntos de inicio y fin (en azul en Fig 4.4), de este modo el seguimiento de esos puntos permitirá realizar el seguimiento de la línea. Una vez se produzca un desplazamiento, la línea rosa se moverá con la cámara, mientras que una segunda línea roja se mantendrá siempre en la posición de referencia, como se puede ver en la Fig 4.5.

Fig. 4.5 Imagen tras realizar dos desplazamientos diferentes, en ambos la línea roja mantiene en la posición de referencia, mientras que la rosa se mueve con la cámara.

El seguimiento de los puntos que definen la línea roja, para poder mantener su posición se realizó utilizando la función “OpticalFlow” de OpenCV. Con esta función se puede realizar un seguimiento de puntos presentes en la imagen, conociendo su posición frame a frame. Gracias a la Kinect se podía obtener el valor de profundidad de los puntos de referencia, expresando los puntos por tanto con coordenadas XYZ. El objetivo final era comparar las líneas en cada instante de tiempo y así obtener la matriz de rotación y de ahí los ángulos de Euler.

4.3 Resultados y conclusiones

En las imágenes mostradas hasta ahora se ha mostrado un funcionamiento correcto del programa, sin embargo los resultados que se obtuvieron resultaron realmente imprecisos y en la mayor parte de las situaciones el programa fallaba, como se mostrará a continuación.

En primer lugar, tanto la detección de líneas como de bordes no resultaban lo suficientemente precisas, dependiendo sus resultados en gran medida de las condiciones de iluminación y del entorno en general, por lo cual sus resultados no eran estables. En las Fig. 4.6 y 4.7 se muestran dos casos en los que la detección de líneas y/o bordes resultó imprecisa

Fig. 4.6 Primer ejemplo de Imagen en la cual se detectaron pocas líneas y bordes del entorno

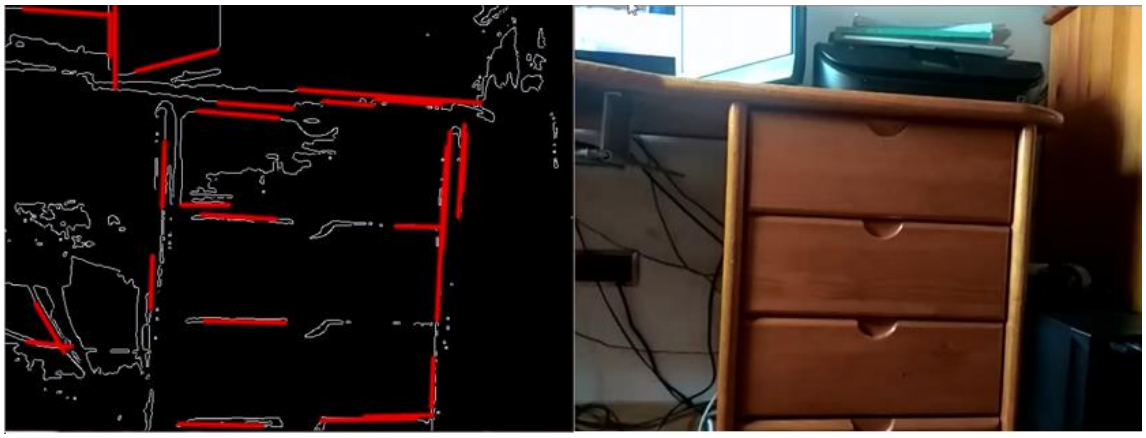

Fig. 4.7 Segundo ejemplo de Imagen en la cual se detectaron pocas líneas y bordes del entorno

Por otro lado, la función encargada de estimar el desplazamiento de los puntos de la línea de referencia también era una fuente de imprecisiones. En muchos casos se perdía posición de la línea de referencia, siendo esta detectada en una posición errónea, como se muestra en la Fig 4.8.

Fig. 4.8 Pérdida de la posición de la línea de referencia (línea roja)

En otros muchos casos la línea cambiada de tamaño o se deformaba tras haberse desplazado, ya que el seguimiento de los puntos que formaban dicha línea fallaba, ejemplo de esto se puede ver en la Fig. 4.9.

Fig. 4.9 Línea de referencia alargada debido al error de la detección

En vista de estos resultados quedó bastante claro que este camino no era una opción adecuada si se buscaba un programa robusto y preciso. Por tanto se decidió optar por desviar el proyecto en otras direcciones que pudiesen realmente satisfacer unos mínimos de funcionalidad. Para ello se decidió orientar este proyecto como una búsqueda entre los algoritmos existentes de la opción más adecuada.

Las conclusiones que se pueden sacar de este primer enfoque fallido es que trasladar la idea de PHVD presente en los aviones a visión por computador no resulta una idea acertada. No resulta adecuado utilizar una única referencia (una línea en este caso) para estimar el desplazamiento producido, por el contrario se deben buscar algoritmos que utilicen un mayor número de referencias lo cual permita una mayor precisión y que además utilicen metodologías más avanzadas.

5 PERSPECTIVE-N-POINT

En este capítulo se aborda todo lo relativo al algoritmo basado en "Perspective-n-Point" que se ha implementado. En primer lugar se ha realizado una revisión del estado del arte de este problema, abordando los diferentes algoritmos existentes para decidir cuál de las opciones disponibles es la más acorde. A continuación se explicará cómo se realizó la implementación del algoritmo. Finalmente se mostrarán y comentarán los resultados obtenidos por el mismo.

5.1 Estado del arte

En esta capitulo se revisará el estado del arte de los diferentes métodos de obtención de la pose mediante "Perspective-n-Point. Abordando en que consiste el problema al que estos algoritmos dan solución y clasificando los diferentes métodos existentes, para así razonar y justificar la decisión de que algoritmo implementar.

5.1.1 Formulación del problema

El problema conocido como "Perspective-n-point" consiste en estimar la posición de una cámara calibrada (es decir, conocidos los parámetros intrínsecos de la misma) conociendo n correspondencias entre puntos de referencia 3D respecto al sistema de referencia del mundo y sus proyecciones 2D en la imagen.

Modelo proyectivo de formación de imágenes.

La forma en la que la cámara forma imágenes en 2D se expresa mediante un modelo matemático proyectivo definido por parámetros extrínsecos e intrínsecos.

Los parámetros intrínsecos (distancia focal, distorsión de la lente…) son parámetros intrínsecos a la cámara con la que se toman las imágenes y se pueden calcular mediante el proceso de calibración [36]. Estos parámetros explican como la cámara afecta sobre la imagen captada. Es decir, si suponemos que sistemas de referencia de cámara y mundo coinciden (Fig 5.1) el modelo matemático estaría únicamente definido por estos parámetros (ecuación 5.1).

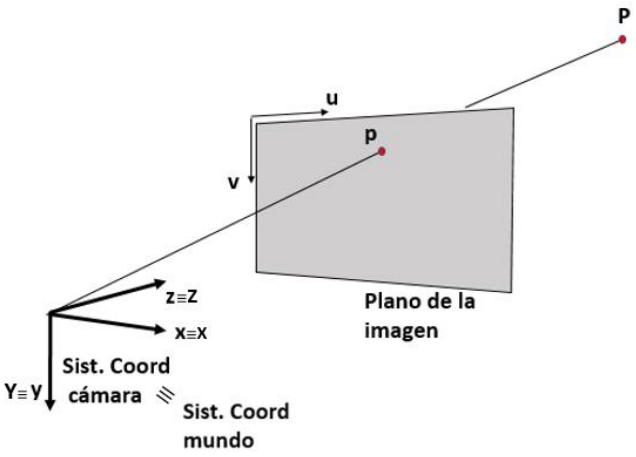

Fig. 5.1 Esquema de los tres sistemas de coordenadas del mundo en el caso en que coincidan sistemas de referencia de mundo y cámara.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (5.1)$$

Siendo la matriz de la ecuación 5.1 la matriz de parámetros intrínsecos.

Sin embargo, el sistema de coordenadas de mundo y cámara nunca coinciden como se ve en la Fig. 5.2.

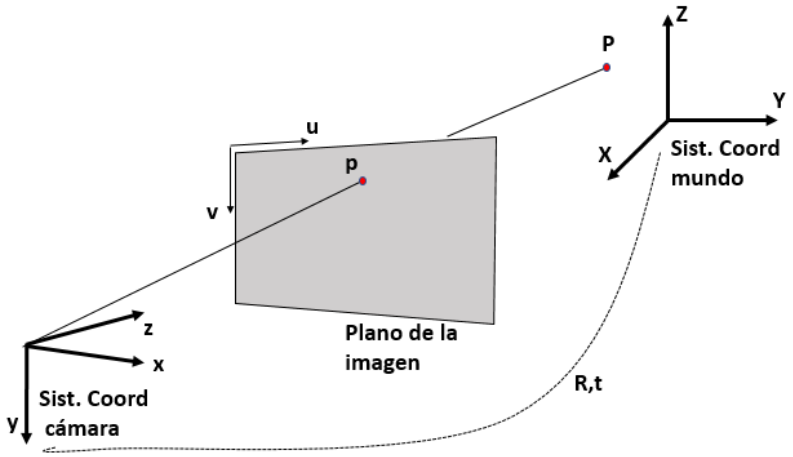

Fig. 5.2 Modelo proyectivo de formación de imágenes cuando los sistemas de referencia de mundo y cámara no coinciden.

Por lo tanto, el modelo proyectivo esta también definido por los parámetros extrínsecos. Estos parámetros son la transformación que explica la traslación y rotación del mundo frente a la cámara. En la ecuación 5.2 se muestra el modelo proyectivo teniendo ambos parámetros en consideración.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (5.2)$$

Siendo la matríz de transformación la matríz de parámetros extrínsecos.

Obtención de posición.

El objetivo de los algoritmos basados en el problema "Perspective-n-point" busca conocer la posición (rotación y traslación) de la cámara respecto a un objeto (o viceversa). Como se recalca en [37] estos algoritmos pueden ser una buena opción de cara a sustituir al sistema de referencia inercial en tareas de obtención de posición en robótica, por su precisión, versatilidad y poco coste económico.

Para conocer esta posición es necesario conocer las coordenadas de dicho objeto respecto al mundo, así como su proyección en la imagen tomada por la cámara (plano de la imagen, coordenadas en pixeles), de este modo, si se conocen con anterioridad los parámetros intrínsecos (calculados mediante un proceso previo de calibración) es posible extraer la matriz de traslación entre mundo y cámara, y por ende, la posición de dicho objeto respecto a la cámara.

5.1.2 Clasificación de métodos

Existen principalmente dos tipos de enfoque dentro de los métodos que buscan solucionar el problema "Perspective-n-point", los métodos iterativos y los métodos no iterativos.

5.1.2.1 Métodos no iterativos

Los métodos no iterativos son aquellos que buscan obtener una solución al problema de manera directa, sin llevar a cabo procesos iterativos.

P3P

Debido a que el sistema de referencia del mundo tiene 6 grados de libertad, se necesitan como mínimo 3 puntos de referencia para poder obtener un número finito de resultados [38] Sin embargo, no se obtiene un único resultado, sino que se obtienen cuatro posibles resultados. Aquellos algoritmos que utilizan solo tres puntos de referencia son los considerados como P3P y se basan en la resolución de ecuaciones cuadráticas para obtener los resultados posibles.

Destacan trabajos como el de Gao, et al [39], que realiza una extensa revisión del problema P3P y propone uno de los métodos que han sido como referencia el cual es el implementado en la librería OpenCV. Otros trabajos más recientes destacados son los realizados por Keip et al [40] y por Persson et al [41], los cuales superan al algoritmo clásico y obtienen buenos resultados de precisión y velocidad.

Este tipo de enfoques suelen ser los más inestables, debido a sus problemas para obtener una única solución. Algunos enfoques añaden más puntos al sistema para poder disminuir el ruido [42] [43]. Sin embargo, a medida que se añaden más puntos el ruido disminuye pero el coste computacional aumenta haciendo que no resulte una buena opción.

EPnP

Introducido por Li et al. [44] el enfoque de EPnP busca poder reducir el ruido al mismo tiempo que se mantiene el coste computacional. Para ello este algoritmo toma un número de puntos $n \geq 4$, pero los expresa como una suma ponderada de cuatro puntos virtuales. De este modo consigue obtener resultados precisos sin necesidad de aumentar el coste computacional añadiendo más números. Se trata de uno de los métodos más populares e incluso se ha propuesto su implementación en la labor de estimación de pose de robots o sistemas móviles[37].

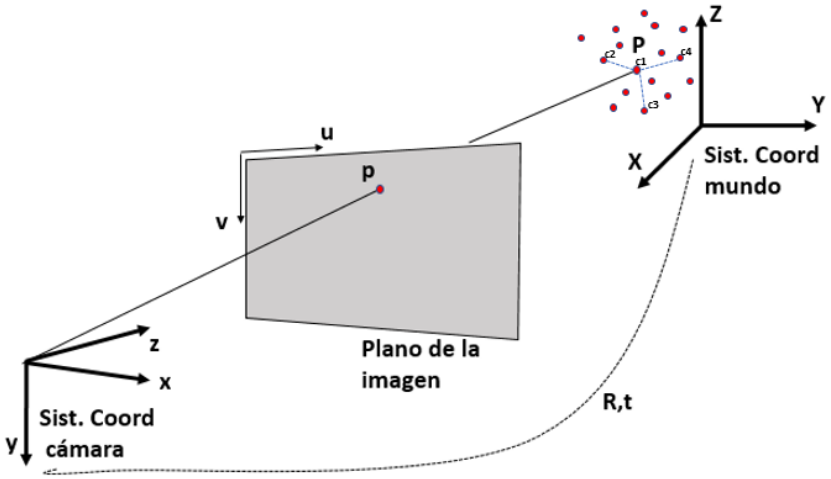

Fig. 5.3 Esquema de EPnP: los cuatro puntos c1,c2,c3 y c4 representan los puntos de control respecto a los cuales se expresan los demás.

Algoritmo basado en DLS (Direct Least Squares)

Este algoritmo introducido por Hersch et al. [45] expone el problema de manera diferente al resto de métodos no iterativos, ya que utiliza mínimos cuadrados para aproximar el modelo de la cámara. La mayoría de métodos que utilizan mínimos cuadrados necesitan un proceso de optimización iterativa, sin embargo este método plantea la función de coste de tal modo que no es necesario llevar a cabo dicha optimización. Se trata de un algoritmo muy escalable (puede utilizar gran cantidad de puntos sin aumentar su coste computacional) y que consigue mayor precisión que el algoritmo basado en EPnP [45].

5.1.2.2 Métodos iterativos

Los métodos iterativos son todo aquellos que plantean el problema en dos partes, realizando primero una estimación inicial (por ejemplo mediante mínimos cuadrados, buscando el mínimo local de una función de coste) y a continuación optimizan la primera estimación minimizando el error. Normalmente este error es el error de reproyección el cual representa la diferencia entre el punto proyectado por el sistema con la estimación realizada y el punto real. El principal inconveniente de este tipo de algoritmos es que al llevar a cabo procesos iterativos, presentan un gran coste computacional, sin embargo en lo respectivo a precisión son la mejor opción.

Entre los algoritmos iterativos el más utilizado es el algoritmo iterativo clásico, basado en optimización mediante un método de optimización conocido como Levenberg-Marquadt [46] que alterna entre el algoritmo Gauss-Newton y el método de pendiente de gradiente para minimizar el error. Este algoritmo realiza una primera estimación mediante mínimos cuadrados y a continuación mediante una minimización iterativa del error de reproyección obtiene el resultado.

Aunque existan otros algoritmos iterativos clásicos [47][48], el algoritmo basado en Levenberg-Marquadt sigue siendo el más utilizado, y obtiene mejores resultados en cuanto a precisión que la mayoría de opciones existentes, superando a los citados DLS y EPnP en precisión [49] [45]. Además, este método cuenta con una gran disponibilidad, y se puede encontrar en la librería OpenCV.

De entre los algoritmos mostrados, se ha decido implementar este por ser el método más utilizado y representativo de entre los algoritmos que buscan resolver el método basado en "Perspective-n-Point".

5.2 Implementación

En este capítulo se describirá en que consistió el proceso de implementación del algoritmo basado en "Perspective-n-Point". La implementación de este algoritmo se divide en dos partes, una primera parte en la cual se describe el proceso de calibración de la cámara, y una segunda parte donde se detallará el proceso de implementación del algoritmo de obtención de la posición.

5.2.1 Calibración

Como se explicó, la formación de imágenes en una cámara se encuentra modelada por las matrices de parámetros extrínsecos e intrínsecos. El proceso de calibración consiste en obtener los parámetros intrínsecos, los cuales no cambiarán nunca para una misma cámara. Se trata de un paso imprescindible para la obtención de la pose.

Cabe mencionar que estos parámetros intrínsecos también fueron necesarios para implementar los algoritmos de odometría visual, pero ha decido detallarse en este capítulo el proceso de calibración, ya que la calibración juega aquí un papel más relevante.

El proceso de calibración se basa en, conociendo las proyecciones de una serie de puntos n en el plano de la imagen, así como las coordenadas 3D de esos mismos poder despejar las matrices de parámetros extrínsecos (que variarán para cada imagen) y la matriz de parámetros intrínsecos (indistintos de la imagen).

Para poder conocer las coordenadas 3D de determinados puntos respecto al mundo, es necesario definir un sistema de referencia del mundo y algunos puntos en el mismo. Para esto es necesaria la utilización de un patrón de calibración reconocible para el programa,

El patrón elegido para este proyecto es un damero (Fig. 5.4). Este patrón tiene unas condiciones geométricas concretas y conocidas, lo que facilita que sea reconocible. Existen otros patrones diferentes, se ha elegido este por ser el más utilizado ya que permite gran precisión en su detección [50].

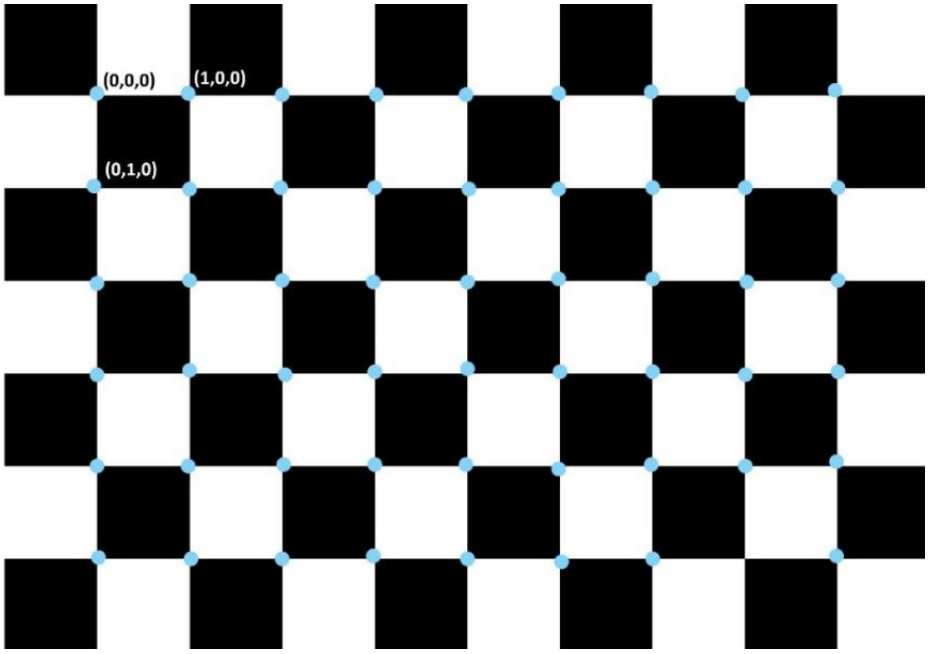

Fig. 5.4 Damero con los puntos de referencia marcados

.

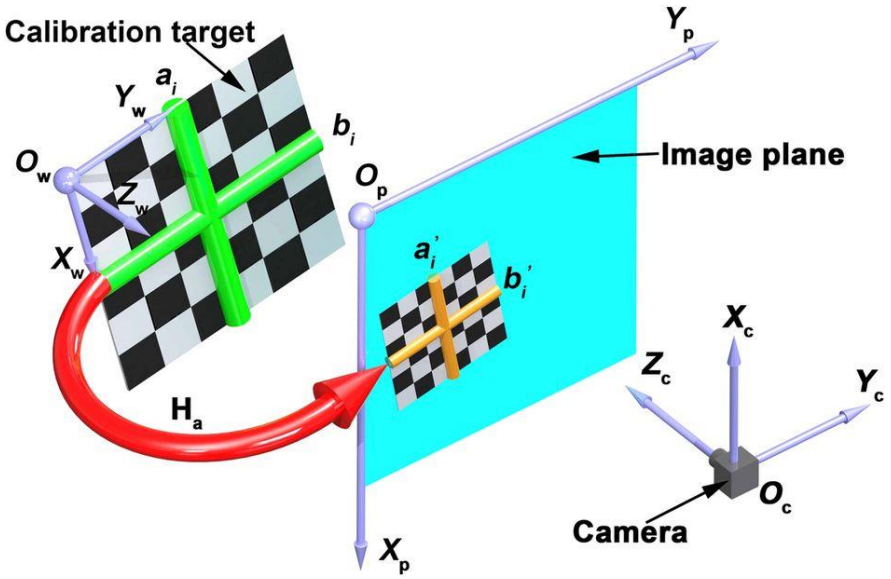

Fig. 5.5 Utilización del patrón como sistema de referencia.

El patrón funciona como sistema de referencia del mundo, pues es la orientación de dicho patrón lo que se busca conocer. Cada esquina interior del patrón representa una coordenada, siendo la esquina superior izquierda la coordenada (0,0,0).

El método de calibración utilizado es el método Zhang [51]. Se ha escogido este método debido a que es el implementado en OpenCV y es un método ampliamente utilizado y probado.

El método de calibración utilizado parte de utilizar una serie de imágenes del damero en diferentes posiciones, utilizando varias imágenes el método puede ajustar de mejor manera el modelo proyectivo de la cámara. Por tanto es importante tomar un número de imágenes adecuado aportando una gran cantidad de posiciones diferentes del patrón de calibración. La función de OpenCV muestra tras el proceso de calibración el error de reproyección (coeficiente RMS) obtenido. Dicho error de reproyección puede ser utilizado para analizar la calidad del proceso de calibración, sin embargo, sus resultados pueden ser engañosos, ya que si se tomar pocas imágenes no se habrán tomado la suficientes orientaciones del patrón, y por tanto el error de reproyección será pequeño, pero la calidad de la calibración no será buena. Por tanto, para analizar cuantas imágenes tomar, se representó gráficamente la evolución del error de reproyección con el número de imágenes tomadas, como se muestra en la Fig 5.6.

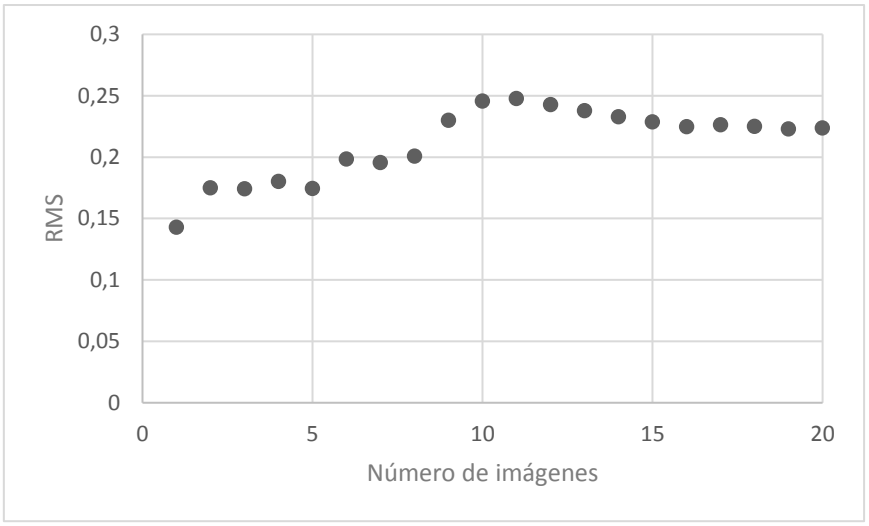

Fig. 5.6 Gráfica que relaciona el error de reproyección con el número de imágenes tomadas.

En la gráfica se puede ver cómo el error de reproyección empieza siendo muy pequeño. Cabe destacar que si se toma una imagen con una perspectiva difícil de captar para el algoritmo, el error de reproyección aumentará, por eso, lo importante es tomar una cantidad de imágenes adecuada con diferentes posiciones para que el error de reproyección no sea engañoso. Como se puede ver en la gráfica, a partir de 16 imágenes, las variaciones del error son muy pequeñas, por lo que podría considerarse válida la utilización de esas 16. Finalmente se usaron 20, lo cual no tendría que tener influencia en los resultados. El RMS obtenido, de entre 0.2 y 0.25 es un muy buen resultado.

Una vez tomadas las imágenes, en OpenCV se crea un vector de 54 (es decir 9x6) puntos 3D. Estos son, de izquierda a derecha y la pare superior a la inferior del tablero, todas las esquinas interiores del mismo, y serán los n puntos de referencia del calibrado (tal y como se muestra en Fig. 5.4, marcando los puntos en azul). El orden de los puntos es importante ya que, si no, el algoritmo que se usará para asociarlos con sus proyecciones en 2D no podrá funcionar. Todos los puntos en 3D tienen su coordenada Z a 0, ya que se considera que están en el plano XY del patrón de calibrado.

Ya conocidos los puntos de referencia en 3D, el siguiente paso fue crear una copia en escala de grises de la imagen para llevar a cabo la detección de la proyección de dichos puntos en el plano de la imagen. La función de OpenCV "findChessboardCorners" encuentra dichos puntos por ser puntos muy característicos en cuanto a intensidad y necesita, por tanto, de la imagen en escala de grises. En la Fig 5.7 se muestra el resultado de la detección de la esquinas del tablero.

Fig. 5.7 Resultados de "FindChessboardCorners" para tres orientaciones diferentes

Una vez se han obtenido las proyecciones en el plano de la imagen, ya se puede llevar a cabo la calibración para obtener los parámetros de la cámara. Para llevar a cabo la calibración se utiliza la función "calibrateCamera". Dicha función conociendo las coordenadas 2D y 3D de las esquinas inferiores calcula los siguientes parámetros

- "RMS": es el error de reproyección, ya mencionado previamente.
- "cameraMatrix": es la matriz de parámetros intrínsecos buscada.
- "distCoeffs": Coeficientes de distorsión radial.
- "rvecs": vectores de rotación del sistema de referencia del mundo en cada imagen.
- "tvecs": vectores de traslación del sistema de referencia del mundo en cada imagen.

Finalmente, todos los parámetros intrínsecos se almacenan en un archivo tipo yml para ser utilizados posteriormente en la obtención de posición.

5.2.2 Obtención de la pose

Una vez obtenidos los parámetros intrínsecos, ya se pueden llevar a cabo la obtención de pose. En este proceso, se llevará a cabo la obtención de parámetros extrínsecos en tiempo real. Para realizar la obtención de la pose se utilizan los parámetros intrínsecos obtenidos en el proceso de calibración.

Al igual que se hizo para la calibración, se debe definir un vector de puntos en tres dimensiones, en el cual se almacenan las 54 coordenadas de puntos de referencia del tablero.

El siguiente paso es utilizar la función "findChessboardCorners", una vez que se detecte el tablero en la imagen, el proceso entrará en un bucle hasta que se deje de detectar. En este bucle se llamará a una serie de funciones continuamente:

- "SolvePnP": Esta función es la encargada de calcular los parámetros extrínsecos mediante el método iterativo. Devuelve los valores de rotación y traslación en forma de vectores. Para funcionar necesita conocer los parámetros intrínsecos.
- "Rodrigues": Aplica inversamente la fórmula de Rodrigues para expresar la rotación en forma de matriz en base a los vectores obtenidos por "SolvePnP".
- "decomposeProjectionMatrix": descompone la matriz de transformación y expresa la rotación mediante los ángulos de Euler.

El resultado final se muestra en la Fig. 5.8, como se puede ver también se muestran los ángulos de Euler resultantes por pantalla. En la Fig. 5.8, el eje azul representa el eje Z, el eje verde corresponde con el eje Y, y el rojo se corresponde con el eje X.

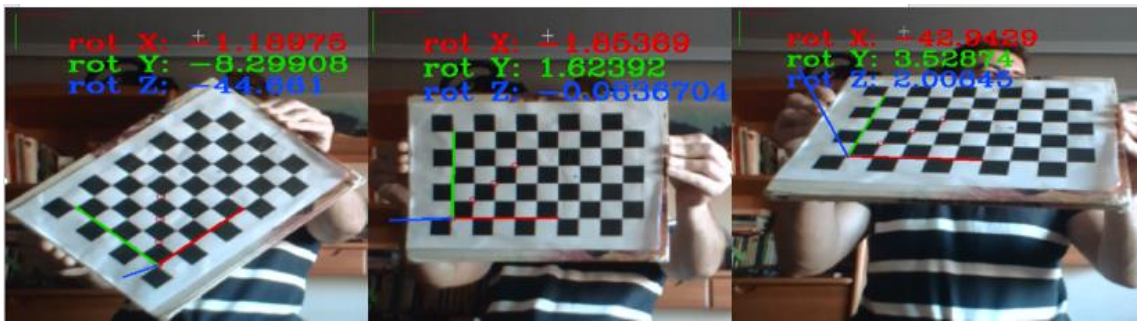

Fig. 5.8 Imágenes obtenidas por el algoritmo, ejemplo de 3 orientaciones diferentes

Una forma de resumir la implementación realizada se muestra en el diagrama de flujo que se muestra en la Fig. 5.9

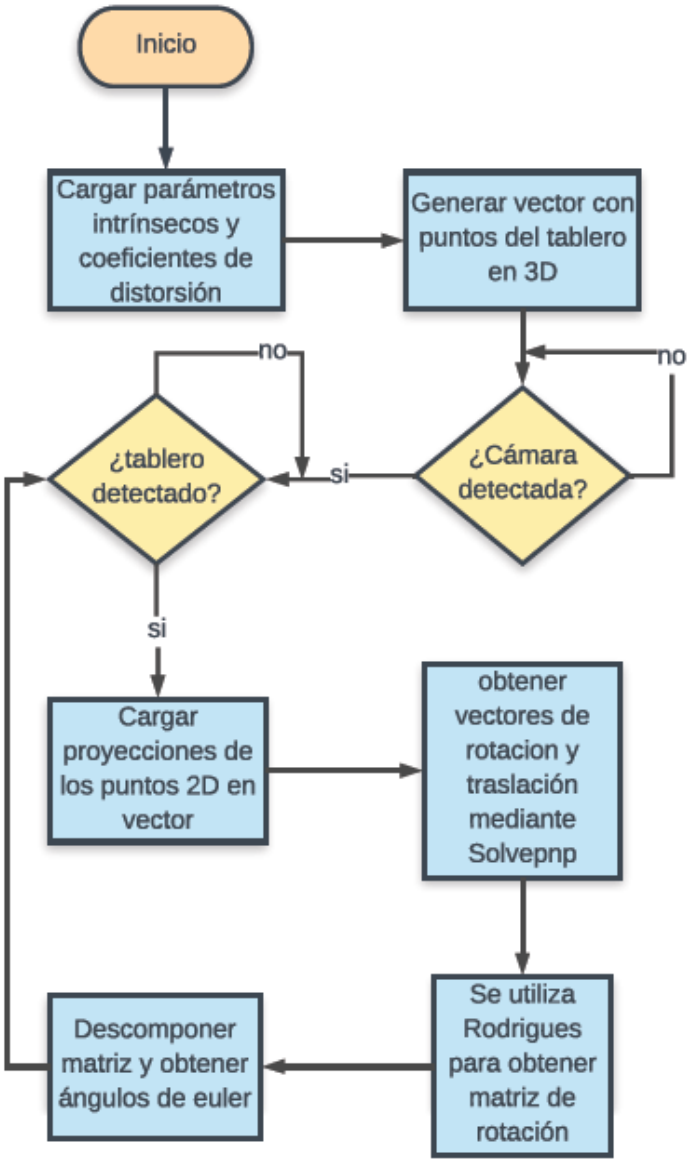

Fig. 5.9 Diagrama de flujo de la implementación del método de obtención de posición.

5.3 Pruebas realizadas

De cara a analizar la viabilidad de implementar algoritmos de visión por computador para controlar el equilibrio en robots humanoides como TEO existen dos parámetros fundamentales a tener en cuenta:

- Precisión: La precisión que estos algoritmos han de presentar ha de ser muy alta. Cualquier fallo en la medida de la orientación puede provocar una caída del robot, ya sea por no detectar una situación de desequilibrio o por detectar una posición de desequilibrio en ausencia de esta, provocando una actuación que puede derivar en una caída del robot. Para analizar la precisión de cada algoritmo, se compararán los resultados con los obtenidos por el sistema de navegación inercial de TEO, ya que es el principal encargado de controlar el equilibrio de TEO. Así se analizará hasta qué punto es interesante la implementación de este tipo de algoritmos para complementar a los detectores de equilibrio ya implementados en el robot.
- "Time complexity" o coste temporal: Es el tiempo que tarda cada algoritmo en recalcular la orientación y dar el resultado. Aquellos algoritmos que presenten los resultados de manera más rápida, reaccionarán antes ante una posición de desequilibrio, por lo que su efectividad será mayor. Se esperaba que entre los tres algoritmos presentados en este proyecto, el algoritmos basado en "Perspective-n-Point" aporte el menos coste temporal ya que al utilizar un patrón de calibración, la complejidad computacional para detectar los puntos de referencia en cada frame será significativamente menor. Por lo que resultará interesante analizar las ventajas e inconvenientes de este algoritmo con respecto a los algoritmos de odometría visual.

Para llevar a cabo las pruebas se realizó un programa de control en lazo abierto (Fig 5.10). La entrada a este era una rotación de la cadera (ángulo de referencia) y la salida la rotación captada por el Sensor inercial y por el algoritmo de visión implementado en cada caso.

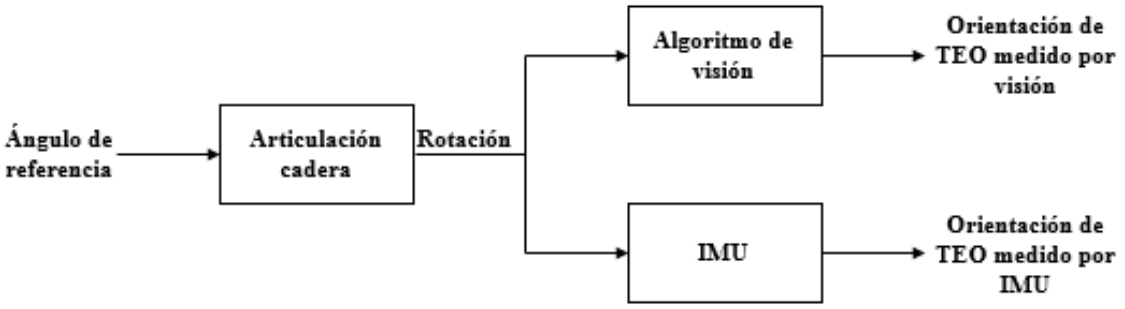

Fig. 5.10 Control implementado para la realización de las pruebas

Solo se realizaron las pruebas en el eje longitudinal de TEO (rotación de la cadera, eje Z en Fig 5.11) debido a que en los ejes transversal (eje Y en Fig 5.11) y anteroposterior (eje X Fig 5.11) la capacidad de rotar de TEO sin perder el equilibrio es demasiado limitada, por lo cual se podrían obtener muy pocos datos. Sin embargo, no solo se obtendrá la rotación en el eje respecto al que se rotará, sino que también se analizará si los algoritmos detectan rotación en los demás ejes, lo cual sería un error de precisión a tener en cuenta.

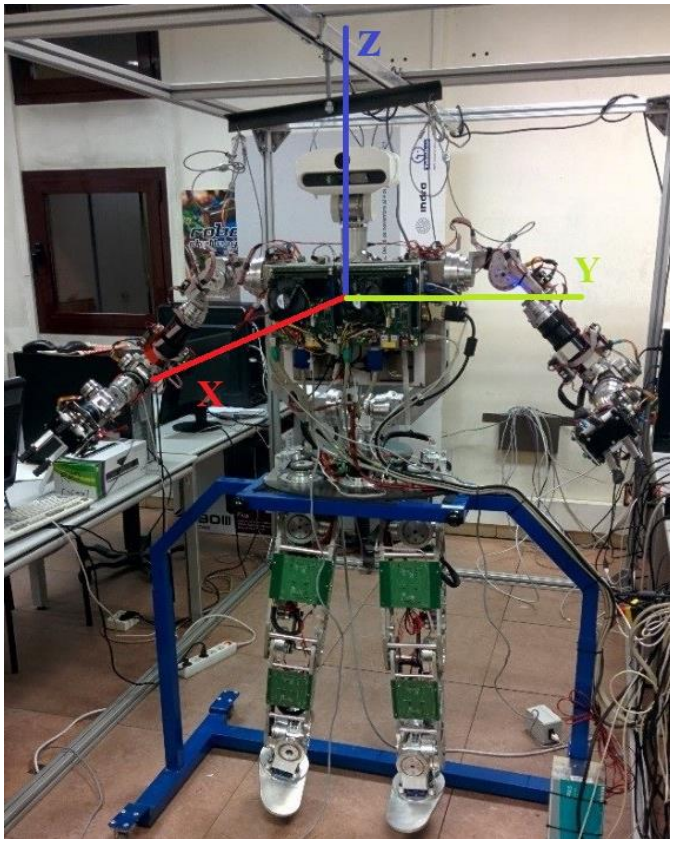

Fig. 5.11 Distribución de los ejes XYZ utilizada en las pruebas

En la primera prueba que se realizó, la entrada era de tipo rampa-escalón, se producían rotaciones de la cadera en el eje Z de TEO sucesivamente hasta llegar a 30 grados. Con esta primera prueba se analizó la precisión de cada algoritmo independientemente del tiempo de respuesta, ya que el ritmo de giro viene definido por la velocidad del algoritmo de visión a la hora de mostrar datos. Es decir, a cada iteración del algoritmo se producirá un ligero desplazamiento y en cada iteración se compara la rotación obtenida por la cámara con el ángulo real al que se ha desplazado TEO.

En la segunda prueba se introduce al sistema una entrada de tipo escalón. Se realiza una rotación de TEO de 30 grados, durante el recorrido de la rotación, se toman datos del IMU y del algoritmo de visión. La velocidad a la que TEO rota la cadera es independiente del algoritmo, por lo cual con esta prueba se puede analizar la velocidad a la hora de tomar datos (el ya mencionado time complexity) y de reaccionar ante un desplazamiento. Una vez TEO llega a 30 grados vuelve a recuperar su posición inicial, así se podrá ver la capacidad de cada algoritmo de retornar a su posición inicial y ver cuanto error acumula. Todas las pruebas se realizaron en las mismas condiciones, en el laboratorio de robótica humanoide de la Universidad Carlos III, con las mismas condiciones de iluminación.

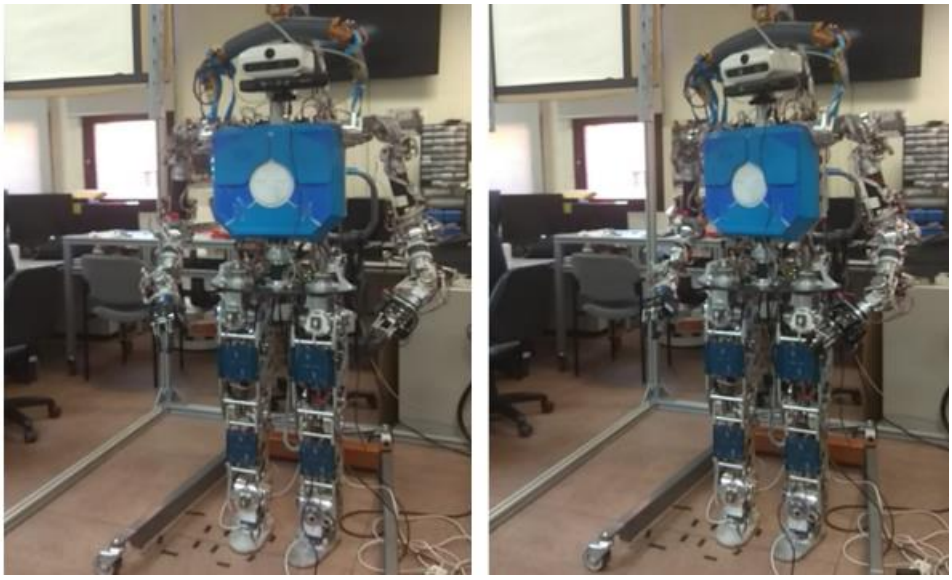

Fig. 5.12 TEO en posición inicial (izquierda), TEO tras rotar 30 grados en el eje Z-longitudinal (derecha)

Para definir un offset se realizaron 30 iteraciones previas, calculándose la media de la orientación obtenida.

Para el caso del algoritmo basado en “Perspective-n-Point”, se colocó el patrón de calibración delante de TEO colgado y nivelado (Fig 5.13).

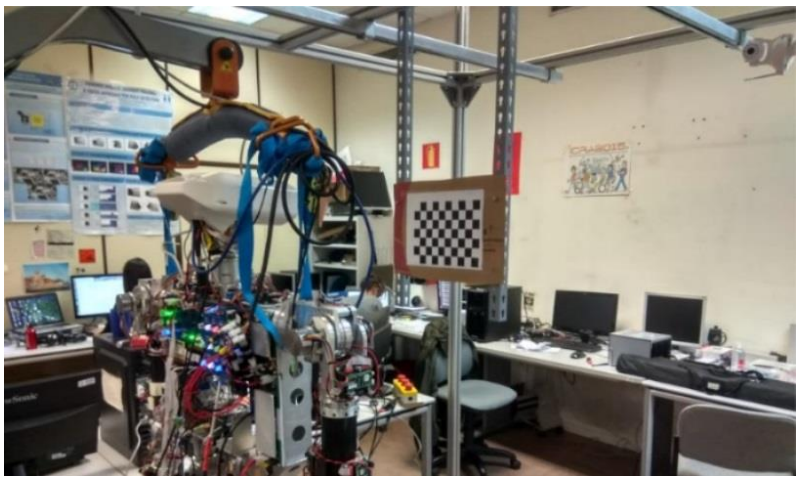

Fig. 5.13 Colocación del patrón de calibración

Para la realización de estas pruebas se utilizan simultáneamente los ordenadores de locomoción y de la cabeza. Mediante un ordenador remoto se maneja el ordenador de la cabeza, desde el cual se lanza dos programas, uno para realizar el control y otro para obtener la posición mediante visión. El programa de control se conecta con el ordenador de manipulación, obteniendo así los resultados del IMU y llevando a cabo la actuación de los motores de la cadera, a su vez los valores de la orientación son enviados mediante el programa de obtención mediante visión y recibidos por el programa de control que los almacena para poder compararlos después.

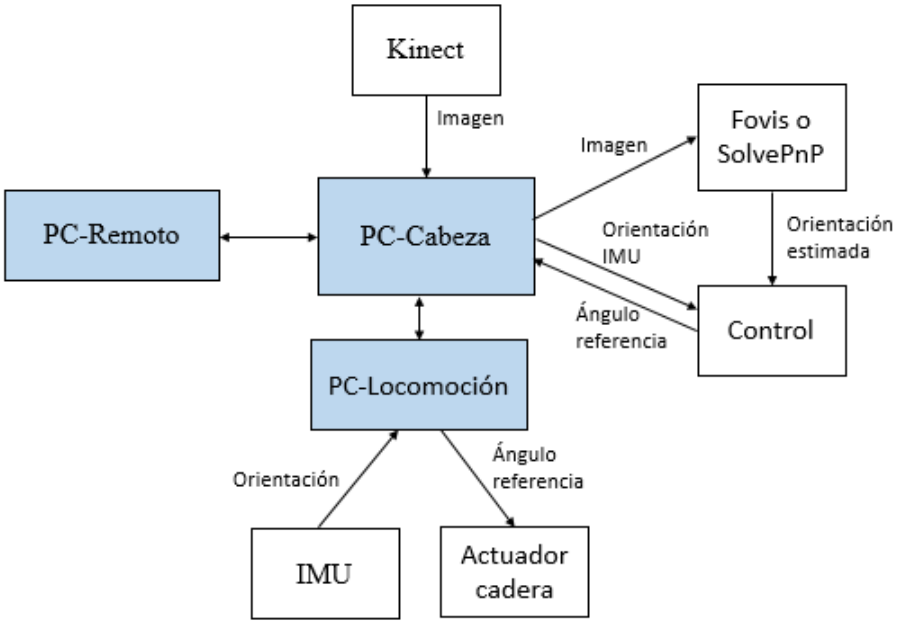

Fig. 5.14 Diagrama de conexiones para la realización de las pruebas

5.4 Resultados “Perspective-n-point”

Para el caso de la primera prueba los resultados obtenidos se pueden ver en las Fig.5.15 y 5.16, donde el ángulo obtenido por la cámara es el definido como “Ángulo SolvePnP”.

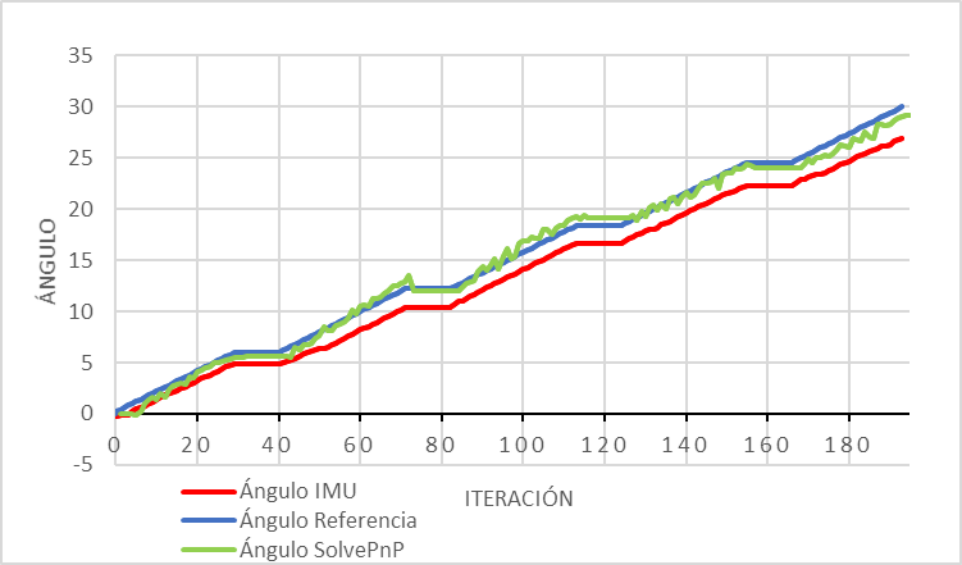

Fig. 5.16 Resultados de la rotación obtenida por el IMU y el algoritmo basado en “perspective-n-point” en la primera prueba para el sentido positivo de rotación en el eje Z.

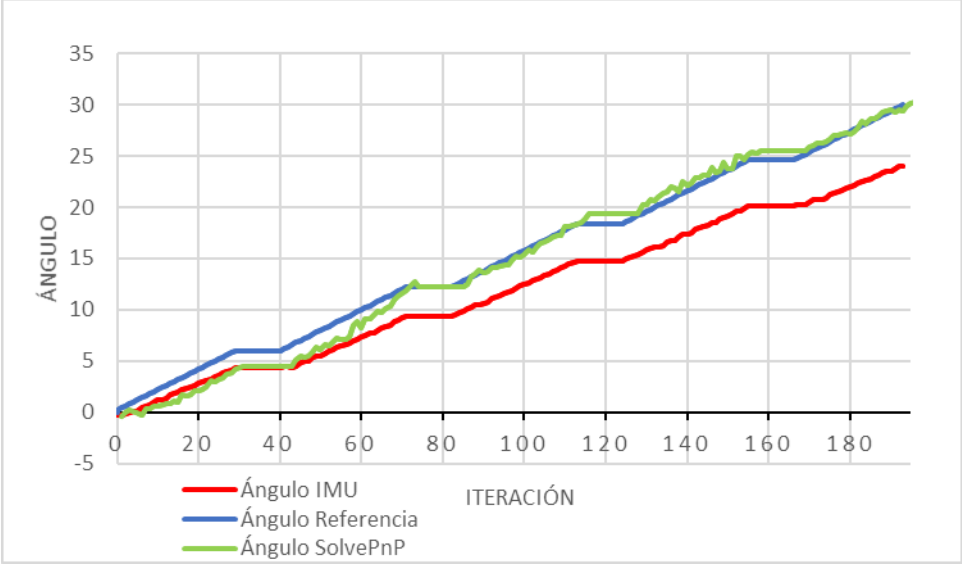

Fig. 5.15 Resultados de la rotación obtenida por el IMU y el algoritmo basado en “perspective-n-point” en la primera prueba para el sentido negativo de rotación en el eje Z.

Como se puede ver, los resultados obtenidos por el algoritmo de visión son bastante precisos, de hecho se puede ver a simple vista que presenta mayor precisión que el IMU. Los resultados que se obtienen con el IMU, especialmente en sentido positivo, se alejan bastante de los resultados obtenidos por el algoritmo de visión. Para poder comparar la precisión de cada método se ha calculado la media de diferencia para cada iteración entre el ángulo obtenido por el algoritmo y el ángulo de referencia, es decir la media de error, así como la desviación típica de dicho error. De igual manera se lleva a cabo para el IMU.

Para la prueba mostrada en las Fig 5.13, es decir, rotación positiva en el eje Z, la media de error fue de 0.535 Grados y la desviación típica fue de 0.383. Para la misma prueba, la media de error obtenida por el sensor inercial fue de 1.755 grados, es decir más de 3 veces mayor que para el algoritmo de visión, con una desviación típica de 0.560.

Para el caso de rotación en sentido negativo (Fig 5.14) la media de error obtenida por el algoritmo de visión fue de 0.859 grados, con una desviación típica de 0.475. En este caso el sensor inercial obtuvo una media de error de 3.235 grados, con desviación típica 1.350, siendo de igual modo un error mucho mayor.

En vista de los resultados, se puede ver que la precisión del algoritmo de visión es muy elevada en comparación con la precisión del sensor inercial. Apreciándose claramente la acumulación de error que sufre el sensor inercial, como se mencionó en el apartado de motivaciones. Dicho error no se aprecia prácticamente en el algoritmo de visión.

Para los dos experimentos anteriores se graficó también la orientación obtenida por la cámara en los ejes en los que no se producía rotación (Fig. 5.17, 5.18, 5.19 y 5.20) es decir los ejes x e y. De este modo se puede ver si presentan error y detectan rotación pese a que TEO no rotase en dichos ejes.

Fig. 5.17 Resultados obtenidos en el eje x en sentido positivo

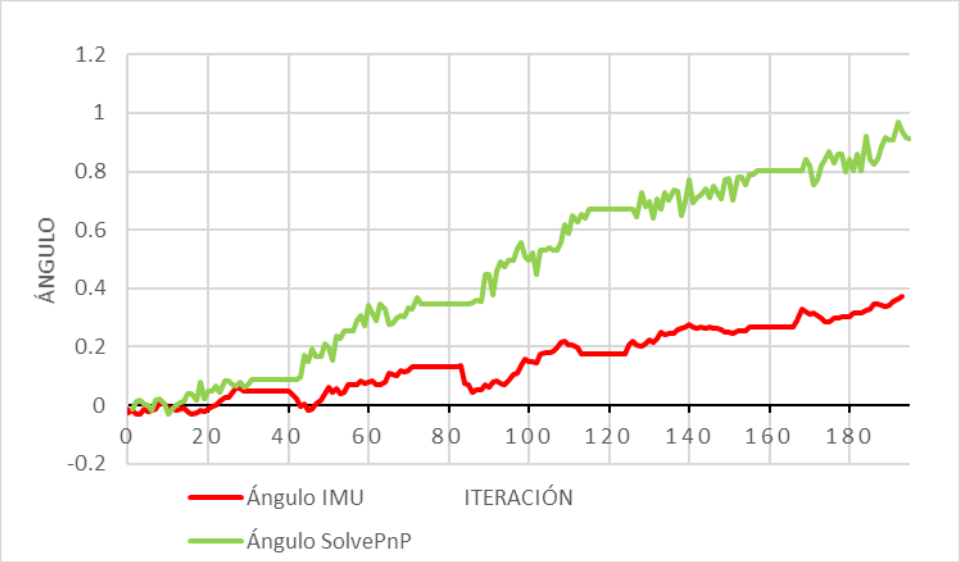

Fig. 5.18 Resultados obtenidos en el eje y en sentido positivo.

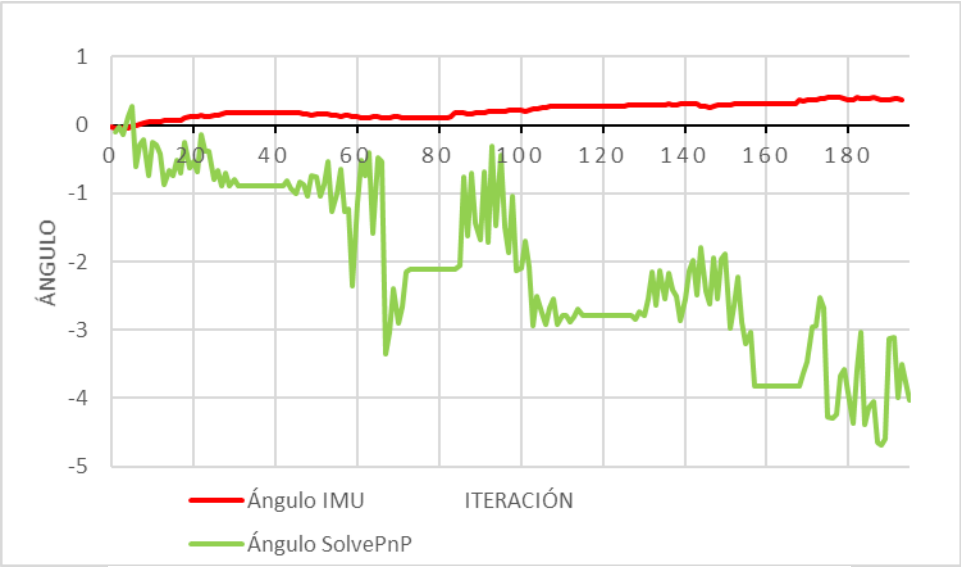

Fig. 5.19 Resultados obtenidos en el eje x en sentido negativo

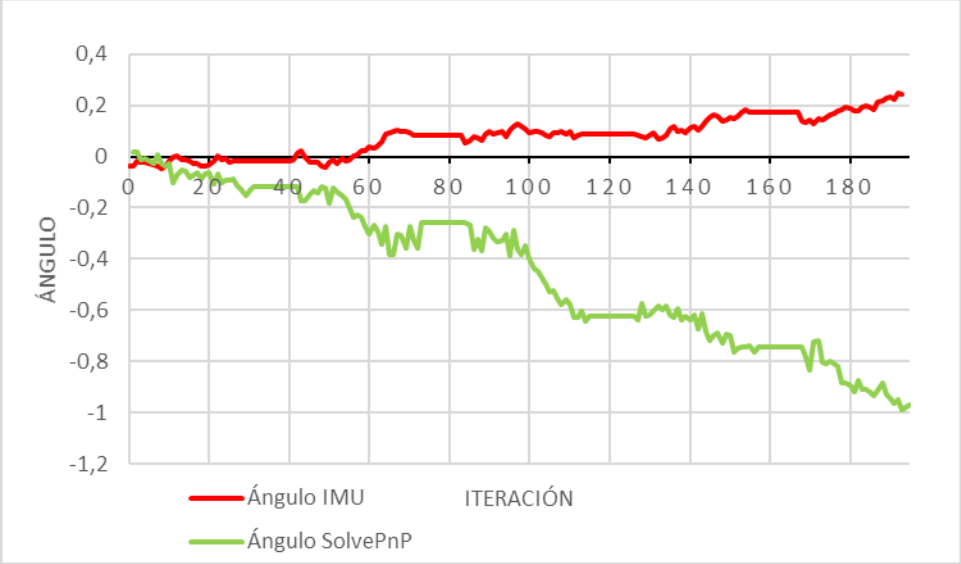

Fig. 5.20 Resultados obtenidos en el eje y en sentido negativo

Los resultados obtenidos para rotación en el eje X (Fig. 5.17 y Fig. 5.19) muestran una media de error obtenida por el algoritmo de visión de 1.221 grados, con desviación típica de 0.052 en sentido positivo frente a una media de error de 0.163 con desviación típica de 0.051 obtenida por el sensor inercial. Para el caso de rotación negativa, los resultados obtenidos por el algoritmo de visión fueron de 2.068 grados de media de error y desviación típica 1.223 frente a una media de error de 0.144 grados con desviación típica de 0.076 del IMU

Para el eje Y (Fig. 5.18 y Fig. 5.20), el algoritmo de visión obtuvo una media de error de 0.470 grados con desviación típica de 0.298, sentido positivo, frente a la media de error de 0.153 grados con desviación típica de 0.108 del sensor inercial. Para la rotación negativa en este mismo eje, los resultados del algoritmo basado en "perspective-n.-point" fueron de 0.470 grados de media de error y desviación típica 0.298, frente a la media de error de 0.153 con desviación típica 0.108 obtenidos por el IMU.

Al contrario que sucedió en las pruebas realizadas en el ángulo rotado, resulta evidente que algoritmo basado en "Perspective-n-Point" presenta menor precisión que el IMU en los ángulos que no sufrieron ninguna rotación.

En cuanto a la segunda prueba, se realizó una única prueba, el resultado se puede ver en la Fig 5.20.

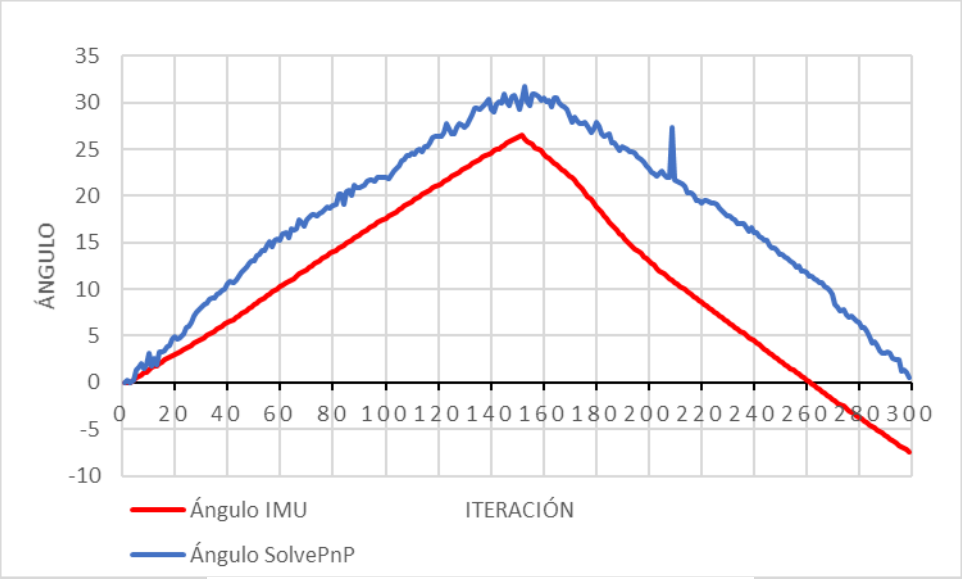

Fig. 5.21 Resultados para la segunda prueba

Como se mencionó previamente esta prueba consistía en rotar la cadera de TEO 30 grados, ida y vuelta, e ir tomando datos a lo largo del recorrido. Como se puede ver en el resultado obtenido, la precisión del algoritmo de visión para este caso es realmente buena, destacando que recupera la posición inicial sin prácticamente error. Además, el "time complexity" es realmente bajo, durante el desarrollo de la prueba se toman 300 iteraciones, lo cual permite detectar cambios en la orientación de manera muy rápida, detectando variaciones realmente pequeñas.

Por otro lado, destacar que existen determinados puntos en los cuales se detectan rotaciones erróneas (entre las iteraciones 200 y 250). También presenta ciertas imprecisiones alrededor de la iteración 150, punto en el cual TEO empieza a rotar en sentido contrario. De nuevo se aprecia el error acumulado por el IMU, no consiguiendo retornar a la posición original por más de 5 grados.

En vista de los diferentes resultados expuestos el algoritmo de visión basado en "perspective-n-point" presenta unos resultados muy buenos, tanto en precisión como en "time complexity". Sin embargo, cabe destacar que esa precisión es menor en los ejes que no se está produciendo rotación. Además, también se debe recordar que se necesita tener en todo momento un patrón de calibración delante, y que los límites de rotación vendrán marcados por la detección o no de ese patrón.

6 ODOMETRÍA VISUAL

En este capítulo se abordará todo lo relativo a los algoritmos de odometría visual. En primer lugar se realiza una revisión del estado del arte, en el que se explica en que consiste el problema a resolver por los algoritmos de odometría visual. A continuación se realizará un recorrido por la evolución de este tipo de algoritmos y una clasificación de los diferentes métodos existentes para la cámara RGB-D y por último se analiza en base a qué criterios se ha decido que algoritmo de odometría visual se ha implementado. Después del estado del arte se expondrá una breve introducción al algoritmo elegido, la implementación y resultados obtenidos por el mismo.

6.1 Estado del arte

En este capítulo, como ya se mencionó se realizará la revisión del estado del arte: introducción, evolución del problema, clasificación de métodos disponibles y selección de algoritmo adecuado para el proyecto.

6.1.1 Introducción

La Odometría visual es un proceso incremental que estima la posición de una cámara utilizando solo la imagen obtenida por la misma [52]. El término de odometría visual fue introducido en 2004 por D.Nister et al [53] . El término de odometría es utilizado debido a la similitud con la odometría clásica, basada en estimar el movimiento de un vehículo integrando el número de vueltas de las ruedas. En el caso de la odometría visual lo que se hace es comparar una imagen o nube de puntos con la anterior, una de referencia o un modelo 3D, al comparar las imágenes o nubes de puntos sucesivamente, se obtienen las transformaciones entre ellas y se actualiza la posición.

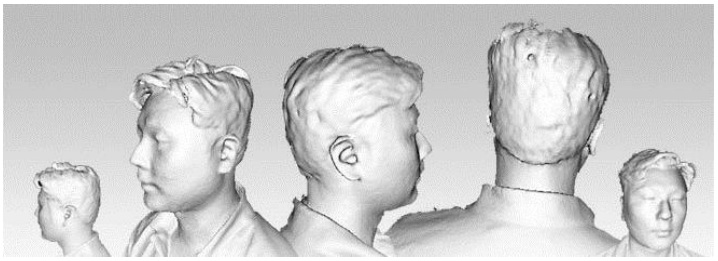

Fig. 6.1 Modelos 3D realizados mediante el algoritmo de V-SLAM KinectFusion, que incorpora un método de odometría visual

La odometría visual está directamente implicada en el mapeado simultaneo V-SLAM (Visual-SLAM), que busca mapear un entorno concreto utilizando las imágenes obtenidas por la cámara, obteniendo modelos 3D como el que se puede ver en la Fig. 6.1. Sin embargo, la odometría se puede separar del mapeado y utilizarse únicamente para conocer la pose de la cámara o el recorrido de esta.

La técnica de odometría visual surgió a principios de los años 80 creada por el investigador en robótica Hans Moravec [54]. El trabajo de Moravec tenía como objetivo la implementación en el robot Rover (Fig. 6.2) utilizado por la NASA en marte en 2004 para poder conocer la posición del mismo en todo momento [55]. Durante esta época se impulsó en gran medida el estudio de la odometría visual, siendo una tecnología muy utilizada actualmente gracias a las cámaras RGB-D.

Fig. 6.2 Recreación del robot Rover en Marte

6.1.2 Formulación del problema

El objetivo es conocer la posición de un determinado robot o vehículo que tiene unida a él una cámara. Los pasos que están involucrados en la odometría visual podrían resumirse de la siguiente forma.

1. La cámara montada en el vehículo o robot toma una imagen o nube de puntos I_k en el instante k. A continuación se desplaza de su posición (en este caso se rota) y toma otra imagen o nube de puntos I_{k+1} en el instante k+1, el objetivo final será conocer la transformación (traslación y rotación) entre sendas imágenes, lo cual también dará la respuesta a que desplazamiento y Rotación ha experimentado la cámara. Se asume que el sistema de referencia de la cámara coincide con el del vehículo o robot cuya posición quiere conocerse. Cabe mencionar que no todos los algoritmos utilizan imágenes, existen algoritmos que usan modelos 3D o nubes de puntos.

2. El primer paso una vez tomadas sendas imágenes es realizar una primera estimación que después se mejorará. Para obtener la primera transformación se busca encontrar las correspondencias entre las dos imágenes o nubes de puntos, para ello existen una gran cantidad de métodos diferentes, normalmente se utilizan puntos característicos de la imagen (que cumplen condiciones especial y por tanto son reconocibles). Una vez encontrada la correspondencia entre imágenes, se puede conocer que transformación se ha realizado para llegar a dicha correspondencia.
3. Una vez realizada la estimación inicial, se utilizan métodos de minimización del error como la reducción del error de reproyección, de esta manera se obtiene una transformación más precisa. La forma en la que se expresa la relación entre las posiciones de la cámara al tomar sendas imágenes se expresa con la matriz de transformación, es decir:

$$T_{k+1,k} = \begin{bmatrix} R_{k+1,k} & t_{k+1,k} \\ 0 & 1 \end{bmatrix} \qquad (6.1)$$

   Donde $R_{k+1,k}$ es la matriz de rotación y $t_{k+1,k}$ la matriz de traslación.

4. La matriz de transformación 6.1 relaciona la posición de la cámara en los momentos k y k+1 (Fig. 6.3), para conocer la posición absoluta de la cámara (es decir, con respecto al momento k=0) se deben multiplicar las matrices sucesivamente actualizando la posición en todo momento. En este proyecto, se utilizará solo la matriz de rotación, pues solo se busca conseguir la orientación de TEO.

Una explicación más detallada del problema se puede encontrar en [52], páginas 84 y 85.

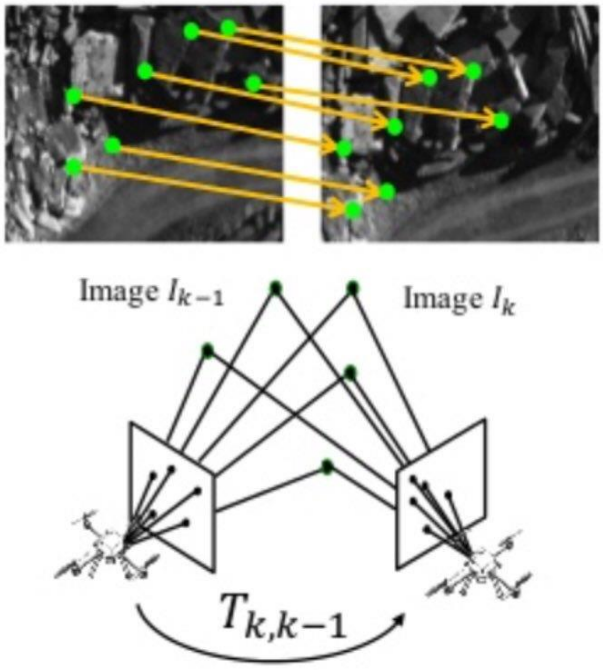

Fig. 6.3 Estimación de la matriz que describe el desplazamiento de un dron en base a conocer la correspondencia entre puntos característicos en dos imágenes

6.1.3 Clasificación de métodos y evolución del problema hasta la aparición de cámaras RGB-D

Existen muchas formas diferentes de clasificar los métodos de odometría visual. Se trata de un campo con un gran estado del arte, existiendo diversos investigadores que han clasificado los algoritmos de diferentes formas. Una clasificación muy popular fue la que hicieron Aggarwal y Nandhakumar en 1998 clasificando los métodos en basados en puntos característicos ("feature based") y basados en flujo óptico ("optical Flow based") [56]. Mas adelante en 2002 otra clasificación sería hecha por los investigadores Sedouka y Kak diferenciando entre algoritmos "map-based", "map-building-based" y "map-less navigation Schemes" [57]. En 2011 los investigadores Scaramuzza y Fraundorfer publicaron dos artículos [52] [58] que han sido muy utilizados como referencia. En dichos artículos diferenciaban entre métodos 2D-2D, 3D-3D y 3D-2D. Para un análisis profundo de las diferentes clasificaciones y como ha cambiado la odometría visual a lo largo de los años desde su origen, la publicación de Shashi Poddar et al. es amplia, detallada y reciente (Abril de 2018)[59].

En términos generales, originalmente existían dos líneas de investigación diferenciadas. Por un lado estaba la odometría visual utilizando cámaras estéreo y por otro lado los algoritmos de visión utilizando cámaras monoculares. La investigación con cámaras estéreo es más amplia debido a que se pueden tomar fotografías tridimensionales mediante el uso de varias cámaras. Destacan trabajos como los de Matthies y Shafer [60] [61]. O el trabajo realizado por Cheng [62][55] ya que fue el encargado de implementar el sistema de odometría visual en el robot Rover ya mencionado en el apartado 6.1.1.

En cuanto a la línea de investigación de odometría visual en cámaras monoculares, presentan el problema de que no se conoce la escala que relaciona medidas reales con las distancias que la cámara capta. Destacan trabajos como el de Besl y Mckay [63] los cuales desarrollaron el algoritmo ICP. Dentro del estado del arte de odometría visual con cámaras monoculares, puede hacerse la diferenciación entre tres tipos de métodos "feature-based" , "appearance-based" y métodos híbridos. En el ya mencionado trabajo de Scaramuzza y Fraundorfer se puede encontrar una revisión de la bibliografía respecto a la odometría visual monocular y estéreo [52].

6.1.4 Odometría visual en cámaras RGB-D

A partir de 2010, con la aparición de las cámaras RGB-D, la investigación en odometría visual se ha centrado en la utilización de estas, ya que estas cámaras pueden obtener a la vez información 2D y 3D y tienen un coste bajo, siendo accesibles para todos los usuarios. En los últimos años se han desarrollado multitud de algoritmos diferentes de odometría visual utilizando cámaras RGB-D. Este proyecto se centrará en la utilización de algoritmos de este tipo, los cuales son accesibles y obtienen resultados bastante precisos.

En los últimos años se han desarrollado una gran cantidad de algoritmos diferentes de odometría visual basados en la utilización de cámaras RGB-D, cada uno de ellos están basados en enfoques o técnicas diferentes. Una clasificación de los distintos métodos que se han desarrollado se puede encontrar en [64] el cual diferencia entre algoritmos basados en imágenes, algoritmos basados en profundidad y algoritmos híbridos.

6.1.4.1 Algoritmos basados en imagen RGB

A esta categoría pertenecen aquellos algoritmos que utilizan principalmente la información aportada por la imagen RGB. Dentro de este tipo de algoritmos se encuentran a su vez otras dos categorías.

Métodos basados en características visuales dispersas

Los algoritmos pertenecientes a método seleccionan determinadas características visuales (normalmente puntos de interés) de la imagen RGB, utilizando algoritmos de detección como HARRIS [65], SURF [66] O FAST[67], por citar tres de los métodos más utilizados habitualmente. La correspondencia entre puntos se encuentra entre las mismas imágenes RGB. Una vez encontradas las correspondencias entre los puntos entre las imágenes, la información de profundidad correspondiente a cada punto de interés es extraído mediante la imagen de profundidad que aporta la cámara RGB-D. Finalmente, mediante el error de reproyección, se estima la matriz de transformación. Un algoritmo característico de esta categoría es Fovis [31].

Métodos basados en características visuales densas

Este método se basa en utilizar la información aportada por toda la imagen, y no solo una serie de características o puntos de interés concretos. Principalmente se apoyan en la suposición de que un mismo punto del mundo se tiene que observar con el mismo brillo en dos imágenes obtenidas por la misma cámara en diferentes posiciones. Buscan encontrar aquella transformación que cumpla dicha suposición. Al igual que los anteriores, utilizan la imagen de profundidad aportada por la cámara RGB-D para encontrar la información de profundidad de los pixeles. A esta categoría pertenece el algoritmo DVO [68].

6.1.4.2 Algoritmos basados en profundidad

Dependen únicamente de la información aportada por la imagen de profundidad. A su vez, se pueden diferenciar tres categorías.

Métodos basados en características 3D

Similares a los a los algoritmos que obtenían características visuales dispersas, con la diferencia de que en este caso las características se obtienen de la imagen de profundidad. Las características que se obtienen de la imagen de profundidad suelen ser geométricas, como líneas, puntos o planos. Este tipo de métodos no son las más recomendables, ya que las limitaciones de las cámaras RGB-D en la detección de profundidad hace que sean computacionalmente costosos, por lo que métodos como [69] pueden obtener buenos resultados pero no son interesantes para este proyecto, ya que no funcionan bien en tiempo real.

Métodos basados Iterative Closest Point

De nuevo, frente a estos algoritmos que utilizan solo una serie de puntos, existen otros que utilizan toda la imagen. Este es el caso de los métodos basados en Iterative Closest Point (ICP) [63]. Los algoritmos basados en ICP se basan principalmente en alinear las nubes de puntos (imagen 3D formada por todas las coordenadas 3D de la imagen, Fig. 6.4) minimizando iterativamente la distancia entre nubes de puntos consecutivas y así obteniendo la transformación entre ambas. Este tipo de odometría es la presente en el algoritmo de generación de modelos 3D "Kinect Fusion" [70]. Actualmente se están desarrollando algoritmos de ICP capaces de trabajar a mayor velocidad como es el caso de FastICP, o GICP [71].

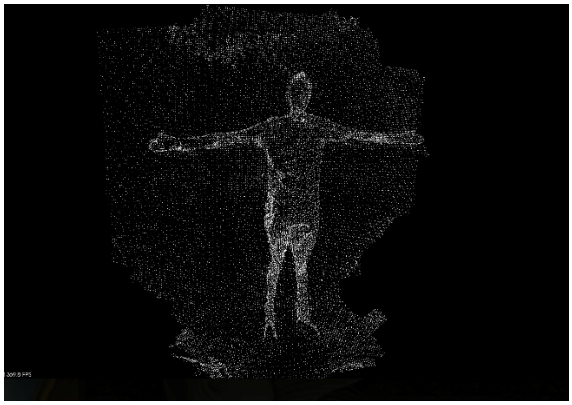

Fig. 6.4 Ejemplo de nube de puntos formada a partir de la imagen de una persona

Métodos basados en transformación de la distribución normal (NDT)

Los algoritmos que se basan en dividir la nube de puntos en varias regiones. Cada región es expresada como una distribución normal describe su superficie. De esta manera encontrar la relación entre dos nubes de puntos diferentes se puede realizar encontrando la correlación entre distribuciones o la pertenencia de un punto concreto a una distribución. Cabe resaltar que este tipo de métodos, pese a ser precisos, suelen tener problemas para hacer una estimación inicial de la transformación, y suelen ser lentos. Para poder implementarlos en tiempo real es necesario combinarlos con métodos de odometría basados en RGB.

Métodos basados en Range-Flow

Los métodos basados de este tipo se basan en la conocida como "range Flow constraint equation" [72] [73]. Considerando un punto en el sistema de coordenadas del mundo (X,Y,Z), se puede expresar la siguiente función de profundidad Z(X,Y,t) siendo la ecuación que exprese la variación de Z:

$$\dot{Z} = Z_t + Z_u\dot{u} + Z_v\dot{v} \qquad (6.1)$$

Siendo $\dot{u}$ y $\dot{v}$ las variaciones de posición de las proyecciones de X e Y (u y v, ver Fig. 6.5) en el plano de la imagen.

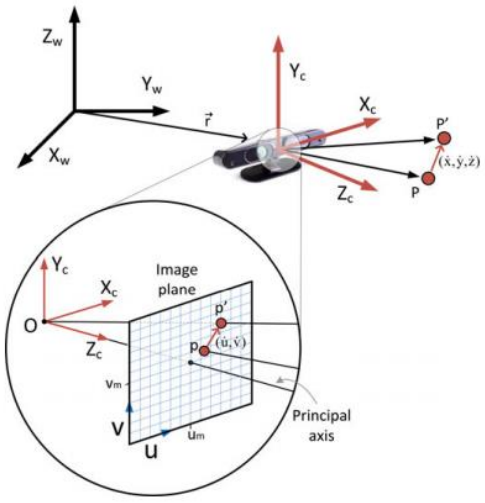

Fig. 6.5 Esquema Rangeflow [66]

Suelen ser métodos muy eficientes, como es el caso de DIFODO [74] pero con una gran dependencia de la calidad de imagen de profundidad y una precisión inferior a lo común.

6.1.4.3 Métodos híbridos

Los métodos híbridos son aquellos que intentan aprovechar tanto la información aportada por la imagen RGB como la de la imagen de profundidad. Dentro de estos métodos, se puede hacer una separación a su vez en dos tipos.

Métodos de dos etapas

Este tipo de métodos se basan en obtener una primera estimación de la transformación utilizando solo la imagen RGB, ya sea mediante puntos característicos dispersos o toda la imagen. Una vez obtenido un punto de partida, minimizan el error apoyándose en la imagen de profundidad utilizando métodos como ICP o NDT. Existen muchos métodos de este tipo que cabe resaltar. Un ejemplo es el algoritmo Realtime NDT [75], que utiliza primero una estimación inicial de la transformación mediante correspondencia de puntos característicos en la imagen RGB y a continuación utiliza transformación a distribución normal para minimizar el error. Otro caso es el algoritmo CCNY [76] que utiliza ICP para minimizar el error de la primera estimación obtenida a partir de la imagen RGB.

Métodos de optimización conjunta

Estos métodos se basan en combinar dos funciones de cálculo de error simultáneamente. Ejemplo de este tipo de método es Kintinuous [77], el cual combina al mismo tiempo fovis e ICP. Este tipo de métodos suelen ser muy costosos computacionalmente.

6.1.5 Selección de algoritmo a implementar.

Como se ha podido ver, existen una gran cantidad de algoritmos de odometría visual disponibles con cámaras RGB-D. Resulta muy importante para este proyecto elegir aquel algoritmo que mejor se ajuste a las condiciones existentes, y que cumpla principalmente lo buscado, es decir: precisión y poco coste computacional (y por tanto un tiempo de muestreo pequeño).

Para decidir qué algoritmo se implementaría se prestó una gran atención al trabajo realizado por Zheng et al. [64] que compararon una gran cantidad de métodos de odometría visual, cada uno de ellos representativos de un enfoque diferente.

Estos algoritmos son: Libviso2, Fovis, DVO, FastICP, Rangeflow, Realtime NDT, CCNY y DEMO. Las pruebas se llevaron a cabo en diferentes datasets, cada dataset con unas condiciones concretas, algunos con iluminación cambiante, velocidad de movimiento de la cámara mayor o menor, pocas referencias externas… La conclusión a la que los autores llegaron es que no era posible decir que un algoritmo concreto fuera mejor que el resto, si no que dependiendo de las condiciones resultaba más interesante uno u otro.

Las condiciones que se tienen que tener en cuenta para decir que algoritmo resulta más interesante son principalmente la iluminación del entorno, la calidad de la imagen de profundidad, el coste computacional y la cantidad de puntos de interés o característicos presentes en el entorno.

Coste computacional

En el artículo citado cada algoritmo presentaba un coste computacional diferente. El coste computacional es de gran importancia para este proyecto, pues este limitará la velocidad a la que el algoritmo puede funcionar, afectando en gran medida a la capacidad de reacción de TEO si pierde la postura de equilibrio. Además hay que tener en cuenta las limitaciones del hardware disponible.

Los algoritmos que presentan un menor uso de CPU son RangeFlow y Fovis. Sin embargo, se ha de tener en cuenta que RangeFlow presenta grandes limitaciones en cuanto a la precisión, pues su poco coste computacional se debe principalmente a que es el algoritmo que utiliza menos técnicas de mejora de Robustez, por lo cual su precisión se encuentra por debajo de los demás. siendo la precisión imprescindible para el objeto del proyecto, RangeFlow no resulta el algoritmo más adecuado a implementar. Por parte de Fovis, pese a su bajo coste computacional, obtiene buena precisión, pudiendo competir con otros algoritmos con costes computacionales muy superiores. Esto también se resalta en [33], donde se probaron diversos algoritmos de odometría en dispositivos móviles con unas características de Hardware inferiores a un ordenador y pese a ello, Fovis consiguió funcionar de manera fluida y con una buena precisión, siendo el algoritmo con mejor relación precisión/coste computacional.

El único problema de Fovis es que requiere de unas buenas condiciones de iluminación, calidad de imagen de profundidad así como de una gran cantidad de puntos de interés en la imagen. A continuación se analizarán las condiciones en las que se realizan las pruebas (laboratorio de robótica humanoide de la universidad Carlos III de Madrid).

Iluminación del entorno

Los algoritmos que obtienen su primera estimación de la transformación principalmente de la imagen RGB, como Fovis, presentan grandes problemas en aquellos entornos en los que la iluminación no es buena. Cabe mencionar que DVO aun en condiciones de iluminación no muy buenas si conseguía unos resultados aceptables. Así mismo, se puede observar en los resultados del artículo que los algoritmos híbridos como Realtime NDT o DEMO también presentan problemas en entornos oscuros. Es en estos entornos en los que los algoritmos basados únicamente en profundidad presentan mayor ventaja frente al resto.

Para analizar si las condiciones del laboratorio son suficientemente buenas como para que no se tengan que descartar los algoritmos basados en imágenes RGB o híbridos, se decidió tomar una serie Fig. 6.6, con la cámara Kinect que se utiliza en el proyecto. todas las imágenes se toman a lo largo del recorrido que la cámara capta durante las pruebas que se realizaron. Las condiciones son similares a en las que se realizaron las pruebas para el algoritmo basado en "Perspective-n-Point", durante el día y con prácticamente la misma luz. Dichas imágenes son convertidas a gris, de este modo, se puede calcular la intensidad de gris media de todos los píxeles de cada imagen lo cual es un indicativo de la iluminación de una imagen. Finalmente se calculó la media de todas las medias obtenidas.

El valor medio de intensidad media por pixel en las imágenes tomadas fue de 151.795 el cual es un valor realmente elevado, por lo que se puede considerar que la calidad de la imagen es óptima para los algoritmos basados en imagen RGB como Fovis.

Fig. 6.6 Cuatro de las imágenes tomadas para analizar las condiciones del laboratorio

Calidad de la imagen de profundidad y cantidad de puntos de interés

Otro factor decisivo a la hora de decidir qué algoritmo se ha de elegir es la calidad de la imagen de profundidad. Las cámaras RGB-D como Kinect presentan una gran debilidad a la hora de obtener la profundidad, ya que para muchos puntos de la nube de puntos no es posible obtener el valor de la profundidad. Esto se debe principalmente al poco rango de alcance que presentan las cámaras RGB-D que se pueden adquirir económicamente. Esta condición afecta a todo tipo de algoritmos, ya que los algoritmos basados en imagen RGB también dependen de la información de profundidad.

Si bien es cierto que para esta condición, ubicar a TEO cerca de objetos cercanos, cuya distancia podría ser captada fácilmente por la Kinect, podría ser una solución, de cara a implementar un algoritmo con más capacidad de adecuarse a condiciones ligeramente adversas, se decidió evaluar en el recorrido realizado por las pruebas para que porcentaje de píxeles se encontraba el valor de la profundidad.

Al mismo tiempo que se realizaba el recorrido obteniendo las imágenes que se utilizaron para medir la iluminación se tomaron los valores de profundidad de los pixeles. Se sumaron todos los píxeles que no tenían información de profundidad y así se calculó el porcentaje que dichos pixeles representaban sobre el total. Finalmente se hizo una media del recorrido y se obtuvo un porcentaje de 49.53%, el cual resultó ser un valor relativamente bajo. Este bajo valor se debe a que en la última parte del recorrido la media fue muy baja, pues como se observa en la última imagen de la Fig. 6.6, existe mucha distancia a los objetos en este último tramo. Sin embargo el valor no es lo suficientemente bajo como para asegurar que no se obtendrán buenos resultados con Fovis, mientras que si es demasiado bajo como para tener en cuenta el algoritmo RangeFlow.

Las dos últimas condiciones de las que dependen los algoritmos de odometría visual son la cantidad de puntos característicos o de interés que se pueden encontrar en la imagen. En entornos con muy pocos elementos externos, como por ejemplo un pasillo, resulta muy difícil encontrar puntos de interés que utilizar para analizar las diferentes imágenes o nubes de puntos. Esto afecta tanto a los algoritmos basados en profundidad como a los a los algoritmos basados en imagen RGB. El laboratorio de Robótica humanoide presenta gran cantidad de elementos como ordenadores y mesas, por lo cual no resultara difícil encontrar buenos puntos de interés.

En vista de todo lo anterior, es claro que el algoritmo que ha de implementarse en este proyecto es Fovis, ya que presenta un coste computacional bajo, y las condiciones del Laboratorio son adecuadas para que este funcione con la precisión adecuada.

6.2 Fovis

El algoritmo Fovis está descrito en el paper "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera" [31]. El algoritmo se desarrolló en un principio para utilizarlo en drones autónomos, tanto para la realización de mapeados como la obtención de la posición en tiempo real.

Las diferentes funciones se encuentran en la librería LibFovis, pudiéndose realizar modificaciones o ser combinado con otros algoritmos. Para este proyecto, la implementación se realizó utilizando el ejemplo de odometría visual disponible en dicha librería, modificando la forma en la que se obtienen los datos finales para obtener los ángulos de Euler a partir de la matriz de traslación.

6.2.1 Resultados

En la Fig. 6.6 se muestran los resultados obtenidos en la primera prueba en el sentido positivo de desplazamiento y en la Fig. 6.7 en sentido negativo.

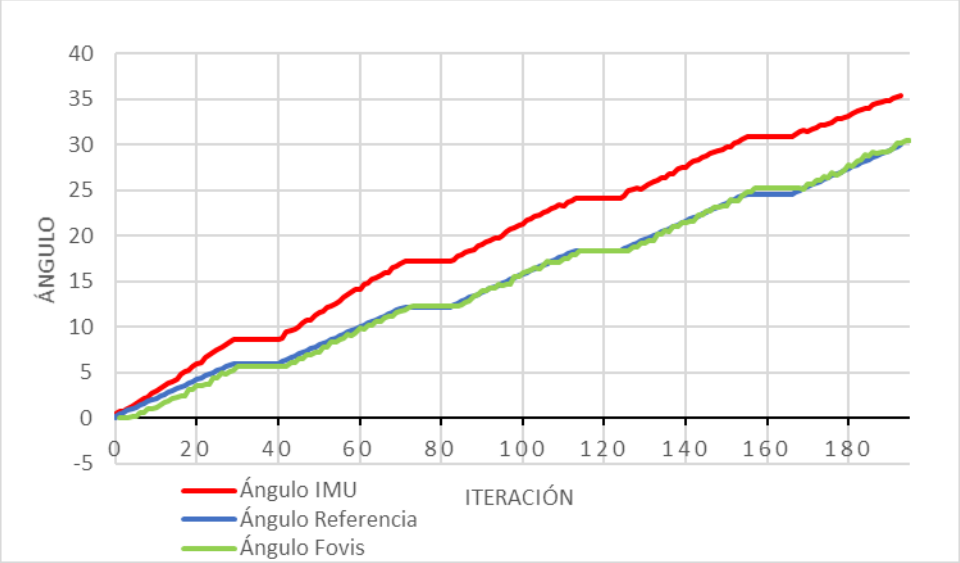

Fig. 6.7 Resultados de Fovis e IMU en la primera prueba para una rotación en sentido positivo en el eje z.

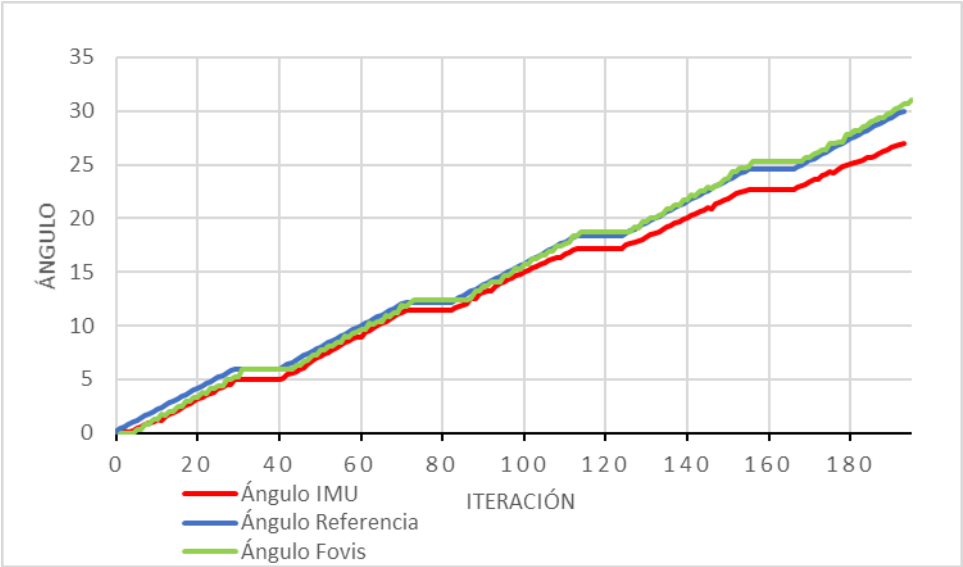

Fig. 6.9 Resultados de Fovis e IMU en la primera prueba para una rotación en sentido negativo en el eje z

Al igual que ocurrió con el algoritmo basado en "Perspective-n-Point", los resultados obtenidos son realmente precisos, superando al Sensor inercial. Las medias de error obtenidas por Fovis son de 0.328 grados en sentido positivo, con desviación típica de 0.328 y de 0.403 grados de media de error con desviación típica de 0.293 para el sentido negativo. Mientras que para el IMU el error obtenido en sentido positivo fue de 4.563 grados con desviación típica de 1.797. En sentido negativo la media de error fue 1.272 grados con desviación típica 0.592.

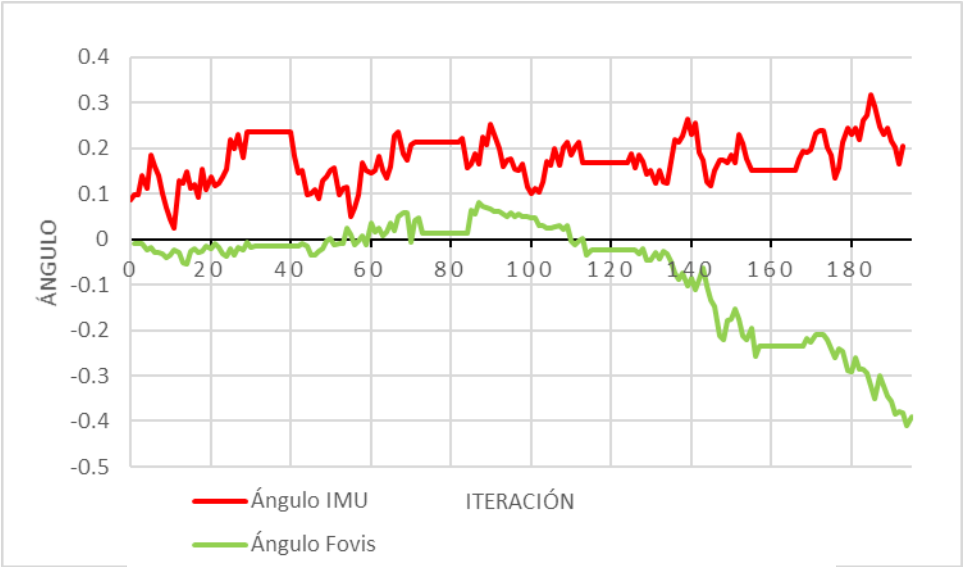

Fig. 6.8 Resultados obtenidos en el eje x en sentido positivo

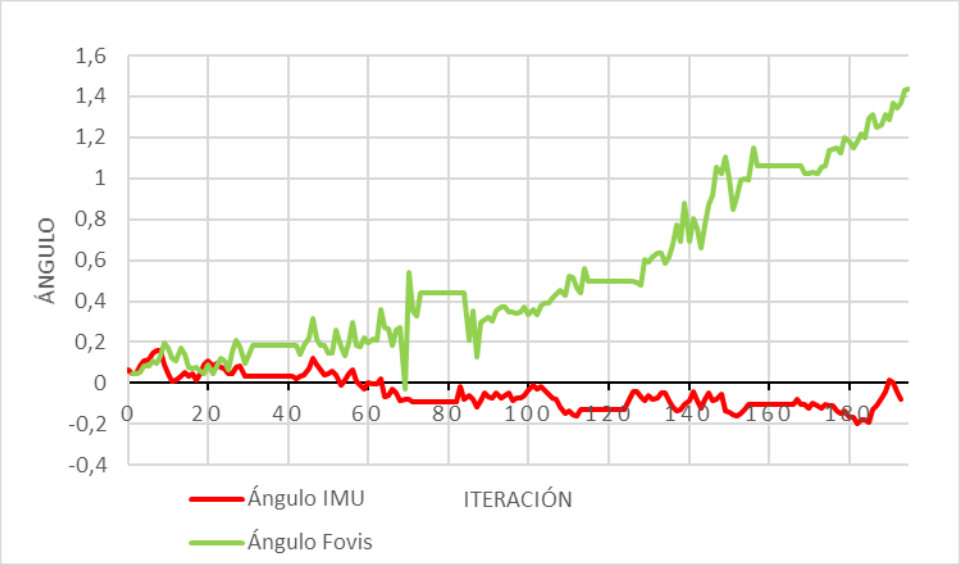

Fig. 6.10 Resultados obtenidos en el eje x en sentido positivo

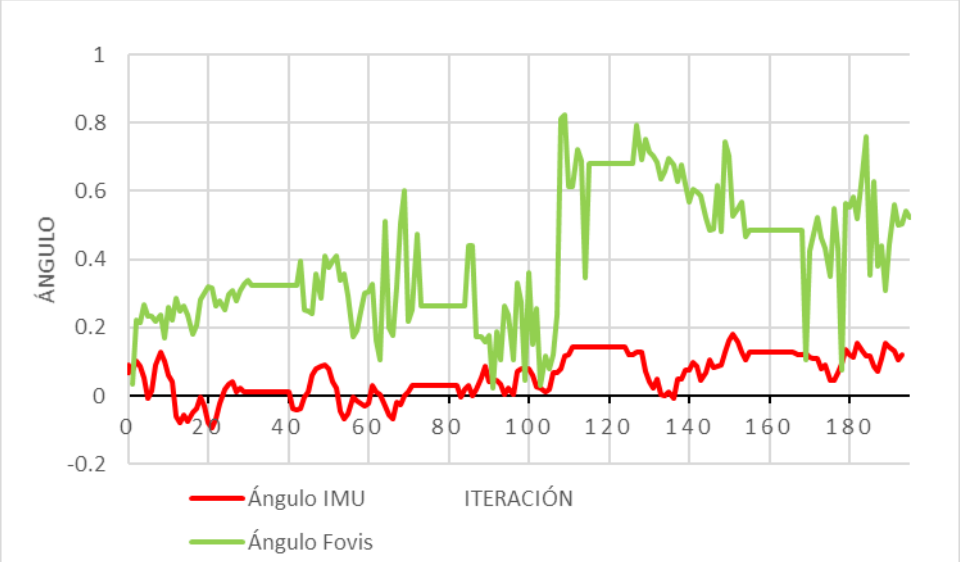

Fig. 6.11 Resultados obtenidos en el eje y en sentido negativo

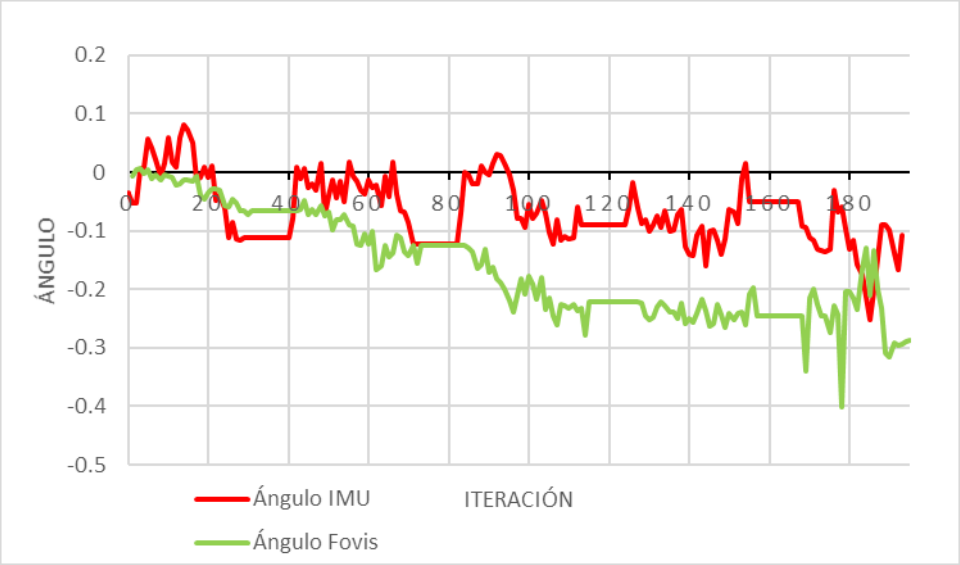

Fig. 6.12 Resultados obtenidos en el eje y en sentido negativo

Al igual que sucedió con el algoritmo basado en "Perspective-n-Point", en estos ejes en los que no se debería detectar movimiento Fovis presenta mayor error que el sensor inercial y este error crece con el tiempo.

En el eje x (Fig. 6.9 y Fig. 6.11) la media de error de Fovis fue de 0.089 grados con desviación típica 0.106 para el sentido positivo (error que va incrementando con el tiempo) y de 0.401 grados con desviación típica 0.188 en sentido negativo.

Por su parte el sensor inercial presentó una media de error 0.173 grados en sentido positivo con una desviación típica de 0.049 y de 0.069 grados con desviación típica de 0.047 en sentido negativo.

En el eje y (Fig. 6.10 y Fig. 6.12) Fovis obtuvo una media de error de 0.532 grados con desviación típica 0.394 en sentido positivo y de 0.163 grados con desviación típica 0.088 en sentido negativo. El IMU obtuvo una media de error de 0.080 grados con desviación típica 0.043 en sentido positivo y de 0.074 grados con desviación típica 0.046.

Los resultados obtenidos en la segunda prueba (rotación de 30 grados y retorno a la posición inicial) se observan en la Fig 6.13

De nuevo, la precisión obtenida por Fovis es realmente buena, como ya se pudo ver en las primeras pruebas. De hecho recupera de manera precisa la posición inicial y muestra una gran simetría entre los resultados obtenidos entre la ida y la vuelta. Sin embargo, es en estas pruebas donde se ve el principal inconveniente o debilidad de Fovis. Y es que pese a ser uno de los algoritmos con menor coste computacional, en el recorrido inducido a TEO en esta prueba Fovis solo consigue obtener 52 iteraciones, frente a las 300 que obtuvo el algoritmo basado en "Perspective-n-Point". El resultado a esto es que, como se aprecia a simple vista, la gráfica que dibujan los resultados de Fovis da una forma muy escalonada.

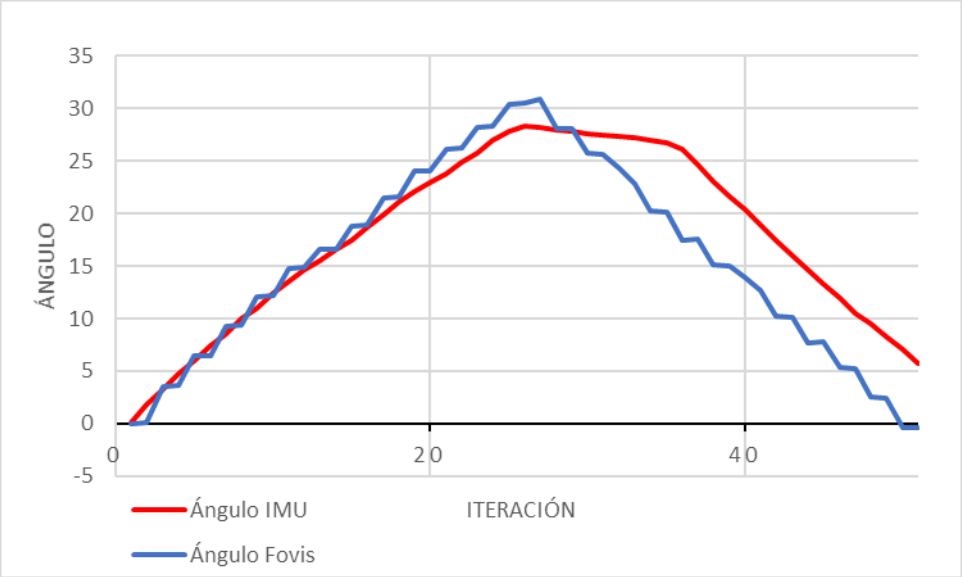

Fig. 6.13 Resultados para la segunda prueba

De nuevo, tal como se puede ver en la Fig 6.13, el Fovis acumula menor error durante el transcurso de las pruebas que el sensor inercial, consiguiendo volver al punto de partida con gran precisión.

6.3 FastICP

En un primer momento antes de analizar las condiciones existentes y optar por Fovis, se decidió utilizar el algoritmo FastICP. Se decidió implementar este algoritmo debido a su gran popularidad y a que estaba disponible en OpenCV 4.0.0.

Una vez implementado dicho algoritmo se llevaron a cabo las pruebas de igual manera quese ha expuesto anteriormente. Sin embargo, tal y como se puede ver en la Fig. 6.14, los resultados obtenidos no fueron precisos.

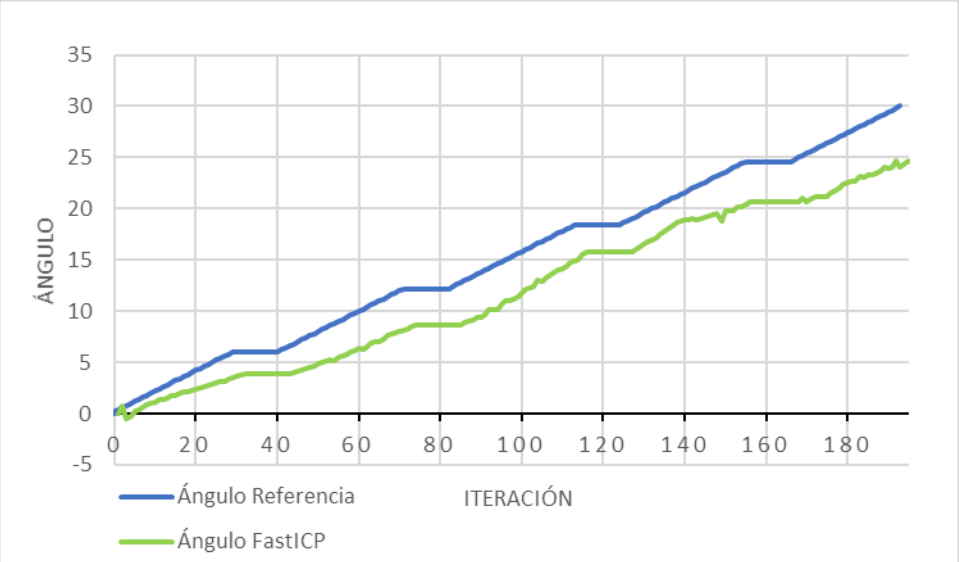

Fig. 6.14 Resultados de FastICP para una rotación en sentido positivo del eje Z

Como se puede ver, los resultados mostrados por FastICP estaban demasiado lejos del ángulo de referencia. De hecho se puede apreciar un gran aumento del error a medida con el paso del tiempo.

En vista de los resultados obtenidos se decidió descartar FastICP y optar entonces por otro algoritmo de odometría visual, el cual se adecuase a las condiciones existentes, llevándose a cabo entonces una revisión más amplia del estado del arte y optándose por elegir Fovis, como se vio previamente. El gran error de FastICP se puede explicar debido principalmente a la baja calidad de la imagen de profundidad, pues como se vio en el apartado 6.4.1.2, los algoritmos de FastICP dependen principalmente de la imagen de profundidad para llevar a cabo la estimación de posición.

7 COMPARACIÓN DE RESULTADOS Y CONCLUSIONES

En este último capítulo se compararán los resultados obtenidos, de ellos se extraerán una serie de conclusiones y valoración del trabajo realizado. Finalmente se analizarán futuros trabajos a realizar siguiendo la idea de control del equilibrio mediante visión por computador establecida en este proyecto.

7.1 Comparación de resultados

Finalmente en la Tabla 2 se recopilan los resultados obtenidos en las pruebas. Los resultados del sensor inercial (IMU) mostrados en la gráfica son la media obtenida por el IMU en las pruebas de Fovis y el método basado en "Perspective -n-Point" (solvePnP); el valor de error de IMU en sentido negativo del eje x es la media obtenida por el IMU en las pruebas realizadas con Fovis y la prueba realizada con SolvePnP en dichos eje y sentido. También se muestra el error de recuperación posición inicial en la prueba 2, indicativo de posible pérdida de precisión en el tiempo. Finalmente se expone la frecuencia de muestreo de cada método.

		IMU	SolvePnP	Fovis
Eje x sentido positivo	$\bar{\theta}(deg)$	0.163	1.221	0.089
	σ	0.051	0.052	0.106
Eje x sentido negativo	$\bar{\theta}(deg)$	0.144	2.068	0.401
	σ	0.076	1.223	0.188
Eje x media	$\bar{\theta}(deg)$	0.153	1.645	0.245
	σ	0.064	0.638	0.147
Eje y sentido positivo	$\bar{\theta}(deg)$	0.122	0.470	0.533
	σ	0.066	0.298	0.394
Eje y sentido negativo	$\bar{\theta}(deg)$	0.114	0.444	0.163
	σ	0.077	0.289	0.088
Eje y media	$\bar{\theta}(deg)$	0.118	0.457	0.348
	σ	0.143	0.293	0.241
Eje z sentido positivo	$\bar{\theta}(deg)$	3.157	0.533	0.328
	σ	1.178	0.383	0.231
Eje z sentido negativo	$\bar{\theta}(deg)$	2.253	0.859	0.402
	σ	0.971	0.475	0.293
Eje z media	$\bar{\theta}(deg)$	2.705	0.697	0.365
	σ	1.074	0.429	0.262
Error recuperación pose inicial		$\theta(deg)$	0.498	0.329
Frecuencia muestreo (Muestras/seg)			9.457	1.639

Tabla 2 Resultados obtenidos por los diferentes algoritmos y el IMU en las pruebas realizadas

En vista de los resultados se puede comprobar que la precisión de Fovis es superior a la precisión del método basado en “Perspective-n-Point, sin embargo la diferencia no es demasiado grande.

En cuanto a los resultados del error en los ejes en los que no se produjo movimiento se puede constatar como ya se vio en las gráficas, que ambos algoritmos presentan mayor error que el sensor inercial. En cualquier caso los resultados vuelven a no ser demasiado negativos, siendo semejantes a los mostrados en el eje que si fue rotado.

Resulta llamativo el hecho de que Fovis presente una ligera ventaja en el error de recuperación de la posición, pues Fovis al ser un método de odometría debería tener un error que se acumule con el tiempo, sin embargo parece que es capaz de mantener una gran precisión. Por su parte, como era de esperar, el algoritmo basado en “Perspective-n-Point” superó en gran medida a Fovis en la frecuencia de muestreo. El sensor inercial, tal y como se mostró en los resultados de las pruebas acumula un gran error como se esperaba.

Fovis ha sido capaz de obtener unos resultados superiores en precisión al método basado en “Perspective-n-Point”. Aunque la frecuencia de muestreo de Fovis sea menor, siendo Fovis capaz de mostrar 1.6 resultados por segundo no se considera que esta desventaja pudiera tener gran influencia a la hora de obtener la posición y controlar el equilibrio de TEO.

En definitiva Fovis resulta una opción más interesante de cara a ser implementado como algoritmo para obtención de la pose y control del equilibrio en robots humanoides. No requiere de ningún tipo de patrón o referencia externa, presenta una gran precisión (Pese a que como se vio en el capítulo 6 la baja calidad de la imagen de profundidad debería haber afectado significativamente a este), la frecuencia de muestreo parece resultas suficientemente alta como para no afectar demasiado al control del equilibrio y no acumula error a lo largo del tiempo como sucede con el IMU.

7.2 Objetivos cumplidos y valoración del trabajo realizado

A continuación se realiza una valoración de la consecución de los objetivos planteados al principio del proyecto, los cuales fueron:

- Revisión del estado del arte: Durante el presente proyecto se ha conseguido recopilar gran cantidad del estado del arte relativo a la obtención de la pose mediante visión que podían encajar en la funcionalidad deseada. Dicha recopilación se ha centrado en los dos enfoques que reúnen los requisitos necesarios para el control de equilibrio mediante visión, es decir, métodos basados en "perspective-n-point" y odometría visual. En base a esta revisión del estado del arte se decidió justificadamente que Fovis y el algoritmo basado en "perspective-n-point" iterativo eran las opciones más adecuadas para obtener los mejores resultados con las condiciones existentes. Por tanto, se considera que este objetivo ha sido cumplido satisfactoriamente.
- Implementación de los algoritmos seleccionados en el robot humanoide TEO: Ambos algoritmos pudieron ser implementados en TEO. Mediante el lenguaje de programación C++ y utilizando YARP para recibir información de la cámara Kinect presente en TEO, así como realizar el lazo abierto de control para desarrollar las pruebas, controlando los motores que permiten la rotación de las articulaciones del robot humanoide.
- Realizar un control en lazo abierto para comparar los resultados obtenidos: Como se mencionó en el punto anterior, las pruebas han podido realizarse correctamente. Dichas pruebas han arrojado una gran cantidad de resultados, los cuales han permitido llevar a cabo una comparación de la cual ha sido posibles extraer conclusiones claras.
- Análisis y comparación de resultados: en vista de lo dicho en el punto anterior, se puede afirmar que se ha podido realizar una comparación satisfactoria de resultados.

Además de los objetivos planteados, cabe resaltar que los resultados obtenidos por ambos algoritmos han sido muy buenos. Especialmente interesantes han sido los resultados de Fovis. Queda bastante claro que los algoritmos de odometría visual son una opción que realmente es útil para el control del equilibrio en robots humanoides mediante visión. Sin necesidad de utilizar ningún tipo de referencias externas prefijadas, Fovis ha mostrado una gran robustez y precisión.

De acuerdo a la consecución de los objetivos planteados y a la gran utilidad que el algoritmo de odometría visual Fovis ha presentado para la misión de control de equilibrio en robots humanoides, el balance que se realiza de este proyecto es positivo.

7.3 Futuros trabajos

En vista de la gran capacidad de Fovis, existe una gran cantidad de posibilidades para continuar este proyecto.

- Analizar más algoritmos de odometría visual: Como se vio en los capítulos 6.1.4 y 6.1.5, existe una gran variedad de algoritmos de odometría visual. Cada uno de estos algoritmos encaja en unas condiciones especiales, por lo cual en diferentes condiciones cabe analizar otras opciones. Una opción muy interesante sería poder analizar algoritmos con un mayor coste computacionalmente, utilizando para ello mejores condiciones de hardware y poder así ver que precisión pueden llegar a presentar algoritmos como Realtime NDT o DEMO.
- Como se explicó en el estado del arte, los algoritmos de odometría visual son utilizados para realizar mapeados y modelos 3D, siendo la obtención de la posición solo una parte de su funcionalidad. En un futuro proyecto podría implementarse un algoritmo de V-SLAM en TEO, de este modo no solo se podría obtener la rotación de la cámara, si no un mapeado completo de un área y conocer la posición del robot en ella en tiempo real. Las posibilidades son realmente amplias.
- En este proyecto se decidió centrarse únicamente en la utilización de la cámara, sin fusionar la información proveniente de otros sensores, para imitar el papel de la visión en el equilibrio en seres humanos. Sin embargo la fusión de información proporcionada por el IMU y por odometría visual es la mejor opción para la

estimación de pose en robots humanoides. Se podría llevar a cabo una fusión del ya implementado Fovis, o utilizar una de las opciones ya existentes de algoritmos de odometría visual con sensor inercial [78].

- Sin duda alguna el futuro trabajo más interesante a realizar es llevar a cabo un control de la posición de TEO en lazo cerrado mediante odometría visual. En el presente proyecto se han realizado pruebas en lazo abierto, es decir, se ha desplazado a TEO y se ha visto cómo reaccionan los algoritmos a dicho desplazamiento, sin embargo no se ha inducido un desplazamiento en TEO en base a información obtenida por el sensor inercial. Por tanto, el futuro proyecto más reciente sería generar una actuación en TEO debido a los resultados del algoritmo de odometría. Una posible opción sería establecer en que límites de rotación debería encontrarse TEO para mantener el equilibrio, de este modo podría introducirse al sistema una rotación (empujando al robot o mediante comandos) por encima de dichos límites y de este modo el algoritmo de odometría visual llevaría a cabo una respuesta, generando una actuación para devolver a TEO a una posición de equilibrio.

BIBLIOGRAFÍA

[1] C. Breazeal *et al.*, «Humanoid Robots as Cooperative Partners for People», *Int. J. Humanoid Robots*, vol. 1, n.º 2, pp. 1-34, dic. 2003.

[2] B. Robins, K. Dautenhahn, R. T. Boekhorst, y A. Billard, «Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills?», *Univers. Access Inf. Soc.*, vol. 4, n.º 2, pp. 105-120, dic. 2005.

[3] M. A. Diftler *et al.*, «Robonaut 2 - The first humanoid robot in space», presentado en 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 2011, pp. 2178-2183.

[4] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, y M. Harnisch, «Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries», *Boston Consult. Group*, vol. 9, n.º 1, pp. 54-89, abr. 2015.

[5] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, y K. Fujimura, «The intelligent ASIMO: system overview and integration», en *IEEE/RSJ International Conference on Intelligent Robots and System*, Lausanne, Switzerland, 2002, vol. 3, pp. 2478-2483.

[6] J. Englsberger *et al.*, «Overview of the torque-controlled humanoid robot TORO», presentado en 2014 IEEE-RAS International Conference on Humanoid Robots, Madrid, Spain, 2014, pp. 916-923.

[7] «Atlas | Boston Dynamics». [En línea]. Disponible en: https://www.bostondynamics.com/atlas. [Accedido: 07-jun-2019].

[8] J. M. Collado, C. Hilario, y J. M. Armingol, «Visión por Computador para Vehículos Inteligentes», presentado en XXIV Jornadas de Automática, León, España, 2003.

[9] «The impact of computer vision on the automotive industry | Articles | Chief Innovation Officer». [En línea]. Disponible en: https://channels.theinnovationenterprise.com/articles/the-impact-of-computer-vision-on-the-automotive-industry. [Accedido: 01-dic-2018].

[10] M. Delvaux, «Draft report with recommendations to the Commission on Civil Law Rules on Robotics (2015/2103(INL))», European Parliament, may 2016.

[11] J. M. Garcia Haro, S. Martinez de la Casa Diaz, y C. Balaguer, «Balance Computation of Objects Transported on a Tray by a Humanoid Robot Based on 3D Dynamic Slopes», presentado en 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), Beijing, China, 2018, pp. 1-6.

[12] R. Q. Boptom, C. R. G., y P. Mitchell, «Visual impairment and falls in older adults: the Blue Mountains Eye Study», *J. Am. Geriatr. Soc.*, vol. 46, n.º 1, pp. 58-64, 1998.

[13] B. E. K. Klein, R. Klein, K. E. Lee, y K. J. Cruickshanks, «Performance-based and self-assessed measures of visual function as related to history of falls, hip fractures, and measured gait time: The beaver dam eye study», *Ophthalmology*, vol. 105, n.º 1, pp. 160-164, ene. 1998.

[14] A. L. Coleman *et al.*, «Higher risk of multiple falls among elderly women who lose visual acuity», *Ophthalmology*, vol. 111, n.º 5, pp. 857-862, may 2004.

[15] D. N. Lee y E. Aronson, «Visual proprioceptive control of standing in human infants», *Percept. Psychophys.*, vol. 15, n.º 3, pp. 529-532, may 1974.

[16] K. Horiuchi, M. Ishihara, y K. Imanaka, «The essential role of optical flow in the peripheral visual field for stable quiet standing: Evidence from the use of a head-mounted display», *PLoS ONE*, vol. 12, n.º 10, oct. 2017.

[17] A. Goswami, «Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point», *Int. J. Robot. Res.*, vol. 18, n.º 6, pp. 523-533, jun. 1999.

[18] G. G. Muscolo, C. T. Recchiuto, y R. Molfino, «Vision and locomotion control systems on a bio-inspired humanoid robot», en *MELECON 2014 - 2014 17th IEEE Mediterranean Electrotechnical Conference*, Beirut, Lebanon, 2014, pp. 380-385.

[19] Santiago Martinez, Juan Garcia-Haro, Juan Victores, Alberto Jardon, y Carlos Balaguer, «Experimental Robot Model Adjustments Based on Force–Torque Sensor Information», *Sensors*, vol. 18, n.º 3, p. 836, mar. 2018.

[20] A. Goswami y V. Kallem, «Rate of change of angular momentum and balance maintenance of biped robots», en *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, New Orleans, LA, USA, 2004, pp. 3785-3790 Vol.4.

[21] A. Kun y W. T. Miller, «Adaptive dynamic balance of a biped robot using neural networks», en *Proceedings of IEEE International Conference on Robotics and Automation*, Minneapolis, MN, USA, 1996, vol. 1, pp. 240-245.

[22] J. Lorente, «Equilibrium control for humanoid robot TEO by intertial perception», Tesis de Master, Universidad Carlos III, Leganes, Madrid, 2018.

[23] O. J. Woodman, «An introduction to inertial navigation», University of Cambridge, Computer Laboratory, UCAM-CL-TR-696, 2007.

[24] D. J. Agravante, J. Pages, y F. Chaumette, «Visual servoing for the REEM humanoid robot's upper body», en *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 5253-5258.

[25] Y. K. Yu, K. H. Wong, M. Ming, y Y. Chang, «Pose Estimation for Augmented Reality Applications Using Genetic Algorithm», *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 35, n.º 6, pp. 1295-1301., dic. 2005.

[26] P. Wunsch, S. Winkler, y G. Hirzinger, «Real-Time Pose Estimation of 3-D Objects from Camera Images Using Neural Networks», en *Proceedings of International Conference on Robotics and Automation*, 1997, vol. 4, pp. 3232-3237.

[27] H. Durrant-Whyte y T. Bailey, «Simultaneous localization and mapping: part I», *IEEE Robot. Autom. Mag.*, vol. 13, n.º 2, pp. 99-110, jun. 2006.

[28] T. Lemaire, C. Berger, I.-K. Jung, y S. Lacroix, «Vision-Based SLAM: Stereo and Monocular Approaches», *Int. J. Comput. Vis.*, vol. 74, n.º 3, pp. 343-364, jul. 2007.

[29] P. Elinas, R. Sim, y J. J. Little, «σSLAM: stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution», en *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, Orlando, FL, USA, 2006, pp. 1564-1570.

[30] G. Nützi, S. Weiss, D. Scaramuzza, y R. Siegwart, «Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM», *J. Intell. Robot. Syst.*, vol. 61, n.º 1-4, pp. 287-299, ene. 2011.

[31] A. S. Huang *et al.*, «Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera», en *Robotics Research*, vol. 100, H. I. Christensen y O. Khatib, Eds. Cham: Springer International Publishing, 2017, pp. 235-252.

[32] D. Nistér, O. Naroditsky, y J. Bergen, «Visual odometry for ground vehicle applications», *J. Field Robot.*, vol. 23, n.º 1, pp. 3-20, ene. 2006.

[33] V. Angladon *et al.*, «An evaluation of real-time RGB-D visual odometry algorithms on mobile devices», *J. Real-Time Image Process.*, 2017.

[34] G. Metta, P. Fitzpatrick, y L. Natale, «Yarp: Yet another robot platform», *Int. J. Adv. Robot. Syst.*, vol. 3, n.º 1, 2006.

[35] R. Malcom, «The Malcolm horizon: History and future», en *NASA. Dryden Flight Research Center Peripheral Vision Horizon Display (PVHD)*, United States, 1984, pp. 11-40.

[36] B. Frank, C. Stachniss, G. Grisetti, K. Arras, y W. Burgard, «Robotics 2 Camera Calibration», Albert-Ludwigs-Universität Freiburg.

[37] X. Xin Lu, «A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation», Journal of Physics Conference Series, vol. 1087, n.º 5, sep. 2018.

[38] R. Horaud, B. Conio, O. Leboulleux, y B. Lacolle, «An analytic solution for the perspective 4-point problem», *Comput. Vis. Graph. Image Process.*, vol. 47, n.º 1, pp. 33-44, jul. 1989.

[39] X.-S. Gao, X.-R. Hou, J. Tang, y H.-F. Cheng, «Complete Solution Classification for the Perspective-Three-Point Problem», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 930-943, ago. 2003.

[40] L. Kneip, D. Scaramuzza, y R. Siegwart, «A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation», en *CVPR 2011*, Colorado Springs, CO, USA, 2011, pp. 2969-2976.

[41] M. Persson y K. Nordberg, «Lambda Twist: An Accurate Fast Robust Perspective Three Point (P3P) Solver», en *Computer Vision – ECCV 2018*, vol. 11208, V. Ferrari, M. Hebert, C. Sminchisescu, y Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 334-349.

[42] L. Zhi y J. Tang, «A Complete Linear 4-Point Algorithm for Camera Pose Determination», *AMSS Acad. Sin.*, vol. 21, pp. 239-249, dic. 2002.

[43] Z. Y. Hu y F. C. Wu, «A note on the number of solutions of the noncoplanar P4P problem», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, n.º 4, pp. 550-555, abr. 2002.

[44] S. Li, C. Xu, y M. Xie, «A Robust O(n) Solution to the Perspective-n-Point Problem», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, n.º 7, pp. 1444-1450, jul. 2012.

[45] J. A. Hesch y S. I. Roumeliotis, «A Direct Least-Squares (DLS) method for PnP», en *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 383-390.

[46] S. Roweis, «Levenberg-Marquardt Optimization», en *Notes, University Of Toronto*, 1996.

[47] C.-P. Lu, G. D.Hager, y E. Mjolsness, «Fast and Globally Convergent Pose Estimation From Video Images», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 610-622, jun. 2000.

[48] P. Y. Lee y J. B. Moore, «Pose Estimation via Gauss-Newton-on-manifold», presentado en Proceedings of the Sixth International Symposium on Mathematical Theory of Networks and Systems, Pacific Grove, California, USA, 2004, pp. 131-135.

[49] E. R. Pi, «Implementation of a 3D pose estimation algorithm», Tesis de Máster, Universitat Politecnica de Catalunya, 2015.

[50] «Tutorial Camera Calibration - BoofCV». [En línea]. Disponible en: https://boofcv.org/index.php?title=Tutorial_Camera_Calibration. [Accedido: 15-jun-2019].

[51] Z. Zhang, «A flexible new technique for camera calibration», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, n.º 11, pp. 1330-1334, nov. 2000.

[52] D. Scaramuzza y F. Fraundorfer, «Visual Odometry [Tutorial]», *IEEE Robot. Autom. Mag.*, vol. 18, n.º 4, pp. 80-92, dic. 2011.

[53] D. Nistér, O. Naroditsky, y J. Bergen, «Visual Odometry», *IEEE Trans. Pattern Anal. Mach. Intell.*, jun. 2004.

[54] H. P. Moravec, «Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.», STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, Informe técnico ADA092604, sep. 1980.

[55] M. Maimone, Y. Cheng, y L. Matthies, «Two years of Visual Odometry on the Mars Exploration Rovers», *J. Field Robot.*, vol. 24, n.º 3, pp. 169-186, mar. 2007.

[56] J. K. Aggarwal y N. Nandhakumar, «On the Computation of Motion from Sequences of Images - A Review», *Proc. IEEE*, vol. 76, pp. 917-935, ago. 1988.

[57] G. N. Desouza y A. C. Kak, «Vision for mobile robot navigation: A survey», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, n.º 2, feb. 2002.

[58] F. Fraundorfer y D. Scaramuzza, «Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications», *IEEE Robot. Autom. Mag.*, vol. 19, n.º 2, pp. 78-90, jun. 2012.

[59] S. Poddar, R. Kottath, y V. Karar, «Evolution of Visual Odometry Techniques», *ArXiv Prepr. ArXiv 180411142*, abr. 2018.

[60] L. Matthies y S. Shafer, «Error Modeling in Stereo Navigation», *IEEE J. Robot. Autom.*, vol. 3, n.º 3, pp. 239-248, jun. 1987.

[61] L. Matthies, «Dynamic Stereo Vision», Tesis doctoral, Carnegie Mellon University, Pittsburgh, PA, USA, 1989.

[62] Y. Cheng, M. W. Maimone, y L. Matthies, «Visual Odometry on the Mars Exploration Rovers», en *IEEE International Conference on Systems, Man ans Cybernetics*, Waikoloa, HI, USA, 2006, vol. 1, pp. 903-910.

[63] P. J. Besl y N. Mckay, «A Method for Registration of 3-D Shapes», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, n.º 2, feb. 1992.

[64] Z. Fang y Y. Zhang, «Experimental Evaluation of RGB-D Visual Odometry Methods», *Int. J. Adv. Robot. Syst.*, vol. 12, n.º 3, p. 26, mar. 2015.

[65] C. Harris y M. Stephens, «A Combined Corner and Edge Detector», en *Procedings of the Alvey Vision Conference 1988*, Manchester, 1988, vol. 15, pp. 147-157.

[66] H. Bay, T. Tuytelaars, y L. Van Gool, «SURF: Speeded Up Robust Features», *Comput. Vis. Image Underst.*, vol. 110, n.º 3, pp. 346-359, jun. 2008.

[67] E. Rosten y T. Drummond, «Machine learning for high-speed corner detection», presentado en European Conference on Computer Vision, Springer, Berlin, Heidelberg., 2006, pp. 430–443.

[68] C. Kerl, J. Sturm, y C. Kerl, «Robust odometry estimation for RGB-D cameras», presentado en 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 3748-3754.

[69] K. Pathak, A. Birk, N. Vaškevičius, y J. Poppinga, «Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping», *IEEE Trans. Robot.*, vol. 26, n.º 3, pp. 424-441, jun. 2010.

[70] R. A. Newcombe *et al.*, «KinectFusion: Real-Time Dense Surface Mapping and Tracking», presentado en ISMAR, 2011, vol. 11, pp. 127-136.

[71] A. Segal, D. Haehnel, y S. Thrun, «Generalized-ICP», *Robot. Sci. Syst.*, vol. 2, n.º 4, pp. 435-443, jun. 2009.

[72] H. Spies, B. Jähne, y J. L. Barron, «Range Flow Estimation», *Comput. Vis. Image Underst.*, vol. 85, n.º 3, pp. 209-231, mar. 2002.

[73] H. Spies y J. L. Barron, «Evaluating the range flow motion constraint», presentado en 16th International Conference on Pattern Recognition, Quebec City, Que., Canada, 2002, vol. 3, pp. 517-520.

[74] M. Jaimez y J. Gonzalez-Jimenez, «Fast Visual Odometry for 3-D Range Sensors», *IEEE Trans. Robot.*, vol. 31, n.º 4, pp. 809-822, ago. 2015.

[75] H. Andreasson y T. Stoyanov, «Real time registration of rgb-d data using local visual features and 3d-ndt registration», presentado en Int. Conf. on Robotics and Automation (ICRA), 2012.

[76] I. Dryanovski y R. G. Valenti, «Fast visual odometry and mapping from RGB-D data», presentado en 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 2305–2310.

[77] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, y J. Mcdonald, «Robust real-time visual odometry for dense RGB-D mapping», presentado en 2013 IEEE International Conference on Robotics and Automation, 2013.

[78] J. Delmerico y D. Scaramuzza, «A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots», presentado en 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 2502-2509.