Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Sistema de odometría visual e inercial con un marcador

Tesis que presenta

Sergio Alberto Herrera Castro

para obtener el grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. Luis Gerardo de la Fraga

Ciudad de México Octubre, 2019

Resumen

En esta tesis se especifican a detalle las consideraciones necesarias para el diseño y construcción de un sistema de odometría visual e inercial. Los componentes del sistema propuesto fueron elegidos de acuerdo a ciertas restricciones de tamaño, precio y poder de cómputo, aportando así, un aparato asequible y escalable para diversas aplicaciones.

El hardware se conforma por un computadora en una sola tarjeta Raspberry Pi 3 junto con su respectivo módulo de cámara y un sensor inercial MPU-6050 de bajo costo y tamaño reducido que va montado a un marcador fiduciario; mismo que es rastreado por el sistema a partir de un punto en el rango de visión. Se diseñó un objeto de acrílico para acoplar el sensor inercial con el marcador haciendo coincidir ambos sistemas de coordenadas. Este objeto permite el uso de marcadores intercambiables y se conecta a la computadora para obtener información a partir de ambos sensores.

El software desarrollado fue codificado en lenguaje C/C++ e incorpora los modelos de odometría visual e inercial aquí descritos para combinar los datos de ambos sensores mediante un filtro Kalman. Estos modelos incluyen el emparejamiento de puntos del modelo del marcador y su imagen en la escena usando el tipo de orden, el cálculo de la pose del marcador mediante una homografía, y la integración de las señales proporcionadas por la unidad de medición inercial para implementar un sistema de navegación inercial.

Abstract

This thesis specifies in detail the considerations necessary for the design and construction of a visual and inertial odometry system. The components of the proposed system were chosen according to certain restrictions of size, price and computing power, thus providing an affordable and scalable device for various applications.

The hardware consists of a single board computer Raspberry Pi 3 along with its respective camera module and an MPU-6050 inertial sensor of low cost and reduced size that is mounted to a fiducial marker; same that is tracked by the system from a point in the range of vision. An acrylic object was designed to couple the inertial sensor with the marker by matching both coordinate systems. This object allows the use of interchangeable markers and connects to the computer to obtain information from both sensors.

The software developed was coded in C/C++ language and incorporates the visual and inertial odometry models described here to combine the data of both sensors using a Kalman filter. These models include the pairing of points of the marker model and its image in the scene using the order type, the calculation of the pose of the marker by means of a homography, and the integration of the signals provided by the inertial measurement unit to implement an inertial navigation system.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca económica que me proporcionó durante estos dos años de maestría.

De igual manera, agradezco al Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV), en particular al Departamento de Computación, por aceptarme en su programa de maestría en ciencias en computación.

Agradezco al Dr. Luis Gerardo de la Fraga por aceptar mi propuesta de tesis, por el tiempo y esfuerzo que dedicó al trabajo.

También agradezco al Dr. Ernesto Olguín Díaz por colaborar en la formulación de la solución y por su apoyo durante mi estancia en el Cinvestav Unidad Saltillo.

A mi familia y amigos.

Índice general

Resumen III

Abstract V

Agradecimientos VII

Índice de figuras XI

Índice de tablas XII

1. Introducción 1
- 1.1. El problema de odometría . . . 1
- 1.2. Motivación . . . 4
- 1.3. Solución propuesta . . . 6
- 1.4. Objetivos generales y específicos de la tesis . . . 7
- 1.5. Metodología . . . 7
- 1.6. Contribuciones . . . 8
- 1.7. Estructura de la tesis . . . 8

2. Antecedentes 9
- 2.1. Revisión bibliográfica . . . 9
- 2.2. Información visual . . . 11
  - 2.2.1. Modelo de la cámara obscura . . . 11
  - 2.2.2. Parámetros intrínsecos de la cámara . . . 13
  - 2.2.3. Parámetros extrínsecos de la cámara . . . 13
  - 2.2.4. Calibración de la cámara . . . 15
- 2.3. Información inercial . . . 20
  - 2.3.1. Sensores inerciales . . . 20
  - 2.3.2. Acelerómetros . . . 21
  - 2.3.3. Giroscópios . . . 22
  - 2.3.4. Navegación inercial . . . 23

3. Diseño y construcción del sistema 25
- 3.1. Plataforma Raspberry Pi . . . 25
- 3.2. Cámara Raspberry . . . 27

3.2.1. Obtención de la imagen 27
3.3. Marcador fiduciario 28
3.3.1. Detección del marcador en la imagen 29
3.3.2. Correspondencia de puntos según el tipo de orden 32
3.4. Unidad de Medición Inercial 34
3.5. Conexión y ensamble del sistema 35
3.6. Montaje del sistema 37

4. Odometría visual-inercial 39
4.1. Odometría visual 39
4.1.1. Calibración fuera de línea 40
4.1.2. Obtención de la pose del marcador 42
4.1.3. Modelo de odometría visual 42
4.2. Odometría inercial 44
4.2.1. Cinemática del cuerpo rígido 44
4.2.2. Calibración de la UMI 46
4.2.3. Cálculo de la posición angular, velocidad y desplazamiento . . 48
4.3. Modelo del sistema visual-inercial 48

5. Filtro Kalman para fusión de datos visuales e inerciales 51
5.1. Modelo del filtro Kalman 51
5.2. Implementación del filtro extendido 53
5.2.1. Cuaternio como modelo de orientación 55
5.2.2. Integración de la velocidad angular mediante un cuaternio . . 56

6. Pruebas y resultados 59
6.1. Prueba del sistema visual 59
6.2. Prueba del sistema inercial 60
6.3. Prueba del sistema acoplado con el filtro Kalman 63
6.4. Resultados filtro Kalman 65

7. Conclusiones 73
7.1. Trabajo a futuro 74

A. Notación 77

B. Parámetros obtenidos en la calibración fuera de línea 79

C. Scripts para simulación del modelo de odometría visual 81

D. Experimentos para determinar la varianza del sistema 89

Bibliografía 91

Índice de figuras

1.1. Réplica de un odómetro romano del siglo I a.C 1
1.2. Odómetro inglés moderno 2
1.3. Ilustración del problema de odometría visual. 3
1.4. Uno de los dos giroscopios para control de rumbo utilizados en los cohetes V2. 3
1.5. Diagrama de los componentes del sistema de odometría visual-inercial 6

2.1. Fotografía del vehículo autónomo desarrollado en [20] 10
2.2. Funcionamiento de una cámara obscura. 11
2.3. Geometría de la cámara obscura 12
2.4. Homografía H que relaciona puntos entre dos planos. 15
2.5. Aparato vestibular humano 20
2.6. Diagrama de un acelerómetro mecánico de un solo eje. 21
2.7. Giroscopio mecánico del siglo XIX, del museo CNAM en Francia. . . 22
2.8. Diagrama de un giroscopio de estructura vibratoria [3]. 23
2.9. Diagrama de un SNI simplificado [3]. 24

3.1. Comparativa de una computadora Raspberry Pi 3 modelo B con una mano humana. 25
3.2. Configuración de los puertos y componentes de una Raspberry Pi 3 modelo B. 27
3.3. Módulo de cámara Raspberry Pi v2, con su respectivo bus de datos para el puerto CSI. 27
3.4. Modelo del marcador propuesto. 28
3.5. Matrices λ de la cubierta convexa del marcador. 32
3.6. Etiquetado y correspondencia de puntos entre el modelo del marcador y el marcador detectado en la imagen. 33
3.7. Unidad de medición inercial MPU-6050 de Invensense 34
3.8. Módulo de cámara conectado a la Raspberry Pi. 35
3.9. Diagrama de cableado entre la Raspberry y el sensor MPU-6050. . . . 35
3.10. Sistemas de coordenadas del marcador y de la UMI 36
3.11. Ensamblaje del objeto con la UMI y el marcador 36
3.12. Interfaz desplegada por el sistema. 37
3.13. Objeto construido para acoplar la UMI con el marcador. 38

3.14. Disposición de los componentes del sistema propuesto. 38
3.15. Sistema con todos sus componentes conectado a la interfaz HDMI. . . 38

4.1. Marcador fiduciario chAruco utilizado para la calibración de la cámara. 40
4.2. Algunas de las fotografías realizadas para la calibración de la cámara. 41
4.3. Diagrama del modelo propuesto de odometría visual 42
4.4. Marcos de referencia asociados a un cuerpo rígido móvil libre en el espacio [8]. 44
4.5. Diagrama del cuerpo rígido móvil libre 47

6.1. Detección del marcador del objeto en la escena. 59
6.2. Capturas del sistema de odometría visual. 60
6.3. Inicialización del sistema inercial. 61
6.4. Integración de las señales del giroscópio por medio de un cuaternio . . 62
6.5. Integración de las señales inerciales con el objeto en reposo. 62
6.6. Capturas de la prueba del filtro Kalman cuando existe oclusión del objeto o no se puede detectar. 63
6.7. Capturas de la prueba del filtro Kalman mientras se mueve el objeto en la escena. 64
6.8. Escenarios para las pruebas del filtro Kalman. 66
6.9. Gráficas de la trayectoria del objeto durante los experimentos en el escenario 1 reiniciando el filtro. 67
6.10. Gráficas de la trayectoria del objeto durante los experimentos en el escenario 1 sin reiniciar el filtro. 68
6.11. Gráficas de la trayectoria del objeto durante los experimentos en el escenario 2 reiniciando el filtro. 69
6.12. Gráficas de la trayectoria del objeto durante los experimentos en el escenario 2 sin reiniciar el filtro. 70

D.1. Disposición de la cámara y el marcador durante los experimentos de la varianza del sistema visual. 89

Índice de tablas

2.1. Clasificación y comparativa de los diferentes tipos de UMI en el mercado [1]. 21

3.1. Especificaciones técnicas de la Raspberry Pi model0 3. 26

6.1. Configuraciones del filtro Kalman utilizadas para los experimentos. . 65

D.1. Listado de experimentos con las distintas poses del marcador con respecto a la cámara. 90
D.2. Valores calculados a partir de $\boldsymbol{\theta}$ con el marcador a 30 cm de la cámara. 90
D.3. Valores calculados a partir de $\mathbf{t}$ con el marcador a 30 cm de la cámara. 90
D.4. Valores calculados a partir de $\boldsymbol{\theta}$ con el marcador a 60 cm de la cámara. 90
D.5. Valores calculados a partir de $\mathbf{t}$ con el marcador a 60 cm de la cámara. 90

Capítulo 1

Introducción

Este primer capítulo comienza describiendo brevemente el problema de odometría y la solución propuesta sobre la que trata esta tesis. Al final de ese capítulo se detalla la metodología empleada para el desarrollo de este trabajo, las contribuciones del mismo y la estructura general del documento.

1.1. El problema de odometría

Un odómetro es un instrumento de medición que calcula la distancia recorrida por un objeto para determinada trayectoria. La raíz de la palabra se deriva del griego *hodos*-camino y *métron*-medida. Así pues, los primeros registros sobre la construcción de odómetros datan del siglo I a.C., y señalan al matemático griego Arquímedes como su inventor [26].

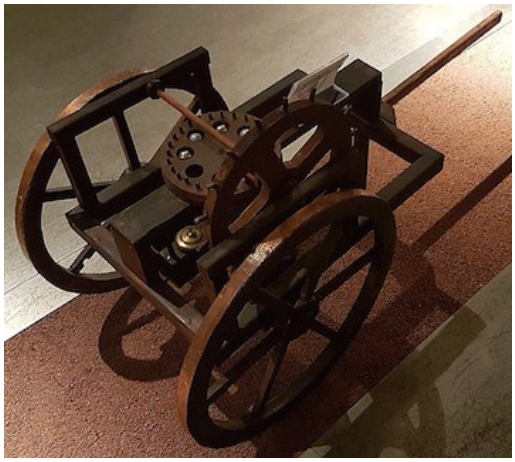


Figura 1.1: Réplica de un odómetro romano del siglo I a.C. Las ruedas tenían un diámetro de 4 pies romanos (29.5cm por pie) y cada 400 revoluciones se obtenía exactamente una milla romana (1481m). Cuando esto sucedía, el mecanismo hacía sonar una piedra dejándola caer contra un contenedor metálico. Al final del proceso de medición, el numero de piedras en el contenedor indican la distancia recorrida

Aunque actualmente existen odómetros más complejos, todos se basan en el mismo principio, una rueda de diámetro conocido que gira fija sobre un eje. A partir del número de revoluciones que se acumulen en el trayecto, es posible calcular la distancia lineal recorrida por el aparato.

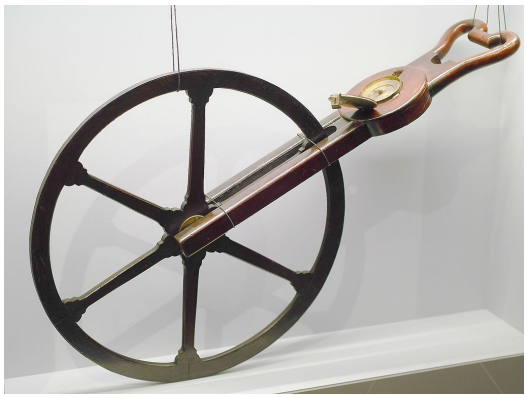


Figura 1.2: Odómetro inglés moderno de una sola mano, fabricado alrededor de 1760.

Sin embargo, los sistemas de odometría actuales no están limitados a utilizar un sólo tipo de instrumento de medición. Durante los años 80, con la inercia de la carrera espacial de la década pasada, se comenzó a desarrollar la idea de incluir cámaras fotográficas en los vehículos de exploración extraterrestre. A partir de una serie de imágenes captadas por una cámara, es posible recuperar la pose relativa y orientación en el espacio tridimensional de la cámara misma; este procedimiento forma parte de una colección de técnicas de fotogrametría conocida como *estructura a partir de movimiento* (EPM).

Surge entonces, el término de odometría visual, retomado por Nister en 2004 para el problema de EPM orientado al ámbito de visión por computadora [20]. De acuerdo con Scaramuzza [24], la odometría visual se enfoca en la estimación secuencial y en tiempo real del desplazamiento de la cámara en el espacio tridimensional.

La principal diferencia y ventaja entre la odometría de vehículos terrestres y la odometría visual, (asumiendo que la cámara está adjunta de manera rígida al cuerpo en movimiento) es que la odometría visual no padece de deslizamientos del vehículo a causa de terreno no uniforme, que añaden error a la medición [20]. También es útil en lugares donde usualmente no se pueden utilizar otro tipo de señales como *GPS* o bien hay poca precisión en las lecturas, ya sea en ambientes submarinos, aéreos o dentro de edificios o cavernas.

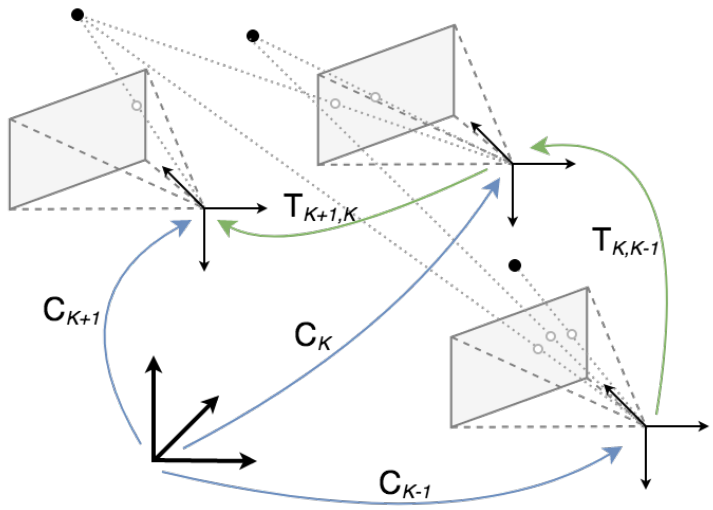


Figura 1.3: Ilustración del problema de odometría visual [24]. Las poses relativas $T_{k,k-1}$ de la cámara se calculan a partir de características visuales, y se concatenan para obtener las poses absolutas C_k con respecto de un marco de referencia inicial en $k = 0$.

Por su parte, los sistemas de navegación inercial (SNI) utilizan sensores inerciales autocontenidos para rastrear la posición y orientación de un objeto relativo a un punto conocido en el espacio, esto sin necesidad de una referencia externa. Aunque su invención se planteó inicialmente para aplicaciones militares (uno de los primeros casos es el del famoso cohete alemán V2 de la Segunda Guerra Mundial), actualmente son utilizados en un sinfín de aplicaciones en vehículos como automóviles, barcos, aviones, submarinos, naves espaciales, vehículos no tripulados y de navegación autónoma.

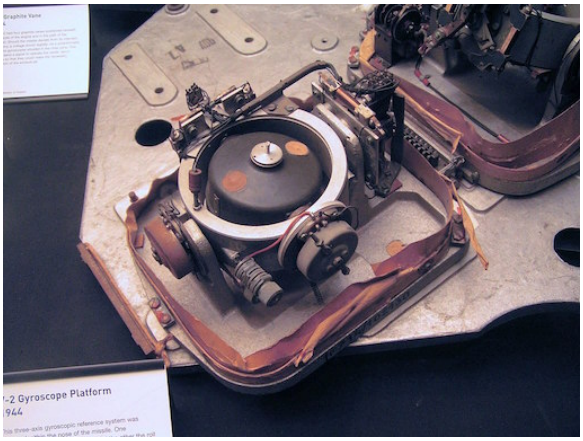


Figura 1.4: Uno de los dos giroscopios para control de rumbo utilizados en los cohetes V2.

Comúnmente, se le denomina Unidad de Medición Inercial (UMI) al conjunto de sensores utilizados en los SNI. Existen tres tipos de sensores que usualmente se incorporan a la UMI: acelerómetros para medir movimientos (de traslación), giroscopios para medir rotaciones, y ocasionalmente, magnetómetros para medir campos magnéticos (útiles para el reconocimiento del norte magnético terrestre). Como detalla Corke en su artículo [3], existen en la naturaleza algunas aves que son capaces de orientarse con el norte magnético del planeta. Sin embargo los sensores magnéticos utilizados en los SNI son propensos a obtener lecturas perturbadas por estructuras metálicas y corrientes eléctricas presentes en los alrededores del espacio de operación.

Generalmente las UMI incorporan más de un sensor de la misma categoría dispuestos de manera ortogonal para obtener mediciones en más de una dimensión. Entonces, para cada instante de tiempo la posición angular del objeto es obtenida con respecto a un marco de referencia local a través de la integración de la señal del giroscopio (con respecto del tiempo) y una vez actualizada la pose con respecto al marco de referencia global, el SNI logra obtener la velocidad lineal y posteriormente la posición del objeto a través de la doble integración de la señal del acelerómetro [3].

1.2. Motivación

Desde la invención del transistor, los circuitos y sensores han visto pasar nuevas técnicas y materiales que los han afectado en tamaño y costo. La micro-integración y producción en masa de este tipo de sensores (visuales e inerciales) dieron pie a que, en años recientes, fueran incluidos por los fabricantes de toda clase de dispositivos móviles con recursos de cómputo y energía restringidos [3].

En la industria, los sensores inerciales han ganado terreno debido a la popularización de aplicaciones basadas en interacciones kinésicas. No obstante, los modelos de orientación actuales basados en sensores puramente inerciales presentan un efecto de deriva (*drift* en inglés) o pérdida debida al ruido inherente en las lecturas, que se acumula con el paso del tiempo, lo que ha obligado a mantener un marco de referencia externo para la corrección de deriva del sistema.

Aunque los sensores inerciales son actualmente muy pequeños y eficientes en términos energéticos, los modelos comerciales de gama media-baja son capaces de operar a frecuencias mayores a los 100 Hz. Estas circunstancias sitúan a este tipo de sensores como suficientemente fiables en movimientos bruscos y por intervalos cortos de tiempo.

Por otro lado, con el abaratamiento y producción en masa de las cámaras digitales, los sistemas de visión y realidad aumentada han sido implementados en toda clase de plataformas. En décadas pasadas, la mayoría de las aplicaciones en el ámbitos empleaban un marcador físico como referencia externa para la calibración de la cámara.

En la actualidad, gran cantidad de trabajos desarrollados emplean detectores de características naturales para extender su rango de operación a ambientes diversos

prescindiendo así del un marcador fiduciario. Aunque estos tipos de detectores pueden adaptarse según el ambiente de operación, gran parte del tiempo de procesamiento se invierte en el empate de características entre las imágenes obtenidas, lo que implica mayor complejidad en el algoritmo y por ende mayor potencia requerida en el sistema [25].

Además, el procesamiento de imágenes requerido para técnicas de odometría visual suele ser muy lento y costoso en términos computaciones, aunado a que padece de factores externos como los cambios iluminación, movimientos repentinos u oclusión del marcador, que limitan o condicionan la capacidad de operación y fiabilidad de los sistemas visuales.

Por esta razón, en este trabajo se propone acotar el problema de la detección y empate de características visuales mediante el uso de un marcador fiduciario empatando los puntos como presenta de la Fraga en [4, 6, 14]. Entonces, calculando la homografía del modelo del marcador con la del marcador detectado en la imagen, es posible corregir el efecto de deriva que padece el sistema inercial.

1.3. Solución propuesta

En este trabajo se detalla el diseño y construcción de un sistema de odometría visual-inercial implementado en un computadora de una sola tarjeta; capaz de calcular la posición y orientación de un objeto, relativo a un punto en el espacio.

El sistema se ejecuta en una Raspberry Pi 3 conectada a su respectiva cámara digital, la cual obtiene imágenes de un marcador prediseñado utilizado como referencia física. Además, se acopló una UMI (Invensense MPU-6050 [15]) al reverso del marcador, que se encuentra de igual forma conectada a la computadora. Con dicho sistema es posible estimar la pose (posición y orientación) del marcador en el espacio, aún cuando existe oclusión del marcador o este sale de la vista de la cámara.

Se despliega también un objeto virtual sobre el marcador mediante una interfaz de realidad aumentada para demostrar la correctitud y fiabilidad del sistema construido.

El sistema propuesto se ilustra mediante el diagrama en la Figura 1.5 y los detalles referentes al diseño y construcción se examinan a detalle en el capítulo 3.

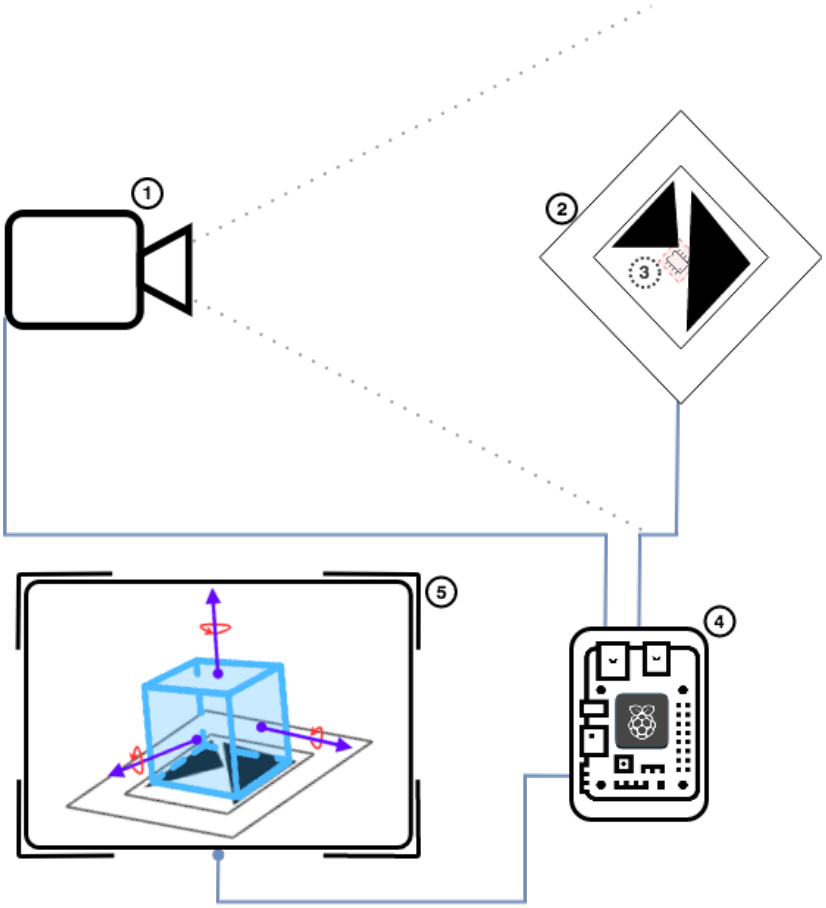


Figura 1.5: Diagrama de los componentes del sistema de odometría visual-inercial: 1) cámara, 2) marcador, 3) UMI (situada al reverso del marcador), 4) computadora Raspberry Pi 3, 5) interfaz de realidad aumentada.

1.4. Objetivos generales y específicos de la tesis

General

Implementar un sistema de odometría visual e inercial en un dispositivo con poder de cómputo y consumo eléctrico restringidos, que sea capaz de calcular de manera congruente y en tiempo real la pose del marcador en el espacio de operación con respecto de una pose inicial que ve la cámara, y desplegar además, un ambiente gráfico de realidad aumentada utilizando los datos del sistema, aún cuando existe oclusión del marcador o este no se puede detectar.

Específicos

1. Analizar de las señales inerciales obtenidas con la UMI.

2. Definir el modelo de odometría visual con un marcador usando el modelo de la cámara obscura.

3. Diseñar un objeto que permita acoplar ambos marcos de referencia, de los sistemas visual e inercial.

4. Formular e implementar un algoritmo para la fusión de señales visuales e inerciales, como es el filtro Kalman.

5. Diseñar una interfaz de realidad aumentada para verificar la correctitud del sistema.

6. Comparar los datos obtenidos con el sistema contra datos de medición reales.

1.5. Metodología

1. Adquirir la información de la UMI a través de la Raspberry Pi.

2. Calibrar la cámara de la Raspberry Pi.

3. Implementar un algoritmo para la detección del marcador.

4. Procesar las imágenes obtenidas por la cámara para obtener la homografía correspondiente al marcador en la escena.

5. Implementar el modelo del filtro Kalman para fusión de datos de ambos sensores.

6. Incorporar una interfaz de realidad aumentada al sistema.

1.6. Contribuciones

- Modelo e implementación de un filtro Kalman para fusionar información visual e inercial, utilizando una computadora en una sola tarjeta (Raspberry Pi 3).
- Interfaz de realidad aumentada como prototipo de prueba de la correctitud y fiabilidad del sistema.
- Sistema de odometría visual e inercial utilizando marcadores fiduciarios basados en tipo de orden [4, 14] y sensores inerciales de gama baja.

1.7. Estructura de la tesis

Esta tesis se compone de siete capítulos y dos apéndices. En este primer capítulo introductorio se presenta el problema de odometría, sus orígenes y líneas de aplicación. Se plantea y motiva una solución al problema y se listan las contribuciones de el presente trabajo.

El segundo capítulo comienza con una revisión del estado del arte, exponiendo algunos prototipos que se consideran cardinales en el ámbito del problema de odometría. Después, el capítulo se extiende a dos amplias secciones donde se exponen los conceptos matemáticos utilizados para la obtención de datos a partir de ambos sensores.

El capitulo tercero se centra en el diseño y construcción de la solución propuesta en esta tesis. Se listan y describen las especificaciones técnicas de los componentes físicos utilizados en el sistema construido; junto con las herramientas de software necesarias para su codificación. Además, se ilustra el montaje y configuración de los dichos componentes en el sistema.

En el cuarto capítulo se exponen los modelos de odometría visual y odometría inercial por separado y finalmente se detalla el modelo solución propuesto para la fusión de datos.

El capítulo quinto trata sobre la implementación del filtro Kalman para la fusión de datos y el sexto en las pruebas y resultados obtenidos con el sistema.

Finalmente, en el séptimo capitulo se discuten las conclusiones de esta tesis, y se enumeran algunos puntos a considerar para el trabajo a futuro.

El apéndice A expone la notación utilizada para este documento.

En el apéndice B se listan los archivos obtenidos por el programa durante la etapa de calibración fuera de línea.

El apéndice C lista los scripts en lenguaje Python necesarios para simular el modelo de odometría visual.

El apéndice D expone los resultados obtenidos por los experimentos para determinar la varianza del sistema visual.

Capítulo 2

Antecedentes

Durante las últimas dos décadas se han desarrollado gran cantidad de trabajos de odometría visual-inercial utilizando diversos métodos para la fusión de datos [24]. Con el fin de contextualizar la solución propuesta, en seguida se exponen algunos de los trabajos más relevantes en el ámbito. Después, se formulan en dos amplias secciones las ecuaciones que describen los principios físicos bajo los que operan ambos sensores, visual e inercial; además, se describen los conceptos matemáticos utilizados para la obtención de información a partir de los datos proporcionados por los mismos.

2.1. Revisión bibliográfica

En [24] se mencionan algunos de los trabajos más antiguos de odometría visual con sistemas computacionales que datan de los años 80, además de la principal diferencia entre sistemas de odometría visual: sistemas estéreo y sistemas monoculares.

La principal diferencia entre los sistemas de odometría estéreo y monoculares es el problema del factor de escala [13]. La escala absoluta puede ser determinada por mediciones directas (p. ej. midiendo algún elemento de la escena), restricciones de movimiento de la cámara o mediante la integración de otro tipo de sensores, como en el caso de las UMI, sensores de presión, GPS o Lidar [3].

El creciente interés en sistemas monoculares es debido a la observación de que, el modelo estéreo puede degenerarse al caso monocular cuando la distancia entre la escena y el objeto es mucho mayor que la distancia entre las dos cámaras montadas en el objeto mismo [24].

Como se menciona anteriormente, en 2004, Nister et. al. proponen un sistema [20] que es capaz de estimar el desplazamiento de un vehículo a partir de fotografías tomadas con cámaras montadas en el mismo, como ilustra la Figura 2.1. Su prototipo funciona tanto para el caso de visión estéreo, como para la visión monocular.

Figura 2.1: Fotografía del vehículo autónomo desarrollado en [20]. Pueden apreciarse las cámaras montadas en la parte frontal.

Cabe destacar el prototipo híbrido que propone Grimm también 2004 [12]. Montando ambos sensores en un bolígrafo y restringiendo el movimiento a una superficie plana, el sistema es capaz de estimar la pose de la cámara con respecto al plano. Además, plantea interesantes aplicaciones como la reconstrucción de imágenes en el plano, idea materializada en la actualidad en la industria por LG con su ratón-escáner LSM-100 [16].

Trabajos realizados por Li et al. en [17] y [18] implementan un filtro Kalman para la fusión de señales visuales e inerciales. Además, en este último consideran la distorsión producida por cámaras de persiana *(rolling shutter)* comúnmente utilizadas en dispositivos de bajo costo y proponen un modelo que corrige la distorsión de la imagen según las mediciones de la UMI.

Tiene lugar mencionar el sistema de captura de movimiento recientemente desarrollado por Neunert et al. [19] en 2016, el cual utiliza varios marcadores fiduciarios dentro de la escena para llevar a cabo la localización y mapeo simultaneo de un dispositivo que lleva acoplado ambos sensores. Los resultados presentados en este trabajo demuestran un sistema resistente a movimientos repentinos y condiciones de iluminación cambiantes.

Como menciona Scaramuzza en [25], la detección y empate de características naturales detectadas en la escena lleva gran parte del tiempo de procesamiento en los sistemas de odometría visual. Anteriormente la mayoría de este tipo de sistemas se ejecutaban fuera de línea, aunque actualmente, existen algoritmos optimizados para detectar y empatar estas características, haciendo posible la ejecución en tiempo real de este tipo de sistemas.

A la par de los sistemas limitados a estos dos tipos de sensores únicamente, como se menciona en [13] y [24], existen sistemas de odometría más sofisticados que cuentan con sensores complementarios como barómetros, altímetros y sensores Lidar que proporcionan en un mayor grado de precisión [3].

2.2. Información visual

Para recuperar la pose de un objeto en un espacio tridimensional a partir de una serie de imágenes captadas por una cámara, es imprescindible describir el proceso proyectivo que se lleva a cabo al capturar una fotografía.

En esta sección, se presentan los conceptos matemáticos que describen la geometría de la cámara a través del modelo de la cámara obscura y la técnica de calibración utilizada en este trabajo.

2.2.1. Modelo de la cámara obscura

Este modelo provee una descripción simple sobre cómo una cámara retrata una escena tridimensional. Formalmente, el modelo de la cámara obscura describe la relación entre un punto tridimensional $\mathbf{P} \in \mathbb{R}^3$ en la escena y su proyección $\mathbf{p} \in \mathbb{R}^2$ en el plano bidimensional de la imagen. Esta relación se representa con la Figura 2.2.

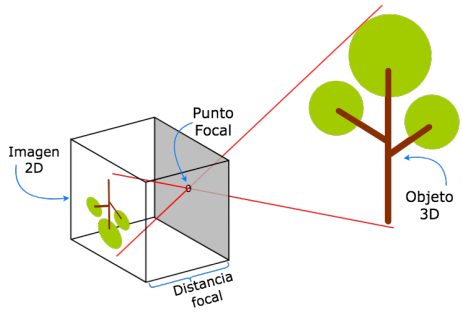


Figura 2.2: Funcionamiento de una cámara obscura.

A continuación se examina el modelo *ideal* de la cámara obscura, que no considera distorsión por el uso de lentes. Se considera la proyección de puntos en el espacio hacia un centro de proyección pasando por un plano. Sea $\mathbf{C}$ el centro de proyección situado en el origen del sistema de coordenadas Euclidiano y f la *distancia focal*; se le conoce como el *plano de proyección* al plano donde $z = f$.

Este modelo propone un punto $\mathbf{P} = [x, y, z]^T$ con coordenadas en el espacio tridimensional, que se proyecta a un punto $\mathbf{p} = [u, v]^T$ situado en el plano de proyección; éste último se localiza donde la línea que une al punto $\mathbf{P}$ con el centro de proyección se encuentra con el plano de proyección. Esto se ilustra con la Figura 2.3.

De la Figura 2.3, se puede deducir por triángulos semejantes lo siguiente:

$$\begin{aligned} \frac{u}{-f} &= \frac{x}{z}, \\ u &= \frac{-fx}{z}. \end{aligned} \tag{2.1}$$

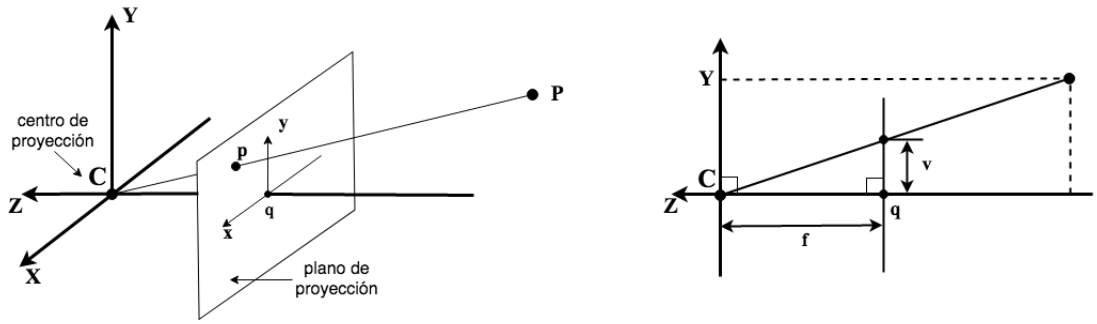


Figura 2.3: Geometría de la cámara obscura. **C** es el centro de proyección y el punto **q** el punto principal.

de manera análoga:

$$\begin{aligned} \frac{v}{-f} &= \frac{y}{z}, \\ v &= \frac{-fy}{z}. \end{aligned} \tag{2.2}$$

Aquí, ambas expresiones (2.1) y (2.2) cumplen con dos restricciones; la primera restringe la posición del plano de proyección, que debe estar adelante de la cámara, por lo tanto $f > 0$. La segunda, $z < 0$, indica que los puntos tridimensionales se encuentran, de igual manera, delante de la cámara.

Proyección usando coordenadas homogéneas

Si se representan los puntos **P** del mundo y los puntos **p** de la imagen como vectores homogéneos, podemos expresar entonces el modelo de la cámara obscura como una transformación lineal. Escrito de forma matricial:

$$\lambda \begin{bmatrix} u \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.3}$$

Aquí, λ es un factor de escala, y se consideran los puntos homogéneos $\mathbf{P} = [x, y, z, 1]^T$ y $\mathbf{p} = [u, v, 1]^T$; la relación existente entre estos puntos es la *matriz de proyección de cámara* M de tamaño 3×4. Entonces podemos escribir el modelo de la cámara obscura (2.3) de manera compacta como

$$\lambda \mathbf{p} = M\mathbf{P}. \tag{2.4}$$

Debido al carácter proyectivo del modelo de la cámara obscura, puede deducirse que se tiene una infinidad de puntos en la escena proporcionales a un punto **p** en la

imagen. Esta relación puede expresarse con un factor de escala λ diferente de cero.

$$\mathbf{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \ \forall \lambda \neq 0 \tag{2.5}$$

2.2.2. Parámetros intrínsecos de la cámara

La expresión (2.3) asume que las coordenadas del punto principal $\mathbf{q}$ son el origen de coordenadas del plano de proyección. Si el punto principal se representa como (u_0, v_0), generalizando el modelo se obtiene:

$$K = \begin{bmatrix} -f & 0 & u_0 \\ 0 & -f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

y la ecuación (2.4) puede reescribirse como:

$$\lambda \mathbf{p} = K[I|0]\mathbf{P} \tag{2.7}$$

donde la matriz K es conocida como *matriz de parámetros intrínsecos* de la cámara, o bien, como *matriz de calibración* de la cámara. La matriz $[I|0]$ representa una matriz identidad de tamaño 3×3 aumentada con un vector columna de ceros.

El sistema de coordenadas bajo el cual se definen a los puntos $\mathbf{P}$ se define como el *sistema de coordenadas de la cámara*.

2.2.3. Parámetros extrínsecos de la cámara

En general, los puntos en el espacio son expresados con respecto a un marco de referencia distinto al de la cámara, conocido como *sistema de coordenadas del mundo*. Ambos sistemas de coordenadas están relacionados mediante una transformación euclidiana, i.e., una rotación y una traslación. Reescribiendo entonces la ecuación de la cámara obscura (2.4) incluyendo estas transformaciones:

$$\lambda \mathbf{p} = K[R|\mathbf{t}]\mathbf{P} \tag{2.8}$$

Escribiendo la expresión anterior de manera explícita:

$$\lambda \begin{bmatrix} u \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & u_0 \\ 0 & -f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{array} \right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.9}$$

donde R es una matriz de rotación de tamaño 3×3 y $\mathbf{t}$ es un vector de traslación de tamaño 3. A la transformación que expresa la matriz aumentada $[R|\mathbf{t}]$ se le conoce como la *matriz de parámetros extrínsecos* de la cámara. Aquí, el punto $\mathbf{P}$ es primero

rotado según R y posteriormente trasladado con respecto al vector $\mathbf{t}$. Finalmente, el punto es proyectado al plano según la matriz K.

Alternativamente, si se supone que la cámara ve hacia el origen del centro de coordenadas $\mathbf{o}$, puede reescribirse la ecuación (2.8) como:

$$\begin{aligned} \lambda\mathbf{p} &= KR[I|R^T\mathbf{t}]\mathbf{P}, \\ \lambda\mathbf{p} &= KR[I|-\mathbf{c}]\mathbf{P}, \end{aligned} \tag{2.10}$$

donde $\mathbf{c}$ es el vector que representa la posición de la cámara.

De (2.10), podemos deducir lo siguiente:

$$\begin{aligned} -\mathbf{c} &= R^T\mathbf{t} \\ \mathbf{c} &= -R^T\mathbf{t} \\ \mathbf{t} &= -R\mathbf{c} \end{aligned} \tag{2.11}$$

Entonces, la matriz R en (2.8) puede calcularse a partir de la posición de la cámara $\mathbf{c}$, considerando que la cámara ve hacia el origen del sistema de coordenadas y un vector $\mathbf{a}$ que indica la dirección de arriba con respecto a la cámara.

$$\begin{aligned} \mathbf{z}_c &= \frac{\mathbf{c}}{||\mathbf{c}||} \\ \mathbf{x}_c &= \frac{\mathbf{a} \times \mathbf{z}_c}{||\mathbf{a} \times \mathbf{z}_c||} \\ \mathbf{y}_c &= \mathbf{z}_c \times \mathbf{x}_c \end{aligned} \tag{2.12}$$

$$R = \begin{bmatrix} \mathbf{x}_c^T \\ \mathbf{y}_c^T \\ \mathbf{z}_c^T \end{bmatrix} \tag{2.13}$$

Puede concluirse que la matriz M es una combinación de las matrices de parámetros intrínsecos y extrínsecos, K y $[R|\mathbf{t}]$. Es trivial notar que el modelo general de la cámara obscura tiene 9 grados de libertad: 3 para K (f,u_0,v_0), 3 para R y 3 para $\mathbf{c}$.

2.2.4. Calibración de la cámara

La calibración de una cámara consiste en calcular la matriz de proyección $K[R|\mathbf{t}]$ a partir de una correspondencia de puntos entre un modelo en la escena y diversas fotografías capturadas por la cámara misma. Existen diversas técnicas de calibración que utilizan objetivos prediseñados o con algún patrón conocido para llevar a cabo la estimación de dichos parámetros. Una de ellas es la calibración mediante una homografía.

Homografía

La homografía es una transformación lineal que determina la correspondencia de un conjunto de puntos entre dos planos. Se representa con una matriz H de tamaño 3×3 y debido a que se suponen planos, la homografía es independiente de la escala, por lo tanto tiene 8 grados de libertad.

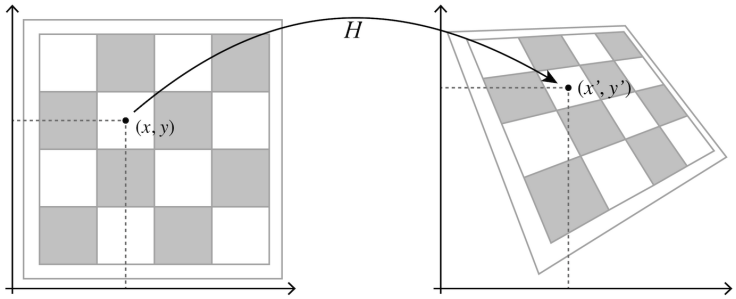


Figura 2.4: Homografía H que relaciona puntos entre dos planos.

Considerando pues, que los puntos de la escena se encuentran en el plano xy podemos asumir que $z = 0$. Entonces, la ecuación de la cámara obscura (2.4) puede escribirse como:

$$\lambda \mathbf{p} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \tag{2.14}$$

o bien,

$$\lambda \mathbf{p} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_4 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.15}$$

Finalmente, tenemos que:

$$\lambda \mathbf{p} = H\mathbf{P}, \tag{2.16}$$

y la ecuación equivalente:

$$\lambda H = K[\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}], \tag{2.17}$$

donde H es una homografía.

Cálculo de la homografía

Para obtener la homografía es necesario tener el empate correspondiente a ambos conjuntos de puntos $\mathbf{p}_i$ y $\mathbf{P}_i$ donde $i = 1, \ldots, n$ para n correspondencias diferentes de puntos.

A partir de la ecuación (2.16) obtenemos:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.18}$$

y debido al factor de escala desconocido λ, podemos fijar el valor de $h_{33} = 1$.

Tenemos entonces:

$$\begin{aligned} \lambda u &= h_{11}x + h_{12}y + h_{13} \\ \lambda v &= h_{21}x + h_{22}y + h_{23} \\ \lambda &= h_{31}x + h_{32}y + 1 \end{aligned} \tag{2.19}$$

Sustituyendo λ en las ecuaciones anteriores:

$$\begin{aligned} u &= h_{11}x + h_{12}y + h_{13} - h_{31}ux - h_{32}uy, \\ v &= h_{21}x + h_{22}y + h_{23} - h_{31}vx - h_{32}vy. \end{aligned} \tag{2.20}$$

Observamos que por cada correspondencia de puntos ($\mathbf{p}_i$,$\mathbf{P}_i$) es posible generar dos ecuaciones. Así que para calcular la homografía es posible construir un vector de incógnitas $\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}]^T$ y resolver el sistema de ecuaciones,

$$A\mathbf{h} = \mathbf{b}. \tag{2.21}$$

La matriz A y el vector $\mathbf{b}$ se construyen a partir de las correspondencias de puntos como en (2.20). De manera explícita:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ & \vdots & & & \vdots & & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nu_n & -y_nu_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nv_n & -y_nv_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix} \tag{2.22}$$

Resulta entonces, un sistema de ecuaciones de tamaño $2n \times 8$ para n correspondencias de puntos. Observamos que el sistema tiene 8 incógnitas, por lo tanto, si por cada correspondencia de puntos se establecen dos ecuaciones, entonces necesitamos al menos 4 correspondencias de puntos.

Normalización de puntos

Para evitar inestabilidades numéricas es necesario transformar los puntos $\mathbf{p}$ y $\mathbf{P}$ a un conjunto de puntos $\mathbf{p}'$ y $\mathbf{P}'$, con media cero y varianza igual a uno.

Esto se logra mediante una trasformación del la forma:

$$\begin{aligned} \mathbf{p}' &= T_1\mathbf{p}, \\ \mathbf{P}' &= T_2\mathbf{P} \end{aligned} \tag{2.23}$$

Donde T_1 y T_2 son matrices de la forma:

$$T_1 = \begin{bmatrix} \frac{1}{\sigma_u} & 0 & -\frac{\bar{u}}{\sigma_u} \\ 0 & \frac{1}{\sigma_v} & -\frac{\bar{v}}{\sigma_v} \\ 0 & 0 & 1 \end{bmatrix}, \qquad T_2 = \begin{bmatrix} \frac{1}{\sigma_x} & 0 & -\frac{\bar{x}}{\sigma_x} \\ 0 & \frac{1}{\sigma_y} & -\frac{\bar{y}}{\sigma_y} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

y donde σ_u, σ_v, σ_x, σ_y son los valores de desviación estándar, mientras que $\bar{u}$, $\bar{v}$, $\bar{x}$, $\bar{y}$ son las medias de los n puntos.

Entonces, la homografía para los puntos trasformados es:

$$\lambda\mathbf{p}' = H'\mathbf{P}' \tag{2.25}$$

sustituyendo los puntos transformados de (2.23) en la ecuación anterior:

$$\begin{aligned} \lambda T_1\mathbf{p} &= H'T_2\mathbf{P} \\ \lambda\mathbf{p} &= T_1^{-1}H'T_2\mathbf{P} \end{aligned} \tag{2.26}$$

Finalmente, la homografía se calcula como:

$$H = T_1^{-1}H'T_2 \tag{2.27}$$

Solución del sistema de ecuaciones

Una forma eficiente de resolver $\mathbf{h}$ para la ecuación (2.21) es por medio de una descomposición QR.

Se propone entonces que:

$$A = QR \tag{2.28}$$

donde Q es una matriz ortogonal, y R es una matriz triangular superior.

Sustituyendo A en la ecuación obtenemos:

$$\begin{aligned} A\mathbf{h} &= \mathbf{b} \\ (QR)\mathbf{h} &= \mathbf{b} \\ (QR)^T(QR)\mathbf{h} &= (QR)^T\mathbf{b} \\ R^TQ^TQR\mathbf{h} &= R^TQ^T\mathbf{b} \\ R\mathbf{h} &= Q^T\mathbf{b} \end{aligned} \tag{2.29}$$

Si consideramos la matriz aumentada B,

$$B = [A|\mathbf{b}] \tag{2.30}$$

y calculamos la descomposición QR:

$$B = [QR|\mathbf{b}] = Q[R|Q^T\mathbf{b}] \tag{2.31}$$

Para la obtención de la homografía, en lugar de calcular $Q^T\mathbf{b}$, podemos construir la matriz aumentada $[A|\mathbf{b}]$ y entonces calcular la descomposición QR de esta matriz. Obtendremos una matriz diagonal, que contiene los valores de $Q^T\mathbf{b}$ en la última columna.

El sistema de ecuaciones resultante $R\mathbf{h} = \mathbf{c}$ podría escribirse de forma matricial como:

$$\begin{bmatrix} r_{11} & \dots & r_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{nn} \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \tag{2.32}$$

Para resolver el sistema, basta con realizar la retrosustitución de las variables h_i, como se muestra en el Algoritmo 1, para encontrar los valores incógnitas para el vector $\mathbf{h}$.

Algoritmo 1 Solución de un sistema de ecuaciones $R\mathbf{h} = \mathbf{c}$ por retrosustitución

Entrada: Matriz triangular superior $R \in \mathbb{R}^{n\times n}$, vector $\mathbf{c} \in \mathbb{R}^n$
Salida: Vector solución $\mathbf{h} \in \mathbb{R}^n$.

```
h_n = c_n / r_nn
i = n − 1
while i > 0 do
    sum = 0
    j = i + 1
    while j ≤ n do
        sum = sum + r_ij × h_j
        j = j + 1
    end while
    h_i = (c_i − sum) / r_ii
    i = i − 1
end while
```

Calibración mediante una homografía

La calibración de la cámara a través de una homografía se logra a partir de las restricciones propuestas por Zhang [30], para el cálculo de los parámetros de la *matriz de proyección de cámara* M.

Retomando la ecuación (2.17), podemos escribir el modelo de la cámara obscura como:

$$\begin{aligned} \lambda H &= K[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \\ \lambda[\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3] &= K[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \end{aligned} \tag{2.33}$$

y las dos restricciones para la matriz R son:

$$\begin{aligned} \mathbf{r}_1^T\mathbf{r}_2 &= 0, \\ |\mathbf{r}_1| = |\mathbf{r}_2| &= 1. \end{aligned} \tag{2.34}$$

los vectores $\mathbf{r}_1$ y $\mathbf{r}_2$ son ortogonales, y deben tener la misma magnitud unitaria.

Entonces, de la ecuación (2.33), tenemos que:

$$\lambda K^{-1}H = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \tag{2.35}$$

Como ya podemos calcular K^{-1}, entonces resolvemos el lado izquierdo de la ecuación:

$$B = K^{-1}H = [\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3] \tag{2.36}$$

Calculamos el factor de escala lambda como:

$$\begin{aligned} \lambda_1 = |\mathbf{b}_1|\ ,\ \lambda_2 = |\mathbf{b}_2| \\ \lambda = \frac{(\lambda_1 + \lambda_2)}{2} \end{aligned} \tag{2.37}$$

y obtenemos entonces:

$$[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] = \frac{1}{\lambda}B \tag{2.38}$$

El tercer vector columna de R' se calcula con el producto cruz de los dos ya encontrados:

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{2.39}$$

Esta matriz $R' = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ no es ortogonal; podemos ortogonalizarla por medio de la descomposición SVD, tal que:

$$R' = UDV^T \tag{2.40}$$

donde U y V^T son matrices cuadradas ortogonales y D es una matriz diagonal.

Finalmente, la matriz de rotación se obtiene a partir de:

$$R = UV^T. \tag{2.41}$$

2.3. Información inercial

En esta sección se detallan los tipos de sensores inerciales utilizados para el desarrollo de esta tesis, así como los principios físicos bajo los que operan. Al final, se describe a grandes rasgos cómo puede usarse la información proporcionada por este tipo de sensores para aplicaciones de odometría puramente inercial.

2.3.1. Sensores inerciales

Los acelerómetros y giroscopios se conocen como sensores inerciales debido a que aprovechan la propiedad de inercia, es decir, la resistencia a un cambio de momento; para percibir movimiento angular en el caso del giroscopio, y cambios en el movimiento lineal para el acelerómetro [3]. Estos sensores no dependen de referencias externas o infraestructura adicional alguna, a excepción del campo gravitacional ubicuo.

Diversos tipos de sensores inerciales generalmente se acoplan en una misma UMI, configurándolos de manera ortogonal para obtener lecturas en varias dimensiones. Así, se pueden obtener valores de aceleración lineal con un acelerómetro, y velocidad angular para el giroscopio.

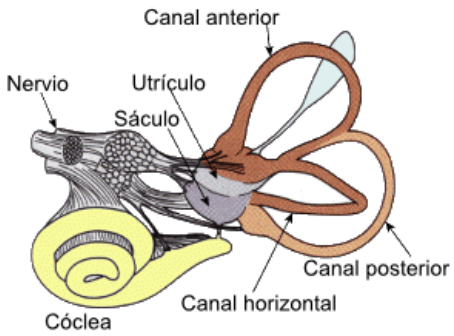


Figura 2.5: Aparato vestibular humano. Los tres canales semicirculares se encuentran dispuestos en tres planos ortogonales [3].

El cuerpo humano cuenta con un sistema sensorial inercial similar, llamado *sistema vestibular*, que está relacionado con el equilibrio y el control espacial. Este aparato está formado por dos órganos: el utrículo y el sáculo, que miden fuerzas gravitacionales e inerciales, proveyendo información a cerca de la inclinación y movimiento de la cabeza. Como se observa en la Figura 2.5, los canales semicirculares están también dispuestos en tres planos ortogonales, útiles para detectar la velocidad angular de la cabeza en el espacio.

Aunque inicialmente los sensores inerciales se fabricaban para fines aeroespaciales y navales, en las últimas dos décadas, la industria automotriz, médica y de entretenimiento ha impulsado el desarrollo de nuevas tecnologías para la producción en masa de sensores inerciales electrónicos y de silicio; cada vez más baratos y pequeños. En la Tabla 2.1 se clasifican los diferentes tipos de UMI disponibles en el mercado.

Sensores MEMS

Es así como surgen los *sensores microelectromecánicos* (MEMS por sus siglas en inglés), que se refieren a dispositivos microscópicos con partes móviles que usan principios electromecánicos para su funcionamiento. Los tamaños, materiales y técnicas de fabricación varían según el sector comercial al que esté orientada la producción de este tipo de sensores inerciales.

Tabla 2.1: Clasificación y comparativa de los diferentes tipos de UMI en el mercado [1].

Clasificación de la UMI	Navegación	Táctico	Industrial	Consumidor
Precio (USD)	>50k	10-20k	0.5-3k	<500
Peso	>2kg	∼0.5kg	0.1-0.2kg	>.010kg
Sesgo del Giroscopio	<0.1°/h	0.1-10°/h	≤1°/s	>1°/s
Sesgo del Acelerómetro	5-10 μg	0.02-0.04mg		
Aplicaciones	Aeronaves y naves espaciales	Militar	Automotriz estabilización y aviación gral	Medicas, entretenimiento, cámaras

La magnitud del ruido y sesgo presente en las mediciones de este tipo de sensores depende directamente de la calidad del sensor utilizado, y por ende, del precio del mismo. Aunque los sensores MEMS se encuentran en el rango de fidelidad más bajo, los precios son bastante asequibles y las mediciones que pueden realizarse con los mismos se consideran lo suficientemente fiables en movimientos repentinos y por intervalos cortos de tiempo.

2.3.2. Acelerómetros

Un acelerómetro es un instrumento que se utiliza para mediar aceleraciones bajo su propio marco de referencia. El principio físico bajo el que opera un acelerómetro se ilustra de manera sencilla en la Figura 2.6.

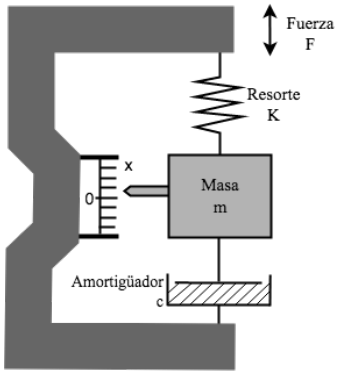


Figura 2.6: Diagrama de un acelerómetro mecánico de un solo eje.

Por la segunda ley de Newton, tenemos que:

$$\mathbf{F} = m\mathbf{a} \tag{2.42}$$

o bien,

$$\mathbf{a} = \mathbf{F}/m \tag{2.43}$$

donde sabemos que si la masa m es constante, se puede establecer una relación entre la fuerza medida $\mathbf{F}$ y la aceleración resultante $\mathbf{a}$. Entonces, un acelerómetro no necesariamente expresa el cambio de velocidad con respecto al tiempo, sino que en realidad mide el peso experimentado por una masa de prueba m contenida dentro del dispositivo.

Existen tres tipos principales de acelerómetros: mecánicos, capacitivos y piezoeléctricos; todos estos se basan en el mismo principio, variando las técnicas y materiales utilizados para la fabricación del instrumento.

De acuerdo con el principio de equivalencia de la relatividad general, los efectos de la gravedad y la aceleración son los mismos, por lo tanto, no podemos determinar si el sensor está sujeto a una componente del vector de gravedad, o si se encuentra acelerando [3]. Se necesitan de otros sensores o suposiciones extra para resolver esta ambigüedad.

2.3.3. Giroscópios

Un giroscopio mecánico se conforma por una rueda giratoria montada sobre dos cardanes que le permiten rotar en cada uno de los tres ejes, como ilustra la Figura 2.7

Figura 2.7: Giroscopio mecánico del siglo XIX, del museo CNAM en Francia.

Este instrumento fue inventado en 1852 por León Foucault. Debido al principio de la conservación angular del momento, dicha rueda presentará una resistencia a los cambios en orientación cuando el giroscopio es sometido a rotación. De modo que la rueda permanecerá en una orientación global constante, mientras que los ángulos entre cardanes adyacentes cambiarán. Este cambio puede ser entonces medido mediante mecanismos específicos.

El efector Coriolis establece que, en un marco de referencia que se encuentra rotando a una velocidad angular $\boldsymbol{\omega}$, una partícula de masa m moviéndose con velocidad $\mathbf{v}$ experimentará una fuerza proporcional $\mathbf{F}$, tal que:

$$\mathbf{F} = 2m(\boldsymbol{\omega} \times \mathbf{v}) \tag{2.44}$$

Para este dispositivo, entonces, la fuerza neta se debe a todas las partículas que comprenden el disco giratorio, que produce un momento $\mathbf{M}$ debido a la aceleración de Coriolis:

$$\mathbf{M} = I\boldsymbol{\omega} \times \mathbf{p} \tag{2.45}$$

donde I es el momento de inercia del disco del giroscopio, $\mathbf{p}$ es la velocidad angular del disco y $\boldsymbol{\omega}$ es la velocidad angular del giroscopio. Entonces, cuando un giroscopio se somete a una rotación, ejerce un momento opuesto proporcional a la velocidad angular; dicho momento es el fenómeno medido en este caso.

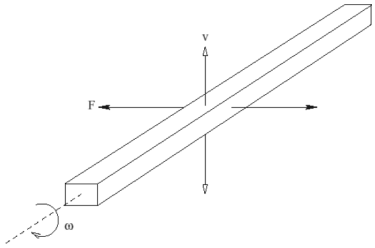


Figura 2.8: Diagrama de un giroscopio de estructura vibratoria [3].

Por su parte, los giroscopios de estructura vibratoria (VSG por sus siglas en inglés), no cuentan con un disco giratorio, sino que funcionan produciendo un movimiento oscilatorio, para medir el efecto Coriolis inducido por la rotación. La Figura 2.8 representa un VSG donde una viga flexible se hace vibrar en el plano vertical. La rotación sobre el eje de la viga induce una fuerza Coriolis que desplaza la viga hacia los lados; siendo este el desplazamiento medido. Los VSG con tecnología MEMS integran varios giroscopios en un mismo chip, y actualmente puede adquirirse en dentro de un amplio rango de precios y precisión.

2.3.4. Navegación inercial

Como se menciona en la parte introductoria de esta tesis, los SNI son dispositivos autocontenidos para posicionamiento y orientación de objetos, por lo cual no necesitan de referencias externas para la estimación y rastreo de los objetos o vehículos sobre los que están montados. Los SNI cuentan con una UMI y una computadora dedicada a la obtención y procesamiento de datos obtenidos de los sensores. De manera general, estos sistemas se dedican a integrar numéricamente las señales discretas obtenidas de los sensores inerciales de la UMI.

Así pues, si se tienen mediciones de aceleración lineal $\mathbf{a}$, para tres ejes ortogonales, puede obtenerse la velocidad lineal del objeto $\dot{\mathbf{p}}$ mediante la integración de los datos

con respecto al tiempo; análogamente la posición $\mathbf{p}$ se integra con respecto al cambio de velocidad:

$$\mathbf{p} = \int \dot{\mathbf{p}} dt = \int \int \ddot{\mathbf{p}} dt = \int \int \mathbf{a} dt \tag{2.46}$$

donde $\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}$ son vectores de aceleración velocidad y posición respectivamente.

De igual manera, con las lecturas de velocidad angular ω, es posible calcular la posición angular θ del objeto para cada instante; sin embargo, esta integración depende directamente del modelo de representación de orientación elegido para el sistema. El proceso que lleva a cabo un SNI se ilustra mediante la Figura 2.9.

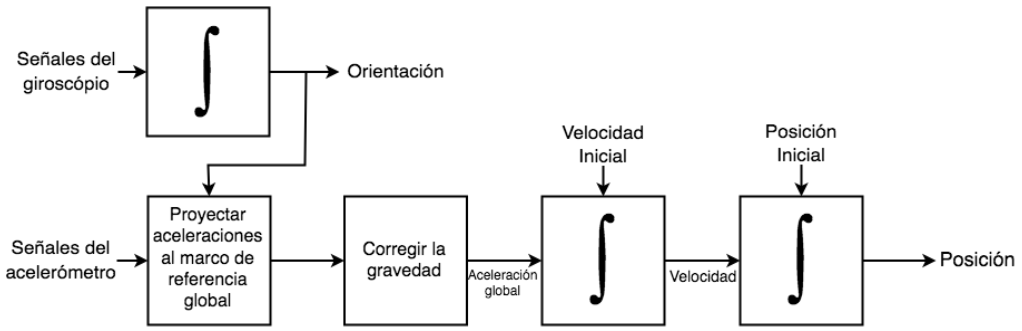


Figura 2.9: Diagrama de un SNI simplificado [3].

Los datos proporcionados por la UMI se miden a partir de su propio marco de referencia, conocido como *marco de referencia no inercial*. Inicialmente, este se encuentra alineado o en alguna pose conocida con un respecto a un marco de referencia en el mundo, llamado *marco de referencia de navegación* o *marco de referencia inercial*. Usando las transformaciones cinemáticas apropiadas, es posible expresar las vectores medidos por la UMI dentro de este marco de referencia global.

Entonces, usando la orientación actual del objeto, es posible separar la componente gravitacional medida por el acelerómetro, para obtener la *aceleración aparente* que experimenta el objeto en cada instante; siendo esta una de las suposiciones mencionadas anteriormente para obtener datos congruentes a partir de un acelerómetro.

No obstante, todos los SNI sufren de errores por integración: el ruido presente en los sensores se integra como error acumulativo para la velocidad y orientación, que con el paso del tiempo se traduce en una posición estimada incorrecta. Por esta razón, este tipo de sistemas están obligados a corregir las estimaciones periódicamente mediante otro tipo de sistema de navegación. Ocasionalmente, las UMI cuentan con sensores auxiliares como altímetros, inclinómetros o magnetómetros que ayudan a corregir los errores acumulados por las mediciones inerciales. A pesar de esto, los SNI son utilizados como sistemas auxiliares que proveen mayor fidelidad a altas frecuencias, y son empleados en un sinfín de aplicaciones, desde aeroespaciales, automotrices, de exploración, médicas y de entretenimiento.

Capítulo 3

Diseño y construcción del sistema

En este capítulo se exponen cada uno de los componentes del sistema, sus especificaciones técnicas y las herramientas de software necesarias para la obtención y procesamiento de datos. Se ilustra el ensamble y montaje de cada uno de los elementos que integran el sistema. Todos los componentes elegidos son de bajo costo y con limitaciones de tamaño, consumo de energía y poder de procesamiento; estos criterios ofrecen un sistema barato, portátil y reducido en dimensiones.

3.1. Plataforma Raspberry Pi

Para este trabajo se propuso utilizar para el procesamiento de datos una computadora en una sola tarjeta (CST) Raspberry Pi. Las CST, como su nombre lo indica, son computadoras completas montadas en un solo circuito. El diseño se centra en integrar al microprocesador, memoria y dispositivos de entrada y salida en una sola placa que suele ser de tamaño y costo reducido. A diferencia de una computadora personal, las CST a menudo no son modulares, es decir, no cuentan con ranuras de expansión para diversas funciones y periféricos. Sin embargo, es cada vez más común encontrar CST con funciones de sonido, red, e incluso gráficos desde la placa base.

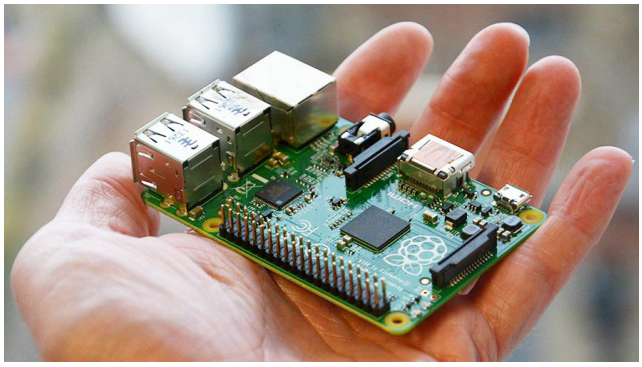


Figura 3.1: Comparativa de una computadora Raspberry Pi 3 modelo B con una mano humana.

Raspberry Pi es una CST de bajo costo y del tamaño de una tarjeta de crédito; desarrollada por la Raspberry Pi Foundation en el Reino Unido y orientada a ambientes académicos, actualmente es una de las plataformas CST más utilizadas por su asequibilidad, potencia y dimensiones.

Concretamente, la computadora elegida es una Raspberry Pi 3 modelo B [9], que tiene un costo aproximado de 35 USD. En la Tabla 3.1 se muestran las especificaciones técnicas de esta plataforma, y en la Figura 3.2 se ilustra la disposición de puertos y demás componentes de la tarjeta.

Tabla 3.1: Especificaciones técnicas de la Raspberry Pi model0 3.

Fecha de lanzamiento	29/Feb/2016
SoC	broadcom BCM2837
CPU	Quad Cortex 64bit @ 1.2GHz
Memoria	1 GB SDRAM
GPU	Broadcom VideoCore IV @ 400 MHz
Almacenamiento	MicroSD
Red	10/100 Mbps Ethernet,
	2.4GHz 802.11n WiFi
	Bluetooth 4 y Bluetooth LowEnergy
Puertos	40 pines GPIO
	4 puertos USB 2.0
	Puerto CSI (Cámara)
	Puerto DSI (Pantalla táctil)
	Puerto MicroSD
	Puerto 3.5mm A/V Análogo
Salida de video	HDMI
Alimentación	5V microUSB
Dimensiones	85.60mm $\times$ 56.5mm $\times$ 17mm

La conexión de la computadora con cada uno de los componentes del sistema propuesto (ver la Figura 1.5, en la página 6) se realizó, de manera general, de la siguiente forma: la UMI se conectó a cuatro puertos GPIO, dos para la transferencia de datos y dos para la alimentación de los sensores. De igual manera, el módulo de la cámara se acopló con un bus de datos al puerto CSI de la tarjeta. La interfaz de realidad aumentada se despliega mediante la salida de video obtenida del puerto HDMI. Se aprovechó la arquitectura de la computadora para ejecutar los sistemas visual e inercial en dos hilos independientes del procesador. Finalmente, ya con todos los periféricos conectados, se procede a alimentar la Raspberry con un eliminador de corriente conectado al suministro eléctrico casero de 115 V.

Los detalles referentes a la comunicación, montaje y configuración para cada uno de los componentes del sistema se exponen en las secciones subsiguientes.

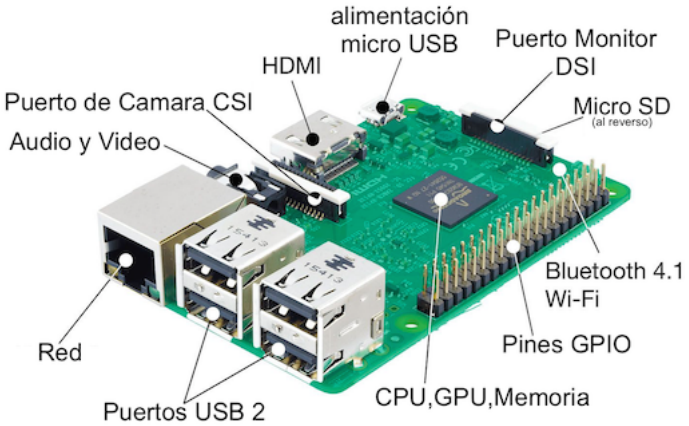


Figura 3.2: Configuración de los puertos y componentes de una Raspberry Pi 3 modelo B.

3.2. Cámara Raspberry

Para la adquisición de fotografías digitales se eligió el módulo de cámara Raspberry [10] en su segunda versión (Figura 3.3). Este módulo está soportado nativamente por la computadora y se conecta a través de puerto CSI de la Raspberry. El hardware especifica un sensor Sony IMX219 de 3280×2464 pixeles (px) para un total de 8 megapixeles, que además soporta video de alta definición. Las dimensiones del módulo publicadas por el fabricante son de 25×23×9mm con un peso neto de 3 gramos. Esta cámara digital tiene un costo aproximado de 25 USD.

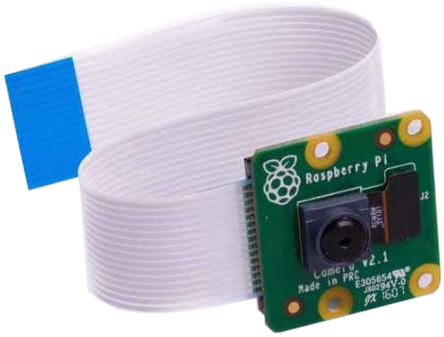


Figura 3.3: Módulo de cámara Raspberry Pi v2, con su respectivo bus de datos para el puerto CSI.

3.2.1. Obtención de la imagen

Para utilizar la cámara es necesario habilitar el puerto CSI de la Raspberry mediante la interfaz de configuración que despliega el comando:

```
sudo raspi-config
```

Una vez habilitada la comunicación, del módulo de la cámara se extraen las imágenes para su procesamiento digital en forma matricial. Para obtener las imágenes a alta frecuencia (30Hz), es necesario acceder al controlador de la cámara mediante código ejecutándose a bajo nivel. Por ello, se utilizó la biblioteca *RaspiCam* [29] de código abierto en C++, facilitando la comunicación con el módulo de la cámara.

Una vez extraídas, las imágenes se procesan utilizando Scimagen [5], como se detallará posteriormente en la Sección 3.3.1.

Dependiendo del controlador, y de la posición de la cámara en la escena, habrá que invertir la imagen para mostrarla en la interfaz visual.

El proceso de calibración fuera de línea para el módulo de cámara utilizado se describe en la Sección 4.1.1, en la página 40, de esta tesis.

3.3. Marcador fiduciario

Se le llama marcador de referencia, o fiduciario, al objeto que se coloca dentro del rango de visión de un sistema de imágenes; para usarlo como punto de referencia o medida conocida. Una regla graduada, entonces, sería un ejemplo sencillo de un objeto fiduciario cotidiano en fotografías. En décadas recientes, se ha popularizado el uso de este tipo de referencias para estimar la posición y orientación relativa entre una cámara y el marcador.

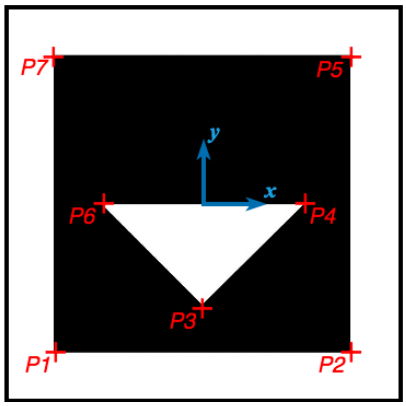


Figura 3.4: Modelo del marcador propuesto. Las anotaciones en color se añaden con fines ilustrativos.

Para este trabajo se propone un marcador fiduciario basado en tipo de orden [4, 14]. El tipo de orden es una función discreta que asigna una orientación a un conjunto de puntos en el plano. Con esto, es posible obtener un identificador único para cada conjunto de puntos que definen un marcador. Además, los fiduciarios generados bajo este esquema son invariantes a transformaciones proyectivas, de traslación y rotación. Aunado a estos factores, este tipo de marcadores pueden ser reproducidos con un impresora común en una hoja de papel, lo que los sitúa como una tecnología ampliamente asequible.

El fiduciario propuesto para esta tesis se puede observar en la Figura 3.4. Está formado por cuatro puntos que demarcan un cuadrado negro, que contrasta con el borde blanco que lo delimita; dentro del cuadrado, se encuentra un triángulo también en blanco, para un total de siete puntos identificables.

Para tener el sistema de coordenadas justo al centro del marcador, se usaron los siguientes siete puntos: (-50,-50), (50,-50), (0,-34), (34,0), (50,50), (-34,0) y (-50,50). El marcador fue impreso en papel con dimensiones de 100×100 mm para el cuadrado negro, posteriormente fue asegurado una superficie plana en el objeto, como se detalla en la Sección 3.5, en la página 35.

Mediante el procesamiento de imágenes obtenidas de la escena por la cámara, es posible detectar este marcador, que sirve para estimar la pose del objeto a partir de una homografía.

3.3.1. Detección del marcador en la imagen

Para el cálculo de la homografía es necesario detectar en la imagen los puntos que componen al marcador previamente diseñado. El marcador de tipo de orden puede ser fácilmente detectado en la escena suponiendo que es el objeto negro más grande en la imagen que se obtiene de la cámara.

Entonces, para detectar los puntos que corresponden al marcador basado en tipo de orden [4], se siguen los siguientes pasos:

1. La imagen obtenida de la cámara en color se convierte a escala de grises.

2. Se aplica una doble operación de suavizado en la imagen a escala de grises.

   El suavizado se realiza mediante un filtro espacial, tal que:

   Se tiene una máscara en forma de matriz cuadrada, $L \in \mathbb{N}^{n \times n}$ tal que n sea impar.

   Dada una imagen de entrada F, la imagen filtrada de salida G se calcula como:

   $$G_{i,j} = \frac{1}{w} \sum_{l=-1}^{1} \sum_{k=-1}^{1} L_{l,k} F_{i-l,j-k}$$

   donde w es la suma de los elementos de L.

   En el sistema se utilizó una máscara de unos de tamaño 3×3, tal que $L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

   Y ambas imágenes F y G son de tamaño 640×480.

3. Se binariza la imagen bajo un umbral global predefinido.

   El umbral define el valor mínimo para convertir bajo un criterio binario, los pixeles de la imagen en escala de grises [0,...,255] a una en blanco y negro [0,1].

4. Extracción de triángulos de la imagen.

   Se busca la componente negra *más grande* de la imagen. Esta componente debe tener cuatro esquinas y contener dentro al triángulo blanco del marcador.

5. Cálculo del perímetro para el triángulo.

   Se aplica una operación morfológica de dilatación al triángulo blanco encontrado dentro del cuadrado negro. El perímetro rodea la silueta del triángulo blanco con un borde de 1 px.

6. Obtención gruesa de las coordenadas de los vértices.

   Para detectar los tres vértices del triángulo, se busca dentro del conjunto de pixeles del perímetro.

   a) El primer pixel encontrado del perímetro es el primer vértice del tríangulo.
   b) El segundo vértice será entonces el pixel del perímetro mas alejado del primer vértice.
   c) El tercer vértice es el pixel mas alejado de la línea que definen los primeros dos vértices.

Debido al posible ruido de la imagen, es indispensable calcular los vértices para cada triángulo con precisión sub-pixel. Esto se logra de la siguiente manera:

1. Se ordenan los vértices de cada triángulo en sentido antihorario. Si los pixeles del perímetro ya se obtuvieron ordenados, se puede omitir este paso.

2. Se extraen los pixeles del perímetro que pertenecen a cada arista.

3. Se ajusta una recta para cada conjunto de puntos correspondiente a cada arista.

   Este problema se resuelve usando el algoritmo de componentes principales. Así es posible calcular el centroide y el vector de dirección para cada arista de cada triángulo [21].

   a) El centroide correspondiente a cada conjunto de puntos se calcula de manera incremental con una matriz de varianza V, inicializada como:

   $$V_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

   y para cada punto $\mathbf{p}$ del segmento que une dos vértices:

   $$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \ \forall i = 1, \ldots, n;$$
   $$V_{i+1} = V_i + \begin{bmatrix} x_i^2 & x_i y_i \\ x_i y_i & y_i^2 \end{bmatrix}.$$

b) El vector de dirección se obtiene calculando el mayor de los autovalores de V_n (hay dos) y su autovector asociado.

4. La nueva posición de cada vértice se recalcula como la intersección de las tres aristas para cada triángulo.

   a) Considere dos rectas $\mathbf{r}_1$ y $\mathbf{r}_2$ definidas a partir de puntos $\mathbf{p}_1$ y $\mathbf{p}_2$ y sus respectivas direcciones $\mathbf{u}$ y $\mathbf{v}$.

$$\begin{aligned}\mathbf{r}_1 &= \mathbf{p}_1 + t\mathbf{u},\\ \mathbf{r}_2 &= \mathbf{p}_2 + s\mathbf{v}.\end{aligned}$$

   b) Entonces ambas rectas se intersectan cuando:

$$\begin{aligned}\mathbf{r}_1 &= \mathbf{r}_2,\\ \mathbf{p}_1 + t\mathbf{u} &= \mathbf{p}_2 + s\mathbf{v}.\end{aligned}$$

   c) Reordenando la ecuación anterior,

$$t\mathbf{u} - s\mathbf{v} = \mathbf{p}_2 - \mathbf{p}_1.$$

   d) Escrito de forma matricial:

$$\begin{aligned}\mathbf{p} &= \begin{bmatrix} p_x \\ p_y \end{bmatrix},\\ \begin{bmatrix} u_x & -v_x \\ u_y & -v_y \end{bmatrix} \begin{bmatrix} t \\ s \end{bmatrix} &= \begin{bmatrix} p_x \\ p_y \end{bmatrix}\end{aligned}$$

   e) Entonces, t y s pueden ser encontradas con la regla de Cramer mediante una división de determinantes,

$$t = \frac{\begin{vmatrix} u_x & -v_x \\ u_y & -v_y \end{vmatrix}}{\begin{vmatrix} p_x & -v_x \\ p_y & -v_y \end{vmatrix}} = \frac{-u_x v_y + v_x u_y}{-p_x v_y + v_x p_y}$$

   f) Basta con sustituir cualquiera de estos valores de t o s en la ecuación de la recta correspondiente para encontrar el punto de intersección. Este punto es entonces el vértice correspondiente con precisión sub-pixel.

3.3.2. Correspondencia de puntos según el tipo de orden

Para llevar a cabo la calibración de una cámara es indispensable empatar los puntos del modelo en la escena (en esta caso el marcador) con los respectivos puntos en la imagen adquirida por cámara.

Mediante la matriz λ calculada por el tipo de orden [4], es posible etiquetar y empatar los puntos del modelo del marcador propuesto, con aquellos detectados en la imagen mediante el procesamiento previamente descrito. El tipo de orden es una medida combinacional de la orientación de los puntos. Esta medida no cambia si a los puntos, que se suponen sobre un plano, se les aplica un escalamiento, traslación, rotación o proyección en perspectiva.

Ambos conjuntos de puntos, del modelo del marcador y la imagen del mismo (Figura 3.6), están ordenados según su matriz λ mínima lexicográfica. Este marcador tiene cuatro puntos en su cubierta convexa; tomando como origen cada uno de estos puntos en la cubierta, se pueden ordenar de forma circular todos los puntos que componen al marcador. De las cuatro matrices λ que se pueden construir (una por cada punto en la cubierta convexa) se toma el orden de los puntos de su matriz mínima lexicográfica [4, 14]. En la Figura 3.5 se ilustran las cuatro matrices lambda posibles y encerrada en el recuadro punteado la matriz λ mínima lexicográfica.

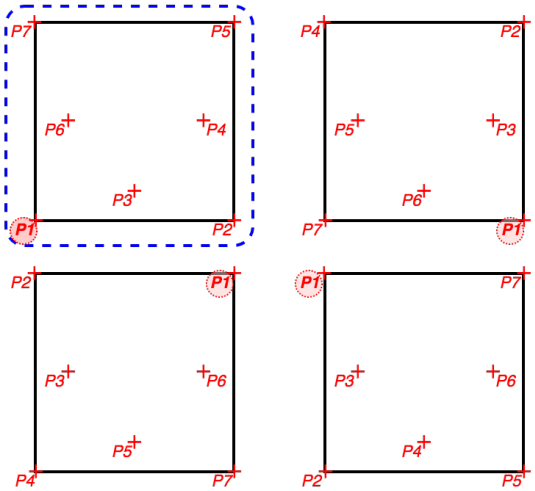


Figura 3.5: Matrices λ de la cubierta convexa del marcador. De las cuatro matrices posibles, la que encierra el recuadro punteado es la matriz λ mínima lexicográfica.

La matriz λ mínima lexicográfica nos indica el ordenamiento de los puntos invariante a rotación, traslación y escalamiento. De esta manera, se empatan los puntos del modelo y la escena, teniendo una relación uno a uno para cada uno de los siete puntos, independientemente de la pose del marcador en la escena que captura la cámara; siempre y cuando éste sea detectado.

Ambos ordenamientos coinciden, tal como se ilustra en la Figura 3.6. La relación obtenida por el tipo de orden se emplea para el cálculo de la homografía como se describe en la página 15, Sección 2.2.4.

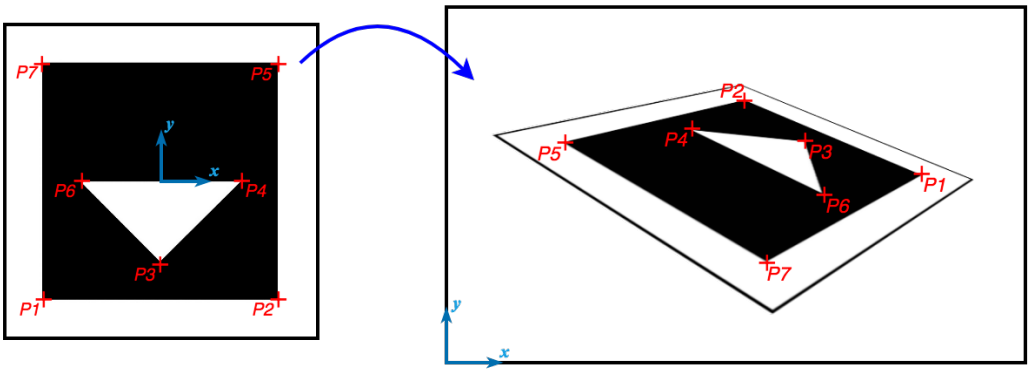


Figura 3.6: Etiquetado y correspondencia de puntos entre el modelo del marcador y el marcador detectado en la imagen.

El cálculo de los puntos en la imagen a nivel subpixel afecta no el orden sino el cómputo de la homografía, por lo que este procedimiento (anteriormente descrito) se considera crucial para la obtención de la pose.

El marcador propuesto ya se encuentra optimizado con la configuración de puntos descrita anteriormente, dado que resiste hasta 18 px de perturbación para el etiquetado correcto de dichos puntos.

3.4. Unidad de Medición Inercial

Para la obtención de datos inerciales se optó por usar la unidad de medición inercial de gama baja MPU-6050 [15] lanzada al mercado por Invensense en 2010. Esta UMI cuesta aproximadamente 3 USD e incluye un acelerómetro y un giroscopio, ambos de tres ejes; contenidos dentro de un mismo circuito integrado QFN con dimensiones de $4 \times 4 \times 0.9$ mm. Puede ser alimentada con tan sólo 3.3V o alternativamente 5V para su operación.

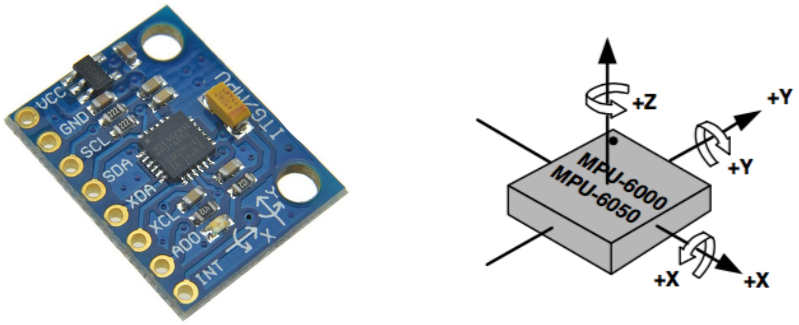


Figura 3.7: Unidad de medición inercial MPU-6050 de Invensense. A la izquierda, el sensor montado en una PCB para facilitar su conexión. A la derecha, la orientación de los ejes de sensibilidad y polaridad de rotación.

Los sensores pueden ser programados para actuar a diferentes rangos de sensibilidad; desde ±250, ±500, ±1000 y ±2000 °/s para el giroscopio, y rangos de ±2g, ±4g, ±8g y±16g para el acelerómetro, con un muestreo de 1 kHz para ambos. Además, el fabricante incluyó un termómetro dentro del chip, útil para medir la temperatura del sensor durante su operación. También, ésta UMI tiene la opción de añadir un magnetómetro como sensor extra mediante un puerto auxiliar.

Este instrumento se conecta a través de cuatro puertos GPIO de la computadora Raspberry, utilizando el protocolo I^2C para la comunicación.

Obtención de datos de la UMI

Para la transferencia de datos entre la UMI y la computadora, es necesario habilitar el protocolo I^2C de la Raspberry. Esto se realiza mediante la herramienta de configuración que puede ser desplegada con el comando de la terminal:

```
sudo raspi-config
```

Puede verificarse el puerto en el que fue activado el protocolo reiniciando la computadora y ejecutando el comando:

```
ls /dev/*i2c*
```

Para la comunicación con el sensor mediante el protocolo I^2C fue utilizada la biblioteca *I2Cdevlib* [23] de código abierto en lenguaje C++.

3.5. Conexión y ensamble del sistema

Es importante señalar que para que todos los componentes funcionen de manera adecuada, todas las conexiones de periféricos a la computadora deben realizarse antes de encenderla y preferentemente con la tarjeta desconectada del suministro eléctrico.

La conexión de la cámara a la Raspberry se realiza mediante un bus de datos de 15 vías que enlaza al módulo con la tarjeta, como se observa en la Figura 3.8. Cabe apuntar que el bus de datos debe ser instalado del lado correcto (note la etiqueta azul que lleva el bus), de lo contrario no funcionará.

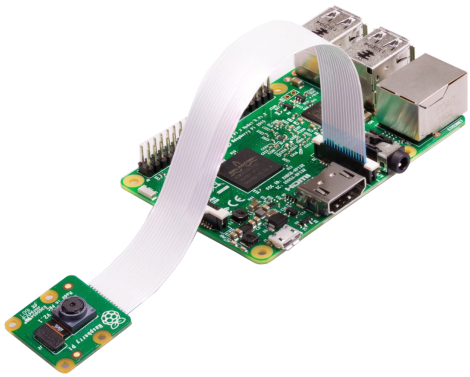


Figura 3.8: Módulo de cámara conectado a la Raspberry Pi.

Por su parte, la UMI utiliza dos puertos GPIO de la Raspberry para la alimentación y dos mas para la transmisión datos mediante el protocolo I^2C. Así las terminales SDA y SCL del sensor se conectan a los puertos GPIO 3 y 5 de la computadora para establecer una comunicación serial. En la Figura 3.9 puede observarse un diagrama del cableado entre la Raspberry y el sensor MPU-6050.

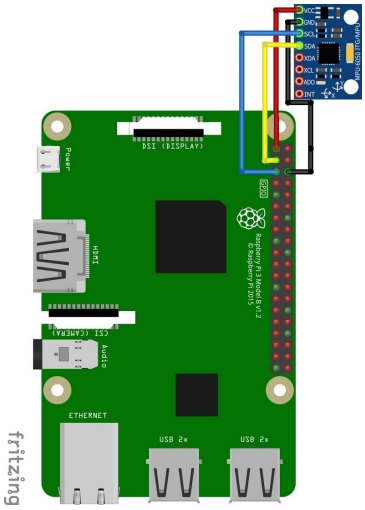


Figura 3.9: Diagrama de cableado entre la Raspberry y el sensor MPU-6050.

Como se especifica en la Figura 1.5 (pág. 6), el sistema debe rastrear al marcador que lleva fija la UMI de alguna manera. Se propone entonces la construcción de un objeto que acopla ambos elementos y relaciona sus sistemas de coordenadas. Si suponemos que el marcador se encuentra en el plano xy con $z = 0$ de su sistema de coordenadas tridimensional, entonces tanto el marcador como la UMI tienen sistemas de coordenadas iguales, ambos de mano derecha.

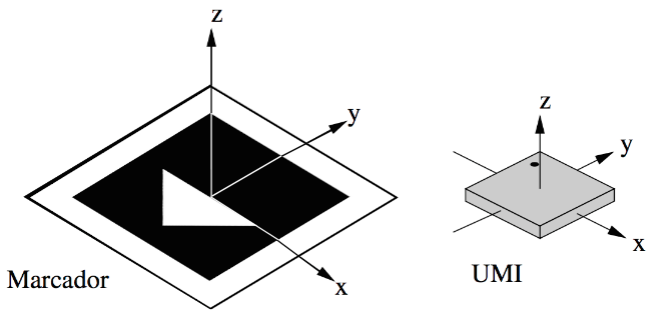
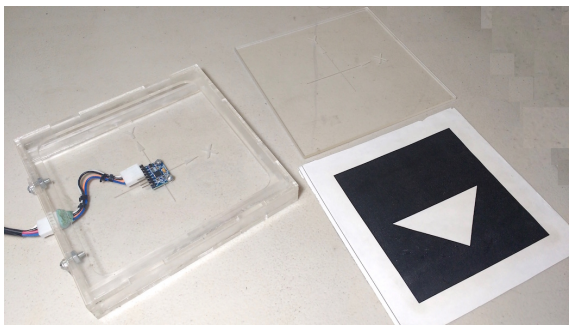
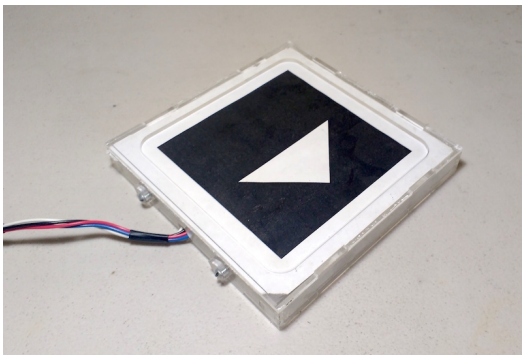


Figura 3.10: Sistemas de coordenadas del marcador y de la UMI. Ambos son iguales.

El objeto adapta ambos marcos de referencia en una estructura de acrílico construida a medida a partir de las dimensiones físicas de los sensores. Se diseñó como una caja cuadrangular de acrílico a la que se atornilla la UMI en el fondo y cuenta con un mecanismo para cambiar el marcador; esto es simplemente una lámina de acrílico que lleva sobrepuesta el marcador, y se desliza por un costado del objeto, para ser asegurada por dos aldabas fijas al lateral. Así, como se puede observar en la Figura 3.11, el marcador de papel intercambiable queda sujeto por la tapa superior del objeto, mientras que la tapa lateral tiene un orificio por el que se conectan los cables de la UMI.



(a) Desglose de los componentes del objeto construido.



(b) Vista del objeto ensamblado.

Figura 3.11: Ensamblaje del objeto con la UMI y el marcador. El marcador se desliza por la parte lateral para quedar sujeto al objeto, haciendo coincidir ambos sistemas de coordenadas.

De esta manera, los sistemas de coordenadas del marcador y la UMI quedan alineados con respecto al eje z, donde existe una traslación de 12 mm entre ambos, que para fines prácticos de esta tesis se considera despreciable.

3.6. Montaje del sistema

El sistema se programó en C/C++ usando la biblioteca de Qt [2] para la interfaz gráfica, mostrada en la Figura 3.12, usando una resolución de 640×480 para las imágenes obtenidas de la cámara. En la interfaz, se dibuja al centro la imagen con un borde gris que la delimita; ésta área permite dibujar sobrepuestos elementos fuera del rango de visión de la cámara. El panel del lado izquierdo contiene algunos botones para la interacción del usuario y además despliega información de la pose estimada por el sistema.

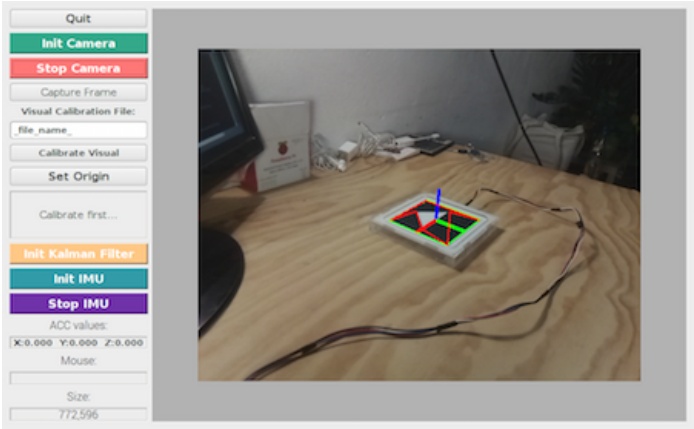


Figura 3.12: Interfaz desplegada por el sistema. En el lado derecho se muestra la imagen captada por la cámara con el ambiente virtual sobrepuesto, del lado izquierdo un panel de control e información del sistema.

Por conveniencia, el módulo de la cámara se montó en un trípode para ajustar su rango de visión con mayor facilidad. Finalmente, la computadora Raspberry se conecta a un monitor externo mediante el puerto HDMI para desplegar la interfaz de realidad aumentada del sistema.

La disposición de todos los componentes del sistema propuesto puede apreciarse en las Figuras 3.13, 3.14, 3.15.

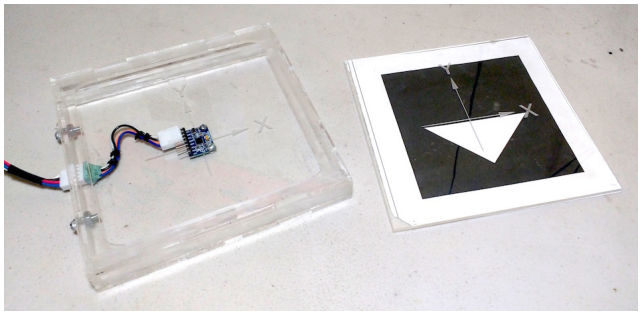


Figura 3.13: Objeto construido para acoplar la UMI con el marcador.

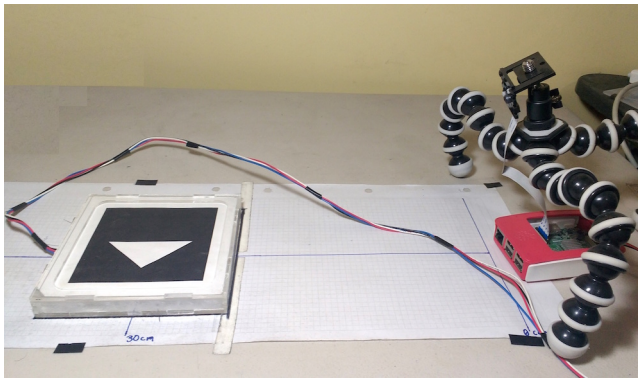


Figura 3.14: Disposición de los componentes del sistema propuesto.

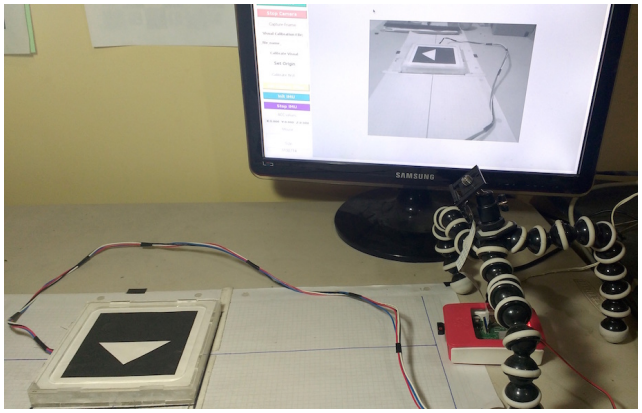


Figura 3.15: Sistema con todos sus componentes conectado a la interfaz HDMI.

Capítulo 4

Odometría visual-inercial

Como se alude en la parte introductoria de este trabajo, los sistemas de odometría actuales no están limitados a un sólo tipo de sensor en específico. Los sistemas de odometría híbridos fusionan la información provista por dos o más sensores diferentes para obtener datos con mayor fiabilidad; explotando las características de unos para compensar las deficiencias de otros, este tipo de sistemas aprovecha los datos obtenidos por cada sensor de manera complementaria.

A continuación se exponen a detalle los modelos propuestos que describen los problemas de odometría visual e inercial de manera independiente. La última sección de este capítulo examina el modelo solución propuesto que fusiona la información de ambos sensores mediante un filtro Kalman.

4.1. Odometría visual

El problema de odometría visual (Figura 1.3), descrito en la parte introductoria de este trabajo, supone dos escenarios equivalentes para el cálculo de las poses consecutivas de la cámara con respecto a la escena. Estos son, cuando hay una cámara moviéndose en una escena fija, y el caso inverso, cuando hay una escena (o para esta tesis, un objeto dentro de la escena) móvil y la cámara permanece inmóvil.

Para ésta tesis se considera el segundo caso, en el que la cámara fija rastrea al marcador propuesto dentro de la escena que ve. Este seguimiento se lleva a cabo por medio de la calibración a partir de una homografía, como se plantea en la Sección 2.2.4.

Debido a que se conocen de antemano las características de la cámara empleada en el sistema, es posible llevar a cabo una calibración fuera de línea para estimar los parámetros físicos del sensor, es decir, calcular la matriz K de parámetros intrínsecos del modelo de la cámara obscura, en la ecuación (2.8), para ésta cámara en particular, aminorando así efectos de distorsión producidos por la lente.

4.1.1. Calibración fuera de línea

Este proceso de calibración se realizó modificando la herramienta de OpenCV *interactive camera calibration application* [22] para poder procesar imágenes tomadas de la cámara Raspberry en su resolución nativa (3280×2464 px). Dicha herramienta sirve para estimar los parámetros físicos de la cámara aprovechando la detección de varios marcadores admitidos, con éstos es posible calcular una homografía para la calibración de la cámara.

En este caso utilizamos un marcador chAruco [11] compuesto por un tablero blanco y negro de 7×5 celdas con un patrón como el de la Figura 4.1. El marcador fue impreso en una hoja de papel con una escala de 35 mm para el lado de cualquier cuadrado negro; y posteriormente fue adherido a una superficie rígida plana.

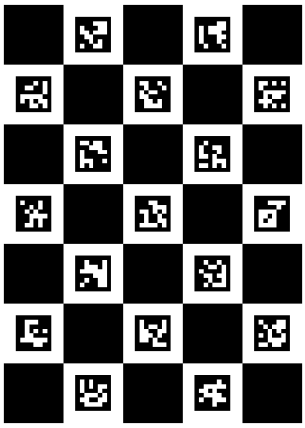


Figura 4.1: Marcador fiduciario chAruco utilizado para la calibración de la cámara.

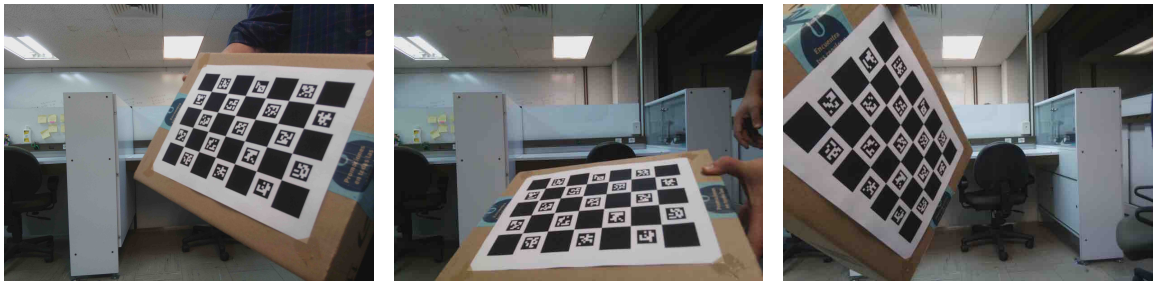
Para la ejecución de la aplicación, es necesario especificar mediante ciertos parámetros las características y el tipo de tablero que la aplicación debe buscar en la escena. Para el tablero propuesto, la aplicación se ejecuta como:

```
opencv_interactive-calibration -h=7 -w=5 --sz=.035 --t=charuco
```

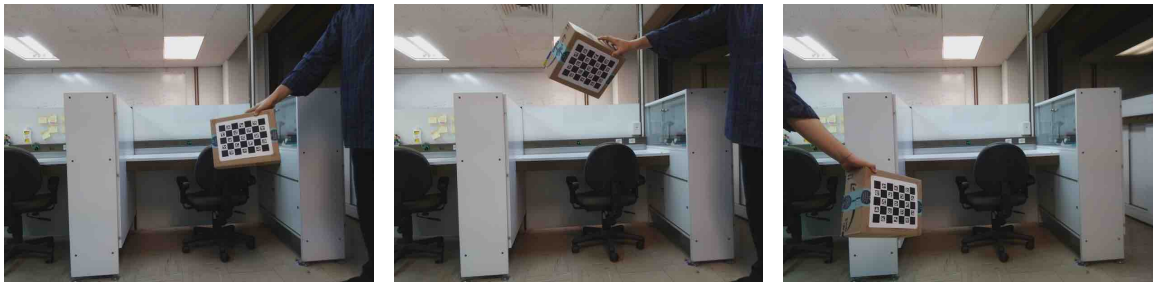
Mientras la aplicación se encuentra en ejecución, basta con posicionar fijamente al marcador dentro del rango visible de la cámara para alimentar al sistema con la pose actual detectada en cada cuadro. Este es un proceso iterativo en el cual se debe intentar cubrir el plano de la imagen de manera uniforme mientras el programa extrae información a partir de todas las vistas capturadas. Así, con un número determinado de vistas, finalmente obtendremos los parámetros de calibración y de distorsión de la cámara, junto con sus respectivos intervalos de confianza.

De manera análoga al procedimiento de Zhang [30], se realizaron dos conjuntos de calibración A y B, de 20 fotografías cada uno, para determinar los parámetros de distorsión, con el marcador a diferentes distancias de cámara. Para el conjunto A, el marcador fue capturado a una distancia aproximada de entre 30 a 50 centímetros,

mientras que para el B, la distancia fue de entre 100 a 150 cm. Algunos ejemplos de ambos conjuntos se muestran en la Figura 4.2.



(a) Fotografías ejemplo del conjunto A de calibración.



(b) Fotografías ejemplo del conjunto B de calibración.

Figura 4.2: Algunas de las fotografías realizadas para la calibración de la cámara. Ver el enlace de internet del conjunto completo en el apéndice B.

De los dos conjuntos de calibración realizados, el programa aceptó 14 y 13 fotografías para los conjunto A y B respectivamente. Los archivos XML con los parámetros obtenidos por el programa para los tres conjuntos analizados A, B y A+B se listan en el apéndice B.

Los datos obtenidos para los valores de distorsión tienen congruencia entre los conjuntos A y A+B, lo que concuerda con la afirmación de Zhang a cerca del sesgo que exhibe el conjunto de calibración B; en el cual, el patrón de calibración sólo ocupa la parte central de la imagen, donde la distorsión por el lente no es relevante y, por lo tanto, no puede estimarse de manera confiable.

Usando los parámetros obtenidos de las 27 imágenes aceptadas del conjunto A+B, obtenemos la distancia focal $f = 2504.0771$ calculada con la resolución nativa de la cámara de 3280×2464 px. Entonces, para la resolución del sistema propuesto de 640×480 px, existe un factor de escala de 1:5.125. Así podemos construir la matriz de calibración K particular de la cámara empleada en el sistema.

$$K = \begin{bmatrix} -488.60 & 0 & 320 \\ 0 & -488.60 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.1.2. Obtención de la pose del marcador

Una vez obtenida la matriz de calibración K particular de la cámara utilizada en el sistema, es posible calcular la pose del marcador detectado en la escena con respecto al modelo del marcador propuesto, como se expuso anteriormente en la Sección 2.2.4, en la página 19. Esta pose relaciona ambos marcos de referencia mediante la matriz de rotación R y el vector de traslación $\mathbf{t}$, que describen la rotación y posterior traslación del modelo del marcador para que se vea como en la imagen.

Si se conocen las dimensiones reales del marcador en la escena, es posible deducir la escala para las unidades del vector $\mathbf{t}$. Así que para este sistema, las unidades utilizadas están en milímetros.

4.1.3. Modelo de odometría visual

Para medir la trayectoria del marcador en el espacio es necesario proponer un modelo en el que se considera una pose inicial para el marcador, y a partir de ésta, se calcula la pose actual del objeto para cada instante. Este modelo se representa mediante la Figura 4.3, dónde se ilustra la referencia inicial M_0 con un marcador desvanecido; se supone una condición para justificar dicho modelo: el marcador con el sistema de coordenadas en rojo, que apunta M_2, se desplaza en el espacio mientras que la cámara permanece inmóvil, y por lo tanto, la pose inicial de referencia tampoco cambia.

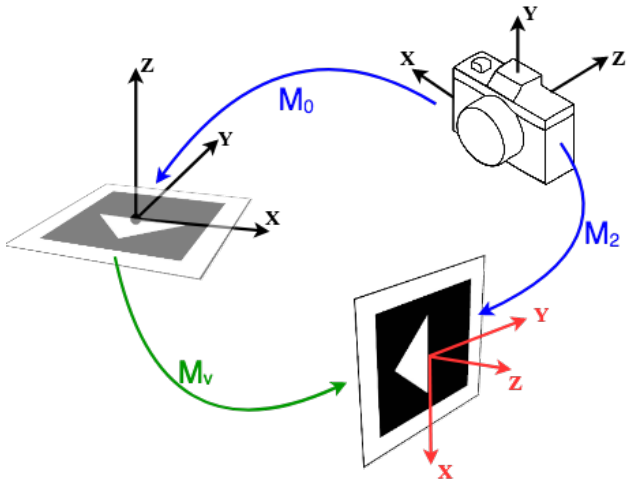


Figura 4.3: Diagrama del modelo propuesto de odometría visual

Del modelo de la cámara obscura, en la ecuación (2.8), en la pág 13, se tienen estas transformaciones para un punto $\mathbf{P}_1$:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = [R|\mathbf{t}]\mathbf{P}_1. \tag{4.2}$$

Si usamos un punto $\mathbf{P}_2$ homogéneo tendríamos que escribir la expresión anterior como:

$$\mathbf{P}_2 = \left[\begin{array}{c|c} R & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right] \mathbf{P}_1, \tag{4.3}$$

esto es, el punto $\mathbf{P}_1$ es rotado por R y luego trasladado según $\mathbf{t}$ para obtener el punto $\mathbf{P}_2$ (esto puede demostrarse fácilmente si usamos matrices de transformación de tamaño 4×4). La idea importante aquí es que la matriz $[R|\mathbf{t}]$ del modelo de la cámara obscura, representa una rotación por R y posterior traslación por $\mathbf{t}$ del punto (o del modelo que ve la cámara).

De esta manera, se inicializa la matriz M_0 del modelo a partir de una pose inicial conocida, tal que $M_0 = [R_0|\mathbf{t}_0]$. A partir de ahí, para cada vista en la que sea posible detectar al marcador, se puede obtener una pose actual $M_2 = [R_2|\mathbf{t}_2]$.

La pose actual del marcador con respecto del marco de referencia definido por la pose inicial M_0, se denota M_v y se calcula entonces como:

$$M_v = R_0^T([R_2|\mathbf{t}_2] - [\mathbf{0}|\mathbf{t}_0]) \tag{4.4}$$

Aquí, $\mathbf{0}$ es una matriz de ceros de $3{\times}3$, y la transformación calculada $M_v = [R_v|\mathbf{t}_v]$ es una matriz de tamaño $3{\times}4$. La expresión anterior (4.4) significa que tenemos que trasladar M_2 al origen (o referencia) $\mathbf{t}_0$ y eliminar la rotación inicial R_0.

La simulación de este modelo de odometría visual puede ejecutarse mediante los scripts en lenguaje Python incluídos en el Apéndice C de esta tesis.

4.2. Odometría inercial

Tal como se describe en la Sección 2.3.4, las técnicas de odometría inercial consideran las mediciones realizadas por una UMI adherida a un cuerpo para calcular su desplazamiento y orientación en el espacio. Mientras que la UMI expresa las mediciones realizadas para cada instante dentro de su propio marco de referencia no inercial, generalmente, los SNI deben expresar la trayectoria del objeto a partir de un marco de referencia inercial, que se encuentra fijo en el espacio. Estas transformaciones entre marcos de referencia se expresan mediante una *transformación Galileana* [8], dando lugar así a las ecuaciones que describen la cinemática del cuerpo rígido.

4.2.1. Cinemática del cuerpo rígido

En la Figura 4.4 se ubican el marco de referencia inercial Σ_0 correspondiente al mundo y el marco de referencia no inercial Σ_1 que corresponde al cuerpo en movimiento. Ambos marcos de referencia están separados desde sus respectivos orígenes por una distancia $\mathbf{d}$ y se relacionan mediante una matriz de rotación R que depende del vector $\boldsymbol{\theta}$, parametrizando el modelo de orientación elegido.

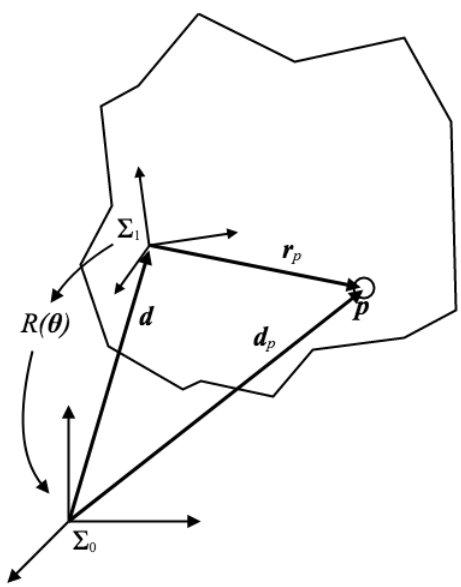


Figura 4.4: Marcos de referencia asociados a un cuerpo rígido móvil libre en el espacio [8].

El vector $\mathbf{r}_p$ representa la posición relativa de cualquier punto $\mathbf{p}$ en el espacio con respecto a Σ_1. Expresando la posición relativa de $\mathbf{p}$ con respecto a Σ_0:

$$\mathbf{d}_p^{(0)} = d^{(0)} + R\mathbf{r}_p^{(1)} \tag{4.5}$$

Si se simplifica la notación eliminando los superíndices que indican los marcos de referencia:

$$\mathbf{d}_p = \mathbf{d} + R\mathbf{r}_p \tag{4.6}$$

Entonces, es trivial deducir que la velocidad lineal del punto $\mathbf{p}$ se obtiene mediante la derivada con respecto del tiempo de la ecuación (4.6),

$$\dot{\mathbf{d}}_p = \dot{\mathbf{d}} + \dot{R}\mathbf{r}_p + R\dot{\mathbf{r}}_p \tag{4.7}$$

La derivada con respecto al tiempo de la matriz de rotación $\dot{R}$, expresa la tasa de cambio en la orientación del objeto con respecto al tiempo, y en consecuencia, la velocidad angular $\boldsymbol{\omega}$ con respecto al marco de referencia no inercial.

Definición 4.2.1. Velocidad angular La velocidad angular $\boldsymbol{\omega} \in \mathbb{R}^3$ de un marco de rotación con respecto a otro es un vector tridimensional que satisface las siguientes condiciones:

$$\dot{R}R^T = [\boldsymbol{\omega}^{(0)}\times] \text{ marco de referencia inercial} \tag{4.8}$$

$$R^T\dot{R} = [\boldsymbol{\omega}^{(1)}\times] \text{ marco de referencia no inercial} \tag{4.9}$$

Las ecuaciones en (4.2.1) pueden utilizarse para expresar la derivada con respecto al tiempo de la matriz de rotación como:

$$\begin{aligned} \dot{R} &= [\boldsymbol{\omega}^{(0)}\times]R \\ &= R[\boldsymbol{\omega}^{(1)}\times] \end{aligned} \tag{4.10}$$

Mientras que la velocidad angular $\boldsymbol{\omega}$ se expresa con respecto a Σ_1, éste vector es único y sus coordenadas pueden ser dadas con respecto a cualquier marco de referencia; a diferencia del vector $\boldsymbol{\theta}$, que expresa una parametrización de la posición angular, y que puede ser diferente entre ambos marcos de referencia.

De esta observación surgen dos conclusiones:

1. La velocidad angular y la derivada con respecto al tiempo de cualquier vector que parametrize la posición no puede ser igual.

$$\boldsymbol{\omega} \neq \dot{\boldsymbol{\theta}}$$

2. La velocidad angular debe de tener alguna relación con la derivada respecto al tiempo del vector de posición angular.

$$\boldsymbol{\omega} = \boldsymbol{\omega}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$$

El operador de pose cinemática J_θ mapea entonces la derivada con respecto al tiempo del vector $\boldsymbol{\theta}$, al vector de velocidad angular en el marco de referencia elegido Σ_i, tal que:

$$\boldsymbol{\omega}^{(i)} = {}^{i}J_\theta(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{4.11}$$

Retomando la ecuación (4.7), podemos expresar la velocidad lineal para cualquier punto del cuerpo omitiendo el término $R\dot{\mathbf{r}}_p$ debido a que la posición $\mathbf{r}_p$ es constante con respecto al marco de referencia del cuerpo, entonces, $\dot{\mathbf{r}}_p = 0$. Reescribiendo la ecuación:

$$\dot{\mathbf{d}}_p = \dot{\mathbf{d}} + \dot{R}\mathbf{r}_p \tag{4.12}$$

Reemplazando $\dot{R}$ según los marcos de referencia inercial y no inercial como en (4.10):

$$\dot{\mathbf{d}}_p = \dot{\mathbf{d}} + [\boldsymbol{\omega}^{(0)}\times]R\mathbf{r}_p = \dot{\mathbf{d}} + \boldsymbol{\omega}^{(0)} \times \mathbf{r}_p^{(0)} \tag{4.13}$$

$$= R\mathbf{v} + R[\boldsymbol{\omega}^{(1)}\times]\mathbf{r}_p = R(\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_p), \tag{4.14}$$

dónde los términos,

$$\mathbf{v} \triangleq \dot{\mathbf{d}}^{(1)} = R^T\dot{\mathbf{d}}$$
$$\boldsymbol{\omega} \triangleq \boldsymbol{\omega}^{(1)} = R^T\boldsymbol{\omega}^{(0)}$$

expresan la velocidad lineal y velocidad angular del cuerpo con respecto del marco de referencia no inercial.

4.2.2. Calibración de la UMI

El proceso de calibración para una UMI consiste en remover la componente de gravedad presente en las medidas de aceleración obtenidas con el instrumento. Para esto es necesario suponer un vector de gravedad $\mathbf{g}_0 = (0, 0, \pm g)^T$ que está alineado perpendicularmente con el plano terrestre, como se ilustra en la Figura 4.5. Aquí, g denota a la constante gravitacional terrestre.

Entonces, el vector de gravedad $\mathbf{g}_v$ medido por la UMI está dado a partir de la matriz de rotación R y la aceleración medida $\mathbf{a}_m$ al momento de la calibración teniendo que:

$$\mathbf{g}_v = R^T\mathbf{a}_m$$

Cumpliéndose entonces que el vector de gravedad del objeto se encuentre siempre alineado con el vector de gravedad terrestre para cada instante, de tal manera que:

$$\mathbf{g}_0 = R\mathbf{g}_v \tag{4.15}$$

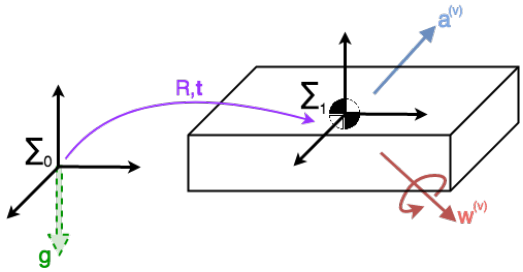


Figura 4.5: Diagrama del cuerpo rígido móvil libre. Los vectores de aceleración $\mathbf{a}$ y velocidad angular $\boldsymbol{\omega}$ están dados con respecto del marco de referencia no inercial. El vector de gravedad es perpendicular al plano terrestre.

A partir de la pose inicial del sistema inercial y de la matriz de rotación que relaciona al marco de referencia inercial con el del objeto, puede obtenerse la medida de aceleración *real* del sistema.

Tenemos entonces que la velocidad lineal del objeto es:

$$\mathbf{v} = R^T \dot{\mathbf{d}}$$

o bien,

$$\dot{\mathbf{d}} = R\mathbf{v}.$$

Derivando esta última ecuación obtenemos la aceleración aparente del objeto como:

$$\begin{aligned} \ddot{\mathbf{d}} &= R\dot{\mathbf{v}} + \dot{R}\mathbf{v} \\ &= R(\dot{\mathbf{v}} + \boldsymbol{\omega} \times \mathbf{v}) \\ &= R(\mathbf{a}_v + \mathbf{g}_v) \end{aligned} \tag{4.16}$$

Donde $\mathbf{a}_m \triangleq (\mathbf{a}_v + \mathbf{g}_v)$, es decir, la aceleración medida por la UMI se compone de la *aceleración real* causada por algún desplazamiento del objeto y la aceleración debida a la gravedad terrestre.

Por lo tanto, para aislar la *aceleración real* del objeto de la medida obtenida por la UMI, basta con remover el vector de gravedad $\mathbf{g}_0$ de la medición según la orientación actual del objeto con respecto del referencial inercial, de tal manera que:

$$\begin{aligned} \ddot{\mathbf{d}} &= R(\mathbf{a}_m - \mathbf{g}_v) \\ &= R\mathbf{a}_m - RR^T\mathbf{g}_0 \end{aligned} \tag{4.17}$$

$$\ddot{\mathbf{d}} = R\mathbf{a}_m - \mathbf{g}_0 \tag{4.18}$$

Sin embargo, éste método implica que la condición (4.15) siempre se cumpla, lo cual no siempre sucede en la práctica. Dichas discrepancias se traducen en aceleraciones aparentes que añaden error a la medición.

4.2.3. Cálculo de la posición angular, velocidad y desplazamiento

La odometría inercial se calcula de acuerdo al diagrama que ilustra la Figura 2.9, en la página 24. De la UMI, se obtiene para cada instante de tiempo dos vectores de mediciones; la aceleración lineal $\mathbf{a}_m \in \mathbb{R}^3$ con unidades en g's y el vector de velocidad angular $\boldsymbol{\omega} \in \mathbb{R}^3$ en $^{\circ}/$s.

A partir de una pose conocida para cierto punto en el espacio, el SNI comienza a leer e integrar los valores de la UMI para el cálculo de la posición angular, velocidad y desplazamiento del objeto en relación con el marco de referencia inicial.

La aceleración aparente del objeto se obtiene de restar la gravedad al vector de aceleración medida como en la ecuación (4.18). El vector de velocidad y posición se obtienen a partir de la doble integración de los valores calculados para la aceleración aparente.

En cambio, el vector de velocidad angular debe de ser expresado en términos del marco de referencia inercial como el cambio en la posición angular respecto al tiempo. De la ecuación (4.11) se tiene entonces que:

$$\dot{\boldsymbol{\theta}} = J_{\theta}^{-1}(\boldsymbol{\theta})\boldsymbol{\omega} \tag{4.19}$$

Si el sistema dinámico se discretiza, pueden integrarse los datos obtenidos numéricamente como:

$$\begin{aligned} \mathbf{d}_{n+1} &= \mathbf{d}_n + \dot{\mathbf{d}}_n \Delta t + \tfrac{1}{2}\ddot{\mathbf{d}}(\Delta t)^2, \\ \boldsymbol{\theta}_{n+1} &= \boldsymbol{\theta}_n + \dot{\boldsymbol{\theta}}_n \Delta t, \\ \dot{\mathbf{d}}_{n+1} &= \dot{\mathbf{d}}_n + \ddot{\mathbf{d}}_n \Delta t \end{aligned} \tag{4.20}$$

donde $\mathbf{d}$, $\dot{\mathbf{d}}$ y $\boldsymbol{\theta}$ son los vectores de posición, velocidad y posición angular respectivamente, mientras que Δt es el intervalo de tiempo que trascurre entre cada muestreo de la UMI.

4.3. Modelo del sistema visual-inercial

Se propone entonces un modelo visual-inercial que considere las mediciones de la UMI para estimar un vector de estado $\mathbf{x}$ que se define como:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\theta} \\ \dot{\mathbf{d}} \end{bmatrix}, \tag{4.21}$$

donde $\mathbf{d}$ y $\dot{\mathbf{d}}$ son la posición y velocidad lineal del objeto y $\boldsymbol{\theta}$ representa la posición angular. Los tres vectores $\mathbf{d}$, $\dot{\mathbf{d}}$ y $\boldsymbol{\theta}$ están en $\mathbb{R}^3$.

Las señales obtenidas de la UMI se representan con el vector $\mathbf{u} = [\mathbf{u}_1^T, \mathbf{u}_2^T]^T = [\mathbf{a}_m^T, \boldsymbol{\omega}^T]^T$, donde $\mathbf{a}_m$ es la aceleración lineal y $\boldsymbol{\omega}$ la velocidad angular medida. Estos dos vectores también están en $\mathbb{R}^3$.

Si se plantea la ecuación diferencial del modelo:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{d}} \\ \dot{\boldsymbol{\theta}} \\ \ddot{\mathbf{d}} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{x}_3 \\ J_\theta^{-1}(\mathbf{x}_2)\mathbf{u}_2 \\ R(\mathbf{x}_2)\mathbf{u}_1 - \mathbf{g}_0 \end{bmatrix} \tag{4.22}$$

Aquí, $\dot{\mathbf{d}}$ y $\dot{\boldsymbol{\theta}}$ son la velocidad lineal y velocidad angular del objeto respectivamente, mientras que $\ddot{\mathbf{d}}$ es la aceleración del mismo.

Si se usa la parametrización ZYZ en angulos de Euler para $\boldsymbol{\theta} = [\alpha, \beta, \gamma]^T$, entonces el operador inverso de pose cinemática J_θ^{-1} se define como:

$$J_\theta^{-1} = \begin{bmatrix} \frac{c_3 c_2}{-s_2} & \frac{s_3 c_2}{-s_2} & 1 \\ -s_3 & c_3 & 0 \\ \frac{c_3}{s_2} & \frac{s_3}{s_2} & 0 \end{bmatrix} \tag{4.23}$$

donde $s_1 = \text{sen}(\gamma)$, $c_1 = cos(\gamma)$, $s_2 = sen(\beta)$, $c_2 = cos(\beta)$, $s_3 = sen(\alpha)$ y $c_3 = sen(\alpha)$. Este operador es el que relaciona velocidad angular del marco de referencia no inercial con la representación de la velocidad angular en coordenadas del marco de referencia inercial [8].

De igual manera, para la matriz de rotación $R(\boldsymbol{\theta}) = R_z(\alpha)R_y(\beta)R_z(\gamma)$. De forma expandida se tiene:

$$\begin{aligned} R(\boldsymbol{\theta}) &= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & c_\beta & 0 \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_3 - c_2c_3s_1 & c_3s_2 \\ c_1c_2s_3 + c_3s_1 & c_1c_3 - c_2s_1s_3 & s_2s_3 \\ -c_1s_2 & s_1s_2 & c_2 \end{bmatrix} \end{aligned} \tag{4.24}$$

Entonces, según el modelo (4.22) la aceleración aparente del objeto $\ddot{\mathbf{d}}$ es obtenida a través de la rotación de la aceleración medida en el marco de referencia inercial mediante la matriz de rotación R, para posteriormente restar el vector de gravedad $\mathbf{g}_0$ correspondiente a la gravedad terrestre.

Entonces para cada instante de tiempo, se resuelve la ecuación diferencial para obtener el vector $\dot{\mathbf{x}}$; para después proceder con la integración numérica de los valores con respecto al tiempo, como en la ecuación (4.20), obteniendo para cada instante un nuevo vector de estado $\mathbf{x}$.

Para el caso inverso, dónde se desea obtener un vector $\boldsymbol{\theta}$ de ángulos de Euler a partir de una matriz de rotación, es crucial considerar el siguiente Algoritmo 2 para evitar singularidades:

Algoritmo 2 Ángulos de Euler ZYZ a partir de una matriz de rotación

Entrada: Matriz de rotación $R \in \mathbb{R}^{3\times 3}$.
Salida: Vector $\boldsymbol{\theta} \in \mathbb{R}^3$ de ángulos de Euler.

1: **if** $r_{33} > 1.0$ **then**
2: $\quad \theta_\alpha = atan^2(r_{21}, r_{11})$
3: $\quad \theta_\beta = 0$
4: $\quad \theta_\gamma = 0$
5: **else**
6: $\quad \theta_\alpha = atan^2(r_{32}, -r_{31})$
7: $\quad \theta_\beta = acos(r_{33})$
8: $\quad \theta_\gamma = atan^2(r_{23}, r_{13})$
9: **end if**

Capítulo 5

Filtro Kalman para fusión de datos visuales e inerciales

Para la combinación de datos entre los modelos visual e inercial se propone el uso de un filtro Kalman para el filtrado y estimación del vector de estado del sistema. El filtro Kalman, también conocido como *estimador lineal cuadrático*, es un algoritmo para la fusión de datos dado un sistema dinámico lineal desarrollado en la década de 1960 por Rudolf Emil Kálmán [28]. Este algoritmo considera el ruido presente en las mediciones del sistema para producir una estimación, tomando en cuenta la varianza de éstas mediciones. Además, este filtro es implementado mediante un algoritmo recursivo, que para cada iteración depende solamente de las mediciones de entrada, el estado calculado previamente y la matriz de incertidumbre que lo caracteriza; esto califica al filtro Kalman como un algoritmo ideal para sistemas limitados y de tiempo real.

5.1. Modelo del filtro Kalman

El filtro Kalman supone un sistema dinámico lineal discretizado en el dominio del tiempo. El modelo del filtro se construye a partir de una cadena de Markov con operadores lineales perturbados por ruido gaussiano. De tal forma, el estado del sistema se representa mediante un vector $\mathbf{x}$ de números reales. Entonces, para cada incremento en el tiempo discreto, se aplica un operador lineal al estado actual para generar un nuevo estado estimado. Esto se expresa con la siguiente ecuación:

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B\mathbf{u}_t + \boldsymbol{\epsilon}_t$$

donde $\mathbf{x}_t$ y $\mathbf{x}_{t-1}$ son vectores de estado, y $\mathbf{u}_t$ es un vector de control para cada instante de tiempo t. Por otro lado, A y B son las matrices de transición de estado y la matriz de control de entrada respectivamente. A es una matriz cuadrada de tamaño $n \times n$, donde n es la dimensión del vector de estado $\mathbf{x}$. La matriz B es de tamaño $n \times m$, dónde m es la dimensión del vector de control $\mathbf{u}$. La variable $\boldsymbol{\epsilon}$ denota un vector gaussiano con media cero y covarianza Q; que describe al ruido del proceso presente

en la transición de estado. Este vector debe ser de la misma dimensión que el vector de estado.

Para determinado instante de tiempo t, se introduce una observación $\mathbf{y}_t$ del estado real del sistema, tal que:

$$\mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}$$

donde, C es la matriz del modelo de observación, mientras que $\mathbf{v}$ es el vector de ruido en la observación, que se asume con media cero y covarianza V.

El filtro Kalman estima de manera recursiva un nuevo estado para cada instante, considerando solamente el vector de estado para el instante previo de tiempo y la observación actual. Usualmente, el estado del filtro se representa usando dos variables: el vector $\mathbf{x}_t$ para el estado estimado por el filtro, y la matriz P de covarianza de error, que representa una medida de precisión para el estado estimado.

De manera general, el filtro se descompone en dos fases: predicción y corrección. La primera de éstas, utiliza el estado estimado del instante de tiempo anterior para producir una estimación del estado en el tiempo actual. A ésta estimación se le conoce como el estado estimado *a priori*, porque aunque es una estimación para el instante de tiempo actual, no considera la información de la última observación para este instante. Por su parte, en la fase de corrección, la predicción vigente *a priori* se combina con la información de la observación actual para mejorar la estimación del vector de estado. A ésta estimación refinada se le denomina como estado estimado *a posteriori*.

Predicción

Estado estimado *a priori*	$\mathbf{x}_{t\|t-1} = A\mathbf{x}_{t-1\|t-1} + B\mathbf{u}_k$
Covarianza de error *a priori*	$P_{t\|t-1} = AP_{t-1\|t-1}A^T + Q$

Corrección

Covarianza de la observación	$S_t = CP_{t\|t-1}C^T + V$
Ganancia Kalman	$K_t = P_{t\|t-1}C^T S^{-1}$
Estado estimado *a posteriori*	$\mathbf{x}_{t\|t} = \mathbf{x}_{t\|t-1} + K_t(\mathbf{y}_t - C\mathbf{x}_{t\|t-1})$
Covarianza de error *a posteriori*	$P_{t\|t} = (I - K_tC)P_{t\|t-1}$

Generalmente, estas dos fases se alternan, con la predicción que estima el estado del sistema y la corrección que considera una observación como medida real de dicho estado. Sin embargo, esto no es imprescindible; si para un instante la observación no se encuentra disponible, puede omitirse el paso de corrección y realizar múltiples predicciones en su lugar. De igual manera, si para un mismo instante se cuenta con más de una observación, pueden realizarse varios pasos de corrección.

5.2. Implementación del filtro extendido

Las suposiciones de linealidad propuestas para la transición de estado en un filtro Kalman son inusuales en la práctica. Sin embargo, la teoría propuesta con el filtro extendido de Kalman permite linearizar la función f asociada a la transición de estado; aproximando mediante una función lineal tangente a f.

Entonces se tiene que, para una función f no lineal que modela el estado de transición de un sistema dinámico; f se puede linealizar mediante una expansión de Taylor de primer orden:

$$f(\mathbf{x},\mathbf{u}) = A\mathbf{x} + B\mathbf{u} \approx \frac{\partial f}{\partial \mathbf{x}}\mathbf{x} + \frac{\partial f}{\partial \mathbf{u}}\mathbf{u} \tag{5.1}$$

Retomando la ecuación (4.22) que describe el modelo propuesto para el sistema, reescribimos la ecuación como:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{d}} \\ \dot{\boldsymbol{\theta}} \\ \ddot{\mathbf{d}} \end{bmatrix} = f(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \mathbf{x}_3 \\ J_\theta^{-1}(\mathbf{x}_2)\mathbf{u}_2 \\ R(\mathbf{x}_2)\mathbf{u}_1 - \mathbf{g}_0 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{d}} \\ J_\theta^{-1}(\boldsymbol{\theta})\boldsymbol{\omega} \\ R(\boldsymbol{\theta})\mathbf{a}_m - \mathbf{g}_0 \end{bmatrix} \tag{5.2}$$

Entonces $f(\mathbf{x},\mathbf{u}) \in \mathbb{R}^9$. Expandiendo la expresión:

$$\begin{aligned}
\begin{bmatrix} \dot{\mathbf{d}} \\ \dot{\boldsymbol{\theta}} \\ \ddot{\mathbf{d}} \end{bmatrix} &= \begin{bmatrix} \dot{d}_x \\ \dot{d}_y \\ \dot{d}_z \\ \begin{bmatrix} \quad J_\theta^{-1} \quad \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \\ \begin{bmatrix} \quad R \quad \end{bmatrix} \begin{bmatrix} a_{m_x} \\ a_{m_y} \\ a_{m_z} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \dot{d}_x \\ \dot{d}_y \\ \dot{d}_z \\ \omega_x \frac{c_3 c_2}{-s_2} + \omega_y \frac{s_3 c_2}{-s_2} + \omega_z \\ -\omega_x s_3 + \omega_y c_3 \\ \omega_x \frac{c_3}{s_2} + \omega_y \frac{s_3}{s_2} \\ a_{m_x}(c_1c_2c_3 - s_1s_3) + a_{m_y}(-c_1s_3 - c_2c_3s_1) + a_{m_z}(c_3s_2) \\ a_{m_x}(c_1c_2s_3 + c_3s_1) + a_{m_y}(c_1c_3 - c_2s_1s_3) + a_{m_z}(s_2s_3) \\ a_{m_x}(-c_1s_2) + a_{m_y}(s_1s_2) + a_{m_z}(c_2) - g \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix}
\end{aligned} \tag{5.3}$$

Así, las derivadas parciales de la función

$$\frac{\partial f}{\partial \mathbf{x}}\mathbf{x} + \frac{\partial f}{\partial \mathbf{u}}\mathbf{u} \approx A\mathbf{x} + B\mathbf{u} \tag{5.4}$$

donde $A \in \mathbb{R}^{9\times 9}$ y $B \in \mathbb{R}^{9\times 6}$.

Calculando la matriz A con respecto a la función f:

$$\frac{\partial f}{\partial \mathbf{x}} = A = \left[\begin{array}{ccc|ccc|ccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & cot_2(\omega_x s_3 - \omega_y c_3) & csc_2^2(\omega_x c_3 + \omega_y s_3) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\omega_x c_3 - \omega_y s_3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & csc_2(\omega_y c_3 - \omega_x s_3) & -cot_2 csc_2(\omega_x c_3 + \omega_y s_3) & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \frac{\partial f_7}{\partial \alpha} & \frac{\partial f_7}{\partial \beta} & \frac{\partial f_7}{\partial \gamma} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial f_8}{\partial \alpha} & \frac{\partial f_8}{\partial \beta} & \frac{\partial f_8}{\partial \gamma} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial f_9}{\partial \alpha} & \frac{\partial f_9}{\partial \beta} & \frac{\partial f_9}{\partial \gamma} & 0 & 0 & 0
\end{array}\right] \tag{5.5}$$

donde,

$$\begin{aligned}
\frac{\partial f_7}{\partial \alpha} =& a_{m_x}(s_3(-c_2)c_1 - c_3 s_1) + a_{m_y}(s_3 c_2 s_1 - c_3 c_1) - a_{m_z}(s_3 s_2), \\
\frac{\partial f_7}{\partial \beta} =& c_3(s_2(a_{m_y} s_1 - a_{m_x} c_1) + a_{m_z} c_2), \\
\frac{\partial f_7}{\partial \gamma} =& s_3(a_{m_y} s_1 - a_{m_x} c_1) - c_3 c_2(a_{m_x} s_1 + a_{m_y} c_1), \\
\frac{\partial f_8}{\partial \alpha} =& a_{m_x}(c_3 c_2 c_1 - s_3 s_1) + a_{m_y}(s_3(-c_1) - c_3 c_2 s_1) + a_{m_z} c_3 s_2, \\
\frac{\partial f_8}{\partial \beta} =& s_3(s_2(a_{m_y} s_1 - a_{m_x} c_1) + a_{m_z} c_2), \\
\frac{\partial f_8}{\partial \gamma} =& c_3(a_{m_x} c_1 - a_{m_y} s_1) - s_3 c_2(a_{m_x} s_1 + a_{m_y} c_1), \\
\frac{\partial f_9}{\partial \alpha} =& 0, \\
\frac{\partial f_9}{\partial \beta} =& c_2(a_{m_y} s_1 - a_{m_x} c_1) - a_{m_z} s_2, \\
\frac{\partial f_9}{\partial \gamma} =& s_2(a_{m_x} s_1 + a_{m_y} c_1).
\end{aligned}$$

Para este caso, la matriz B de control de entrada no se calcula debido a que no es requerida para la variante extendida del filtro, en la cual la transición de estado se lleva a cabo mediante la función f descrita en (4.22) y la posterior integración planteada en (4.20).

Se requiere entonces que el sistema se inicialice en una pose preliminar con el marcador dentro del rango de visión de la cámara, donde pueda se reconocido. Además, el marcador debe estar en una posición tal que el eje Z de su sistema de coordenadas se encuentre perpendicular al plano terrestre. Esta presunción es necesaria para la obtención de la aceleración aparente del objeto como se describe en la página 46,

Sección 4.2.2. Así es posible calcular un vector de estado inicial $\mathbf{x}_0$ dónde se asume posición y velocidad lineal cero.

Para el filtro Kalman, se propone el modelo de observación siguiente:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix}. \tag{5.6}$$

Aunque el sistema visual no mide la velocidad lineal del objeto directamente, se propone usar un modelo de observación el cual asume que, cuando el sistema visual puede capturar el objeto, este se encuentra a velocidad lineal cero con cierto rango de varianza. Este modelo de observación permite controlar la acumulación de error causada por la integración del ruido presente en las mediciones de la UMI.

5.2.1. Cuaternio como modelo de orientación

La integración numérica descrita en la sección 4.2.3 (pág. 48) para los valores del vector de posición angular $\boldsymbol{\theta}$ usando la paramerización propuesta en ángulos de Euler puede presentar singularidades. Por esta razón, se consideran los cuaternios para la integración de la velocidad angular mediante un modelo que evade singularidades para la representación de un vector de posición angular como se detalla en [7, 27].

Los cuaternios fueron propuestos en 1843 por Sir William Hamilton para expresar una orientación en el espacio tridimensional mediante un conjunto de tres números imaginarios y una variable real; que parametrizan el espacio completo de orientaciones sin singularidades, pero no tienen significado físico directo [8].

Un cuaternio, denotado $\mathbf{q}$, se parametriza como:

$$\mathbf{q} = \left[q_w, q_x, q_y, q_z\right]^T \in \mathbb{R}^4, \tag{5.7}$$

donde q_w representa la parte real y los valores q_x, q_y, q_z la parte imaginaria. Un cuaternio cumple siempre con la norma unitaria tal que:

$$||\mathbf{q}|| = 1 \quad \Leftrightarrow \quad \mathbf{q}^T\mathbf{q} = ||\mathbf{q}||^2 = q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1. \tag{5.8}$$

Una matriz de rotación R puede obtenerse a partir de un cuaternio, con respecto a la siguiente expresión:

$$R(\mathbf{q}) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_xq_y - q_wq_z) & 2(q_xq_z + q_wq_y) \\ 2(q_xq_y + q_wq_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_yq_z + q_wq_x) \\ 2(q_xq_z - q_wq_y) & 2(q_yq_z - q_wq_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{5.9}$$

Por el contrario, para convertir una matriz de rotación a un cuaternio, se utiliza la siguiente formulación:

$$\begin{aligned} q_w &= \frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1} \\ q_x &= \operatorname{sgn}(r_{32}-r_{23})\frac{1}{2}\sqrt{r_{11}-r_{22}-r_{33}+1} \\ q_y &= \operatorname{sgn}(r_{13}-r_{31})\frac{1}{2}\sqrt{-r_{11}+r_{22}-r_{33}+1} \\ q_z &= \operatorname{sgn}(r_{21}-r_{12})\frac{1}{2}\sqrt{-r_{11}-r_{22}+r_{33}+1} \end{aligned} \tag{5.10}$$

donde sgn() denota a la función signo.

La conversión de un cuaternio a la representación ZYZ de ángulos de Euler se realiza por medio de una matriz de rotación, tomando en cuenta el Algoritmo 2, en la página 50, para evitar singularidades.

5.2.2. Integración de la velocidad angular mediante un cuaternio

Como ya se mencionó anteriormente, la integración de la velocidad angular $\boldsymbol{\omega}$ medida por la UMI requiere de la información sobre la posición angular actual del objeto para cada instante. Se propone, de igual manera, utilizar un cuaternio para realizar esta operación de la siguiente manera:

1. Se debe calcular una orientación inicial $\mathbf{q}_0$ ya sea a partir del cuaternio identidad $\mathbf{q}_I = [1, 0, 0, 0]^T$, o bien, considerando una matriz de rotación inicial R_0 y obteniendo el cuaternio correspondiente como en (5.10).

2. Se construye el cuaternio de velocidad angular $\mathbf{q}_{\boldsymbol{\omega}}$ con los valores medidos por la UMI, tal que:

$$\mathbf{q}_{\boldsymbol{\omega}} = [0, \omega_x, \omega_y, \omega_z]^T = \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{5.11}$$

3. La integración del nuevo cuaternio $\mathbf{q}_{k+1}$ se obtiene de la multipliciación de cuaternios, denotada por el operador $\otimes$:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \frac{\Delta t}{2}\mathbf{q}_k \otimes \mathbf{q}_{\boldsymbol{\omega}} \tag{5.12}$$

   donde Δt representa el intervalo de tiempo transcurrido entre los muestreos de la UMI.

   La expresión anterior (5.12) puede escribirse de forma matricial como:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \frac{\Delta t}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & [\boldsymbol{\omega}]_\times \end{bmatrix}\mathbf{q}_k \tag{5.13}$$

o bien, de forma expandida, como:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \frac{\Delta t}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \mathbf{q}_k \tag{5.14}$$

4. La precisión finita del cómputo de estas operaciones puede provocar inestabilidades numéricas que se traducen en un cuaternio no unitario [8]. Ésta situación puede remediarse fácilmente renormalizando el cuaternio calculado para cada iteración, como:

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{||\mathbf{q}||} \tag{5.15}$$

5. Finalmente, el cuaternio obtenido puede convertirse a una matriz de rotación como se formula en la ecuación (5.9), y posteriormente, obtener los nuevos ángulos de Euler.

Capítulo 6

Pruebas y resultados

A continuación se describen las pruebas realizadas a partir de los modelos propuestos para el sistema visual e inercial por separado, así como los resultados obtenidos de la interfaz visual para cada una de éstas. Después se exponen los resultados del sistema acoplado mediante el filtro Kalman como se propuso en el capítulo anterior. Aquí se incluyen capturas de la interfaz gráfica desplegada por el sistema y gráficas de trayectoria que validan la correctitud y fiabilidad para cada caso.

6.1. Prueba del sistema visual

Primeramente, se considera el modelo de odometría puramente visual ilustrado en la página 42 con la Figura 4.3. Si el marcador es detectado en la escena, el sistema dibuja sobrepuesta a la imagen, una línea en rojo que une a todos los puntos que componen al marcador propuesto (véase Figura 3.4). Entonces, de ser posible el cómputo de una homografía dados estos puntos, se despliega un conjunto de ejes ortogonales acotados dentro de un recuadro verde, como en la Figura 6.1, que es el cuadrado del marcador y su sistema de coordenadas reproyectado en la escena a partir de la pose calculada.

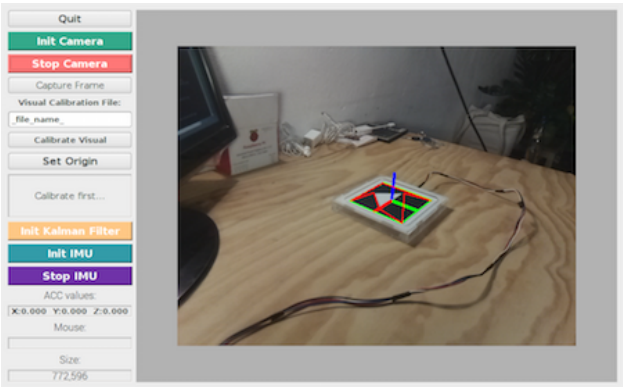


Figura 6.1: Detección del marcador del objeto en la escena. Al lado izquierdo se ubica la interfaz de control desplegada por el sistema

Una vez que el usuario captura la pose M_0 que define el origen del sistema, se superpone a la imagen un sistema de coordenadas en rojo,verde y azul inscritos dentro de un recuadro turquesa que referencian esta pose inicial; y en la parte lateral izquierda de la interfaz, se despliegan los valores de posición lineal y angular (en milímetros y grados respectivamente) para cada uno de los ejes X,Y,Z con respecto al origen previamente definido. Esto se ilustra en la Figura 6.2.

Figura 6.2: Capturas del sistema de odometría visual. En el lateral izquierdo se despliega la información relativa a la pose calculada del objeto.

Estos datos se extraen de la transformación definida por la matriz calculada M_v, y se actualizan para cada cuadro capturado por la cámara en el que sea posible detectar al marcador. Como se explica en la sección 4.1.3 (página 42) que describe a este modelo de odometría visual, la pose calculada M_v depende directamente de la pose de referencia M_0, y por lo tanto, también de que la cámara no cambie de posición.

Las pruebas realizadas con esta configuración se implementaron en un hilo del procesador ejecutándose a 30 Hz.

6.2. Prueba del sistema inercial

En esta prueba, se propone aislar la integración de los datos proporcionados por el sistema inercial a partir de un estado preliminar conocido. La inicialización del sistema considera entonces una pose inicial del objeto con respecto a la cámara, que

es calculada con el marcador, como en la Figura 6.1. Dada ésta observación, es posible obtener una estimación de la pose del objeto con respecto a la pose de referencia a partir de la integración de las señales proporcionadas por la UMI.

Para este caso, el sistema se inicializa con una matriz de rotación identidad $R_0 = I$, y un vector de traslación cero $\mathbf{t}_0 = [0, 0, 0]^T$; de manera que el marco de referencia inicial y el estimado por la UMI se encuentran inicialmente alineados como en la Figura 6.3.

Figura 6.3: Inicialización del sistema inercial a partir de una referencia visual. Los marcos de referencia de ambos sistemas se encuentras alineados.

Para evitar singularidades con la matriz de rotación identidad, el vector $\boldsymbol{\theta}$ que parametriza la posición angular, se inicializa como: $\boldsymbol{\theta}_0 = [0.0, 0.00001, 0.0]^T$. Si se considera la integración de la velocidad angular utilizando cuaternios, entonces se inicializa el cuaternio identidad $\mathbf{q}_0 = [1, 0, 0, 0]^T$.

En las subsiguientes Figuras 6.4 y 6.5, el sistema de coordenadas en rojo, verde y azul inscrito en el recuadro turquesa representa a la pose de referencia con la se inicializa al sistema, mientras que el sistema de coordenadas análogo inscrito en el recuadro violeta denota a la pose estimada a partir de los datos inerciales. Con esta información, es posible desplegar un objeto virtual en la interfaz, que se visualiza como un cubo en el sistema.

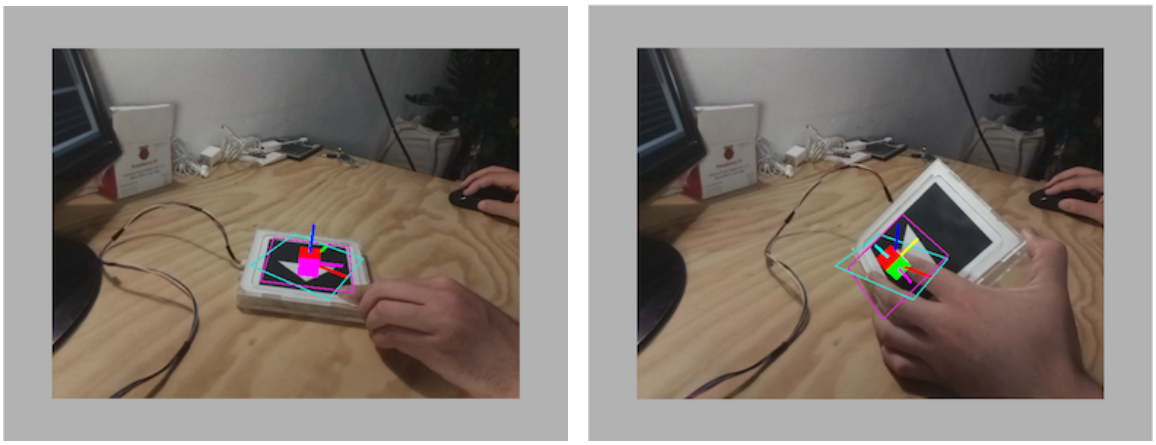


Figura 6.4: Integración de las señales del giroscópio por medio de un cuaternio. El cubo inscrito en el recuadro violeta representa la pose angular estimada.

(a) Pose estimada después de 1 segundo. (b) Pose estimada después de 3 segundos.

Figura 6.5: Integración de las señales inerciales con el objeto en reposo. Las capturas muestran la deriva producida por el sistema a diferentes instantes con una frecuencia de integración de 500 Hz.

Las pruebas realizadas con el sistema inercial se probaron a frecuencias de ejecución de 100 y 500 Hz, implementándose en un hilo del procesador independiente al que utiliza el sistema visual.

6.3. Prueba del sistema acoplado con el filtro Kalman

La fusión de datos proporcionados por ambos sistemas, visual e inercial, se lleva a cabo con el modelo del filtro Kalman descrito en el Capítulo 5 de esta tesis. Este algoritmo permite compensar las deficiencias inherentes de ambos modelos obteniendo así un estado estimado para cada iteración del sistema; aún cuando existan cambios de iluminación repentinos, oclusión del marcador o movimientos bruscos del objeto.

Este sistema acoplado se inicializa de igual manera que el sistema puramente visual, pero en lugar de introducir directamente la pose observada por la cámara, se considera un vector de observación $\mathbf{y} \in \mathbb{R}^9$, tal que:

$$\mathbf{y} = \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\theta} \\ \dot{\mathbf{d}} \end{bmatrix}, \tag{6.1}$$

donde $\mathbf{d}$ y $\boldsymbol{\theta}$ son la posición lineal y angular registradas por el sistema visual para cada instante, mientras que se considera que el vector de velocidad $\dot{\mathbf{d}}$ es igual a cero.

Para estimar los valores de la varianza del sistema utilizados en la matriz V del filtro, se consideraron los experimentos descritos en el apéndice D de esta tesis.

A continuación, las Figuras 6.7 y 6.6 ilustran capturas del sistema en dónde la pose de referencia se dibuja de igual manera que en la prueba del sistema visual, mientras que el cubo reproyectado en la imagen representa el estado estimado por el filtro.

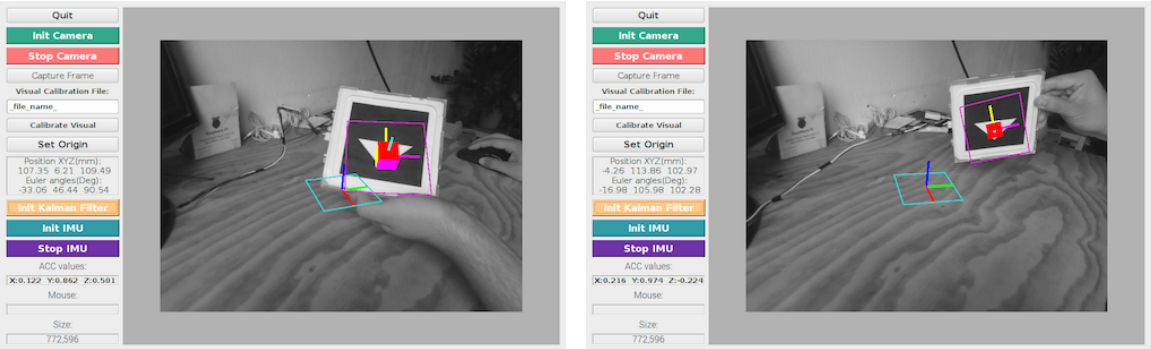


Figura 6.6: Capturas de la prueba del filtro Kalman cuando existe oclusión del objeto o no se puede detectar.

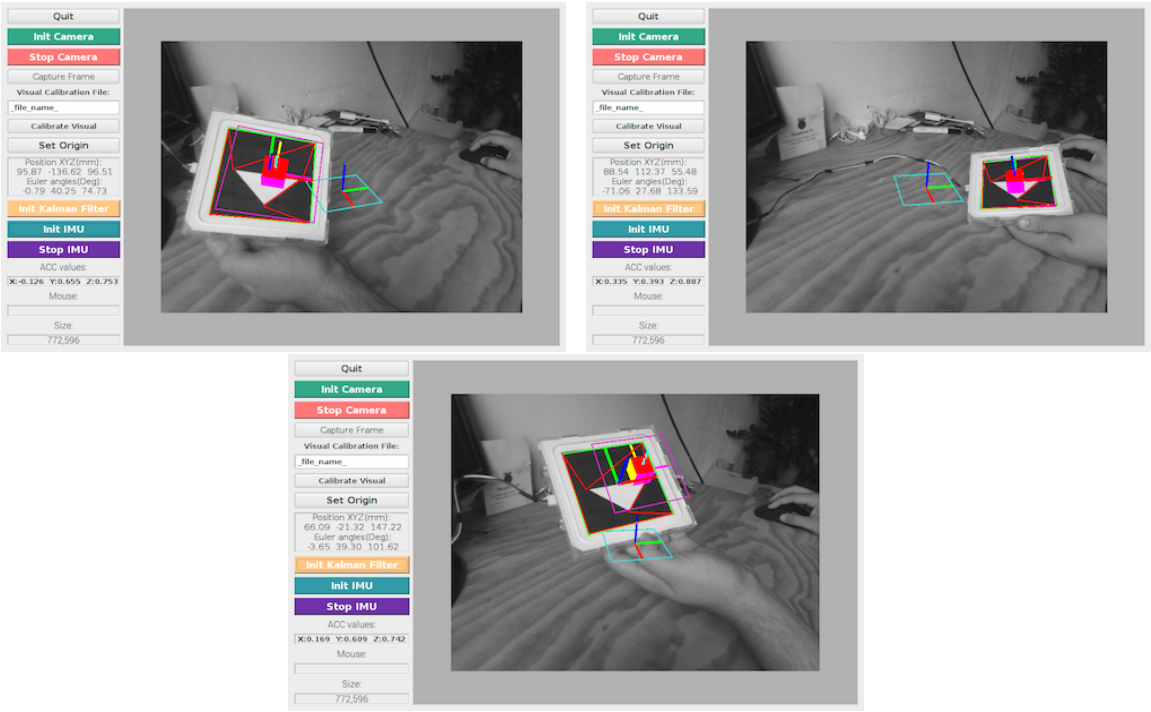


Figura 6.7: Capturas de la prueba del filtro Kalman mientras se mueve el objeto en la escena. En estas tres capturas se observan los tres sistemas de coordenadas, el de referencia, el del marcador detectado y el del estado estimado por el filtro ilustrado con un cubo.

6.4. Resultados filtro Kalman

Para medir la fiabilidad y correctitud del sistema se construyeron dos escenarios experimentales en los que se midieron durante 5 segundos las trayectorias del objeto guiado por una escuadra, utilizando diferentes combinaciones para los valores del filtro. Este intervalo de tiempo se consideró suficiente para mover el objeto sobre la trayectoria a velocidad constante, permitiendo aún la detección por parte del sistema visual.

En todos los experimentos el sistema visual se ejecutó a una frecuencia de 30 Hz, mientras que el sistema inercial y el filtro Kalman se ejecutaron en un hilo independiente a 500 Hz. Durante los 5 segundos de cada prueba, el filtro Kalman se muestreó a una frecuencia de 100 Hz, obteniendo así, 500 muestras por experimento.

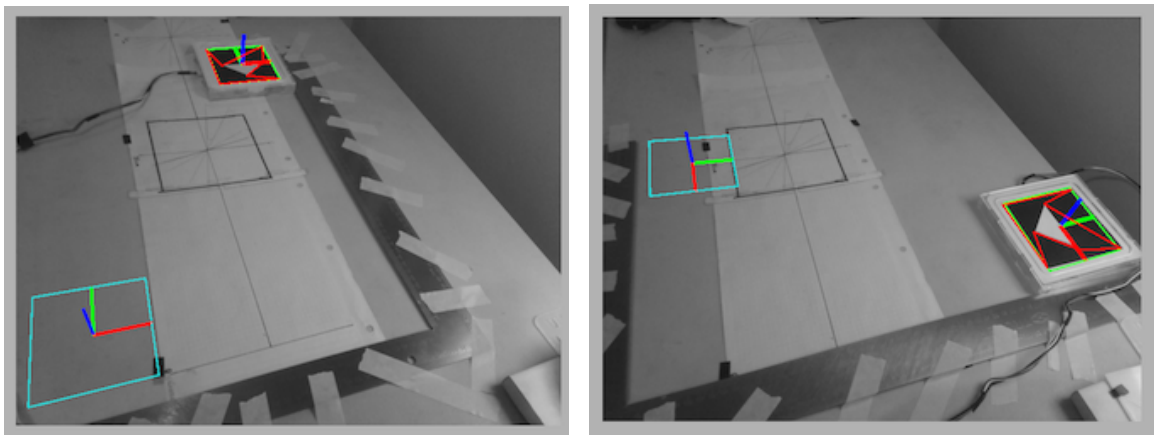
Concretamente, se proponen las tres configuraciones descritas en la Tabla 6.1; dónde la configuración 3 es la más sensible a los datos inerciales, mientras que la 1 responde mayormente a la información visual. Estas configuraciones fueron determinadas de manera experimental.

Tabla 6.1: Configuraciones del filtro Kalman utilizadas para los experimentos.

		Configuración		
		1	2	3
Matriz de covarianza Q	$\sigma_{\mathbf{d}}$	0.1	0.01	0.001
	$\sigma_{\boldsymbol{\theta}}$	0.000001	0.0000001	0.0000001
	$\sigma_{\dot{\mathbf{d}}}$	0.01	0.01	0.01
Matriz de covarianza V	$\sigma_{\mathbf{d}}$	0.1	0.01	0.01
	$\sigma_{\boldsymbol{\theta}}$	0.1	0.01	0.01
	$\sigma_{\dot{\mathbf{d}}}$	1.0	10.0	10.0

En los experimentos, se fijó con cinta adhesiva una escuadra de metal a la mesa. Entonces el marcador se coloca en un extremo de la escuadra dónde se inicializa la pose de referencia del sistema. Después, se procede desplazar el objeto al otro extremo de la escuadra arrastrándolo sobre la superficie usando el borde de la escuadra como guía. Para ambos escenarios, el objeto se desplaza sobre el plano, primero en el eje positivo X hasta topar con la parte interna de la escuadra, y después se mueve en el sentido positivo del eje Y hasta llegar al final de la trayectoria.

Los dos escenarios, como los ve la cámara, con las poses de referencia iniciales y finales de la trayectoria propuesta se ilustran en la Figura 6.8.



(a) Escenario 1 (b) Escenario 2

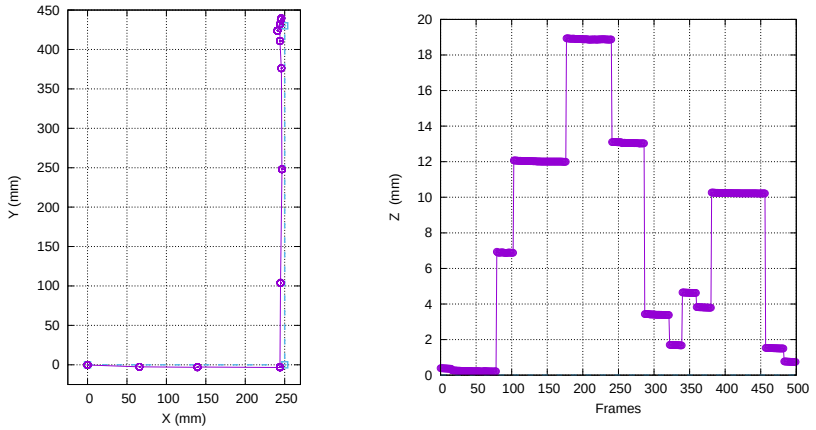
Figura 6.8: Escenarios para las pruebas del filtro Kalman. El sistema de coordenadas inscrito en el recuadro azul denota la pose de referencia inicial, mientras que el marcador se encuentra al final de la trayectoria delimitada por la escuadra. El objeto se desplaza sobre el plano, primero en el eje X y luego en Y, siguiendo la orilla de la escuadra mostrada en las Figuras.

Además, para ambos escenarios, los experimentos con cada configuración del filtro se efectuaron de dos maneras: la primera, reiniciando el filtro cada que se tiene una pose visual válida, y la segunda, ejecutando el filtro con las mediciones de odometría obtenidas a partir de la pose de referencia sin reiniciar para todo el experimento.

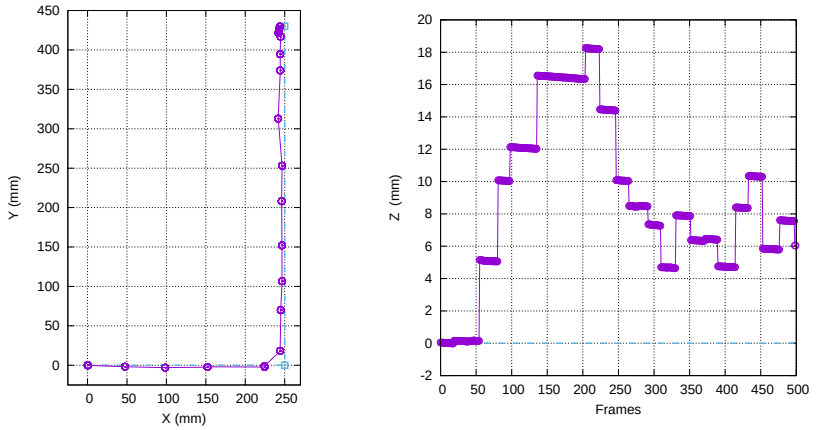
En todas las gráficas subsiguientes de las Figuras 6.9, 6.10, 6.9 y 6.10, los tres pares de gráficas en cada Figura corresponden a cada una de las configuraciones del filtro para un experimento en concreto.

Cada circulo violeta en las gráficas representa una muestra obtenida del vector **d** de posición a partir del estado del sistema. Las gráficas del lado izquierdo despliegan la trayectoria del objeto en el plano XY, mientras que las del lado derecho, los valores de posición para el eje Z con respecto al tiempo. La línea punteada en azul en las gráficas representa el valor esperado de la trayectoria según la trayectoria del experimento. Para el caso de las gráficas del lado derecho, esta línea se encuentra siempre en 0 dado que el objeto se desplazó sobre el plano.

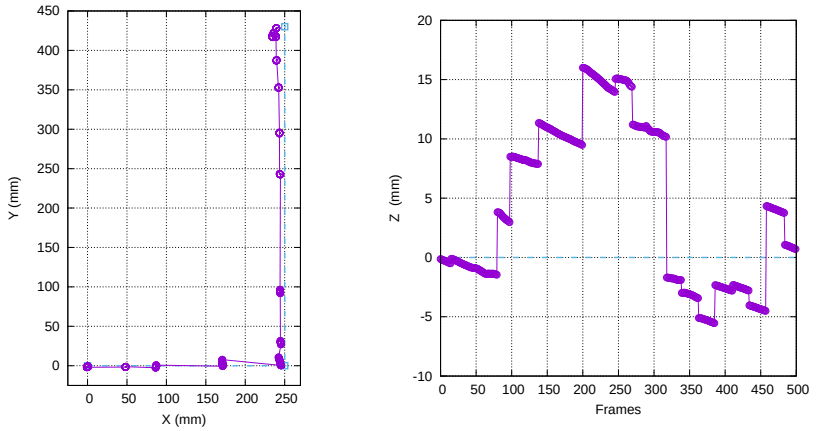
Para el primer escenario, la trayectoria del objeto en el plano se consideró de aproximadamente 250 mm sobre el eje X y 430 mm sobre el eje Y. En las Figuras 6.9 y 6.10 se ilustran las gráficas de los resultados obtenidos para la trayectoria del objeto en este primer escenario con las distintas configuraciones del filtro Kalman con y sin reinicio respectivamente.



(a) Trayectoria en el escenario 1 para la configuración 1 del filtro con reinicio.

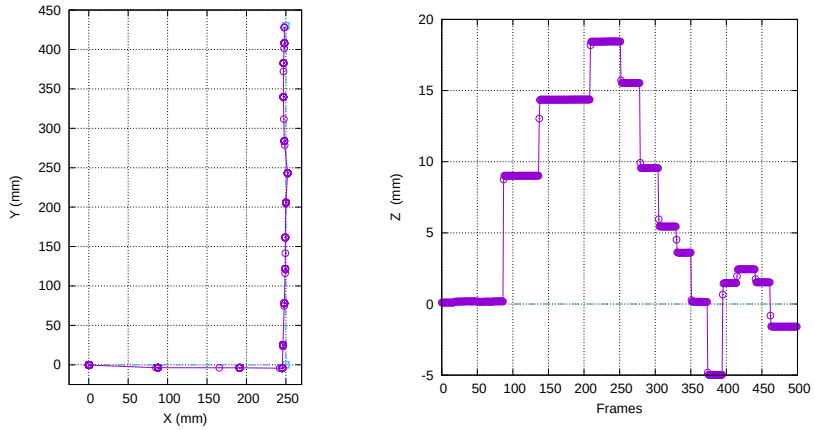


(b) Trayectoria en el escenario 1 para la configuración 2 del filtro con reinicio.

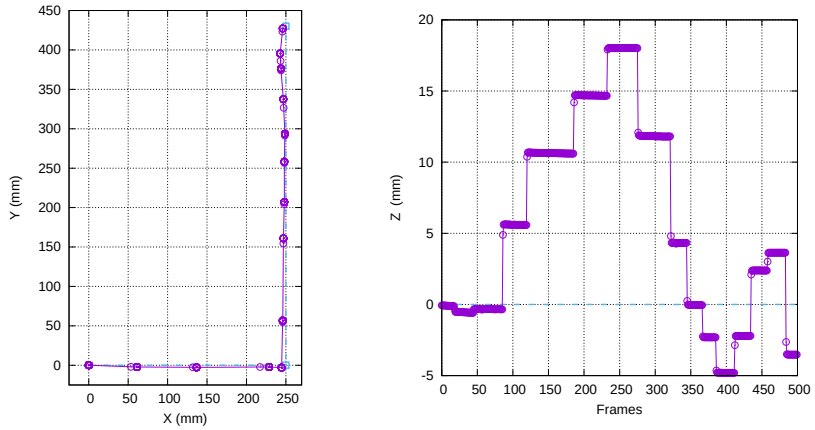


(c) Trayectoria en el escenario 1 para la configuración 3 del filtro con reinicio.

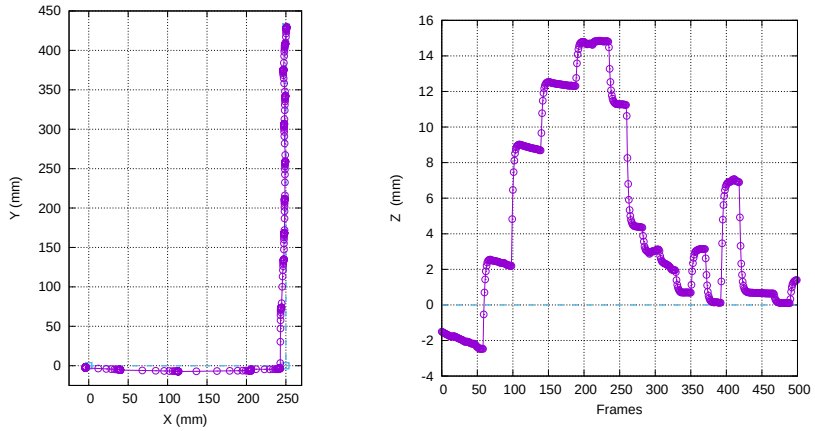
Figura 6.9: Gráficas de la trayectoria del objeto durante los experimentos en el escenario 1 reiniciando el filtro.



(a) Trayectoria en el escenario 1 para la configuración 1 del filtro sin reinicio.



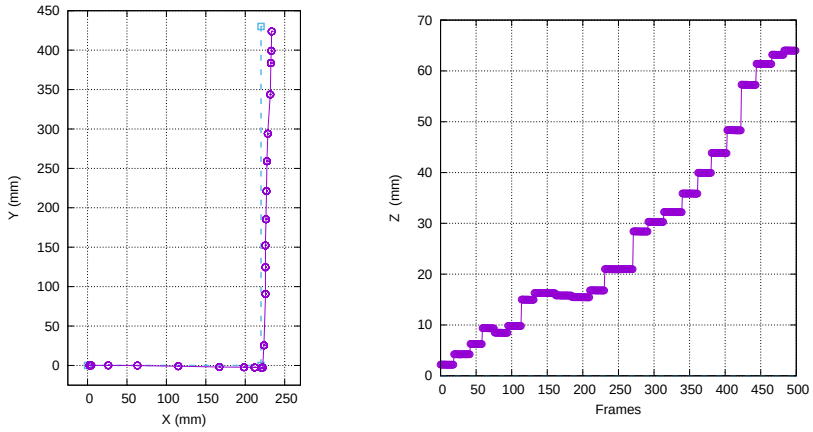
(b) Trayectoria en el escenario 1 para la configuración 2 del filtro sin reinicio.



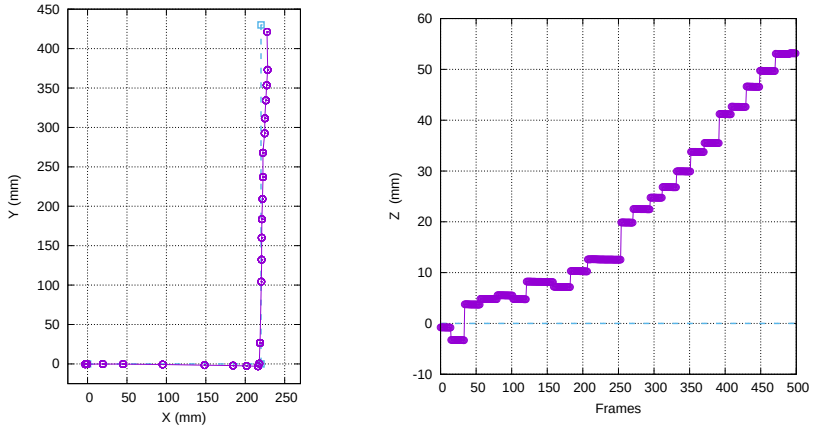
(c) Trayectoria en el escenario 1 para la configuración 3 del filtro sin reinicio.

Figura 6.10: Gráficas de la trayectoria del objeto durante los experimentos en el escenario 1 sin reiniciar el filtro.

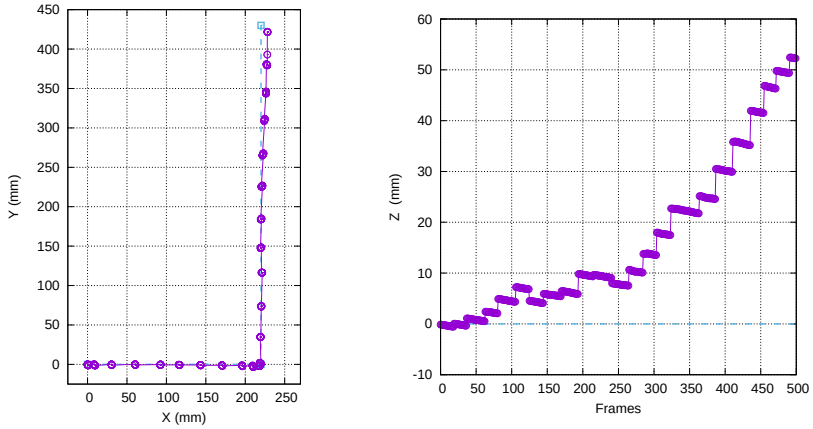
Para el segundo escenario, la trayectoria del objeto en el plano se consideró de aproximadamente 215 mm sobre el eje X y luego 430 mm sobre el eje Y. Los resultados para este escenario se ilustran en la Figuras 6.11 y 6.12 de manera análoga anterior.



(a) Trayectoria en el escenario 2 para la configuración 1 del filtro con reinicio.

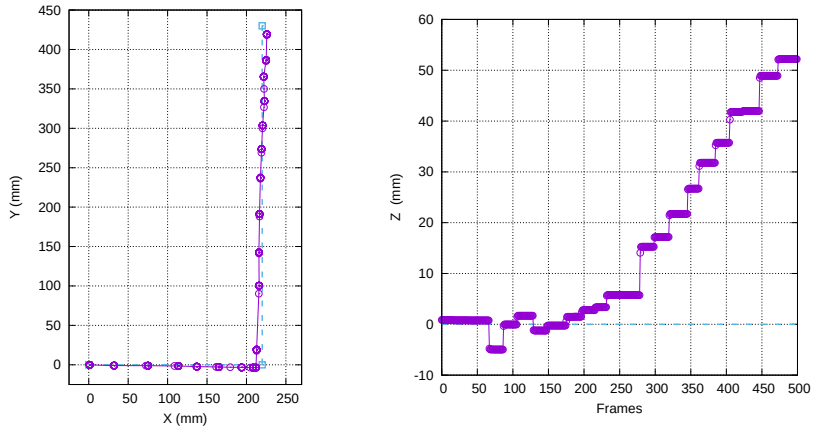


(b) Trayectoria en el escenario 2 para la configuración 2 del filtro con reinicio.

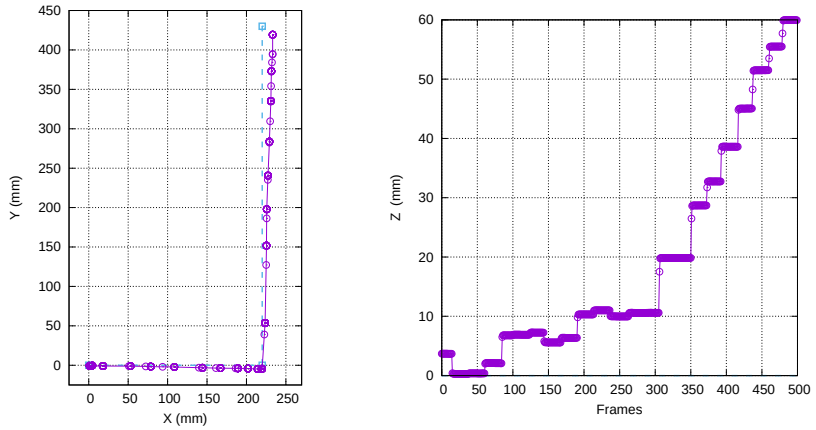


(c) Trayectoria en el escenario 2 para la configuración 3 del filtro con reinicio.

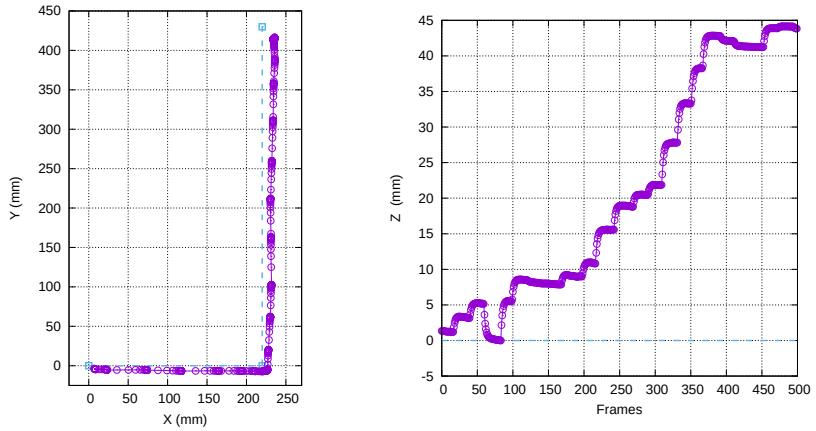
Figura 6.11: Gráficas de la trayectoria del objeto durante los experimentos en el escenario 2 reiniciando el filtro.



(a) Trayectoria en el escenario 2 para la configuración 1 del filtro sin reinicio.



(b) Trayectoria en el escenario 2 para la configuración 2 del filtro sin reinicio.



(c) Trayectoria en el escenario 2 para la configuración 3 del filtro sin reinicio.

Figura 6.12: Gráficas de la trayectoria del objeto durante los experimentos en el escenario 2 sin reiniciar el filtro.

Las gráficas muestran la trayectoria estimada por el filtro para cada una de las tres configuraciones en ambos escenarios de prueba descritos.

Puede observarse que:

1. Dependiendo de la configuración de los valores del filtro Kalman, se obtiene una estimación que se inclina a confiar más en las lecturas de cualquiera de los dos sensores utilizados.

2. Las configuraciones con valores menores de covarianza, para la matriz de covarianza Q, presentan una mejor dispersión de datos a lo largo de las trayectorias.

3. El filtro Kalman permite tener información sobre el estado del sistema a alta frecuencia, esto se traduce en transiciones entre estados más suaves.

4. También se ilustra que el error está directamente relacionado con la posición de la cámara con respecto a los puntos que describen la trayectoria del objeto.

5. Para el primer escenario, los valores del error para la posición en Z nunca sobrepasan los 19 mm, sin embargo, en el segundo escenario pueden llegar a presentarse mediciones incorrectas hasta por 60 mm.

Capítulo 7

Conclusiones

Esta tesis describe las consideraciones necesarias para el diseño y construcción de un sistema de odometría visual e inercial de bajo costo. De igual manera, se tiene en cuenta el peso y dimensiones de cada uno de los componentes del mismo; así como los aspectos relacionados al consumo eléctrico y de potencia de cómputo de los ya antes mencionados. El sistema construido es de bajo costo y de dimensiones reducidas.

Como se detalla en los capítulos anteriores, se explotan las características de la computadora en una sola tarjeta Raspberry Pi 3, que permite acoplar los dos sensores utilizados en el sistema y que, además, cuenta con 4 núcleos para el procesamiento simultáneo de datos. Esto hace posible que el sistema se ejecute en tiempo real a pesar de las limitaciones de hardware.

El sistema en su conjunto es relativamente pequeño y portátil, aunque quizás, las conexiones y cables entre los componentes del mismo son los que podrían afectarlo en maniobrabilidad y rango de operación.

Se utilizó el módulo de cámara oficial de Raspberry por su fácil conexión y compatibilidad con la computadora del mismo fabricante. Por su parte, el sensor inercial elegido es bastante pequeño y puede montarse con facilidad a cualquier superficie por medio de dos tornillos que sujetan la tarjeta del sensor. El tamaño y costo de ambos sensores es considerablemente reducido para las prestaciones que ofrecen; esto los hace fácilmente adaptables a distintas plataformas.

Cabe mencionar que cuando se diseñó el prototipo del sistema no se consideró que la longitud máxima de los cables conectores sugerida por el protocolo I^2C, es de 1 metro. Esto limita de manera importante el rango de operación del sistema.

Para simplificar el emparejamiento de puntos necesario para el cálculo de la homografía por parte del sistema visual, se usó un marcador fiduciario basado en tipo de orden como se detalla en la sección 3.3.2, página 32. En el proceso de detección del marcador en la escena, se considera el procedimiento descrito en la tesis [21], aunque como se menciona en la misma, la cámara de la Raspberry produce cambios de iluminación aún cuando el marcador y fuente de luz no cambian. Esto se traduce en ruido para los datos obtenidos con dicho sistema visual.

La calibración fuera de línea de la cámara es de crucial importancia para aminorar la carga de trabajo del procesador al calcular la homografía, además de mejorar la precisión en el cómputo de ésta. Aunque la homografía pueda calcularse de manera satisfactoria con el modelo de la cámara obscura ideal, la distorsión de la lente afecta de manera significativa al modelo de odometría visual propuesto. Además, la posición de la cámara con respecto a la pose inicial de referencia repercute de manera significativa en el cálculo de la odometría visual.

Del mismo modo, el sensor inercial debe calibrarse para mitigar el ruido en las lecturas. Los valores de calibración dependen de cada sensor en específico y los valores del sesgo estimado pueden variar según la temperatura interna del sensor.

Se utilizó un filtro Kalman para fusionar las mediciones visuales e inerciales. Los valores de covarianza utilizados para el filtro afectan directamente la estimación y transición entre estados del mismo, y por consecuencia, el rango y frecuencia de los movimientos que pueden registrarse. La selección de estos valores dependerá de la aplicación a la que se oriente el sistema.

El sistema funciona de manera satisfactoria para aplicaciones de realidad aumentada o interacciones kinésicas simples, sin embargo, las limitaciones de los sensores y la computadora podrían afectar de manera relevante la alta precisión requerida en tareas de odometría para robots u otros agentes móviles.

Aunque actualmente el sistema es funcional, se reconocen varias mejoras posibles que se listan a continuación.

7.1. Trabajo a futuro

Aquí se enumeran los distintos puntos que se proponen para trabajos a futuro relacionados con el sistema descrito en esta tesis:

1. Incorporar controles de umbralización de imagen para que el proceso de detección del marcador pueda adaptarse más fácilmente a ambientes con diferente iluminación.

2. Aunque el objeto de acrílico se diseño de tal manera que pudieran intercambiarse los marcadores, ésto no se consideró en la herramienta de software desarrollada.

3. Adaptar el sistema para que sea compatible con diferentes tipos de cámaras y sensores inerciales.

4. Usar la información del vector de gravedad para la corrección de la pose inercial. Adicionalmente podría usarse un magnetómetro para estimar el norte terrestre.

5. Considerar la distorsión y aberración causada por el lente en el modelo de la cámara obscura.

6. Replantear el modelo del filtro Kalman para parametrizar el vector de posición angular $\boldsymbol{\theta}$ con un cuaternio $\mathbf{q}$ y evitar singularidades.

7. También se propone un filtro Kalman que considere el sesgo en los sensores como parte de los valores estimados dentro del vector de estado del filtro.

8. Se plantea incorporar el sistema a una plataforma móvil con la computadora y los sensores autocontenidos en un mismo armazón; el sistema computa la odometría del objeto con respecto a un punto en el espacio.

Apéndice A

Notación

Aquí se expone de manera general la notación utilizada en este documento.

- Las matrices se denominan siempre con una letra mayúscula, p.ej. M.
- Los vectores columna se denotan con una letra en negritas, p.ej. $\mathbf{t}$.
- Un punto en 2D se representa con la letra minúscula $\mathbf{p}$, mientras que uno 3D se denota como $\mathbf{P}$.

Algunos operadores utilizados en esta tesis:

$\int$	Integral.
$R(\cdot)$	Matriz de rotación.
K	Matriz de calibración de la cámara.
H	Matriz de homografía.
$\mathbf{t}$	Vector columna de traslación.
$\boldsymbol{\theta}$	Vector columna de posición angular.
$\mathbf{d}$	Vector columna de posición lineal.
$\dot{\mathbf{d}}$	Vector columna de velocidad lineal.
$\dot{\boldsymbol{\theta}}$	Vector columna de velocidad angular en el marco de referencia inercial.
$\dot{\boldsymbol{\omega}}$	Vector columna de velocidad angular en el marco de referencia de la UMI.
$\mathbf{x}$	Vector de estado del sistema.
q	Cuaternio.

Apéndice B

Parámetros obtenidos en la calibración fuera de línea

El conjunto completo de las 40 fotografías a resolución nativa (3280×2464 pixeles) capturadas con la cámara de la Raspberry para la calibración usando un marcador chAruco pueden descargarse en el enlace de internet:
http://computacion.cs.cinvestav.mx/~sherrera/raspiCam_calibSet.tar.gz

Los siguientes tres archivos XML contienen los parámetros obtenidos por el programa de calibración para los tres conjuntos analizados: A, B y A+B.


```xml
<?xml version="1.0"?>
<opencv_storage>
<calibrationDate>"Wed Feb  6 21:12:06 2019"</calibrationDate>
<framesCount>14</framesCount>
<cameraResolution>
  3280 2464</cameraResolution>
<cameraMatrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    2.5088859492258089e+03 0. 1.6098593933721459e+03 0.
    2.5088859492258089e+03 1.2101871296095715e+03 0. 0. 1.</data></cameraMatrix>
<cameraMatrix_std_dev type_id="opencv-matrix">
  <rows>4</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    0. 2.2906405846749529e+01 2.3312365782221470e+01
    2.2873702099162163e+01</data></cameraMatrix_std_dev>
<dist_coeffs type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    1.8765099894418336e-01 -4.9565350314175466e-01
    -7.3613056707817585e-03 -9.8684046953726647e-03
    4.7625815050776293e-01</data></dist_coeffs>
<dist_coeffs_std_dev type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    2.7266710749527016e-02 1.4086071095550792e-01 3.6548034461080991e-03
    3.4285785368681617e-03 2.2394373275200227e-01</data></dist_coeffs_std_dev>
<avg_reprojection_error>1.5073008293479266e+00</avg_reprojection_error>
</opencv_storage>
```


Listado B.1: Archivo de calibración generado con el conjunto A.

```xml
<?xml version="1.0"?>
<opencv_storage>
<calibrationDate>"Wed Feb  6 21:19:57 2019"</calibrationDate>
<framesCount>13</framesCount>
<cameraResolution>
  3280 2464</cameraResolution>
<cameraMatrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    2.3641684706590413e+03 0. 1.6199344680384243e+03 0.
    2.3641684706590413e+03 1.2835190552206891e+03 0. 0. 1.</data></cameraMatrix>
<cameraMatrix_std_dev type_id="opencv-matrix">
  <rows>4</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    0. 4.4416211591194752e+01 2.0137262655410591e+01
    2.3783514969621105e+01</data></cameraMatrix_std_dev>
<dist_coeffs type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    1.9511286138819106e-01 -6.0428543622129482e-01
    2.7707723018091514e-03 -3.7156558890164967e-03
    5.8306244547654118e-01</data></dist_coeffs>
<dist_coeffs_std_dev type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    3.2107953972530495e-02 1.3059753137586513e-01 2.8394091035455547e-03
    2.5214154178364533e-03 1.8910120937064004e-01</data></dist_coeffs_std_dev>
<avg_reprojection_error>4.7167391671757319e-01</avg_reprojection_error>
</opencv_storage>
```


Listado B.2: Archivo de calibración generado con el conjunto B.


```xml
<?xml version="1.0"?>
<opencv_storage>
<calibrationDate>"Wed Feb  6 21:31:17 2019"</calibrationDate>
<framesCount>25</framesCount>
<cameraResolution>
  3280 2464</cameraResolution>
<cameraMatrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    2.5040771423389979e+03 0. 1.6128502125971950e+03 0.
    2.5040771423389979e+03 1.2167776265332961e+03 0. 0. 1.</data></cameraMatrix>
<cameraMatrix_std_dev type_id="opencv-matrix">
  <rows>4</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    0. 1.7305059158569723e+01 1.7742205438186346e+01
    1.6787971611303529e+01</data></cameraMatrix_std_dev>
<dist_coeffs type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    1.9724988796931556e-01 -5.4168923692580073e-01
    -5.8635946581960885e-03 -8.7320628764757362e-03
    5.3624909927373865e-01</data></dist_coeffs>
<dist_coeffs_std_dev type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>d</dt>
  <data>
    2.0389206999651036e-02 1.0427333468108022e-01 2.4968987298009089e-03
    2.4979387484191273e-03 1.6653353358768924e-01</data></dist_coeffs_std_dev>
<avg_reprojection_error>1.1708560774681489e+00</avg_reprojection_error>
</opencv_storage>
```


Listado B.3: Archivo de calibración generado con el conjunto A+B.

Apéndice C

Scripts para simulación del modelo de odometría visual

Los siguientes scripts en lenguaje Python permiten simular el modelo de odometría visual propuesto en la sección 4.1.3.

Para proyectar los puntos del modelo del marcador según el modelo de la cámara oscura, se utiliza el Listado C.1, en la página 83.

Para ejecutar este script es necesario especificar el ángulo a rotar en Z y la traslación en el plano XY de los puntos del modelo del marcador.

```
python proyecta.py angulo dx xy > p1
```

Luego se recuperan estos datos de transformación, mediante el Listado C.2, en la página 84; que calcula la homografía entre los puntos del marcador y los puntos transformados por el programa en el Listado C.1.

```
python prueba.py p1
```

El Listado C.3, es usado por el programa en el Listado C.2.

Un ejemplo de la rotación y traslación del marcador recuperadas por el programa. Se ejecuta:

```
python proyecta.py 30 20 -15 > p1
python prueba.py p1 > s1
```

Y la salida de ambos programas:

```
cat p1
# c^T : [[400   0 170]]
# t^T : [[   0.            0.          -434.62627624]]
453.461230 202.488564
326.890862 279.435193
127.646248 238.465714
225.758205 142.257679
346.569729 174.214150
335.548901 233.141474
281.031715 410.950389

cat s1
453.46123 202.488564
326.890862 279.435193
127.646248 238.465714
225.758205 142.257679
B: [[   5.00000003e-01    8.66025409e-01   -1.50000000e+01]
[  -3.38737735e-01    1.95570322e-01   -7.82281299e+00]
[   7.97029940e-01   -4.60165470e-01   -4.16219661e+02]]

R1: [[   5.00000003e-01    8.66025409e-01    7.39618192e-09]
[  -3.38737735e-01    1.95570322e-01    9.20330916e-01]
[   7.97029940e-01   -4.60165470e-01    3.91140647e-01]]

R: [[   5.00000007e-01    8.66025400e-01    7.39618180e-09]
[  -3.38737738e-01    1.95570320e-01    9.20330916e-01]
[   7.97029948e-01   -4.60165467e-01    3.91140647e-01]]

c: [ 336.58965262 -177.00962325  170.00000397]

[[   5.00000007e-01    8.66025400e-01    7.39618180e-09   -1.50000000e
    +01]
[  -3.38737738e-01    1.95570320e-01    9.20330916e-01   -7.82281299e
    +00]
[   7.97029948e-01   -4.60165467e-01    3.91140647e-01   -4.16219661e
    +02]]

[R|t]:
[[   8.66025401e-01   -5.00000007e-01    7.75527020e-09    1.99999992e
    +01]
[   5.00000007e-01    8.66025400e-01    7.39618180e-09   -1.50000000e
    +01]
[  -1.04143519e-08   -2.52764634e-09    1.00000000e+00   -6.50171684e
    -07]]
Euler angles:
0.0 0.0 30.0000004089
```

```
1 # encoding: utf8
2
3 # Se proyecta el marcador.
4 # se rota en z con el angulo especificado y
5 # se translada segun dx y dy
6 #
7 # La camara esta fija en el punto
8 # c = np.array( [  [400], [0], [170] ] )
9 #
10 # 14.06.2019
11 # 19.06.2019
12 #
13 # Fraga
14
15 import numpy as np
16 import numpy.linalg as la
17 import math
18 import sys
19
20 n = len( sys.argv )
21 if n != 5 :
22   print "Args: theta2 theta1 dx dy"
23   sys.exit(1)
24
25 theta2 = float( sys.argv[1] )
26 theta1 = float( sys.argv[2] )
27 dx = float( sys.argv[3] )
28 dy = float( sys.argv[4] )
29
30
31 K = [ [ 1000.0,    0.0, -320.0], #  640
32       [    0.0, 1000.0, -240.0], #  480
33       [    0.0,    0.0,   -1.0] ]
34
35
36 P = [ [ 50.0,  50.0,  0.0, 1.0 ],
37       [-50.0,  50.0,  0.0, 1.0 ],
38       [-50.0, -50.0,  0.0, 1.0 ],
39       [ 50.0, -50.0,  0.0, 1.0 ],
40       [ 50.0,   0.0,  0.0, 1.0 ],
41       [  0.0,  25.0,  0.0, 1.0 ],
42       [  0.0,   0.0, 80.0, 1.0  ] ]
43
44 # Lo rotamos alrededor de z
45 ang = math.radians( theta1 )
46 R1 = np.eye( 3 )
47 R1[0,0] =  math.cos(ang)
48 R1[0,1] = -math.sin(ang)
49 R1[1,0] =  math.sin(ang)
50 R1[1,1] =  math.cos(ang)
51
52 # Lo rotamos alrededor de y
53 ang = math.radians( theta2 )
54 R2 = np.eye( 3 )
55 R2[0,0] =  math.cos(ang)
56 R2[0,2] =  math.sin(ang)
57 R2[2,0] = -math.sin(ang)
58 R2[2,2] =  math.cos(ang)
59
60 R = np.dot( R2, R1 )
61
62
63 p1 = np.array( [ [0.0], [0.0], [0.0] ] )
64 n = len( P )
65 i = 0
66 while i < n :
67   p1[0][0] = P[i][0]
68   p1[1][0] = P[i][1]
69   p1[2][0] = P[i][2]
70   p1 = np.dot( R, p1 )
71   P[i][0] = p1[0][0]
72   P[i][1] = p1[1][0]
73   P[i][2] = p1[2][0]
74   i += 1
75
76 # Transladamos el cuerpo
77 i = 0
78 while i < n :
79   P[i][0] += dx
80   P[i][1] += dy
81   i += 1
82
83 # Modelo de la camara obscura
84 # lp = K [ R|t ] P, o
85 # lp = K R [ I| -c ] P
86 # viendo al punto O y el vector arriba es a = [0 0 1]'
87 a = np.array( [ [0], [0], [1] ] )
88
89 c = np.array( [  [400], [0], [170] ] )
```

```
90 print "# c^T :", np.transpose( c )
91
92 z = c/la.norm(c)
93
94 # x = a x z
95 xp = np.cross( a, z, axis=0 )
96 x = xp/la.norm(xp)
97
98 y = np.cross( z, x, axis=0 );
99
100 R = []
101 R = x.transpose()
102 R = np.append( R, y.transpose(), axis=0 )
103 R = np.append( R, z.transpose(), axis=0 )
104
105 I = np.eye(3,4)
106 I[:,3] = -c[:,0]
107
108 # M = [ I, -c]
109 # M = K * R * [ I, -c];
110 M = np.dot( R, I )
111 M = np.dot( K, M )
112
113 t = np.dot( R, -c )
114 print "# t^T :", np.transpose( t )
115
116 i = 0
117 while i < 7 :
118   p = P[i]
119   p1 = np.dot( M, np.transpose(p) )
120   u = p1[0]/p1[2]
121   v = p1[1]/p1[2]
122   print( "{0:3.6f} {1:3.6f}".format(u, v) )
123
124   i += 1
```

Listado C.1: Archivo 'proyecta.py'.

```
1 # encoding: utf-8
2 import homografia
3 import numpy as np
4 import numpy.linalg as la
5 import math
6 import sys
7
8 n = len( sys.argv )
9 if n != 2 :
10   print "Args: nombre_archivo_puntos"
11   sys.exit( 1 )
12
13 nombre = sys.argv[1]
14
15 p1 =[[ 50, 50],
16       [-50, 50],
17       [-50,-50],
18       [ 50,-50]]
19
20 p2 = np.zeros( (4,2 ) )
21
22 homografia.lee_archivo_puntos( nombre, p2 )
23
24 H = homografia.homografia( p1, p2 )
25
26 # Comprobacion
27 pp = [ [0.0], [0.0], [0.0] ]
28 i = 0
29 n = len( p1 )
30 while i < n:
31   pp[0][0] = p1[i][0]
32   pp[1][0] = p1[i][1]
33   pp[2][0] = 1.0
34
35   p3 = np.dot( H, pp )
36
37   u = p3[0][0]/p3[2][0]
38   v = p3[1][0]/p3[2][0]
39   print u, v
40
41   i += 1
42
43 K = [[ 1000.0,     0.0,   -320.0], #  640
44       [     0.0, 1000.0,   -240.0], #  480
45       [     0.0,     0.0,     -1.0] ]
46
47 Ki = [[ 1.0/K[0][0],          0.0,  K[0][2]/K[0][0] ], #  640
48        [          0.0, 1.0/K[1][1],  K[1][2]/K[1][1] ], #  480
49        [          0.0,          0.0,      -1.0          ] ]
50
```

```
51 A = np.dot( Ki, H )
52
53 #
54 l1 , l2 , l3 = la.norm( A, axis=0 )
55 l = (l1+l2)/2.0
56
57 #
58 B = A / l
59 print "B:", B
60 print
61 #
62 R1 = np.zeros( (3,3) )
63 R1[:,0] = B[:, 0 ]
64 R1[:,1] = B[:, 1 ]
65 R1[:,2 ] = np.transpose( np.cross( B[:,0], B[:,1] ) )
66 print "R1:", R1
67 print
68 U, D, V = la.svd( R1 )
69
70 R = np.dot( U, V )
71 print "R:", R
72 print
73
74 c = np.dot( -np.transpose(R), B[:,2] )
75 print "c:", c
76 print
77
78 R0 =  [[ 0.,          1.,          0.        ],
79  [-0.39114064,  0.,          0.92033092],
80  [ 0.92033092,  0.,          0.39114064]]
81
82 T0 = [ [0.00000000e+00],
83        [6.51258739e-08],
84       [-4.34626278e+02] ]
85
86 M = np.eye( 3, 4 )
87 M[:,0] = R[:,0]
88 M[:,1] = R[:,1]
89 M[:,2] = R[:,2]
90 M[:,3] = B[:,2]
91 print M
92 print
93
94 M[0][3] = M[0][3] - T0[0][0]
95 M[1][3] = M[1][3] - T0[1][0]
96 M[2][3] = M[2][3] - T0[2][0]
97
98 M1 = np.dot( np.transpose( R0 ), M )
99 print "[R|t]:"
100 print M1
101
102 print "Euler angles:"
103 if M1[2][2] > 0.99999 :
104   print "Singularidad"
105   theta1 = math.atan2 ( M1[1][0], M1[0][0] ) # %* 180.0/pi
106   theta2 = 0.0
107   theta3 = 0.0
108 else :
109   theta1 = math.atan2 ( M1[2][1], -M1[2][0] ) # %* 180.0/pi
110   theta2 = math.acos( M1[2][2] )      # %* 180.0/pi
111   theta3 = math.atan2 ( M1[1][2], M1[0][2] )  # %* 180.0/pi
112
113 theta1 = math.degrees( theta1 )
114 theta2 = math.degrees( theta2 )
115 theta3 = math.degrees( theta3 )
116 print theta3, theta2, theta1
```

Listado C.2: Archivo 'prueba.py'.

```
1 # encoding: utf8
2 import math
3 import numpy as np
4 import numpy.linalg as la
5
6 def mediaVarianza( M ) :
7
8   rows = len( M )
9   cols = len( M[0] )
10
11   media = [0.0, 0.0]
12   varianza = [0.0, 0.0]
13   i = 0
14   while i < cols :
15     # Calcumos medias y sigmas de los puntos p1
16     med = 0.0
17     sig = 0.0
18     j = 0
19     while j<rows :
```

```
      if  cols == 1 :
        v = M[j]
      else :
        v = M[j][i]

      med += v
      sig += v*v
      j += 1

    media[i] = med/rows
    varianza[i] = sig/rows - media[i]*media[i]
    i += 1

  return ( media, varianza )

# Calcula la homografia de dos conjuntos de puntos en 2D
#
def homografia( p1, p2 ) :
  # Calcumos medias y sigmas de los puntos p1 y p2
  med1, sig1 = mediaVarianza( p1 )
  med2, sig2 = mediaVarianza( p2 )

  sig1[0] = math.sqrt(sig1[0])
  sig1[1] = math.sqrt(sig1[1])
  sig2[0] = math.sqrt(sig2[0])
  sig2[1] = math.sqrt(sig2[1])

  n = len( p1 )
  # print n
  A = np.zeros( (2*n, 9) )
  i = 0
  while i < n :
    u1 = ( p1[i][0] - med1[0] ) / sig1[0]
    v1 = ( p1[i][1] - med1[1] ) / sig1[1]

    u2 = ( p2[i][0] - med2[0] ) / sig2[0]
    v2 = ( p2[i][1] - med2[1] ) / sig2[1]

    k = 2*i
    A[ k ][ 0 ] = u1
    A[ k ][ 1 ] = v1
    A[ k ][ 2 ] = 1.0
    A[ k ][ 3 ] = 0.0
    A[ k ][ 4 ] = 0.0
    A[ k ][ 5 ] = 0.0
    A[ k ][ 6 ] = - u1 * u2
    A[ k ][ 7 ] = - v1 * u2
    A[ k ][ 8 ] = u2

    k = 2*i + 1
    A[ k ][ 0 ] = 0.0
    A[ k ][ 1 ] = 0.0
    A[ k ][ 2 ] = 0.0
    A[ k ][ 3 ] = u1
    A[ k ][ 4 ] = v1
    A[ k ][ 5 ] = 1.0
    A[ k ][ 6 ] = - u1 * v2
    A[ k ][ 7 ] = - v1 * v2
    A[ k ][ 8 ] = v2

    i += 1

  # Ax = b
  # QR x = b
  # (QR)^{-1} QR = (QR)^{-1} b
  # R^{-1} Q^T Q R = R^{-1} Q^T b
  # R = Q^T b
  # A = QR
  # A^{-1} = (QR)^{-1} = R^{-1} Q^T
  R = la.qr( A, 'r' )
  # print R

    # Retrosustitucion
  vh = np.zeros( 8 )

  vh[7] = R[7][8]/R[7][7]

  i = 6
  while i>=0 :
    suma = 0.0;
    j = i + 1
    while j<8 :
      suma += R[i][j] * vh[j]
      j += 1

    vh[i] = ( R[i][8] - suma )/R[i][i]
    i -= 1

  Hp = [ [ vh[0], vh[1], vh[2] ],
```

```
109          [ vh[3], vh[4], vh[5] ],
110          [ vh[6], vh[7],  1.0 ] ]
111
112   T1 = [ [ 1.0/sig1[0],       0.0    , -med1[0]/sig1[0] ],
113          [     0.0      , 1.0/sig1[1], -med1[1]/sig1[1] ],
114          [     0.0      ,       0.0    ,         1.0       ] ]
115
116   T2i = [ [ sig2[0],       0    ,  med2[0]    ],
117           [    0     ,    sig2[1],  med2[1]    ],
118           [    0     ,       0    ,      1       ] ]
119
120   # l p2 = H p1
121   # l T2 p2 = H T1 p1
122
123   H = np.dot( np.dot( T2i, Hp ), T1 )
124
125   return H
126
127 def lee_archivo_puntos( nombre, M ) :
128   try :
129     arch = open( nombre, "r" )
130   except:
131     print "No se puede abrir el archivo", nombre
132     exit(2)
133
134   i = 0
135   while 1 :
136     linea = arch.readline( )
137     if linea == "" :
138       break
139
140     if linea[0] == "#" :
141       continue
142
143     if i == 4 :
144       break
145
146     d = linea.split( )
147     n = len( d )
148     if n == 2 :
149       M[i][0] = float( d[0] )
150       M[i][1] = float( d[1] )
151
152     i += 1
```

Listado C.3: Archivo 'homografia.py'.

Apéndice D

Experimentos para determinar la varianza del sistema

Enseguida se describen los experimentos realizados para determinar los valores de varianza del sistema visual.

Los experimentos fueron realizados en una mesa paralela al suelo colocando la cámara de manera perpendicular al plano con la ayuda de un tripié, como se muestra en la figura D.1.

El marcador se hizo colocó frente a la cámara a dos diferentes distancias, de 30 y 60 centímetros y se hizo variar el ángulo de inclinación para cada uno de los ejes, tomando 100 muestras para cada posición. Para las muestras recopiladas en el experimento, se calculó la varianza y desviación estándar para cada uno de los valores de pose extraídos a partir del cálculo de la homografía.

Figura D.1: Disposición de la cámara y el marcador durante los experimentos de la varianza del sistema visual.

Lo que se muestra en la siguiente tabla son los promedios de los valores obtenidos para los vectores $[\boldsymbol{\theta}^T, \mathbf{t}^T]^T$ de pose calculados en cada experimento. Los experimentos se enumeran en la tabla siguiente D.1.

Tabla D.1: Listado de experimentos con las distintas poses del marcador con respecto a la cámara.

Distancias: 30 o 60 cm			
Eje X	Eje Y	Eje Z	→ Eje del marcador a rotar
0º	0º	0º	→ Esta fila es un solo experimento
15º	15º	45º	
30º	30º	90º	
60º	60º	135º	

Las siguientes cuatro tablas D.2, D.3, D.4 y D.5 muestran los resultados de los experimentos.

Tabla D.2: Valores calculados a partir de $\boldsymbol{\theta}$ con el marcador a 30 cm de la cámara.

$\boldsymbol{\theta}$ con el marcador a 30 cm							
Angulo X/Y	Eje X		Eje Y		Eje Z		Angulo Z
	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	
0º	0.103507	0.282443	0.103507	0.282443	0.103507	0.282443	0º
15º	0.0463994	0.190867	0.0565918	0.211034	0.334081	0.49133	45º
30º	0.027909	0.163702	0.0100239	0.0906983	0.641676	0.681123	90º
60º	0.314961	0.498774	0.012735	0.0946974	0.188824	0.368576	135º

Tabla D.3: Valores calculados a partir de $\mathbf{t}$ con el marcador a 30 cm de la cámara.

$\mathbf{t}$ con el marcador a 30 cm							
Angulo X/Y	Eje X		Eje Y		Eje Z		Angulo Z
	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	
0º	0.0218557	0.117428	0.0218557	0.117428	0.0218557	0.117428	0º
15º	0.028674	0.132134	0.0194762	0.128341	0.0146787	0.0852555	45º
30º	0.531574	0.513576	0.0162044	0.0911696	0.0285002	0.138193	90º
60º	4.1456	1.42421	0.0741579	0.18821	0.00601849	0.0674882	135º

Tabla D.4: Valores calculados a partir de $\boldsymbol{\theta}$ con el marcador a 60 cm de la cámara.

$\boldsymbol{\theta}$ con el marcador a 60 cm							
Angulo X/Y	Eje X		Eje Y		Eje Z		Angulo Z
	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	
0º	1.69391	1.07997	1.69391	1.07997	1.69391	1.07997	0º
15º	0.0978365	0.302318	0.930105	0.884257	1.74075	1.1255	45º
30º	0.6173032	0.791724	0.785123	0.767962	0.588123	0.635178	90º
60º	2.24842	1.43139	0.013272	0.108492	0.377648	0.737152	135º

Tabla D.5: Valores calculados a partir de $\mathbf{t}$ con el marcador a 60 cm de la cámara.

$\mathbf{t}$ con el marcador a 60 cm							
Angulo X/Y	Eje X		Eje Y		Eje Z		Angulo Z
	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	$\sigma^2_{\boldsymbol{\theta}}$	$\sigma_{\boldsymbol{\theta}}$	
0º	0.249311	0.341556	0.249311	0.341556	0.249311	0.341556	0º
15º	0.740484	0.574939	1.84131	0.873785	0.242756	0.381652	45º
30º	1.633481	1.243561	0.47575	0.466275	0.113494	0.219352	90º
60º	12.0182	2.29656	4.40862	1.36023	0.024265	0.114512	135º

Bibliografía

[1] Haiyang Chao, Calvin Coopmans, Long Di, and YangQuan Chen. A comparative evaluation of low-cost IMUs for unmanned autonomous systems. *2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Sept 2010.

[2] The Qt Company. Qt web page. https://www.qt.io/. Acceso: 22/Ago/2019.

[3] Peter Corke and Jorge Lobo. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26(6):519–535, 2007.

[4] Heriberto Cruz-Hernández and Luis Gerardo de la Fraga. A fiducial tag invariant to rotation, translation, and perspective transformations. *Pattern Recognition*, 81:213–223, 2018.

[5] Luis Gerardo de la Fraga. Una biblioteca para procesamiento de imagen: scimagen. http://cs.cinvestav.mx/~fraga/Softwarelibre/Scimagen.tar.gz. Acceso: 02/Ago/2019.

[6] Luis Gerardo de la Fraga and Heriberto Cruz Hernandez. Point set matching with order type. In José Francisco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa, José Arturo Olvera-López, and Sudeep Sarkar, editors, *Pattern Recognition*, pages 229–237, Cham, 2018. Springer International Publishing.

[7] David Diel. Stochastic constraints for vision-aided inertial navigation. Master's thesis, Massachusetts Institute of Technology, Enero 2005.

[8] Ernesto Olguín Díaz. *3D Motion of Rigid Bodies: A Foundation for Robot Dynamics Analysis*. Studies in Systems, Decision and Control. Springer International Publishing, 2018.

[9] RaspberryPi Foundation. Raspberry pi 3 model b. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Acceso: 02/Oct/2019.

[10] RaspberryPi Foundation. Raspberry pi camera module v2. https://www.raspberrypi.org/products/camera-module-v2/. Acceso: 02/Oct/2019.

[11] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco Madrid-Cuevas, and Manuel Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47:2280–2292, 06 2014.

[12] Marco Grimm and Rolf-Rainer Grigat. Real-time hybrid pose estimation from vision and inertial data. In *Proceedings of the First Canadian Conference on Computer and Robot Vision*, May 2004.

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2nd edition, 2003.

[14] Heriberto Cruz Hernández and Luis Gerardo de la Fraga. Order type dataset analysis for fiducial markers. *Data in Brief*, 20:1068–1072, 2018.

[15] InvenSense Inc. *MPU-6000/MPU-6050 Product Specification*, 08 2013. Rev. 3.4.

[16] LG. Lg scanner mouse lsm-100. https://www.lg.com/uk/computer-accessories/lg-LSM-100-scanner-mouse. Acceso: 16/Ago/2019.

[17] Mingyang Li and Anastasios I. Mourikis. Vision-aided inertial navigation for resource-constrained systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012.

[18] Mingyang Li and Anastasios I. Mourikis. Vision-aided inertial navigation with rolling-shutter cameras. *The International Journal of Robotics Research*, 33(11):1490–1507, 2014.

[19] Michael Neunert, Michael Blösch, and Jonas Buchli. An open source, fiducial based, visual-inertial motion capture system. *2016 19th International Conference on Information Fusion (FUSION)*.

[20] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 26(6):652–659, 2004.

[21] Daybelis Jaramillo Olivares. Prototipo de un sistema de ojo de halcón. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Ago 2018.

[22] OpenCV. Interactive camera calibration application. https://docs.opencv.org/master/d7/d21/tutorial_interactive_calibration.html. Acceso: 15/Feb/2019.

[23] Jeff Rowberg. I2C device library. https://www.i2cdevlib.com/. Acceso: 02/Ago/2019.

[24] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics and Automation Magazine*, 18(4):80–92, 2011.

[25] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, 19(2):78–90, 2012.

[26] André Wegener Sleeswyk. Vitruvius' odometer. *Scientific American*, 245(4):188–201, 1981.

[27] Joan Solà. Quaternion kinematics for the error-state Kalman filter. working paper or preprint, March 2015.

[28] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[29] Cédric Verstraeten. Raspicam: C++ API for using raspberry camera (with opencv). https://github.com/cedricve/raspicam. Acceso: 02/Ago/2019.

[30] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.