



SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación
y Desarrollo Tecnológico

Tesis de Maestría

Sistema de Navegación Inercial Asistido por Visión para
Robots Móviles Terrestres

presentada por

Ing. Larisa Navarrete Olmedo

como requisito para la obtención del grado de
Maestra en Ciencias de la Computación

Director de tesis

Dr. José Ruiz Ascencio

Codirector de tesis

Dra. Andrea Magadán Salazar

Cuernavaca, Morelos, México. enero de 2019.

Cuernavaca, Morelos a 06 de diciembre del 2018
OFICIO No. DCC/245/2018

Asunto: Aceptación de documento de tesis

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial de la **Ing. Larisa Navarrete Olmedo**, con número de control M16CE011, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado **"Sistema de navegación inercial asistido por visión para robots móviles terrestres"** y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS



Dr. José Ruiz Ascencio
Doctor en Ciencias
5009035

CO-DIRECTORA DE TESIS



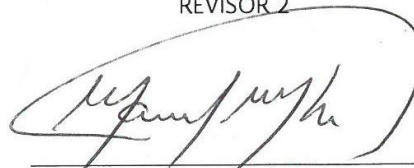
Dra. Andrea Magadán Salazar
Doctorado en Ciencias
Computacionales
10654097

REVISOR 1



Dr. Raúl Pinto Elías
Doctor en Ciencias en la Especialidad de
Ingeniería Eléctrica
3890453

REVISOR 2



Dr. Manuel Mejía Lavalle
Doctor en Ciencias Computacionales
8342472

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/Imz



SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MEXICO

Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 11 de diciembre de 2018
OFICIO No. SAC/569/2018

Asunto: Autorización de impresión de tesis

ING. LARISA NAVARRETE OLMEDO
CANDIDATA AL GRADO DE MAESTRA EN CIENCIAS
DE LA COMPUTACIÓN
PRESENTE

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado "Sistema de Navegación Inercial Asistido por Visión para Robots Móviles Terrestres", ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

ATENTAMENTE
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

SEP TecNM
CENTRO NACIONAL
DE INVESTIGACIÓN
Y DESARROLLO
TECNOLÓGICO
SUBDIRECCIÓN
ACADÉMICA

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

cenidet[®]
Centro Nacional de Investigación
y Desarrollo Tecnológico

Interior Internado Palmira S/N, Col. Palmira, C. P. 62490, Cuernavaca, Morelos.
Tel. (01) 777 3 62 77 70, ext. 4106, e-mail: dir_cenidet@tecnm.mx

www.tecnm.mx | www.cenidet.edu.mx



Dedicatoria

A dios

*Por haberme dado salud para lograr mis objetivos y permitirme
culminar una meta más.*

A mis padres

Por su amor, apoyo incondicional y motivación constante.

A mis hermanos y sobrinos

Por su amor, alegría y confianza otorgada día a día.

A ti amor

*Por apoyarme, motivarme y no dejarme vencer en los peores
momentos.*

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (**CONACYT**) por el apoyo económico otorgado para realizar mis estudios de maestría.

Al Centro Nacional de Investigación y Desarrollo Tecnológico (**CENIDET**) por permitirme ser parte de esta gran institución y brindarme la oportunidad de realizar mis estudios de maestría.

A mi director de tesis, el Dr. José Ruiz Ascencio por su asesoría, orientación y apoyo brindado durante el desarrollo de esta tesis. En especial por su confianza, consejos y paciencia.

A mi codirectora, la Dra. Andrea Magadán por su orientación y atención a mis consultas, así como su paciencia durante la escritura de esta tesis.

A mis revisores de tesis, Dr. Manuel Mejía y Dr. Raúl Pinto por sus comentarios, observaciones y correcciones realizadas para la mejora de esta tesis.

A mis amigos José Luis, Keila y Nicolas por su amistad y apoyo incondicional.

Resumen

El sistema de navegación inercial o INS por sus siglas en inglés (inertial navigation system) brinda datos precisos de posición, velocidad y dirección angular a corto plazo. Sin embargo, debido a la doble integración de las aceleraciones para obtener la posición del móvil, incluso pequeños errores cometidos se van acumulando y, por lo tanto, la incertidumbre crece proporcionalmente al tiempo y al espacio recorrido. Por tal motivo, el INS requiere de algún método que reduzca continuamente dicha incertidumbre.

Esta tesis presenta un estudio y la implementación de técnicas de odometría inercial con la unidad de medición inercial (IMU por sus siglas en inglés), con el objetivo de fusionar sus mediciones con datos visuales y con esto lograr reducir la incertidumbre en la estimación de la posición de un robot móvil obtenida por el INS.

Como resultado de este trabajo se deriva una heurística capaz de controlar las integraciones de los componentes x y y del INS, mediante la detección del estado (“movimiento” o “reposo”) en ambos ejes por medio de datos visuales. Los experimentos se realizaron en entornos con condiciones controladas, registrando un error menor a los 0.08 m en trayectorias de longitud total de 1.1 m.

Abstract

The inertial navigation system or INS provides accurate position, speed and angular direction data in the short term. However, due to the double integration of the accelerations to obtain the position of the mobile, small errors are accumulated and therefore, the uncertainty grows proportionally to time and space traveled. For this reason, the INS requires some method that continuously reduces this uncertainty.

This thesis presents a study and implementation of inertial odometry techniques with inertial measurement unit (IMU), with the goal of merging their measurements with visual data and thereby reduce the uncertainty in the estimation of the position of a mobile robot obtained by the INS.

As a result of this work, a heuristic capable of controlling the integrations of the x and y components of the INS is derived, through the detection of the state (“movement” or “rest”) in both axes by means of visual data. The experiments were conducted in environments with controlled conditions, recording an error lower than 0.08 m in a trajectory of 1.1 m total length.

Índice general

Resumen	i
Abstract	ii
Índice de Figuras	vi
Índice de Tablas	ix
Glosario de Términos	x
Capítulo 1 Introducción	1
1.1 Introducción a la navegación inercial asistida por visión	1
1.2 Antecedentes	2
1.3 Descripción del problema	3
1.4 Objetivo general.....	3
1.5 Objetivos específicos	3
1.6 Metodología de solución.....	4
1.6.1 Prototipo construido	4
1.6.2 Sistema desarrollado.....	5
1.7 Organización de la tesis	5
Capítulo 2 Contexto teórico	6
2.1 Navegación inercial	6
2.1.1 Giroscopios MEMS	8
2.1.2 Acelerómetros MEMS	9
2.1.3 Unidad de Medición Inercial (IMU).....	11
2.2 Detección de movimiento por flujo óptico	12
2.3 Fusión de datos inerciales con visuales	14
Capítulo 3 Estado del arte	17
3.1 Sistemas que fusionan los datos de la IMU	17
3.2 Sistemas que fusionan los datos de la IMU con datos GPS.....	18
3.3 Sistemas que fusionan los datos de la IMU con datos visuales	20
3.4 Discusión	31
Capítulo 4 Diseño y construcción del prototipo	33
4.1 Diseño del prototipo.....	33

4.2	Componentes principales.....	34
4.2.1	Raspberry Pi 3	34
4.2.2	IMU Adafruit 9 DoF.....	35
4.2.3	Arduino DUE.....	36
4.2.4	Cámara Pi v2	37
4.3	Conexiones y software empleado	38
4.3.1	Conexión entre la IMU Adafruit 9 DoF y Arduino DUE.....	38
4.3.2	Conexión entre Cámara Pi v2 y Raspberry Pi 3	39
4.3.3	Conexión entre Arduino DUE y Raspberry Pi 3	40
4.3.4	Conexión entre la computadora y la Raspberry Pi 3	40
4.4	Carcasa.....	41
4.5	Construcción	42
4.6	Discusión	43
Capítulo 5 Desarrollo del sistema		44
5.1	Arquitectura del sistema	44
5.2	Módulo inercial.....	45
5.2.1	Calibración	47
5.2.2	Filtro paso bajo	48
5.2.3	Pose inicial.....	49
5.2.4	Integración de la velocidad angular.....	50
5.2.5	Resta de la gravedad.....	51
5.2.6	Doble integración de la aceleración lineal.....	51
5.3	Módulo visual	52
5.3.1	Adquisición de la imagen	53
5.3.2	Regiones de interés (ROI)	54
5.3.3	Flujo óptico.....	55
5.3.4	Detección del estado.....	56
5.3.5	Detección de la dirección	57
5.4	Módulo fusión de datos.....	59
5.5	Interfaz gráfica.....	61
5.6	Discusión	63

Capítulo 6 Experimentación y resultados	64
6.1 Descripción general de los experimentos	64
6.1.1 Escenario utilizado para los experimentos ejecutados de forma manual	65
6.1.2 Escenario utilizado para los experimentos ejecutados sobre un carro de control 66	
6.2 Módulo inercial.....	67
6.3 Módulo visual	72
6.4 Módulo fusión de datos.....	73
6.4.1 Experimento manual número 1.....	74
6.4.2 Experimento manual número 2.....	75
6.4.3 Experimento manual número 3.....	77
6.4.4 Experimento a control remoto número 1	78
6.4.5 Experimento a control remoto número 2.....	79
6.4.6 Experimento a control remoto número 3.....	80
6.5 Comparación	83
6.6 Discusión	84
Capítulo 7 Conclusiones y trabajos futuros	85
7.1 Conclusiones	85
7.2 Objetivos alcanzados	86
7.3 Productos	87
7.4 Aportaciones	87
7.5 Trabajos futuros	88
Referencias	89
Anexos.....	94
1. Instalación de la biblioteca Raspicam	94
2. Instalación del módulo libqt5serialport5	94
3. Instalación y configuración de TightVNC.....	95
4. Experimentación del módulo inercial.....	96
5. Experimentación del módulo visual	102

Índice de Figuras

Figura 1.1. Esquema general del sistema de navegación inercial asistido por visión.	4
Figura 1.2. Arquitectura del prototipo construido.	5
Figura 1.3. Organización del software desarrollado.	5
Figura 2.1. Chip ASIC integrado por un acelerómetro MEMS triaxial (Santana, Van Den Hoven, et al. (2012)).	7
Figura 2.2. Sistema de Navegación Inercial (INS).	7
Figura 2.3. Estructura del giroscopio MEMS (Zheng et al. (2016)).	8
Figura 2.4. Sistema de coordenadas de un giroscopio de tres ejes.	9
Figura 2.5. Estructura del acelerómetro MEMS (Andrejašič (2008)).	10
Figura 2.6. Sistema de coordenadas de un acelerómetro de tres ejes.	10
Figura 2.7. Sistema de coordenadas de una IMU 6 DoF.	11
Figura 2.8. Estimación del desplazamiento entre las imágenes I y J (Tomažič & Škrjanc (2015)).	13
Figura 2.9 Fusión de datos de los sistemas sensoriales del cuerpo humano (Fuentes varias (2018)).	14
Figura 3.1. Experimento núm. 4 (Ibrahim & Moselhi (2016)). a) ruta utilizada para el experimento. b) ruta estimada por el sistema desarrollado.	18
Figura 3.2. Algoritmo de detección de orientación y posición (Elarabi & Suprem (2015)).	18
Figura 3.3. Resultados obtenidos por la GUI desarrollada (Suprem et al. (2017)). a) datos IMU. b) datos IMU corregidos. c) fusión GPS/IMU.	19
Figura 3.4. Experimentación (Suwandi et al. (2017)). a) ruta recta, donde la línea roja es la ruta predefinida y la línea amarilla son posiciones GPS. b) Resultados obtenidos por los sistemas GPS, IMU y la fusión de datos.	19
Figura 3.5. SLAM aerotransportado (Kim & Sukkarieh (2007)). a) Algoritmo directo. b) algoritmo indirecto.	20
Figura 3.6. Vistas detalladas de SLAM en tiempo real durante ciclos sucesivos del vuelo en trayectoria ovalada (Kim & Sukkarieh (2007)).	21
Figura 3.7. Resultados (Karam et al. (2010)). a) Robucat. b) trayectorias generadas por los enfoques con corrección de factor de escala local (línea azul) y global (línea verde) en comparación con la trayectoria obtenida por RTK-GPS (línea roja).	22
Figura 3.8. Descripción general del sistema de navegación (Barrett et al. (2013)).	23
Figura 3.9. Resultados de simulación del algoritmo (Vincenzo et al. (2013)). a) guiñada. b) cabeceo. c) alabeo.	23
Figura 3.10. Sistema GA-ScaViSLAM (Babu et al. (2014)). a) Dispositivo de mano integrado por una Bumblebee XB3 (A), IMU NavChip (B) y una NUC (C). b) Diagrama del flujo de datos del sistema.	24

Figura 3.11. Experimento C, con datos <i>ground truth</i> en rojo y datos generados en azul (Babu et al. (2014)). a) camino recorrido. b) ScaViSLAM. c) Ga-ScaViSLAM.....	25
Figura 3.12. Resultados del sistema (Y. Zhang et al. (2014)). a) posicionamiento a corta distancia. b) posicionamiento a larga distancia.	25
Figura 3.13. Ubicaciones reales del pez robótico en el secuestro (J. Zhang et al. (2014)). a) ubicación antes del secuestro. b) ubicación después del secuestro.	26
Figura 3.14. Estimación de la ubicación del pez robótico en el secuestro (J. Zhang et al. (2014)). a) estimación de coordenadas x . b) estimación de coordenadas y	26
Figura 3.15. Trayectoria peatonal obtenida mediante el sistema de localización (Tomažič & Škrjanc (2015)).	27
Figura 3.16. Resultados de simulación (L. Zhang et al. (2016)). a) pista teórica. b) valores calculados mediante la combinación de IMU con flujo óptico.	28
Figura 3.17. Estructura del sistema de navegación OFS/SINS (X. Liu et al. (2016)).	28
Figura 4.1. Raspberry Pi 3 modelo B (Raspberry Pi Org. (2018)b).	34
Figura 4.2. IMU Adafruit 9 DoF (Adafruit Industries (2018)).	35
Figura 4.3. Arduino DUE (Arduino Org (2018)a).	36
Figura 4.4. Cámara Pi v2 (Raspberry Pi Org. (2018)a).	37
Figura 4.5. Esquema de conexión entre la IMU Adafruit 9 DoF y Arduino DUE.	38
Figura 4.6. Conexión de la Cámara Pi v2 a la Raspberry Pi 3.	39
Figura 4.7. Esquema de conexión entre el Arduino DUE y la Raspberry Pi 3.	40
Figura 4.8. Diseño de la carcasa del prototipo.	41
Figura 4.9. Prototipo construido.	42
Figura 4.10. Ubicaciones de los sensores del prototipo. a) IMU 9 DoF. b) Cámara Pi v2.	42
Figura 5.1. Sistema de navegación inercial asistido por visión.	45
Figura 5.2. Sistema de coordenadas (x, y, z) del módulo inercial.	46
Figura 5.3. Algoritmo de navegación inercial (Woodman (2007)).	46
Figura 5.4. Resultado de la calibración del acelerómetro.	48
Figura 5.5. Filtrado de las aceleraciones lineales.	49
Figura 5.6. Sistema de coordenadas (x, y, z) del módulo visual.	52
Figura 5.7 Regiones de interés de la imagen.	54
Figura 5.8. Comportamiento del flujo óptico ante diferentes movimientos.	57
Figura 5.9. Sistema de referencia (x, y, z) del prototipo.	59
Figura 5.10. Interfaz gráfica del sistema de navegación inercial asistido por visión.	62
Figura 5.11. Diagrama de flujo del sistema de navegación inercial asistido por visión.	63
Figura 6.1. Experimentos ejecutados.	65
Figura 6.2. Escenario utilizado para los experimentos hechos de forma manual.	65
Figura 6.3. Escenario utilizado para los experimentos hechos sobre el carro de control.	66
Figura 6.4. Información contenida en el archivo creado por el sistema desarrollado.	66
Figura 6.5. Trayectorias obtenidas por el sistema. a) Prueba 4.5 registrada como la peor estimación. b) Prueba 3.10 registrada como la mejor estimación.	69
Figura 6.6. Gráficas de desplazamiento vs tiempo de las pruebas 4.5 y 3.10.	69

Figura 6.7. Trayectorias obtenidas por el sistema. a) Prueba 11.3 registrada como la peor estimación. b) Prueba 2.9 registrada como la mejor estimación.	70
Figura 6.8. Gráficas de desplazamiento vs tiempo de las pruebas 11.3 y 2.9.	70
Figura 6.9. Trayectorias obtenidas por el sistema. a) Prueba 6.4 registrada como la peor estimación. b) Prueba 7.4 registrada como la mejor estimación.	71
Figura 6.10. Trayectorias obtenidas por el sistema. a) Prueba 6.4. b) Prueba 7.4.....	71
Figura 6.11. Algoritmo de detección del movimiento ante diferentes velocidades. a) 0.18 m/s. b) 0.40 m/s. c) 0.69 m/s.	73
Figura 6.12. Algoritmo de detección de movimiento afectado por la pérdida de puntos característicos debido a la altura de la cámara y a la alta velocidad.	73
Figura 6.13. Resultado del experimento manual núm. 1.	74
Figura 6.14. Gráfica de desplazamiento vs tiempo del experimento manual núm. 1.....	75
Figura 6.15. Resultado del experimento manual núm. 2.	76
Figura 6.16. Gráfica de desplazamiento vs tiempo del experimento manual núm. 2.....	76
Figura 6.17. Resultado del experimento manual núm. 3.	77
Figura 6.18. Gráfica de desplazamiento vs tiempo del experimento manual núm. 3.....	78
Figura 6.19. Resultado del experimento a control remoto núm. 1.	79
Figura 6.20. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 1.	79
Figura 6.21. Resultado del experimento a control remoto núm. 2.	80
Figura 6.22. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 2.	80
Figura 6.23. Resultado del experimento a control remoto núm. 3.	81
Figura 6.24. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 3.	81
Figura 6.25. Resultados del algoritmo de detección de movimientos en una dimensión durante recorridos ejecutados por el carrito de control remoto. a) reposo. b) atrás. c) adelante. d) derecha. e) izquierda.	82
Figura 6.26. Resultados del algoritmo de detección de movimientos en dos dimensiones durante recorridos ejecutados por el carrito de control remoto. a) adelante/izquierda. b) adelante/derecha.	82
Figura 6.27. Resultados del algoritmo de detección de movimientos ante pérdidas de puntos característicos durante recorridos ejecutados por el carrito de control remoto.	82
Figura 6.28. Resultados obtenidos por INS (línea amarilla) e INS/Visión (línea turquesa) a partir de la trayectoria predefinida (línea negra). a) trayectoria del recorrido. b) trayectoria del recorrido con acercamiento para mejorar su visualización.....	83
Figura 6.29. Gráfica de desplazamiento vs tiempo de las mediciones obtenidas por INS e INS/Visión.	84

Índice de Tablas

Tabla 3.1. Tabla comparativa del estado del arte.	29
Tabla 4.1. Especificaciones técnicas de la Raspberry Pi 3.	35
Tabla 4.2. Especificaciones técnicas de la IMU Adafruit 9 DoF.	36
Tabla 4.3. Especificaciones técnicas del Arduino DUE.	37
Tabla 4.4. Especificaciones técnicas de la Cámara Pi v2.	38
Tabla 4.5. Especificaciones de conexión entre la IMU Adafruit 9 DoF y Arduino DUE.	39
Tabla 5.1. Estados de los ejes z y x del sistema de visión detectados por el módulo de visión.	59
Tabla 5.2. Acoplamiento de datos inerciales con visuales.	60
Tabla 6.1. Experimentos del módulo inercial.	67
Tabla 6.2. Resultados de los experimentos del módulo inercial.	68
Tabla 6.3. Experimentos de detección de reposo y movimiento.	72
Tabla 6.4. Resultados de los experimentos de detección de reposo y movimiento.	72
Tabla 6.5. Mediciones predefinidas y estimadas del experimento manual núm. 1.	74
Tabla 6.6. Mediciones predefinidas y estimadas del experimento manual núm. 2.	75
Tabla 6.7. Mediciones predefinidas y estimadas del experimento manual núm. 3.	77
Tabla 6.8. Trayectoria predefinida y estimada del experimento a control remoto número 1.	78
Tabla 6.9. Trayectoria predefinida y estimada del experimento a control remoto número 2.	79
Tabla 6.10. Trayectoria predefinida y estimada del experimento a control remoto número 3.	80
Tabla 6.11. Comparación entre los resultados obtenidos por INS e INS_Visión.	84
Tabla 7.1. Comentarios de los objetivos específicos.	86

Glosario de Términos

AROCCAM

Software desarrollado para diseñar e implementar aplicaciones de fusión de datos. El software gestiona los sensores no sincronizados y las observaciones diferidas de una manera que permite al usuario fusionar la información, teniendo en cuenta la fecha de percepción del entorno (Tessier, Cariou, et al. (2006)).

Corriente Alterna (CA)

Corriente eléctrica donde los electrones libres se mueven oscilando con la misma amplitud en ambos sentidos. La corriente alterna fluye con una variación continua de sentido e intensidad (Bastian, Eichler, et al. (2001)).

Corriente Continua (CC)

Corriente eléctrica donde el número de electrones que se mueve en un segundo en una misma dirección permanece inalterable. La corriente continua fluye en un solo sentido y con intensidad constante (Bastian et al. (2001)).

Cross-correlation

Correlación cruzada, es la similitud entre las observaciones como una función del intervalo de tiempo entre ellas. Es una herramienta matemática para encontrar patrones que se repiten, como la presencia de una señal periódica oscurecida por el ruido, o la identificación de la frecuencia fundamental que falta en una señal implícita en sus frecuencias armónicas. A menudo se usa en el procesamiento de señales para analizar funciones o series de valores, como señales de dominio de tiempo (Kale & Limaye (2014)).

Brújula digital

Componente que determina la orientación absoluta del dispositivo (o usuario) de la mejor manera posible según el sistema de coordenadas del mundo. La brújula digital requiere esencialmente de dos sensores para funcionar: el acelerómetro y el magnetómetro (Tomažič & Škrjanc (2015)).

Direction Cosine Matrix (DCM)

Matriz de cosenos directores también llamada matriz de rotación, es una matriz ortogonal, que desempeña un papel fundamental en la representación de la actitud (orientación respecto al sistema de referencia de navegación) de la aeronave utilizada principalmente para calcular los ángulos de Euler, a través de las mediciones de los giroscopios, acelerómetros, magnetómetros y/o GPS (Wang, Azaizia, et al. (2018)).

Degrees of Freedom (DoF)

Grados de libertad es el número de variables de posición independientes que deberían especificarse para ubicar todas las partes del mecanismo (Craig (2004)).

Extended Kalman Filter (EKF)

Algoritmo recursivo muy utilizado en sistemas no lineales, donde la distribución de estado se propaga analíticamente a través de la linealización de primer orden, debido a que la media posterior y la covarianza podrían estar dañadas (Vincenzo, Rosario, et al. (2013)).

Exteroceptivo

En navegación robótica, los datos exteroceptivos proporcionan una percepción del entorno. Los datos exteroceptivos son obtenidos mediante sensores de visión, láseres, entre otros (Wawrzyński, Mozaryn, et al. (2015)).

Filtro de Kalman

En teoría, es un estimador de lo que se llama el “problema cuadrático lineal”, que es el problema de estimar el estado instantáneo de un sistema dinámico lineal perturbado por un ruido blanco, usando mediciones linealmente relacionadas con el estado, pero corrompidas por el ruido blanco (Grewal & Andrews (2008)).

Global Positioning System (GPS)

Sistema que permite determinar la posición de un vehículo dentro de aproximadamente tres metros de su posición verdadera. El sistema se sirve de una red de 24 satélites en órbita sobre el planeta Tierra, a 20,200 km de altura, con trayectoria sincronizada. El sistema proporciona una manera simple y conveniente de determinar la ubicación global del vehículo, sin embargo, tiene un inconveniente importante: los receptores GPS solo son ideales para trabajar en áreas exteriores relativamente abiertas. No funcionan de manera confiable en entornos con poca visibilidad satelital, como áreas muy boscosas, áreas urbanas, bajo el agua, cuevas o dentro de edificios (Barrett, Gennert, et al. (2013)).

Inertial Measurement Unit (IMU)

Dispositivo electrónico que mide la aceleración y la tasa de rotación de una plataforma con altas tasas de actualización, que luego pueden transformarse y procesarse para proporcionar su posición, velocidad y actitud (Kim & Sukkarieh (2007)).

Inertial Navigation System (INS)

Sistema de ayuda a la navegación que usa una IMU y una computadora para estimar continuamente la posición, orientación y velocidad de un objeto en movimiento sin necesidad de referencias externas (Qazizada & Pivarčiová (2016)).

Circuito interintegrado (I^2C)

Bus de comunicación en serie que utiliza dos líneas para transmitir la información: SDA para los datos y SCL para la señal del reloj. Su nombre viene de Inter-Integrated Circuit (Inter-Circuitos Integrados) y es muy utilizado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas embebidos (Ferrer (2015)). Además, existen relaciones simples maestro/esclavo entre todos los componentes y cada dispositivo conectado al bus es direccionable por software mediante una dirección única (i2c-bus.org (2018)a).

Localización autónoma

Proceso de determinación de la posición de la plataforma, sin el uso de información a priori externa a la plataforma, excepto por lo que la plataforma detecta sobre su entorno (Kim & Sukkarieh (2007)).

Microelectromechanical System (MEMS)

Es la tecnología de dispositivos muy pequeños impulsados por la electricidad con al menos algunas de sus dimensiones en el rango micrométrico. Los MEMS típicos consisten en componentes con un tamaño de 1 a 100 μm ; el dispositivo MEMS completo generalmente varía en tamaño desde una estructura de 20 μm a 1 mm . Por lo general, consta de un microprocesador como unidad central para el procesamiento de datos y varios componentes que interactúan con el exterior, por ejemplo, sensores de presión, acelerómetros o giroscopios (Mamilla & Chakradhar (2014)).

Monte Carlo Localization (MCL)

Es un tipo de filtro Bayesiano, que se aproxima numéricamente a la estimación óptima del estado del sistema con un número de partículas ponderadas. El MCL es aplicable a casi todos los sistemas dinámicos no lineales y distribuciones de ruido (J. Zhang, Wang, et al. (2014)).

Multi-State Constraint Kalman Filter (MSCKF)

Algoritmo de navegación inercial visual basado en EKF que realiza una fusión, estrechamente acoplada, entre una IMU y las medidas de la cámara sobre una ventana deslizante de poses de cámara que tiene un tamaño fijo (Heo, Cha, et al. (2017)).

Navegación a estima

Permite conocer la posición aproximada en la que se encuentra el móvil partiendo de la posición en la que se realizó la última medición, la dirección, el valor de la velocidad y tiempo transcurrido desde esa última medición (Álvarez (2016)).

Odometría inercial

Técnica de navegación autónoma en la que, las mediciones proporcionadas por acelerómetros y giroscopios, se utilizan para rastrear la posición y la orientación de un objeto en relación con un punto de partida, orientación y velocidad conocidos (Qazizada & Pivarčiová (2016)).

Odometría visual

Proceso de estimación del movimiento a partir de la entrada visual por sí sola, sin necesidad de conocer previamente la escena ni el movimiento (Nister, Naroditsky, et al. (2004)).

Oriented FAST and Rotated BRIEF (ORB)

Algoritmo resultado de la fusión de los algoritmos FAST y BRIEF. El algoritmo FAST se usa para detectar las esquinas en una imagen y BRIEF se utiliza como descriptor de características en la estructura binaria. El método de detección es mucho más rápido en comparación con SIFT y SURF (Lye, Nisar, et al. (2014)).

Podómetro

Contador de pasos que en la mayoría de los casos se basa únicamente en el acelerómetro, pero también puede usar otros sensores para su funcionamiento a fin de aumentar la confiabilidad de la detección de pasos (Tomažič & Škrjanc (2015)).

Propioceptivo

En navegación robótica, los datos propioceptivos describen el movimiento del vehículo. Los datos propioceptivos son obtenidos mediante sensores como encoders rotatorios, acelerómetros, giroscopios, entre otros (Wawrzyński et al. (2015)).

Robot móvil pequeño

Robot móvil se define como un sistema electromecánico capaz de desplazarse de manera autónoma sin estar sujeto físicamente a un solo punto. Posee sensores que permiten monitorear a cada momento su posición relativa a su punto de origen y a su punto de destino. Normalmente su control es en lazo cerrado. Su desplazamiento es proporcionado mediante dispositivos de locomoción, tales como ruedas, patas, orugas, etc. (Barrientos Sotelo, García Sánchez, et al. (2007)). En este trabajo, robot móvil pequeño se refiere a uno capaz de desplazarse con un paquete de sensores inercial-visual de aproximadamente 0.30 kg de masa.

Region of interest (ROI)

Se define como el delineado aproximado para el objeto de interés o como una región rectangular que contiene tanto el objeto de interés como cierto fondo. El análisis de imagen dentro del área rectangular definida por ROI resulta en un menor costo computacional y una tasa de error prevista a medida que las áreas de la imagen innecesarias se reducen considerablemente (Koundal, Vishraj, et al. (2016)).

Real Time Kinematic- Global Positioning System (RTK-GPS)

Sistema de posicionamiento global cinemático en tiempo real que permite una precisión constante de 1 *em* para las aplicaciones de navegación (Karam, Hadj-Abdelkader, et al. (2010)).

Simultaneous Localization and Mapping (SLAM)

Técnica usada por robots y vehículos autónomos para construir mapas en línea, al tiempo que utiliza el mapa generado para estimar y corregir errores en la solución de navegación obtenida (Kim & Sukkariah (2007)).

Two Wire Interface (TWI)

Bus de comunicación idéntico a I^2C . El nombre TWI fue presentado por Atmel y otras compañías para evitar conflictos con cuestiones de marcas relacionadas con I^2C . Los dispositivos TWI son compatibles con los dispositivos I^2C a excepción de algunas particularidades, como la difusión general o el direccionamiento de 10 bits (i2c-bus.org (2018)b).

Unmanned Aerial Vehicle (UAV)

Vehículo aéreo (que incluye ala fija, ala giratoria o plataforma de dirigible) que puede sostener su vuelo a lo largo de una ruta prescrita sin un piloto a bordo. La tecnología UAV tiene aplicaciones comprobadas en muchas áreas tales como monitoreo y protección ambiental, vigilancia meteorológica e investigación del clima, agricultura, exploración y explotación de minerales, sistema de objetivos aéreos, vigilancia aérea para operaciones terrestres militares y misiones de reconocimiento (Budyono (2008)).

Unscented Kalman Filter (UKF)

Algoritmo recursivo derivado de EKF, se utiliza para linealizar una función no lineal de una variable aleatoria a través de una regresión lineal entre n puntos extraídos de la distribución previa de la variable aleatoria (Vincenzo et al. (2013)).

Capítulo 1

Introducción

Este primer capítulo es una introducción a la navegación inercial asistida por visión, en el cual se señalan los principios de los sistemas de navegación inercial y las características de los sistemas de visión. A continuación, se mencionan los antecedentes, se describe el problema de los sistemas de navegación que fusionan datos inerciales con datos visuales. Y finalmente se define el objetivo y se presenta la metodología de solución de la tesis.

1.1 Introducción a la navegación inercial asistida por visión

En los principios de la navegación, la ubicación del móvil era establecida por métodos basados en referencias externas. Sin embargo, eran susceptibles a errores debido a cambios en las referencias externas provocados por condiciones ambientales. Entre estos métodos se encuentran el uso de las observaciones celestes, las corrientes marinas, los sensores magnéticos y la navegación por radio, que podían ser afectados por cielos nublados, mares tempestuosos, perturbaciones magnéticas y bloqueos (Tazartes (2014)). Ante este problema, nace la navegación inercial como un medio capaz de navegar sin necesidad de referencias externas. La navegación inercial se basa en mediciones de aceleraciones lineales y velocidades angulares obtenidas de acelerómetros y giroscopios respectivamente durante un intervalo de tiempo para calcular mediante *estima* en qué medida y dirección se ha desplazado el móvil desde la última posición conocida (Qazizada & Pivarčiová (2016)).

Los antecedentes de la navegación inercial se encuentran en la invención de los primeros giroscopios, popularizados por ser utilizados para demostrar el movimiento de rotación de la Tierra en el siglo XIX. A comienzos del siglo XX, el uso de giroscopios permitía detectar la dirección de dicho movimiento. Los primeros sistemas de navegación inercial (INS) se desarrollaron en la Segunda Guerra Mundial, empleando giroscopios y acelerómetros para guiar cohetes V2. Además, con la llegada de naves espaciales, misiles y aviones comerciales

Los sistemas de navegación inercial fueron adquiriendo rápidamente un uso más generalizado (Tazartes (2014)), (Amezcuca & Pineda (2012)). Estos sistemas eran de gran peso y alto costo, además, requerían gran asistencia manual para realizar su configuración. El progreso en la tecnología MEMS dio origen a sensores inerciales de bajo costo y tamaño pequeño, permitiendo la combinación de acelerómetros y giroscopios en dispositivos electrónicos llamados IMUs. Este avance volvió más factible al sistema inercial e impulsó su uso en la detección de movimientos y orientación. Sin embargo, los sensores MEMS no son tan precisos como los fabricados con técnicas tradicionales, por lo que cualquier error en las mediciones se va acumulando, generando una importante deriva o diferencia entre la posición estimada por el sistema de navegación. Es por ello, que el sistema de navegación inercial requiere de otros sistemas o instrumentos de navegación para corregir o atenuar los errores producidos (Vincenzo et al. (2013), Ferrer (2015)). Por otro lado, el abaratamiento de las cámaras digitales, la mejora de calidad y su facilidad de uso, propició su utilización en robots y vehículos autónomos, que, aunque tienen una tasa de muestreo relativamente lenta debido al proceso computacional requerido para procesar los datos de la imagen, sus mediciones son bastante confiables ante movimientos moderados. Por lo tanto, en la actualidad se desarrollan sistemas de navegación que fusionan datos inerciales con datos visuales, aprovechando la confiabilidad de los datos obtenidos por las cámaras ante movimientos moderados y la capacidad de seguimiento ante movimientos rápidos y/o bruscos de los sensores inerciales (Barrett et al. (2013)).

En este trabajo se propone estudiar desde la perspectiva de un programa de inteligencia artificial (IA) orientado a visión artificial, las capacidades y limitaciones del IMU miniaturizado como base de un sistema de navegación para un robot móvil con visión.

1.2 Antecedentes

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) en el departamento de Ciencias Computacionales en la línea de Inteligencia Artificial, en la temática de visión robótica, se han abordado los temas de visión y fusión de datos visuales. Unos de los trabajos desarrollados en el CENIDET y considerados como antecedentes de este tema se describen brevemente a continuación.

En la tesis de Vergara (2015) se desarrolló un sistema de visión artificial que permite a un robot móvil realizar un recorrido autónomo sobre una trayectoria, la cual está definida por una secuencia de imágenes extraídas del video capturado durante un recorrido de entrenamiento. El sistema desarrollado está integrado por un par de cámaras montadas en un arreglo de configuración binocular como único sensor; utiliza técnicas y algoritmos de visión binocular; y consta de tres fases principales: entrenamiento, localización y navegación autónoma. Los experimentos realizados en un entorno de interiores con condiciones controladas y sin obstáculos muestran que el sistema es capaz de determinar en qué parte de

la trayectoria se encuentra el robot y finalmente permite repetir la trayectoria de entrenamiento de principio a fin de forma autónoma con un error menor a 10 cm.

En la tesis de Gómez (2013) se realizó un estudio de dos métodos de fusión de información para mejorar la estimación de pose de un robot móvil que cuente con un sistema de visión. La información de pose se obtiene utilizando odometría visual y odometría mecánica, que posteriormente es utilizada en los métodos de fusión. Los métodos de fusión que fueron estudiados y probados son Intersección de Covarianzas y Filtro de Partículas, de los cuales el filtro de partículas refleja ser mejor para la estimación de pose.

1.3 Descripción del problema

Los sistemas de navegación inercial integrados principalmente por IMUs basados en tecnología MEMS se volvieron más factibles e impulsaron su uso en la detección de movimientos y orientación. Sin embargo, los sensores MEMS no son tan precisos como los fabricados con técnicas tradicionales, por lo que cualquier error en las mediciones se van acumulando durante las integraciones realizadas para obtener la posición del móvil, generando una importante deriva o diferencia entre la posición real y la estimada por el sistema de navegación inercial.

En este trabajo se pretende reducir dicha diferencia o deriva controlando las integraciones por componentes del INS mediante la detección del movimiento utilizando la información del flujo óptico.

1.4 Objetivo general

Estudio e implementación de técnicas de odometría para las señales de la unidad de medición inercial aplicables a robots móviles pequeños con visión.

1.5 Objetivos específicos

- Conocer y comprender el funcionamiento de la unidad de medición inercial.
- Estudiar técnicas de detección de movimiento a partir de imágenes.
- Construir un prototipo integrado por una unidad de medición inercial y una cámara.
- Implementar técnicas de odometría inercial.
- Implementar técnicas de odometría inercial con visual.
- Proponer una heurística que permita trabajar, en diferentes momentos, con datos del IMU o visuales, dependiendo de los escenarios de navegación, y cuantificar la reducción del error de dicha heurística.

1.6 Metodología de solución

Se desarrolló e implementó un sistema de navegación inercial asistido por visión en un prototipo integrado principalmente por un microcontrolador equipado con un sensor inercial y un sensor de visión, con el objetivo de estudiar las capacidades y limitaciones de la IMU en robots móviles pequeños con visión.

Para la navegación, el prototipo es montado sobre el robot móvil de manera paralela al suelo cuidando la visibilidad del sensor de visión. El control y la administración se realiza mediante instrucciones ejecutadas desde una computadora conectada remotamente al microcontrolador contenido en el prototipo, tal como se muestra en la Figura 1.1.

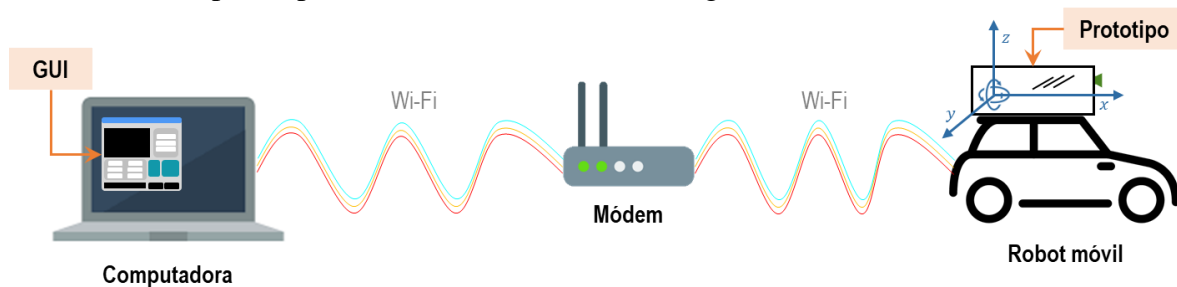


Figura 1.1. Esquema general del sistema de navegación inercial asistido por visión.

La experimentación del sistema se realizó de forma manual y utilizando un carro de control remoto bajo condiciones controladas.

1.6.1 Prototipo construido

La arquitectura del prototipo está conformada por una *IMU Adafruit 9 DoF*, un *Arduino DUE*, una *Raspberry Pi 3* y su *Cámara pi v2* (Figura 1.2). El prototipo se controla mediante instrucciones ejecutadas desde una computadora conectada remotamente a la Raspberry Pi. El funcionamiento de cada componente se resume de la siguiente manera: el arduino gestiona, calibra y filtra las aceleraciones lineales y velocidades angulares detectadas por la IMU. La Raspberry gestiona y procesa las imágenes capturadas por la cámara; gestiona y procesa las aceleraciones lineales y velocidades angulares mediante interrupciones al arduino; y acopla los datos inerciales con los visuales para estimar la orientación del robot móvil durante la navegación.

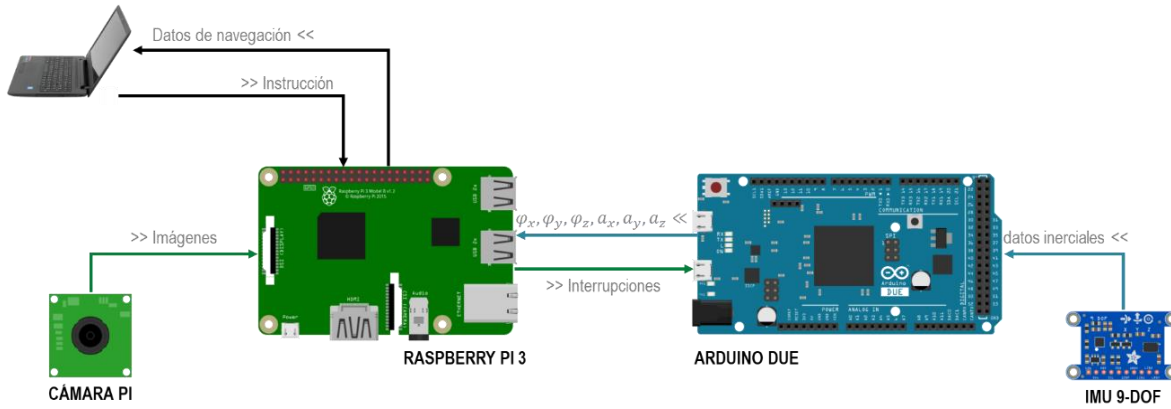


Figura 1.2. Arquitectura del prototipo construido.

1.6.2 Sistema desarrollado

El sistema de navegación inercial asistido por visión para robots móviles terrestres de ruedas se compone de tres módulos: visual, inercial y fusión de datos (Figura 1.3). Por un lado, el módulo visual emplea las imágenes captadas por la cámara para detectar el estado de los ejes x y y del prototipo mediante el cálculo del flujo óptico, dicho estado puede ser “reposo” o “movimiento”. Por otro lado, el módulo inercial utiliza los datos entregados por la IMU para calcular la odometría del prototipo mediante integraciones de aceleraciones lineales y velocidades angulares. Por último, el módulo de fusión de datos realiza el acoplamiento entre los datos del módulo visual e inercial para la estimación de la orientación y posición del prototipo.

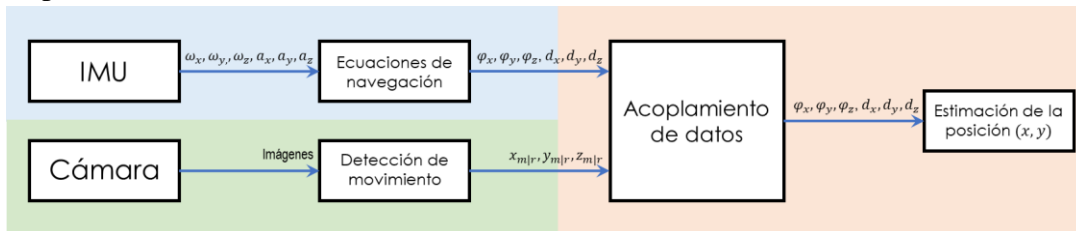


Figura 1.3. Organización del software desarrollado.

1.7 Organización de la tesis

El presente documento se compone de siete capítulos, además de referencias y anexos. En el capítulo dos se presentan los fundamentos teóricos necesarios para la comprensión de esta tesis. En el capítulo tres se presenta una revisión de los trabajos que conforman el estado del arte. En el capítulo cuatro se muestra el prototipo construido, se describen sus componentes y el ambiente de trabajo. En el capítulo cinco se detalla el desarrollo de los módulos que conforman el sistema de navegación desarrollado. En el capítulo seis se describe la experimentación realizada y se presentan los resultados obtenidos. Para finalizar, en el capítulo siete se mencionan las conclusiones y los trabajos futuros.

Capítulo 2

Contexto teórico

En este capítulo se presentan los fundamentos teóricos de la navegación inercial y la detección de movimiento mediante visión, necesarios para la comprensión del documento. En la navegación inercial se mencionan y describen los componentes principales de un sistema de navegación inercial (INS). En el caso de la detección del movimiento a partir de datos visuales, se mencionan tres de las técnicas más utilizadas destacando la técnica del flujo óptico. Además, se define el concepto de fusión de datos y se señala su aplicación en sistemas de navegación que trabajan con datos inerciales y visuales.

2.1 Navegación inercial

La navegación inercial es una técnica de navegación autónoma que utiliza las mediciones proporcionadas por sensores inerciales para rastrear la posición y orientación de un objeto en relación con un punto de partida conocido. Comúnmente, la navegación inercial utiliza acelerómetros y giroscopios para medir aceleración lineal y velocidad angular respectivamente.

Los acelerómetros y giroscopios en el pasado eran de complicada configuración, gran peso y alto costo, por lo que su uso se limitaba a aplicaciones avanzadas como buques, misiles y naves espaciales. En la actualidad, se fabrican acelerómetros y giroscopios basados en Sistemas Microelectromecánicos (MEMS), con un tamaño que va desde un micrómetro a un milímetro, por lo que pueden ser colocados en un pequeño chip de silicio formando un sensor como tal. La ventaja de los sensores inerciales MEMS es que son pequeños, livianos y tienen un bajo consumo de energía y tiempos de puesta en marcha. Sin embargo, no son tan precisos como los fabricados con técnicas tradicionales, aunque su rendimiento va mejorando rápidamente (Woodman (2007), Qazizada & Pivarčiová (2016)). La Figura 2.1, muestra un acelerómetro MEMS de 3 ejes integrado a un chip ASIC.

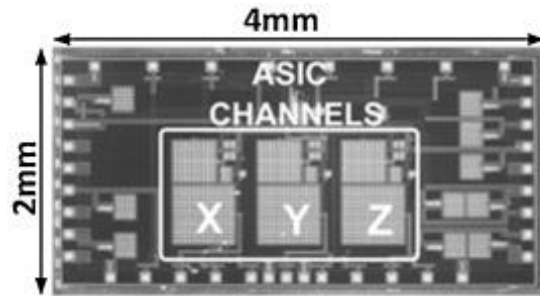


Figura 2.1. Chip ASIC integrado por un acelerómetro MEMS triaxial (Santana, Van Den Hoven, et al. (2012)).

Los sensores inerciales pueden ser combinados para formar una Unidad de Medición Inercial (IMU), esto con el objetivo de obtener datos más informativos que serían imposibles de conseguir usando únicamente la información de cada sensor por separado. La IMU más usual es la que combina un acelerómetro y un giroscopio de tres ejes cada uno.

Las mediciones entregadas por la IMU permiten la navegación en relación con el espacio inercial (sin gravedad presente), dado los valores iniciales de velocidad, posición y orientación. En la navegación terrestre es necesario compensar la gravedad y la rotación de la Tierra. Las mediciones se deben integrar para obtener la velocidad lineal, la orientación y la posición, las ecuaciones utilizadas para las integraciones se denominan ecuaciones de navegación.

De acuerdo con Qazizada & Pivarčiová (2016), la combinación de una IMU y una computadora que ejecuta ecuaciones de navegación se denomina Sistema de Navegación Inercial (INS). En la Figura 2.2 se muestran los elementos que componen el INS.

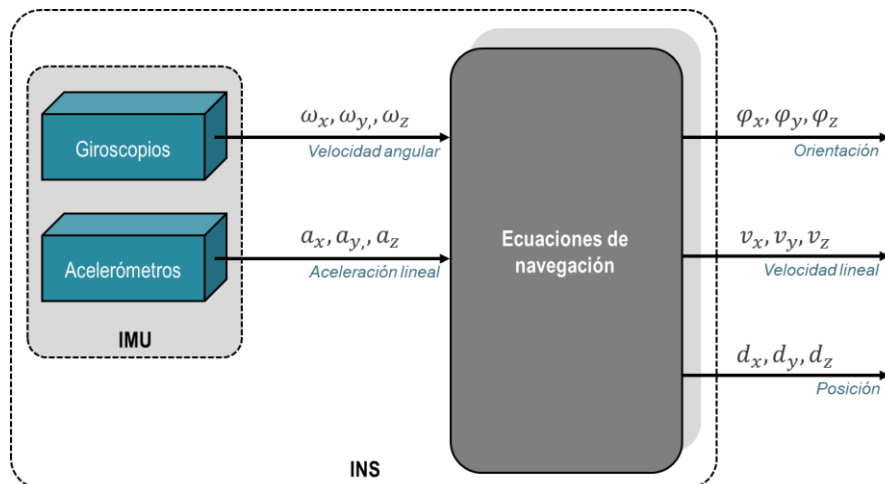


Figura 2.2. Sistema de Navegación Inercial (INS).

Los sistemas de navegación inercial tienen la ventaja de ser *propioceptivos*, es decir, no necesitan de referencias externas para describir su movimiento. Sin embargo, sus mediciones

son afectadas por un error acumulativo debido a que las estimaciones de dirección y posición se realizan partiendo de la posición en la que se realizó la última medición. Esto provoca que cualquier error en la medición se vaya acumulando de punto a punto, generando una diferencia entre la posición real del sistema y la posición estimada por el INS. Por lo que, en su mayoría son inadecuados para un posicionamiento preciso durante un periodo prolongado. Es por esto, que los INS se apoyan en otros instrumentos o sistemas para corregir o atenuar los errores producidos (Vincenzo et al. (2013), Ferrer (2015)).

2.1.1 Giroscopios MEMS

Sensor electromecánico capaz de medir la velocidad angular a la que está sometido en uno, dos o tres ejes (Suprem, Deep, et al. (2017)). Los giroscopios MEMS hacen uso del efecto Coriolis, que establece que en un marco de referencia que gira a velocidad angular (ω), una masa (m) que se mueve con velocidad (v) experimenta una fuerza (Woodman (2007)):

$$F_c = -2m(\omega \times v) \tag{2.1}$$

De acuerdo con Zheng, Li, et al. (2016) y Varesano (2011) el giroscopio MEMS está compuesto de una masa de prueba unida al marco rígido a través de amortiguadores y resortes, como se muestra en la Figura 2.3. El marco rígido está sujeto a una velocidad angular (Ω) sobre el eje de rotación (z), se produce una aceleración de Coriolis a lo largo del eje de detección (y) mientras la masa de prueba se conduce a resonancia a lo largo del eje de accionamiento (x), que es perpendicular a los ejes de sentido y rotación. La aceleración de Coriolis es proporcional tanto a la velocidad angular aplicada como a la amplitud de velocidad de la masa en movimiento a lo largo del eje motriz. Por lo tanto, la velocidad angular puede determinarse a través de la vibración del eje de detección.

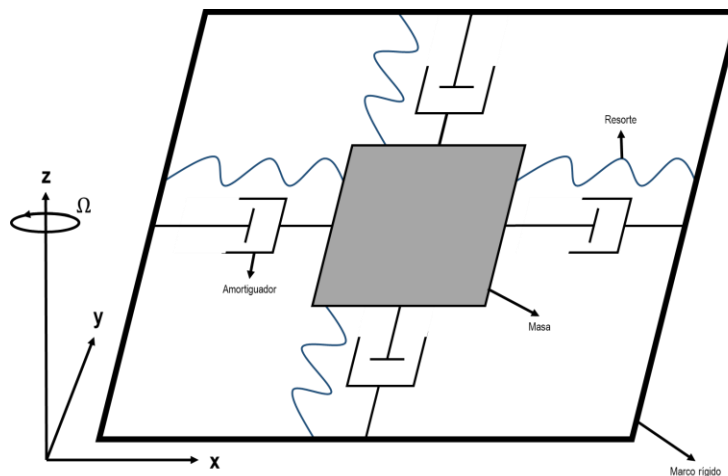


Figura 2.3. Estructura del giroscopio MEMS (Zheng et al. (2016)).

En un giroscopio de dos o tres ejes, este tipo de estructura se replica, con el cambio de orientación oportuno para cada uno de los ejes. Los giroscopios disponibles normalmente

son de tres ejes, permitiendo conocer la magnitud y dirección de la velocidad angular. El sistema de coordenadas de un giroscopio de tres ejes se ilustra en la Figura 2.4.

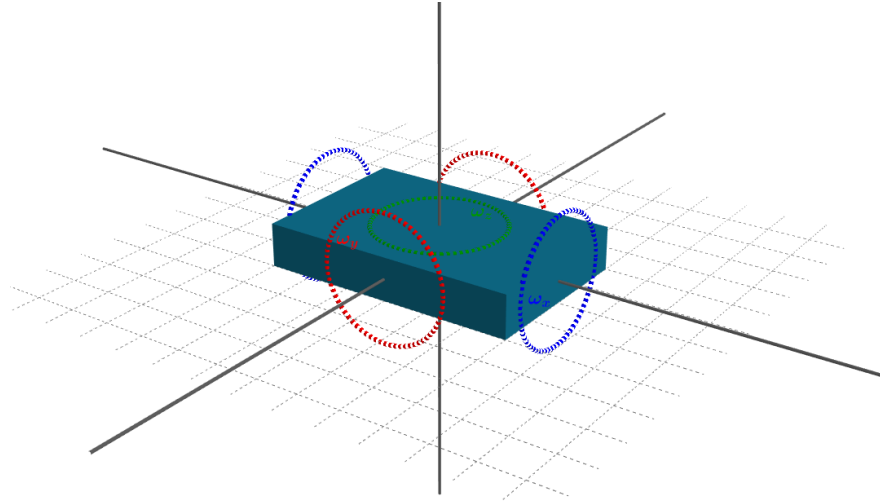


Figura 2.4. Sistema de coordenadas de un giroscopio de tres ejes.

A pesar de que los giroscopios MEMS no son tan precisos, son frecuentemente utilizados en aplicaciones robóticas. Sin embargo, en su utilización se debe lidiar con inconvenientes como el sesgo (valor medio del giroscopio sin rotación (ε)) medido en $^{\circ}/h$ causado por la integración de la velocidad angular para obtener la orientación $\theta(t)$. Dicho sesgo produce un error denominado “deriva”, el cual crecerá en el tiempo (t) (Zaidner & Shapiro (2016)):

$$\theta(t) = \varepsilon \cdot t \quad (2.2)$$

Este error puede estimarse tomando mediciones a largo plazo sin rotación y en la mayoría de los casos puede suponerse que su tasa permanece constante, por lo que se puede restar de la salida.

2.1.2 Acelerómetros MEMS

Sensor electromecánico capaz de medir las fuerzas de aceleración a la que está sometido en uno o más ejes, relativamente insensible a las direcciones ortogonales (Qazizada & Pivarčiová (2016)).

De acuerdo con Andrejašič (2008) y Varesano (2011) el acelerómetro MEMS típico está construido a base de silicio, compuesto de una masa de prueba móvil con placas que se conectan a través de un sistema de suspensión mecánica a un marco de referencia, como se muestra en la Figura 2.5. Las placas móviles y las placas exteriores fijas representan condensadores. La masa de prueba se fija a través de resortes k_s en el sustrato, permitiendo únicamente movimientos hacia arriba y hacia abajo. Las capacitancias de espacio libre entre la placa móvil y dos placas exteriores fijas c_1 y c_2 , son funciones de los desplazamientos

correspondientes a x_1 y x_2 . La deflexión de la masa es provocada por la aceleración y se mide usando la diferencia de capacitancia, esta diferencia es detectada por la electrónica, que lo convierte en un valor de aceleración. Si la aceleración es cero las capacitancias c_1 y c_2 son iguales.

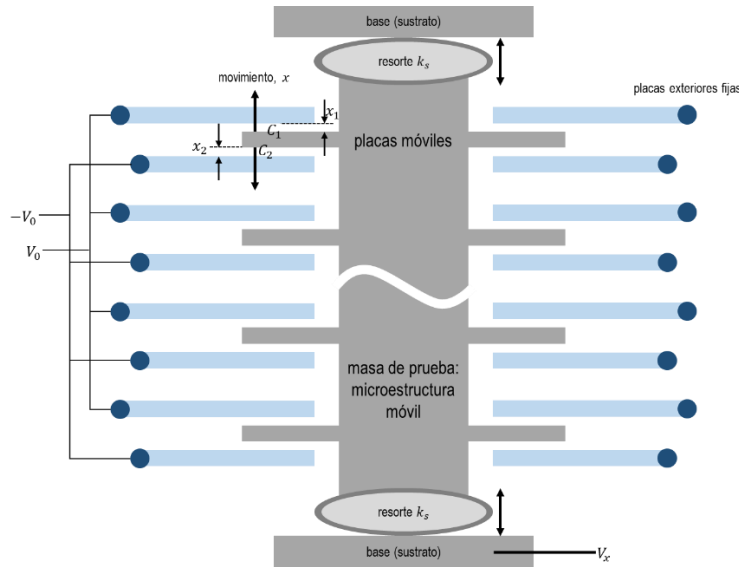


Figura 2.5. Estructura del acelerómetro MEMS (Andrejašič (2008)).

En un acelerómetro de dos o tres ejes, este tipo de estructura se replica, con el cambio de orientación oportuna para cada uno de los ejes. Al igual que los giroscopios, normalmente los acelerómetros disponibles son de tres ejes, permitiendo conocer la magnitud y dirección de la aceleración lineal. El sistema de coordenadas de un acelerómetro de tres ejes se ilustra en la Figura 2.6.

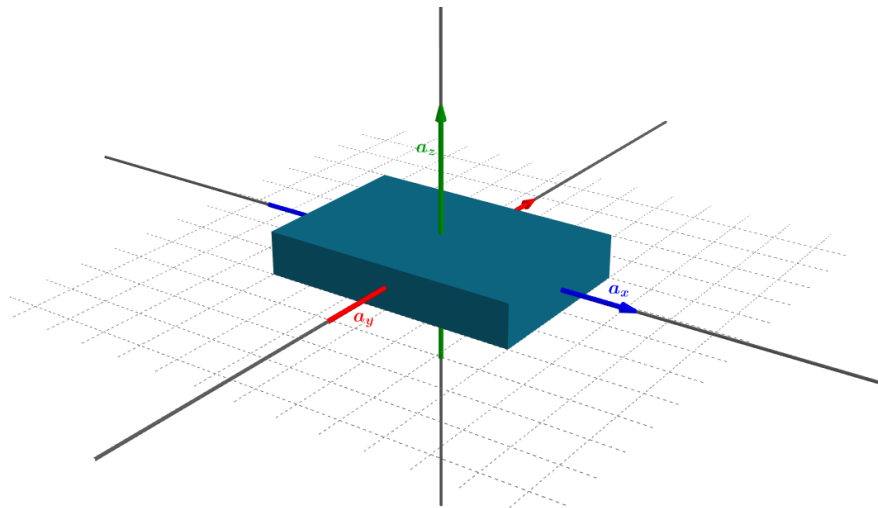


Figura 2.6. Sistema de coordenadas de un acelerómetro de tres ejes.

Los acelerómetros detectan fuerzas de aceleración estáticas como la gravedad y fuerzas dinámicas como las vibraciones y los movimientos. La aceleración originada por la fuerza de gravedad es de 9.80665 m/s^2 aproximadamente y es registrada constantemente por el

sensor. El registro de la gravedad puede emplearse para detectar condiciones de caída libre e incluso para determinar la orientación del sensor. Además, dicho conocimiento de orientación es esencial para compensar la gravedad.

Los acelerómetros MEMS al igual que el giroscopio MEMS presenta lecturas sesgadas que causan errores en las mediciones. Por lo que, al integrar doblemente las aceleraciones lineales con sesgo (ε) para obtener la posición $d(t)$, el resultado es un error creciente en el tiempo(t) (Zaidner & Shapiro (2016)):

$$d(t) = \varepsilon \cdot \frac{t^2}{2} \quad (2.3)$$

Sin embargo, dicho sesgo puede ser modelado mediante mediciones a largo plazo en aceleración cero, para disminuir los errores en la medición.

2.1.3 Unidad de Medición Inercial (IMU)

La unidad de medición inercial o IMU (del inglés Inertial Measurement Unit) es un dispositivo electrónico típicamente compuesto por tres giroscopios ortogonales y tres acelerómetros ortogonales que miden velocidad angular y aceleración lineal respectivamente.

Las IMUs se utilizan como componentes esenciales en los sistemas de navegación de barcos, aviones, helicópteros, transbordadores, satélites o cualquier móvil sin posibilidad de referencias externas. La IMU más común es la de 6 grados de libertad o DoF (del inglés degrees of freedom) combinando un acelerómetro de tres ejes y un giroscopio de tres ejes. Actualmente, es fácil encontrar IMUs de 9 DoF que añaden una brújula magnética de tres ejes, así como IMUs 10 DoF que añaden un barómetro para la estimación de la altura del sensor. En la Figura 2.7 se ilustra el sistema de coordenadas de una IMU 6 DoF.

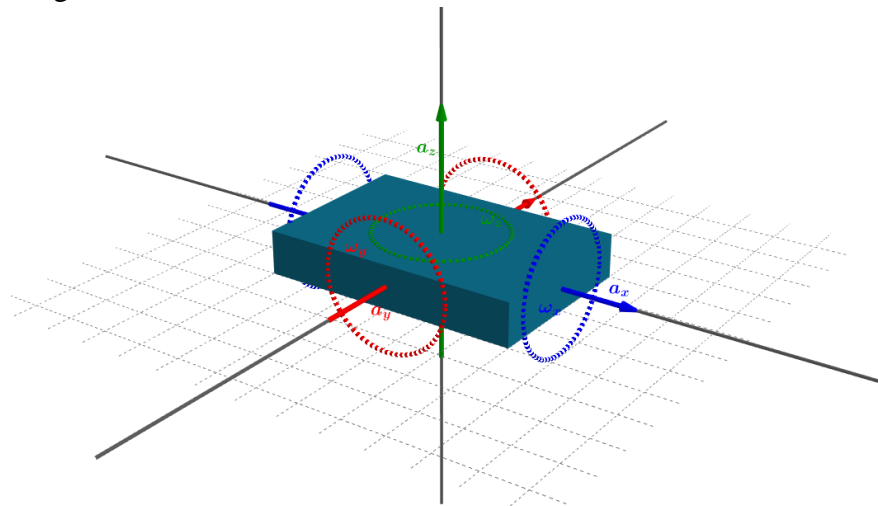


Figura 2.7. Sistema de coordenadas de una IMU 6 DoF.

En un sistema de navegación inercial los datos inerciales entregados por la IMU suelen ser controlados y administrados por un microprocesador, conocido como sistema embebido. En dicho sistema se ejecutan los cálculos necesarios para obtener las estimaciones requeridas.

2.2 Detección de movimiento por flujo óptico

La detección de movimiento es el proceso de detectar un cambio en la posición de un objeto en relación con su entorno o un cambio en el entorno en relación con un objeto (Ortega, Molina, et al. (2016)). La detección de movimiento es frecuentemente utilizada por aplicaciones de visión por computadora, cuyo propósito consiste en extraer los objetos en movimiento en el tiempo t en una secuencia de video. Entre sus aplicaciones más comunes se encuentra la videovigilancia, el control de tráfico y peatones y el reconocimiento de lenguaje de señas específicas para aplicaciones de robótica (navegación automática, detección de obstáculos o guía de misiles).

De acuerdo con Elharrouss, Moujahid, et al. (2015) y Sengar & Mukhopadhyay (2017) las técnicas de detección de movimiento se pueden dividir en tres categorías: diferencia de trama, sustracción de fondo y flujo óptico. La diferencia de trama consiste en calcular la diferencia de píxeles entre dos o más imágenes consecutivas de una secuencia de video. Esta técnica puede detectar objetos en movimiento en diferentes entornos desafiantes; sin embargo, a veces los objetos detectados están incompletos y mal presentados debido a la lentitud o pausa del movimiento. La sustracción de fondo consiste en construir un modelo de la escena estática (fondo) y comparar mediante una función de umbralización cada imagen de la secuencia de video con el modelo construido para distinguir las regiones de movimiento (u objetos en movimiento). Esta técnica detecta eficientemente los objetos en movimiento, sin embargo, falla en la detección en diferentes escenarios debido a la falta de un modelo de fondo generalizado. Por último, el flujo óptico se basa en el vector de flujo para calcular el movimiento aparente del objeto entre cuadros consecutivos. Esta técnica es muy robusta en la detección de las regiones objetivo en movimiento debido a que proporciona toda la información sobre el movimiento, sin embargo, proporciona resultados inexactos en entornos con variación de iluminación.

Para la realización de esta tesis, se trabajó en un ambiente controlado, por lo que se utilizó el flujo óptico como técnica de detección de movimiento. A continuación, se define el concepto de flujo óptico, se mencionan las técnicas para su cálculo y se explica la técnica empleada.

El **flujo óptico** es una forma de describir el movimiento aparente de la superficie, los objetos y los bordes que ocurre debido al movimiento relativo entre el observador y la escena (Tomažič & Škrjanc (2015)). El propósito del flujo óptico es determinar el movimiento entre dos imágenes consecutivas que se capturan en el tiempo t y $t + \Delta t$. Existen diferentes técnicas propuestas para su cálculo, por ejemplo, el análisis de la correlación de distintos

fotogramas del vídeo con una región de interés a seguir o técnicas diferenciales que se basan en el cálculo de las derivadas espaciales y temporales de las imágenes.

En las técnicas diferenciales se encuentra el método de Lucas y Kanade, el cual se basa en la aproximación local de Taylor de la señal de la imagen pudiéndose implementar utilizando el rastreador de características Kanade-Lucas-Tomasi (KLT). De acuerdo con Tomažič & Škrjanc (2015) el algoritmo KLT ordinario es para movimientos pequeños y el algoritmo KLT piramidal es para movimientos más grandes. En el primero, el flujo óptico se estima entre dos imágenes consecutivas en escala de grises con tamaño $N_x \times N_y$ píxeles, que se denotan con I y J . Si el punto $u = (u_x, u_y)$ se conoce en la imagen I , entonces el objetivo del algoritmo es encontrar el punto v en la imagen J , donde los valores de $I(u)$ e $J(v)$ son similares ($v = u + d = (u_x + d_x, u_y + d_y)$). En el punto u el flujo óptico es igual a $d = [d_x d_y]^T$. El flujo óptico actual se evalúa con el estimador $\varepsilon(d)$, que se define como:

$$\varepsilon(d) = \varepsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I(x, y) - J(x + d_x, y + d_y))^2 \quad (2.4)$$

donde w_x en w_y son parámetros que determinan el tamaño de la ventana de integración: $(2w_x + 1) \times (2w_y + 1)$ (Figura 2.8).

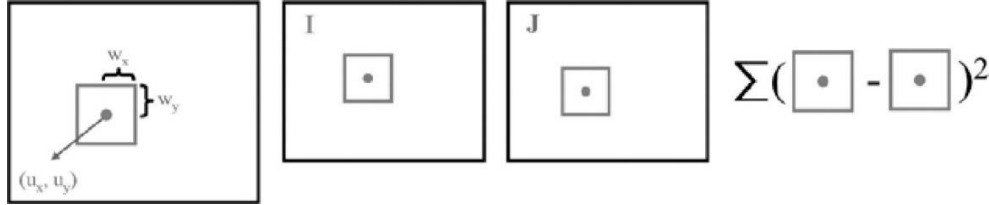


Figura 2.8. Estimación del desplazamiento entre las imágenes I y J (Tomažič & Škrjanc (2015)).

En el segundo, el flujo óptico se estima en una pirámide de imágenes $I^0 \rightarrow I^1 \rightarrow I^2 \rightarrow I^3 \rightarrow I^{Lm} \rightarrow \dots$ ($L_m: 2 \sim 4$), donde el flujo óptico se calcula primero en el nivel más alto y luego en niveles más bajos hasta el nivel cero. En cada nivel se considera una estimación preliminar del movimiento desde el nivel superior. El flujo óptico en el nivel individual se calcula usando la matriz de gradiente:

$$G = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (2.5)$$

y el vector de desajuste de imagen:

$$b = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I & I_x \\ \delta I & I_y \end{bmatrix} \quad (2.6)$$

como:

$$d^L = G^{-1}b \quad (2.7)$$

donde I_x en I_y son las derivadas parciales de la imagen I y $\delta I(x, y) = A(x, y) - B(x, y)$ es la diferencia entre dos subimágenes. El flujo óptico final se obtiene como:

$$d = \sum_{L=0}^{L_m} 2^L d^L \quad (2.8)$$

donde d^L es el desplazamiento estimado en el nivel L-ésimo.

2.3 Fusión de datos inerciales con visuales

La fusión de datos es una integración de varios datos de sensores para obtener mejores resultados (Zaidner & Shapiro (2016)). Se usa ampliamente en sistemas militares, vigilancia, control de procesos y robótica. Además, es de particular importancia en el desarrollo de sistemas autónomos en todas sus aplicaciones. El mejor ejemplo de fusión de datos se da en el cerebro humano (Figura 2.9), el cual fusiona la información entregada por los sistemas sensoriales (vista, oído, gusto, olfato y tacto) del cuerpo para percibir lo que está en el ambiente e intentar derivar el conocimiento.

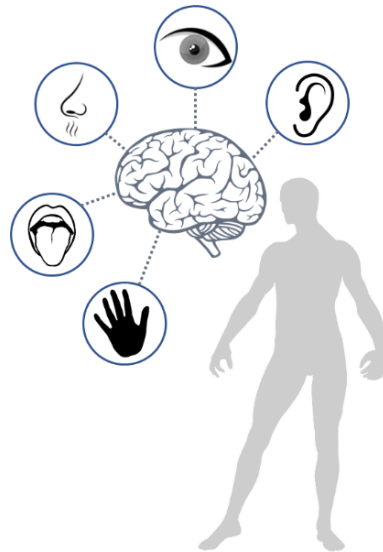


Figura 2.9 Fusión de datos de los sistemas sensoriales del cuerpo humano (Fuentes varias (2018)).

La fusión de datos provenientes de múltiples sensores provee ventajas significativas sobre una fuente única de datos. Entre las ventajas destaca la reducción de la incertidumbre e incremento de la precisión con la cual las características son percibidas, así como el aumento de la confiabilidad en caso de errores o fallas de un sensor. Además, el uso de múltiples tipos de sensores puede incrementar la precisión con la cual un objetivo puede ser observado y caracterizado.

En robótica, la fusión de datos es utilizada en el reconocimiento de objetos, la construcción de mapas y la localización. En el tema de la navegación, para que un sistema de navegación tenga éxito, el vehículo debe conocer su posición actual (localización) en todo momento. Para esto, los principales sistemas de localización empleados son el Sistema de Posicionamiento Global (GPS), los sistemas basados en visión y el Sistema de Navegación Inercial (INS). El GPS proporciona una manera simple y conveniente de determinar la ubicación global del vehículo; sin embargo, estiman la posición dentro de los tres metros de la posición verdadera, aproximadamente. En los sistemas basados en visión, la odometría visual (VO) normalmente detecta y rastrea puntos característicos entre cuadros en una secuencia de video para estimar el movimiento relativo entre fotogramas. Sin embargo, el procesamiento de imágenes requiere de gran esfuerzo computacional, por lo que ante cambios repentinos de escena se pierde la mayoría o todos los puntos característicos provocando errores en la estimación del movimiento. El INS por su parte, calcula la posición, orientación y velocidad sin necesidad de referencias externas, siendo capaz de medir con precisión los cambios rápidos tanto en las tasas de rotación angular como en las aceleraciones lineales. Sin embargo, pequeños errores en la medición de aceleración lineal y velocidad angular se integran en errores progresivamente mayores (Ferrer (2015)).

En la actualidad, con el progreso de la tecnología MEMS se encuentran IMUs de tamaños pequeños, ligeras y baratas. Sin embargo, sus mediciones presentan ruido que al ser integradas acumulan errores que deben corregirse periódicamente. El GPS es utilizado frecuentemente como sensor complementario, sin embargo, no funciona de manera confiable en entornos con poca visibilidad satelital. Otros métodos, últimamente estudiados, son los basados en visión, que, a pesar de sus limitaciones, la precisión de sus estimaciones de posición es confiable. Por lo tanto, un sistema de navegación inercial asistido por visión (VINS) puede aprovechar la capacidad de seguimiento ante movimientos rápidos y/o bruscos durante períodos cortos de los sensores inerciales y la confiabilidad de las mediciones de visión (Hesch, Kottas, et al. (2014)).

De acuerdo con Fang & Zheng (2018), la navegación a partir de datos inerciales y visuales se puede clasificar en basada en filtrado y basada en optimización. El enfoque basado en filtrado requiere menos recursos computacionales al marginar estados pasados, pero puede tener un rendimiento ligeramente inferior debido a la fijación temprana de puntos de linealización. Mientras que el segundo enfoque basado en optimización da como resultado un mejor rendimiento a expensas de mayores demandas de cómputo. Para las aplicaciones en tiempo real es recomendado abordar el método de fusión basado en filtrado. De acuerdo con Kim & Sukkarieh (2007), los métodos de fusión basados en filtrado se agrupan en ligeramente acoplados y estrechamente acoplados. Los sistemas que emplean el primer enfoque utilizan dos módulos independientes ejecutados a diferentes velocidades, uno de estimación de pose basado en la visión y otro de propagación IMU. Ambos módulos intercambian información, sin embargo, debido a la falta de *cross-correlation* (correlación



cruzada) entre ambos bloques, estos sistemas son incapaces de corregir derivas en las estimaciones. Mientras que los sistemas que utilizan el segundo enfoque obtienen mejores resultados de navegación, mediante la combinación de datos inerciales con visuales en un filtro único y óptimo.

Capítulo 3

Estado del arte

En este capítulo se presenta una revisión de distintos trabajos realizados en el área de navegación robótica, específicamente para la estimación de la posición del móvil mediante fusión de datos inerciales con otros sistemas de navegación. Los trabajos del estado del arte se clasificaron en tres grupos dependiendo de los datos a fusionar: sistemas que fusionan los datos de la IMU, sistemas que fusionan los datos de la IMU con datos GPS y sistemas que fusionan los datos de la IMU con datos visuales.

3.1 Sistemas que fusionan los datos de la IMU

Estos sistemas de navegación son independientes de cualquier infraestructura, por lo tanto, estiman la dirección y posición en base a las mediciones de rumbo y velocidad entregadas por una IMU montada en el vehículo. Sin embargo, las estimaciones no son tan precisas, por lo que requieren de algoritmos inteligentes para corregir los errores de deriva.

En el artículo de Ibrahim & Moselhi (2016) se presenta un sistema de localización en interiores utilizando un microcontrolador equipado con una IMU 9 DoF y un sensor de presión barométrica. El sistema se compone de tres módulos: medición inercial, medición de altitud y localización. El primer módulo procesa y fusiona los datos inerciales utilizando un algoritmo de *Matriz de Coseno Directores* (DCM). El segundo módulo calcula la altitud con base en la presión barométrica medida. Por último, el tercer módulo estima la posición con base en los desplazamientos, el rumbo y la altitud mediante el *Filtro de Kalman Extendido* (EKF). En la Figura 3.1 se muestran los resultados obtenidos por el sistema desarrollado.

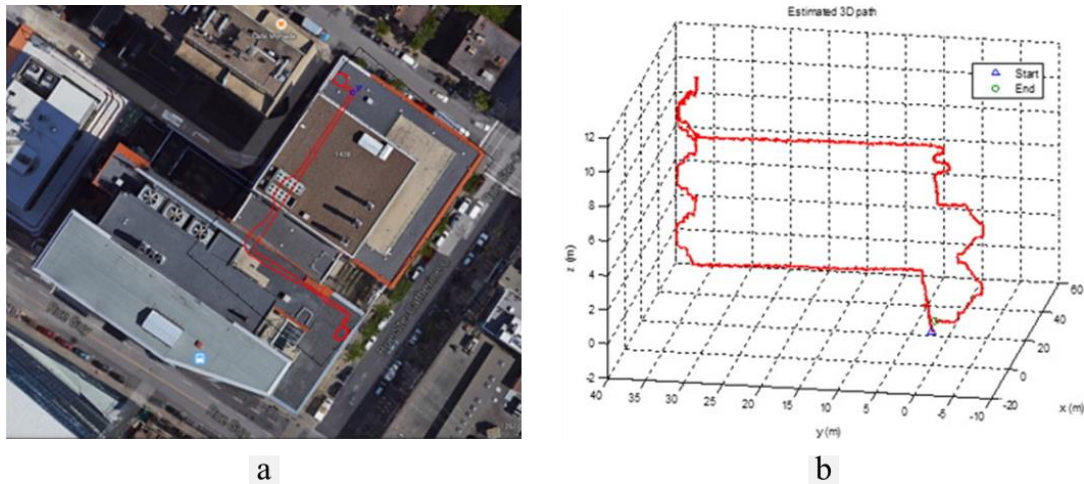


Figura 3.1. Experimento núm. 4 (Ibrahim & Moselhi (2016)). a) ruta utilizada para el experimento. b) ruta estimada por el sistema desarrollado.

3.2 Sistemas que fusionan los datos de la IMU con datos GPS

Estos sistemas de navegación estiman la dirección y posición del vehículo mediante la fusión de datos entregados por la IMU y el GPS. Fueron desarrollados para aumentar la precisión en la estimación de la posición, así como para mejorar la navegación en entornos con interferencias significativas para el GPS, tales como áreas urbanas, dentro de edificios u otras ubicaciones con barreras físicas entre el dispositivo receptor y los satélites GPS.

En el artículo de Elarabi & Suprem (2015) se presenta un algoritmo de detección de orientación y posición que fusiona la información de los sensores inerciales MEMS de un teléfono inteligente. El algoritmo se resume en cinco actividades: obtención de la aceleración, cálculo de la orientación, reorientación de los vectores de aceleración, resta de la gravedad de los valores de aceleración e integración de los valores de velocidad para determinar el desplazamiento (Figura 3.2). Este algoritmo se desarrolló con el objetivo de ser utilizado en la fusión de datos GPS y datos IMU para dispositivos pequeños en el contexto de las aplicaciones de internet de las cosas y/o seguimiento peatonal.

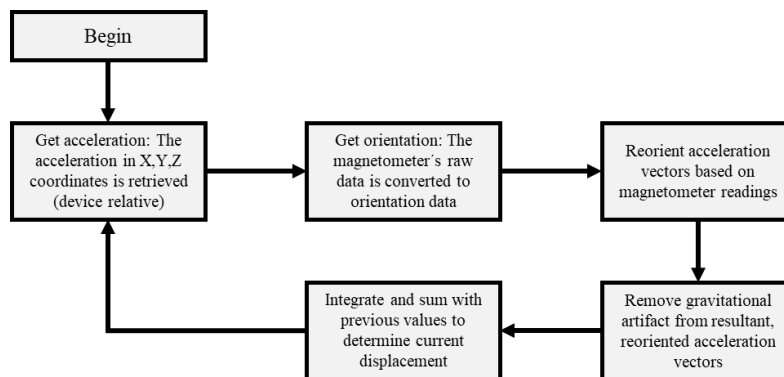


Figura 3.2. Algoritmo de detección de orientación y posición (Elarabi & Suprem (2015)).

En el artículo de Suprem et al. (2017) se presenta una GUI basada en MATLAB que fusiona los datos de la IMU y el GPS para mejorar la navegación en áreas de acceso denegado. En tales áreas, ocurren interferencias que generan la pérdida de datos GPS, provocando huecos en los datos de navegación. En la GUI desarrollada, estos huecos son llenados con los datos proporcionados por el algoritmo de detección de orientación y posición presentado en Elarabi & Suprem (2015). Los valores resultantes se pasan a través de un *Filtro Savitzky-Golay* para eliminar los elementos ruidosos. En la Figura 3.3 se muestran los resultados obtenidos por la GUI desarrollada.

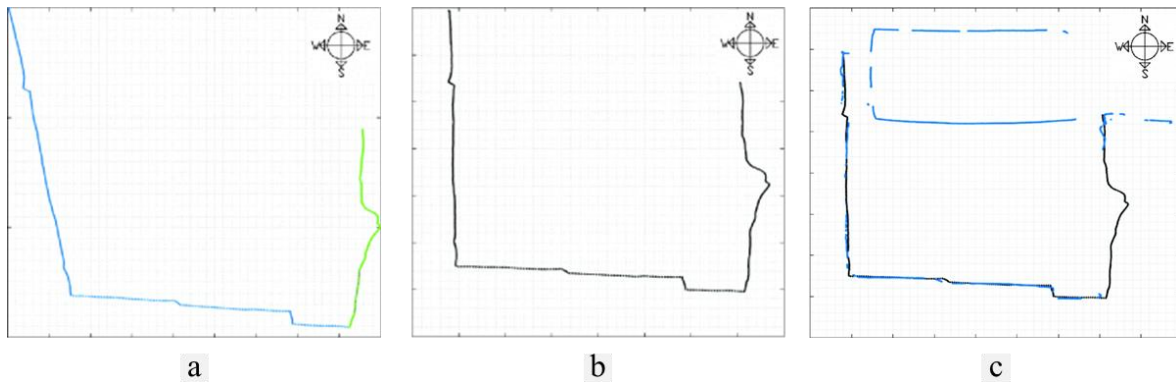


Figura 3.3. Resultados obtenidos por la GUI desarrollada (Suprem et al. (2017)). a) datos IMU. b) datos IMU corregidos. c) fusión GPS/IMU.

En el artículo de Suwandi, Kitasuka, et al. (2017) se presenta una estrategia de fusión de datos GPS/IMU para la localización de vehículos terrestres. La estrategia combina velocidad, distancia y ángulo yaw (obtenidos a partir de las integraciones de los datos entregados por la IMU), posición GPS, velocidad GPS y rumbo GPS mediante un tipo de *Filtro de Kalman*. Para la experimentación se usó un automóvil equipado por una IMU 9 DoF y un receptor GPS. La Figura 3.4 muestra el escenario experimental que consiste en una ruta recta con una distancia de 100 metros; además, se muestra la comparación entre los resultados obtenidos por el sistema de fusión de datos, el GPS, y la IMU con los datos ground truth obtenidos de la ruta experimental. Nota: la frecuencia de muestreo del receptor GPS es de 1 Hz y 33 Hz de la IMU.

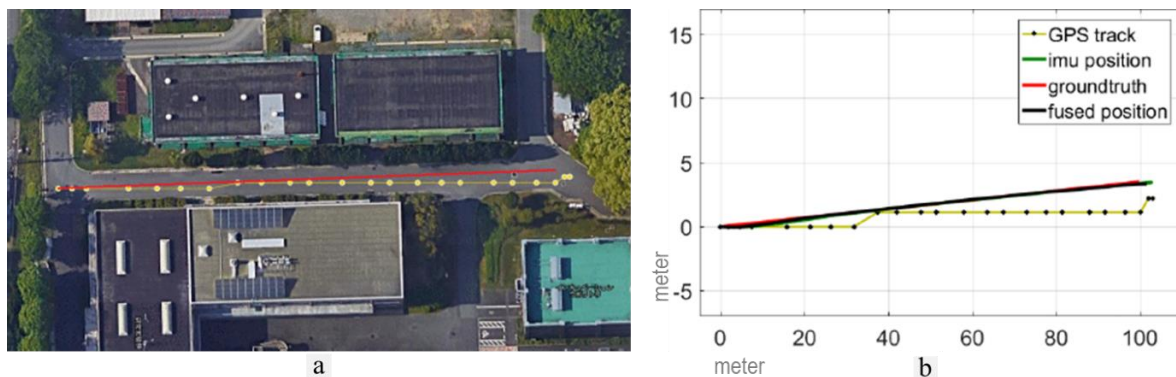


Figura 3.4. Experimentación (Suwandi et al. (2017)). a) ruta recta, donde la línea roja es la ruta predefinida y la línea amarilla son posiciones GPS. b) Resultados obtenidos por los sistemas GPS, IMU y la fusión de datos.

3.3 Sistemas que fusionan los datos de la IMU con datos visuales

Estos sistemas de navegación estiman la dirección y posición del vehículo con base en los datos entregados por IMUs y cámaras. Por un lado, la IMU puede realizar el seguimiento de movimientos de alta frecuencia entregando los datos a una alta velocidad de muestreo, pero sus mediciones derivan con el tiempo. Por otro lado, las cámaras entregan los datos con una tasa de muestreo relativamente lenta debido al proceso computacional que se requiere para procesar los datos de la imagen, pero sus mediciones solo derivan en movimientos bruscos. Estas características vuelven a ambos enfoques ideales para trabajar juntos, por lo que, la fusión de datos IMU y cámara permite explotar las mejores características de ambos enfoques, proporcionando un mejor rendimiento en la estimación de la dirección y posición.

En el artículo de Kim & Sukkarieh (2007) se presentan dos algoritmos de SLAM aerotransportado para vehículos aéreos no tripulados (UAV). Ambos algoritmos fusionan datos de una IMU con datos de un sistema de visión. Sin embargo, uno ha sido formulado directamente para modelos dinámicos no lineales y de observación, usando un Filtro de Kalman Extendido (EKF) y el otro se ha formulado indirectamente para modelos de dinámica/observación de errores linealizados, utilizando un Filtro de Kalman lineal (KF) (Figura 3.5). En ambos casos, una IMU proporciona la aceleración lineal y la velocidad angular del vehículo. El sensor de observación proporciona el rango, el rumbo y la elevación de las características observadas. En forma directa, el filtro acepta datos brutos del IMU y los pasa a un modelo 6 DoF no lineal, y el EKF procede a través del proceso de predicción y actualización de los estados del vehículo y las ubicaciones de las características. En forma indirecta, el bucle inercial está separado del filtro, por lo tanto, se emplean ecuaciones de navegación inercial para transformar los datos inerciales brutos en mediciones de velocidad, orientación y posición. El modelo dinámico del KF es un modelo de error tanto del vehículo como de las características observadas, por lo que, cuando ocurre una observación, también se genera una observación pronosticada, que se basa en la ubicación actual del vehículo y de la característica como se indica en el mapa. El KF usa la diferencia entre las observaciones pronosticadas y reales para estimar los errores inerciales y de características, dichos errores se vuelven a enviar al INS y al mapa para realizar nuevas correcciones.

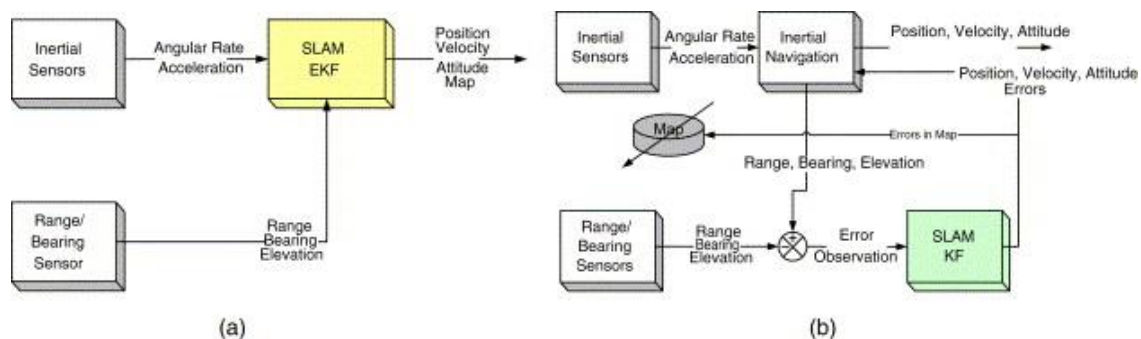


Figura 3.5. SLAM aerotransportado (Kim & Sukkarieh (2007)). a) Algoritmo directo. b) algoritmo indirecto.

Al momento de la publicación del trabajo aún no se había implementado la estructura indirecta debido a restricciones de prueba de vuelo, por lo que sólo se presentan resultados de la estructura directa. El algoritmo SLAM aerotransportado (directo) se implementó en una plataforma de UAV (Brumby MKIII), equipado con sensores conectados a la computadora de control de vuelo a través de un enlace en serie o un sistema de bus local. Los sensores principales fueron una IMU de Inertial Science (6 DoF) de tamaño pequeño y ligero, el cual generaba datos a una velocidad de 400 Hz a la computadora de control de vuelo y SLAM a través de un enlace en serie RS-422 durante las operaciones de vuelo. Y una cámara monocroma Sony de bajo costo, con resolución de 600 líneas horizontales y una salida de video de 25 Hz o 50 Hz, dependiendo del modo entrelazado. La cámara estaba montada apuntando hacia abajo y proporcionaba observaciones de características al nodo SLAM integrado. Para las pruebas se colocaron características artificiales (láminas de plástico blanco) en el suelo con una separación mayor a 50 m. Se utilizó un algoritmo de umbral de intensidad simple para extraer estas características de las imágenes en tiempo real, con esto se genera el rumbo y un valor estimado para el rango en función del tamaño de las características observadas. El tiempo de prueba de vuelo fue alrededor de 30 min, la altura nominal de vuelo fue de 100 m sobre el suelo con una velocidad de desplazamiento máxima de 40 m/s aprox. El vehículo se sometió a un vuelo autónomo en una trayectoria ovalada, en el cual el sistema construye un mapa a medida que recorre la trayectoria. El resultado se muestra en la Figura 3.6, en la cual se observa que la re-observación sucesiva de las características mejora continuamente la precisión de navegación y mapeo, por lo que la incertidumbre del mapa disminuye y se vuelve menos sensible a la suma de más información.

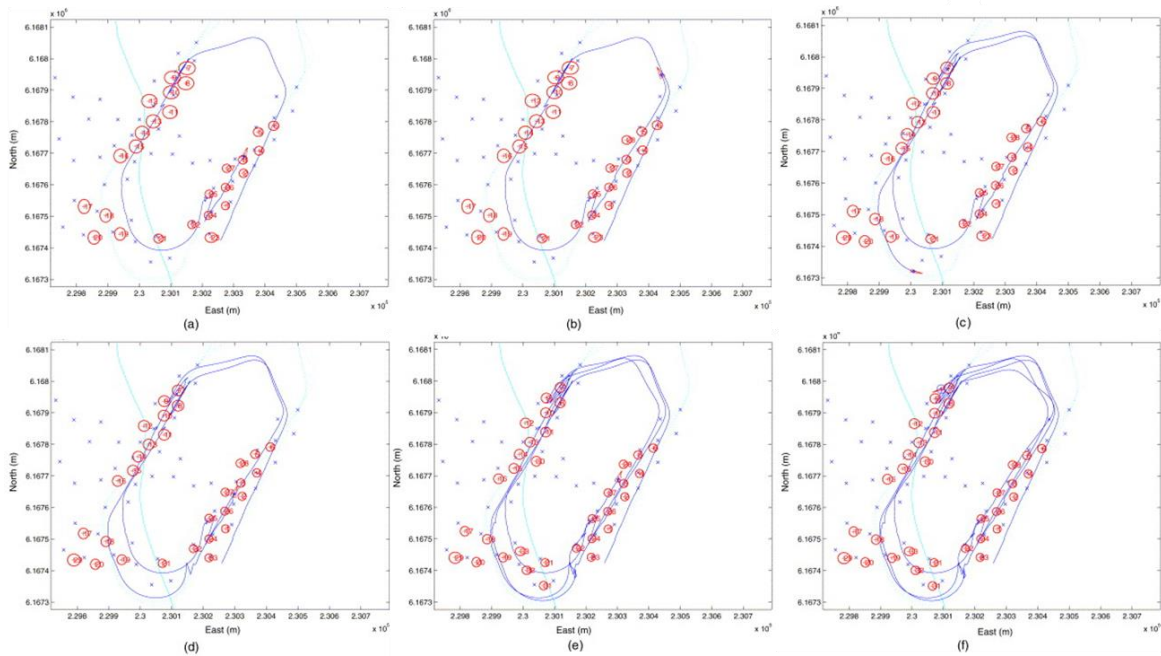


Figura 3.6. Vistas detalladas de SLAM en tiempo real durante ciclos sucesivos del vuelo en trayectoria ovalada (Kim & Sukkarieh (2007)).

En el artículo de Karam et al. (2010) se presenta un enfoque de localización basado en la fusión provista por sensores propioceptivos y exteroceptivos mediante *EKF*. El enfoque opera en dos fases distintas: predicción y actualización. En la fase de predicción, el estado anterior se usa para generar el estado actual del vehículo con el modelo del triciclo. En seguida, en la fase de actualización la observación actual se utiliza para corregir el estado previsto con fines de precisión. El enfoque fue implementado y probado en un vehículo eléctrico llamado Robucat (Figura 3.7a), vehículo limitado a una velocidad de 5 m/s y equipado con un sensor odométrico en las ruedas, una cámara y dos computadoras. La cámara proporciona imágenes en escala de grises de 512x314 píxeles. La primera computadora, llamada de bajo nivel, trata las variables de control (velocidad y ángulo de dirección) proporcionadas por los sensores propioceptivos y envía mediciones a la segunda computadora, llamada de alto nivel, a través de un enlace Ethernet. Los algoritmos de localización del vehículo (basado en la visión y la fusión) se implementan en la computadora de alto nivel en lenguaje C++ utilizando la biblioteca AROCCAM. En cada uno de los experimentos se grababa una secuencia de video de referencia y se reconstruía un mapa en 3D. En el algoritmo de reconstrucción de mapas 3D el sistema de coordenadas se define por el primer fotograma de referencia, a continuación, se establece el factor de escala local cada dos fotogramas claves utilizando información odométrica (distancia). Por último, la posición estimada por la visión se actualiza usando la información odométrica. En la Figura 3.7b se muestran las trayectorias generadas por los enfoques de corrección de factor de escala local y global en comparación con la trayectoria real (RTK-GPS).

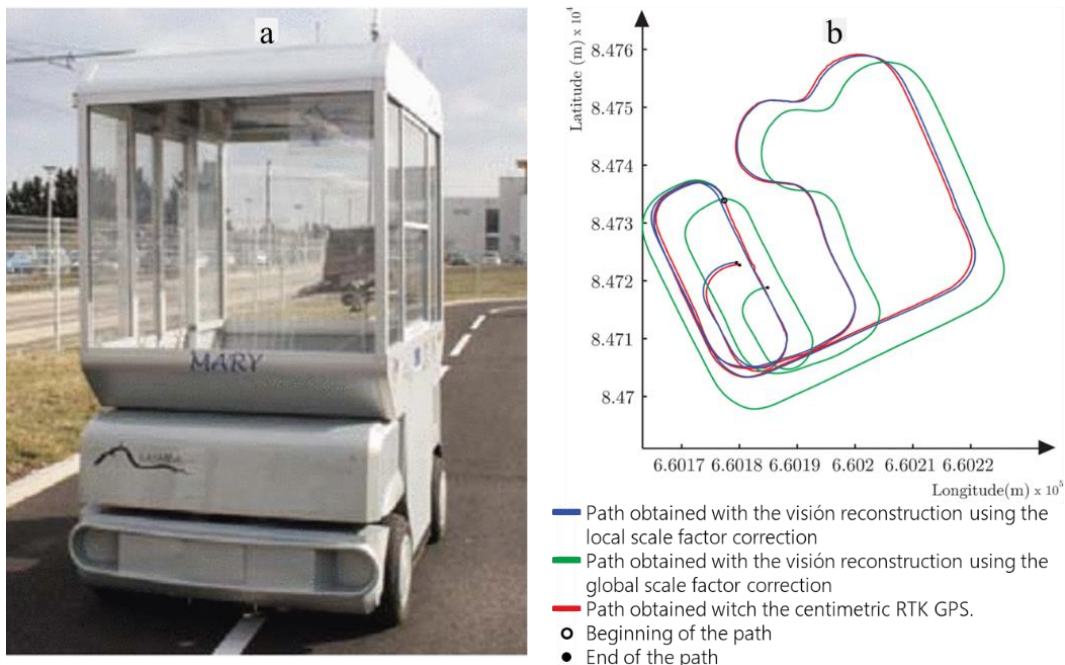


Figura 3.7. Resultados (Karam et al. (2010)). a) Robucat. b) trayectorias generadas por los enfoques con corrección de factor de escala local (línea azul) y global (línea verde) en comparación con la trayectoria obtenida por RTK-GPS (línea roja).

En el artículo de Barrett et al. (2013) se presenta el diseño de un sistema de navegación híbrido para vehículos autónomos terrestres que utilizan una unidad de medición inercial de bajo costo y un sistema de cámaras estéreo. La información del sistema de cámaras se procesa y se utiliza para estimar y corregir las derivas a largo plazo que aparecen en los datos de la unidad de medición inercial, esto se realiza con un *Filtro de Kalman Extendido* (EKF). El funcionamiento del sistema de navegación híbrido se muestra en la Figura 3.8.

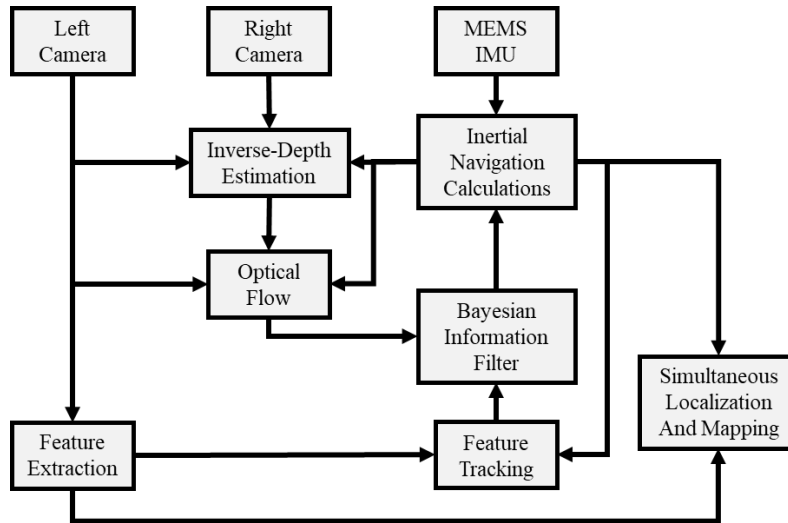


Figura 3.8. Descripción general del sistema de navegación (Barrett et al. (2013)).

En el artículo de Vincenzo et al. (2013) se presenta un algoritmo de integración de mediciones proporcionadas por sensores inerciales, GPS y un sistema de video para la estimación de la posición y altitud de un vehículo aéreo no tripulado (UAV). La fusión de datos se realiza a través de un *Filtro de Kalman "Unscented"* (UKF). Se llevaron a cabo experimentos de simulación con el fin de probar su desempeño en la estimación de la altitud del UAV en términos de ángulos yaw, pitch y roll. En los resultados se observó que la estimación de los ángulos de pitch y roll muestran mejores desempeños que la estimación de yaw, tal como se muestra en la Figura 3.9. El desempeño en la estimación del ángulo yaw puede mejorarse empleando los datos del magnetómetro. Nota: la frecuencia de muestreo del receptor GPS es de 10 Hz, 100 Hz de la IMU y 5 Hz de la cámara.

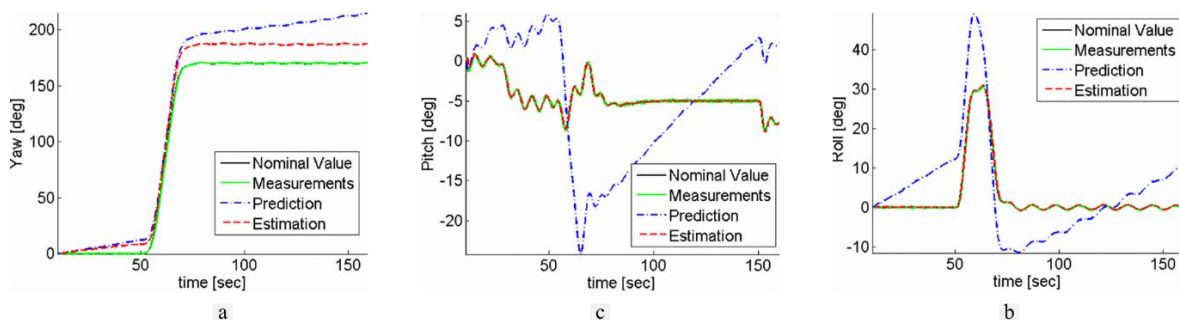


Figura 3.9. Resultados de simulación del algoritmo (Vincenzo et al. (2013)). a) guiñada. b) cabeceo. c) alabeo.

En el artículo de Babu, Cyganski, et al. (2014) se presenta un sistema de hardware portátil para el mapeo de interiores empleando un algoritmo de SLAM visual escalable asistido por Giroscopio (GA-ScaViSLAM) que mejora la robustez y precisión del algoritmo de SLAM visual escalable (ScaViSLAM) en interiores. El dispositivo de mano diseñado consiste en una unidad de procesamiento Next Unit of Computing (NUC) utilizada para recopilar datos de una IMU NavChip y de una cámara estéreo Bumblebee XB3 (Figura 3.10a). Para sincronizar la cámara con la IMU, se genera un disparador (captura de imagen estereofónica) para cada n lecturas del giroscopio, cuidando que el tiempo de exposición de la cámara este por debajo del periodo de disparo de la IMU para garantizar que se genere un cuadro de imagen para cada disparo de IMU. Además, para garantizar que no haya pérdida de datos tanto en las imágenes como las lecturas del giroscopio se obtienen utilizando hilos independientes y se almacenan para el procedimiento fuera de línea (Figura 3.10b).

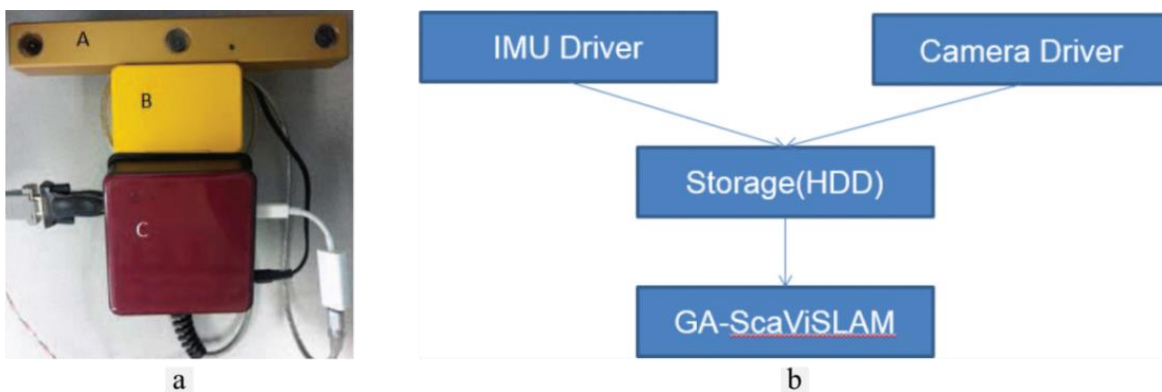


Figura 3.10. Sistema GA-ScaViSLAM (Babu et al. (2014)). a) Dispositivo de mano integrado por una Bumblebee XB3 (A), IMU NavChip (B) y una NUC (C). b) Diagrama del flujo de datos del sistema.

El sistema de visión utiliza un rastreador denso para estimar la transformación entre dos cuadros y un algoritmo Levenberg Marquardt (LM) para encontrar una transformación que minimice el error de proyección posterior de los puntos de la escena. Los datos del giroscopio son empleados para estimar la postura inicial de la cámara, e iniciar el algoritmo LM con dicha información. La adición de la información del giroscopio permite mejorar la duración de seguimiento de una característica y, por lo tanto, mejora la precisión de coincidencia. Además, las coincidencias cuyo error de reproyección es grande no se utilizan para la optimización lo que impide la propagación del error de coincidencia. Para la experimentación la frecuencia de la IMU fue de 200 Hz y 10 Hz de la cámara, por lo que se generaba una imagen estéreo por cada 20 lecturas IMU ($n = 20$). En el experimento C, se transitó por un camino con una distancia total de 77 m, con pasillos estrechos, mal iluminados y con paredes sin textura. En el cual el Ga-ScaViSLAM obtuvo un error RMS de 0.6 m, mucho menor al del ScaViSLAM con 1.85 m. En la Figura 3.11 se muestra la ruta recorrida, así como las generadas por ScaViSLAM y Ga-ScaViSLAM.

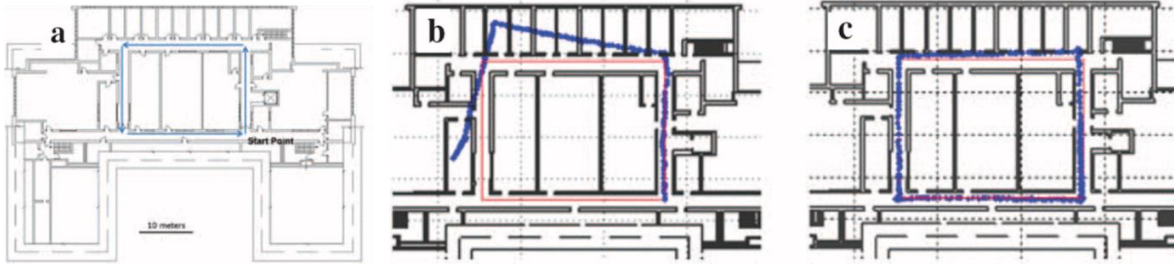


Figura 3.11. Experimento C, con datos *ground truth* en rojo y datos generados en azul (Babu et al. (2014)). a) camino recorrido. b) ScaViSLAM. c) Ga-ScaViSLAM.

En el artículo de Y. Zhang, Tan, et al. (2014) se presenta un sistema de posicionamiento en interiores que integra datos provenientes de una cámara y una IMU. En el sistema, la tasa de muestreo de la cámara y los algoritmos de extracción de características y coincidencia empleados en la estimación de la posición en presencia de movimientos lentos y rápidos se realiza de la siguiente manera: ante los primeros se emplea una frecuencia de 15 fps (cuadros por segundo) y la ejecución de Canny Edge Detector, Harris Feature Point Detector y Sift Descriptor; y ante los segundos una frecuencia de 60 fps y la ejecución del flujo óptico. En el caso de movimientos súper rápidos a corto plazo donde la cámara no capta imagen se recurre al posicionamiento IMU mediante el Filtro de Kalman. Además, ante el movimiento estático los datos visuales retroalimentan el sistema para corregir los errores producidos por los datos inerciales. La frecuencia de muestreo de la IMU utilizada en el sistema es de 100 Hz. Los experimentos realizados fueron: posicionamiento a corta distancia y posicionamiento a larga distancia. En el primero se realizó una trayectoria en línea recta con una distancia de 35 m, y en el segundo se realizó una ruta en forma de rectángulo con una longitud aproximada de 210 m. Los resultados se muestran en la Figura 3.12, donde en a) la línea azul punteada representa la ruta predefinida y la roja la trayectoria obtenida por el método. En b) la línea negra punteada es la ruta predefinida, la roja es la trayectoria obtenida mediante datos visuales y la azul es la trayectoria obtenida por el sistema desarrollado.

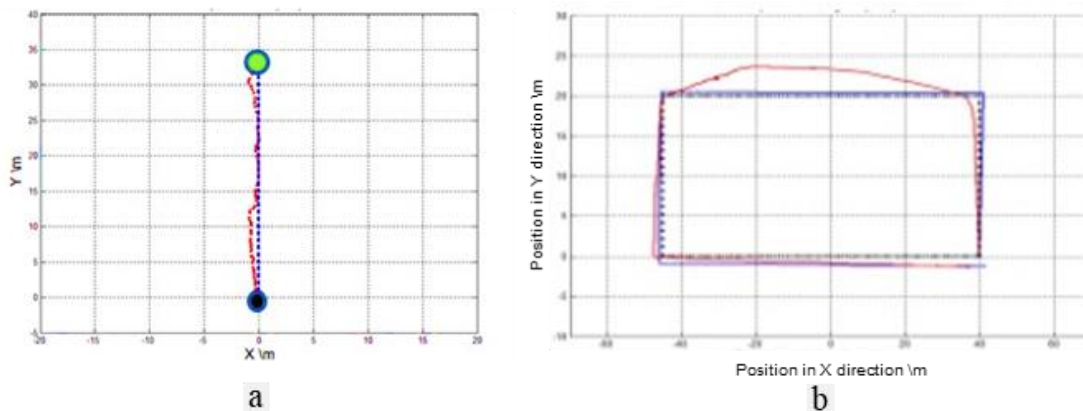


Figura 3.12. Resultados del sistema (Y. Zhang et al. (2014)). a) posicionamiento a corta distancia. b) posicionamiento a larga distancia.

En el artículo de J. Zhang et al. (2014) se presenta un algoritmo de **Localización Monte Carlo** (MLC) para robots submarinos en miniatura. El algoritmo se implementó en un pez robótico autónomo equipado con una cámara digital y una IMU 9 DoF. Para la ejecución del algoritmo MLC, primeramente, se propone un algoritmo de procesamiento de imágenes subacuáticas para detectar los puntos de referencia artificiales y calcular la distancia y el ángulo entre el robot y el punto de referencia. Después, se calcula la odometría del robot en base a los datos entregados por la IMU. Finalmente, la información de distancia y ángulo, junto con la odometría del robot extraída de la IMU, sirven como entradas para realizar la localización en línea en el MLC. Los experimentos se realizaron en una piscina de 200×300 cm, marcada con seis puntos de referencia ubicados alrededor. Uno de los experimentos realizado fue el secuestro, que consistía en cambiar de posición al pez robótico sacándolo del agua. En la Figura 3.13 se muestran las ubicaciones reales del pez robótico en el secuestro.

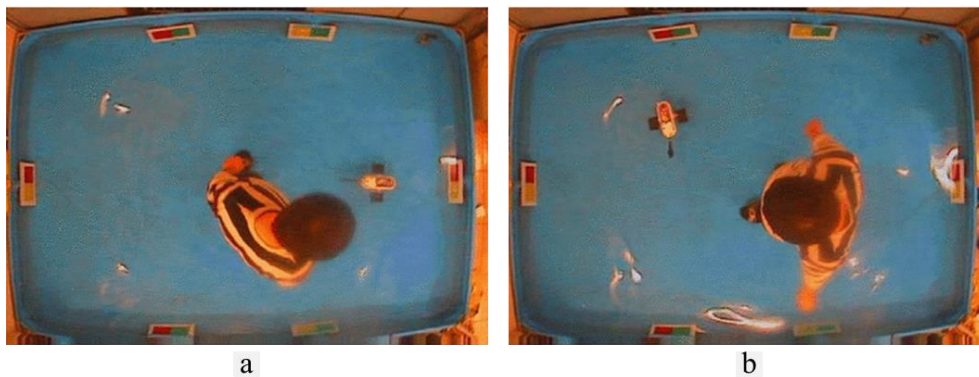


Figura 3.13. Ubicaciones reales del pez robótico en el secuestro (J. Zhang et al. (2014)). a) ubicación antes del secuestro. b) ubicación después del secuestro.

En la Figura 3.14 se muestran los resultados obtenidos por el algoritmo MLC durante el experimento del secuestro del pez robótico. En las Figuras 3.14 a) y b) el punto A, es la primera posición; la etapa AB presenta estimaciones estables con una ligera fluctuación; en el punto B, el robot es sacado del agua; la etapa BC es causada por el movimiento inadvertido del robot cuando es agarrado por el experimentador; la etapa CD es el proceso del secuestro; y en el punto D, el robot es llevado al agua.

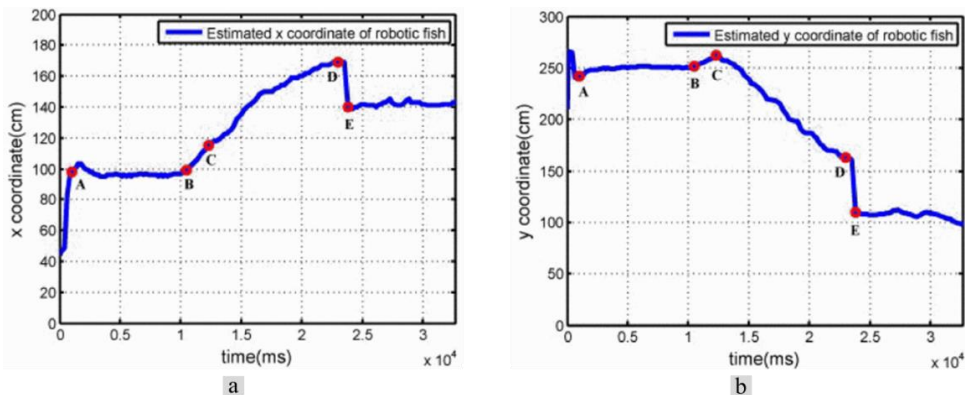


Figura 3.14. Estimación de la ubicación del pez robótico en el secuestro (J. Zhang et al. (2014)). a) estimación de coordenadas x . b) estimación de coordenadas y .

En el artículo de Tomažič & Škrjanc (2015) se presenta un sistema de localización peatonal implementado en un teléfono inteligente (Samsung Galaxy S4). El núcleo del sistema es la odometría visual monocular, que permite determinar con precisión las posiciones relativas, a menos que haya demasiados giros bruscos en la ruta a navegar. Como sistema complementario se implementó el sistema de navegación inercial compuesto básicamente de una brújula digital que determina el rumbo absoluto y un podómetro que determina la longitud de la ruta recorrida contando los pasos. Con el fin de obtener un sistema de localización aún más robusto y preciso, se fusionan ambos enfoques mediante el uso del *Filtro de Kalman Extendido* (EKF). En la Figura 3.15 se muestra la trayectoria obtenida mediante la fusión de odometría visual y el sistema de navegación inercial. La trayectoria corresponde al recorrido peatonal de un circuito cerrado con una longitud de 27 m.

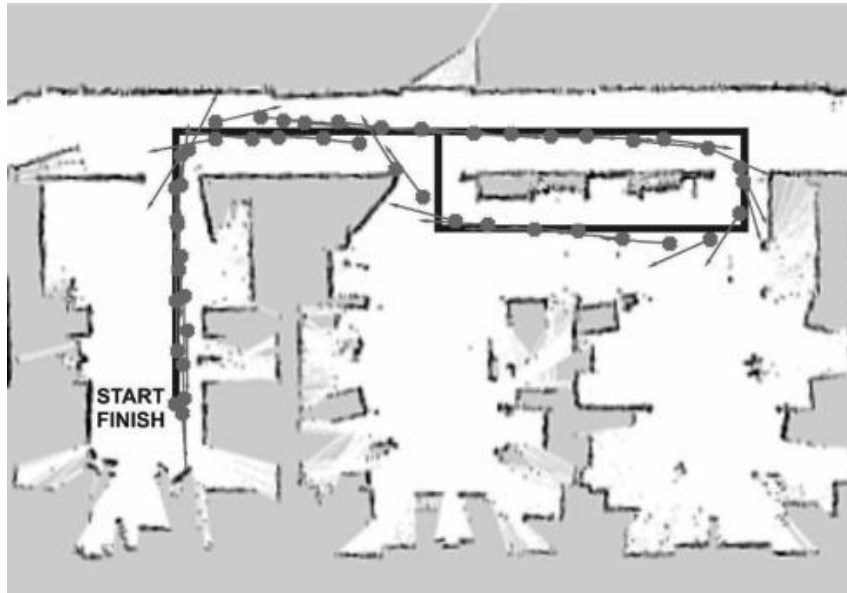


Figura 3.15. Trayectoria peatonal obtenida mediante el sistema de localización (Tomažič & Škrjanc (2015)).

En el artículo de L. Zhang, Xiong, et al. (2016) se presenta un método para la estimación de la orientación de vehículos aéreos no tripulados (UAV) equipados con una cámara y una IMU. El método consiste en tres módulos: odometría inercial, medición de orientación y corrección de la orientación. El primer módulo estima la odometría del UAV mediante la integración de datos inerciales. El segundo módulo estima la orientación del UAV mediante el flujo óptico, obtenido de los datos visuales. Por último, el tercer módulo corrige los errores de orientación en la navegación inercial en base a las mediciones visuales mediante el *Filtro de Kalman*. La frecuencia de muestreo de la IMU es mucho más rápida que la de la cámara, por lo que los datos inerciales y visuales se alinearon en el tiempo. Por lo tanto, la corrección de los errores en las mediciones inerciales se realizaba cuando se disponía de mediciones visuales. En la Figura 3.16, se muestran las condiciones y los resultados obtenidos durante una prueba de simulación. La prueba consistía en volar hacia adelante en estilo senoidal durante un periodo de 200 segundos. Los resultados de la simulación muestran un mejor rendimiento en comparación con una navegación puramente inercial.

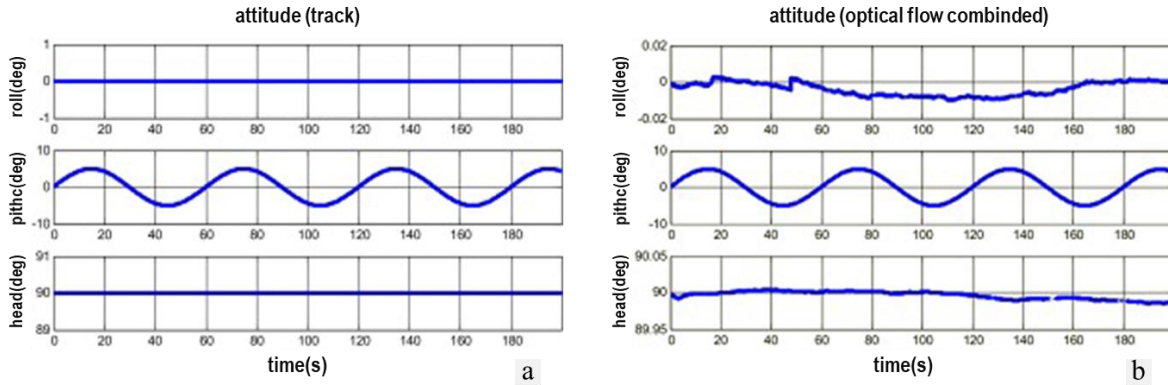


Figura 3.16. Resultados de simulación (L. Zhang et al. (2016)). a) pista teórica. b) valores calculados mediante la combinación de IMU con flujo óptico.

En el artículo de X. Liu, Chen, et al. (2016) se presenta un sistema de navegación inercial asistido por imágenes para vehículos no tripulados (UAV). El UAV es equipado con una IMU y varios sensores de flujo óptico con ubicaciones diferentes para detectar el flujo óptico desde diferentes direcciones. La estructura del sistema propuesta se muestra mediante un diagrama de bloques en la Figura 3.17. El bloque SINS está conformado por la IMU encargada de proporcionar mediciones de orientación, posición y velocidad. El bloque OFS es integrado por los sensores de flujo óptico, encargado de proporcionar el flujo óptico medido. En el bloque Predicción del Flujo óptico, se realiza la predicción del flujo óptico utilizando las mediciones del SINS. La observación del flujo óptico se obtiene al restar el flujo óptico medido por el OFS al flujo óptico predicho por el SINS. Por último, el bloque EKF, es el encargado de fusionar la información de flujo óptico e inercial mediante un *Filtro de Kalman Extendido* (EKF), para corregir los errores del sistema de navegación inercial.

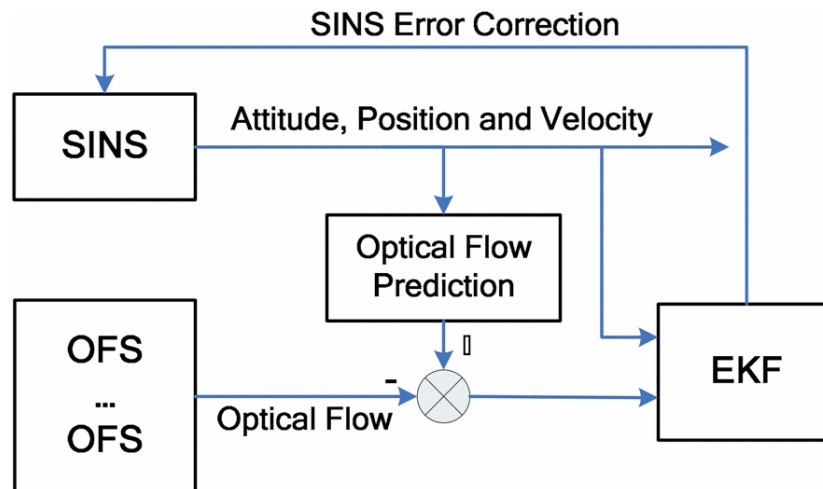


Figura 3.17. Estructura del sistema de navegación OFS/SINS (X. Liu et al. (2016)).

En la Tabla 3.1 se muestran las características y técnicas empleadas en cada uno de los trabajos revisados dentro del estado del arte (en orden cronológico).

Tabla 3.1. Tabla comparativa del estado del arte.

Artículo	Aplicación	Datos de navegación	Métodos de procesamiento	Métodos de fusión
Kim & Sukkarieh (2007)	Vehículo aéreo no tripulado	IMU 6 DoF Visión	<ul style="list-style-type: none"> (Datos inerciales crudos) Umbral de intensidad 	Filtro de Kalman Extendido (EKF)
Karam et al. (2010)	Vehículo autónomo terrestre	Sensor odométrico Visión	<ul style="list-style-type: none"> Tricycle model Harris corner detector 	Filtro de Kalman Extendido (EKF)
Barrett et al. (2013)	Vehículo autónomo terrestre	IMU 6 DoF Visión	<ul style="list-style-type: none"> Ecuaciones INS (Integración) Flujo óptico 	Filtro de Kalman Extendido (EKF)
Vincenzo et al. (2013)	Vehículo aéreo	IMU GPS Visión	<ul style="list-style-type: none"> Flujo óptico 	Filtro Kalman “Unscented” (UKF)
Babu et al. (2014)	Peatón	Giroscopio Visión (estéreo)	<ul style="list-style-type: none"> Cuaterniones Rastreador denso con Levenberg Marquardt 	
Y. Zhang et al. (2014)	Vehículo autónomo terrestre	IMU Visión	<ul style="list-style-type: none"> Ecuaciones INS (Integración) Flujo óptico 	
J. Zhang et al. (2014)	Robot acuático	IMU 9 DoF Visión	<ul style="list-style-type: none"> Ecuaciones INS (Integración) Segmentación basada en el modelo HSV, para la detección y seguimiento de puntos de referencia (landmarks). 	Localización de Monte Carlo (MLC)
Elarabi & Suprem (2015)	Teléfono inteligente	GPS IMU 9 DoF	<ul style="list-style-type: none"> Ecuaciones INS (Podómetro) 	Filtro Savitzky-Golay

Artículo	Aplicación	Datos de navegación	Métodos de procesamiento	Métodos de fusión
Tomazič & Škrjanc (2015)	Teléfono inteligente	IMU 9 DoF Visión	<ul style="list-style-type: none"> • Brújula digital • Podómetro • Flujo óptico 	Filtro de Kalman Extendido (EKF)
Ibrahim & Moselhi (2016)	Peatón	IMU 9 DoF Barómetro	<ul style="list-style-type: none"> • Ecuaciones INS (Integración) 	Matriz de Cosenos Directores (DCM) Filtro de Kalman Extendido (EKF)
L. Zhang et al. (2016)	Vehículo aéreo no tripulado	IMU Visión	<ul style="list-style-type: none"> • Ecuaciones INS (Integración) • Flujo óptico 	Filtro de Kalman
X. Liu et al. (2016)	Vehículo aéreo no tripulado	IMU 6 DoF Visión	<ul style="list-style-type: none"> • Ecuaciones INS (Integración) • Flujo óptico 	Filtro de Kalman Extendido (EKF)
Suprem et al. (2017)	Teléfono inteligente	GPS IMU 9 DoF	<ul style="list-style-type: none"> • Ecuaciones INS (Podómetro) 	Filtro Savitzky-Golay
Suwandi et al. (2017)	Vehículo autónomo terrestre	GPS IMU 9 DoF	<ul style="list-style-type: none"> • Ecuaciones INS (Integración) 	Filtro de Kalman

Continuación de la Tabla 3.1.

3.4 Discusión

De acuerdo con la revisión de los artículos que conforman el estado del arte, los sistemas que fusionan datos inerciales con visuales se desarrollan para vehículos terrestres, aéreos y acuáticos como para su uso en peatones. Estos son implementados sobre sistemas de visión monocular o estereoscópica y utilizados tanto en ambientes de interiores como de exteriores.

Los sistemas que integran el primer grupo son utilizados para estimar la posición del vehículo por tramos cortos para evitar el crecimiento del error en el sistema IMU. En el trabajo de Ibrahim & Moselhi (2016) se emplea un algoritmo de Matriz de Coseno Directores para fusionar los datos inerciales y un Filtro de Kalman Extendido para estimar la posición en base a los datos inerciales fusionados. Los sistemas del segundo grupo se desarrollan con el objetivo de aumentar la precisión en las estimaciones del sistema GPS, así como el poder navegar por entornos de GPS denegado. Por lo tanto, el GPS es el sistema principal y el INS el sistema complementario. En el trabajo de Suprem et al. (2017) las mediciones de orientación y posición del INS son utilizadas para llenar los huecos en los datos de navegación GPS, al final los valores resultantes se pasan a través de un Filtro de Savitzky-Golay para eliminar los elementos ruidosos. Mientras que Suwandi et al. (2017) utiliza un Filtro de Kalman para combinar los datos GPS/IMU. Por último, los sistemas del tercer grupo aprovechan la confiabilidad de las mediciones visuales y la capacidad de seguimiento ante altas velocidades de los sensores inerciales. De acuerdo con la clasificación de sistemas ligera y estrechamente acoplados, la gran mayoría de los trabajos estudiados corresponden a ligeramente acoplados y sólo uno de los trabajos figura como estrechamente acoplados.

El trabajo de Kim & Sukkariéh (2007) es un sistema estrechamente acoplado que mediante un filtro EKF procede a través de predicción y actualización de los estados del vehículo y las ubicaciones de las características estima la posición del vehículo. Mediante la re-observación sucesiva de las características la precisión de navegación y el mapeo mejoran continuamente. En los sistemas ligeramente acoplados comúnmente se utiliza el Filtro de Kalman y sus derivados (Filtro de Kalman Extendido (EKF) y Filtro de Kalman Unscented (UKF)). Entre ellos se encontraron los siguientes trabajos: L. Zhang et al. (2016) que estima la odometría a partir de datos inerciales y utiliza el flujo óptico para estimar y corregir los errores de orientación en la navegación inercial mediante un Filtro de Kalman. Barrett et al. (2013), X. Liu et al. (2016) y Karam et al. (2010) utilizan mediciones visuales para estimar y corregir las derivas del INS mediante un Filtro de Kalman Extendido. Los dos primeros emplean la información del flujo óptico, mientras que el tercero emplea Harris corner detector. Tomažič & Škrjanc (2015) también fusiona el sistema de odometría visual monocular basado en el flujo óptico con las mediciones de rumbo y longitud del INS mediante EKF con el fin de obtener un sistema de localización aún más robusto y preciso. Y Vincenzo et al. (2013) que incluye mediciones GPS a la fusión de datos inerciales y visuales a través de un filtro UKF. Además de los sistemas que emplean el filtro de Kalman, también se encontraron sistemas

que emplean filtros bayesianos como en el trabajo de J. Zhang et al. (2014), en el cual se ejecuta un algoritmo de Localización Monte Carlo, cuyas entradas son información de distancia y ángulo entre el robot y el punto de referencia proporcionados por visión y la odometría del robot obtenida a partir de datos de la IMU.

Dos trabajos diferentes a los anteriormente comentados son los sistemas desarrollados por Y. Zhang et al. (2014) que trabaja dependiendo de la velocidad del movimiento y Babu et al. (2014) que no utiliza ningún Filtro de Kalman. En el primer trabajo, tanto los datos visuales como inerciales son utilizados de forma independiente en diferentes momentos dependiendo de la velocidad del movimiento. En el caso de movimientos lentos y rápidos estima la odometría mediante datos visuales, ante los primeros ejecuta Canny Edge Detector, Harris Feature Point Detector y Sift Descriptor, y ante los segundos ejecuta el flujo óptico. En el caso de movimientos super rápidos a corta distancia recurre a la odometría inercial mediante un Filtro de Kalman. Además, ante el movimiento estático los errores inerciales son corregidos mediante datos visuales. Y en el caso del movimiento estático, se utilizan datos visuales para corregir los errores del INS. Y en el segundo trabajo, se emplea un giroscopio para estimar la posición inicial del vehículo e iniciar el algoritmo LM permitiendo encontrar una transformación que minimice el error de proyección posterior de los puntos de la escena. Con la adición del giroscopio se mejora la duración de seguimiento de una característica y, por lo tanto, mejora la precisión de coincidencia.

En este trabajo se desarrolló un sistema de navegación inercial asistido por visión que utiliza el enfoque ligeramente acoplado. Dicho sistema trabaja con datos inerciales y la información del flujo óptico como en los trabajos de Barrett et al. (2013), Vincenzo et al. (2013), L. Zhang et al. (2016) y X. Liu et al. (2016) con la diferencia de que no utiliza Filtros de Kalman para la fusión de información, sino que trabaja de forma parecida a los trabajos de Babu et al. (2014) y Y. Zhang et al. (2014), realizando un acoplamiento de datos, que en base a la información visual determina en qué momento integrar la información inercial. El sistema desarrollado emplea el algoritmo presentado por Elarabi & Suprem (2015) para fusionar los datos de la IMU, con la diferencia que utiliza un giroscopio en lugar de un magnetómetro, por lo que, las ecuaciones son diferentes. Además, para que no haya pérdida de datos, tanto las imágenes como las lecturas del IMU se obtienen utilizando hilos independientes, como en Babu et al. (2014). Y como en Y. Zhang et al. (2014), el movimiento estático de los datos visuales es utilizado para evitar el crecimiento de la deriva.

Es importante destacar que la gran mayoría de trabajos desarrollados para la fusión de datos inerciales con visuales utilizan el enfoque ligeramente acoplados, esto debido a su ejecución en tiempo real y su búsqueda por mantenerse en un bajo costo. Ya que los sistemas estrechamente acoplados demandan más poder de cómputo, lo que demanda tecnología de mayor costo. Además, con el avance en la tecnología tanto en precisión en los sensores inerciales MEMS como la calidad en los sistemas de visión se ven muy prometedores.

Capítulo 4

Diseño y construcción del prototipo

En este capítulo se presenta el diseño y la construcción del prototipo utilizado para la evaluación de las técnicas de odometría inercial y detección de movimiento. El prototipo está equipado por microcontroladores y sensores visuales e inerciales, necesarios para la estimación de su orientación y posición durante la navegación.

4.1 Diseño del prototipo

El propósito del diseño del prototipo es poder ser montado en robots móviles para brindarles información sobre su navegación. La intención de montaje en robots móviles demanda el menor peso y tamaño posible. Por lo que, se tenía que contar con componentes pequeños y ligeros, que cubrieran los requerimientos necesarios para alcanzar el objetivo del proyecto, sin excederse en costos. Entre los principales componentes se encuentra una IMU Adafruit 9 DoF, un Arduino DUE, una Raspberry Pi 3 y una Cámara pi v2. Un dato interesante de la arquitectura es que funcionaría sin la necesidad del arduino. Sin embargo, el arduino se incorpora con el objetivo de distribuir y reducir la carga de procesamiento a la Raspberry Pi. Además, se pretende que la arquitectura del prototipo se utilice en trabajos futuros, por lo que, la distribución de trabajo beneficiaría a los proyectos en que se desee incorporar más componentes como motores u otros sensores. La carcasa del prototipo fue construida a base de acrílico transparente, que en conjunto con todos los componentes pesa alrededor de 0.30 kg. Además, gracias a las características de cada uno de los componentes, es fácil acceder al sistema del prototipo mediante una conexión remota.

A continuación, se describen los componentes principales del prototipo, se detallan las conexiones establecidas, se muestra la carcasa construida y se presenta el prototipo final.

4.2 Componentes principales

Las características y especificaciones de los componentes principales que integran el prototipo se muestran a continuación.

4.2.1 Raspberry Pi 3

Placa computadora de bajo costo desarrollada en Reino Unido por la Fundación Raspberry Pi, construida con base en el procesador BCM2837 ARMV8 de 64 bits con 1.2 GHz de velocidad. Para su funcionamiento, es necesario un medio de almacenamiento (microSD) con NOOBS y una fuente de alimentación de 2.1 A (Raspberry Pi Org. (2018)b).

Como se observa en la Figura 4.1, la Raspberry Pi 3 está integrada de todos, o la mayoría de los componentes de una computadora. Además, dispone de pines de conexión de entradas y salidas digitales para sensores o actuadores.

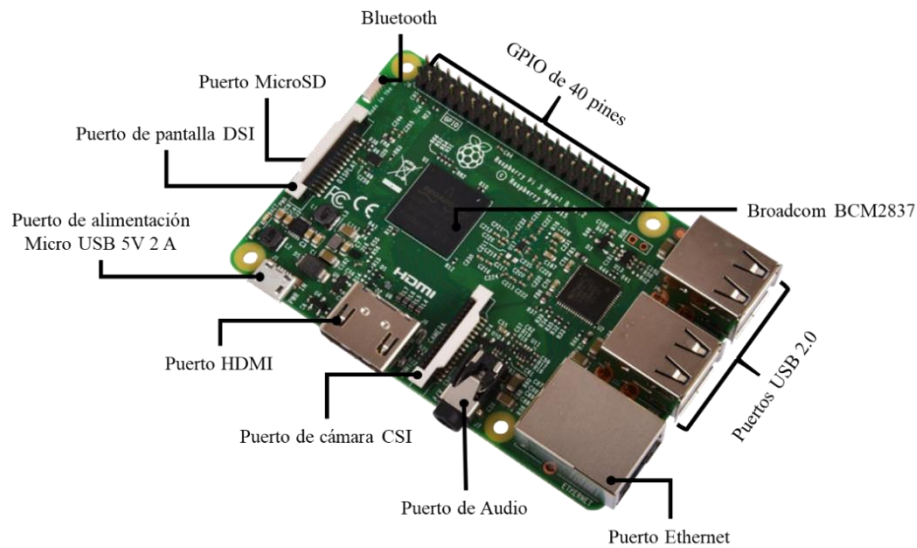


Figura 4.1. Raspberry Pi 3 modelo B (Raspberry Pi Org. (2018)b).

En la Tabla 4.1 se detallan las especificaciones técnicas de la Raspberry Pi 3.

Tabla 4.1. Especificaciones técnicas de la Raspberry Pi 3.

Especificaciones técnicas	
Procesador	Quad Core 1.2GHz Broadcom BCM2837 de 64 bits
RAM	1GB
Conectividad	<ul style="list-style-type: none"> ▪ Puerto Ethernet 10/100 Baset ▪ LAN inalámbrica BCM43438 ▪ Bluetooth Low Energy (BLE) ▪ GPIO de 40 pines ▪ 4 puertos USB 2.0 ▪ Salida de audio y video ▪ HDMI ▪ Puerto de cámara CSI ▪ Puerto de pantalla DSI ▪ Puerto MicroSD
Dimensiones	86.9mm x 58.5mm x 19.1mm / 3.4" x 2.3" x 0.8"
Peso	41.2g / 1.5oz

4.2.2 IMU Adafruit 9 DoF

Unidad de medición inercial de 9 DoF (9 grados de libertad) que combina un L3GD20 giroscopio y un LSM303DLHC acelerómetro y magnetómetro, permitiendo capturar nueve tipos de movimientos o datos relacionados con la orientación: 3 ejes de datos del acelerómetro (aceleración lineal), 3 ejes de datos del giroscopio (velocidad angular) y 3 ejes del magnetómetro (orientación magnética) (Adafruit Industries (2018)). En la Figura 4.2 se señalan los sensores que integran la Adafruit 9 DoF.

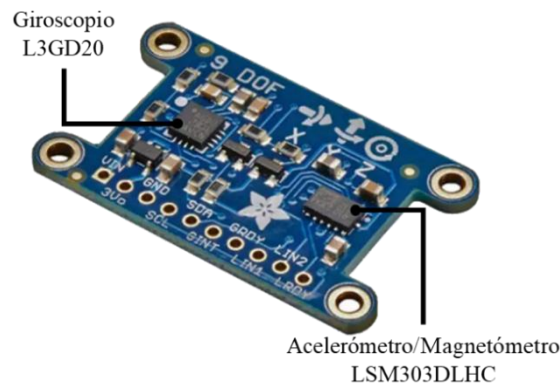


Figura 4.2. IMU Adafruit 9 DoF (Adafruit Industries (2018)).

En la Tabla 4.2 se detallan las especificaciones técnicas de la IMU Adafruit 9 DoF.

Tabla 4.2. Especificaciones técnicas de la IMU Adafruit 9 DoF.

Especificaciones técnicas	
Comunicación	I^2C , utiliza sólo dos cables de comunicación (SDA y SCL)
Dimensiones	38mm x 23mm/ 1.5" 0.9"
Peso	2.8g

4.2.3 Arduino DUE

Placa electrónica basada en el microcontrolador ARM Cortex-M3 de 32 bits programable a través del IDE de Arduino con una versión superior a la 1.5.0 (Arduino Org (2018)a). A diferencia de la mayoría de las placas Arduino, la placa Arduino DUE funciona a 3.3 V y basta con conectarlo a una computadora con un cable Micro USB o alimentarlo con un adaptador CA/CC.

Arduino DUE cuenta con dos pines TWI: SDA 20 y SCL 21, utilizadas para comunicarse con dispositivos I^2C . En la Figura 4.3 se muestra la placa Arduino DUE, en ella se señalan los pines SDA y SCL, el microcontrolador y los puertos de alimentación y programación.

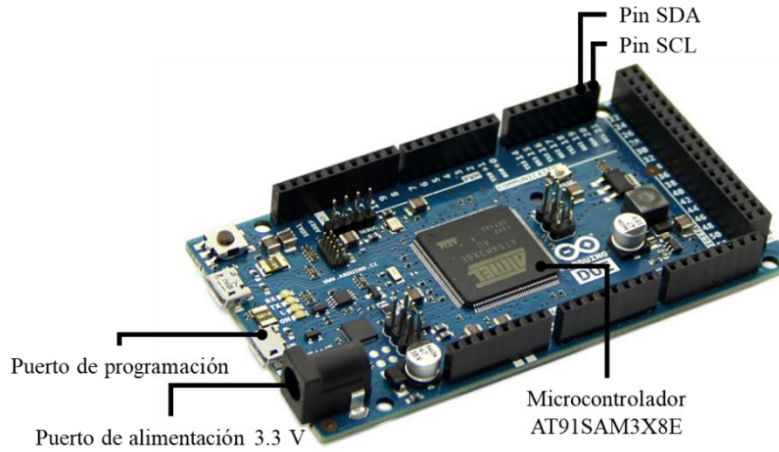


Figura 4.3. Arduino DUE (Arduino Org (2018)a).

En la Tabla 4.3 se detallan las especificaciones técnicas del Arduino DUE.

Tabla 4.3. Especificaciones técnicas del Arduino DUE.

Especificaciones técnicas	
Microcontrolador	AT91SAM3X8E
Voltaje de funcionamiento	3.3 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límites)	6-20 V
Pines digitales I/O	54 pines (12 proveen salida PWM)
Pines de entrada analógica	12
Pines de salida analógica	2 (DAC)
Corriente máxima (entradas y salidas)	130 mA
Corriente máxima salida regulada de 3.3 V	800 mA
Corriente máxima salida regulada de 5 V	800 mA
Memoria Flash	512 KB
SRAM	96 KB (en dos bancos de: 64 KB y 32 KB)
Velocidad del reloj	84 MHz
Dimensiones	101.52 mm x 53.3 mm
Peso	36 g

4.2.4 Cámara Pi v2





Módulo de cámara con un sensor de imagen Sony IMX219 de megapíxeles y lente de foco fijo. Es capaz de capturar imágenes y videos sin sonidos. Se conecta mediante un cable plano de 15 cm al puerto CSI en la Raspberry Pi (Raspberry Pi Org. (2018)a). En la Figura 4.4 se muestra la Cámara Pi v2.



Figura 4.4. Cámara Pi v2 (Raspberry Pi Org. (2018)a).

En la Tabla 4.5 se muestran las especificaciones del esquema de conexión correspondiente a la Figura 4.5.

Tabla 4.5. Especificaciones de conexión entre la IMU Adafruit 9 DoF y Arduino DUE.

Especificación de conexión			
Arduino Due		Adafruit 9 DOF	
SCL 21		SCL	
SDA 20		SDA	
3.3 V		VIN	
GND		GND	

Las bibliotecas necesarias para que Arduino haga uso de la IMU Adafruit 9 DoF son:

- *Adafruit Sensor Library*
- *Adafruit LSM303 Library*
- *Adafruit L3GD20 Library*
- *Adafruit 9DoF*

Dichas bibliotecas pueden ser descargadas desde la página oficial de Adafruit: <https://learn.adafruit.com/adafruit-9-dof-imu-breakout/software>.

4.3.2 Conexión entre Cámara Pi v2 y Raspberry Pi 3

La conexión de la Cámara Pi v2 a la Raspberry Pi 3, se realiza mediante el puerto CSI. Es importante verificar que la cámara esté conectada en la orientación correcta, de modo que el lado del cable con las letras verdes este hacia arriba, tal como se muestra en la Figura 4.6.

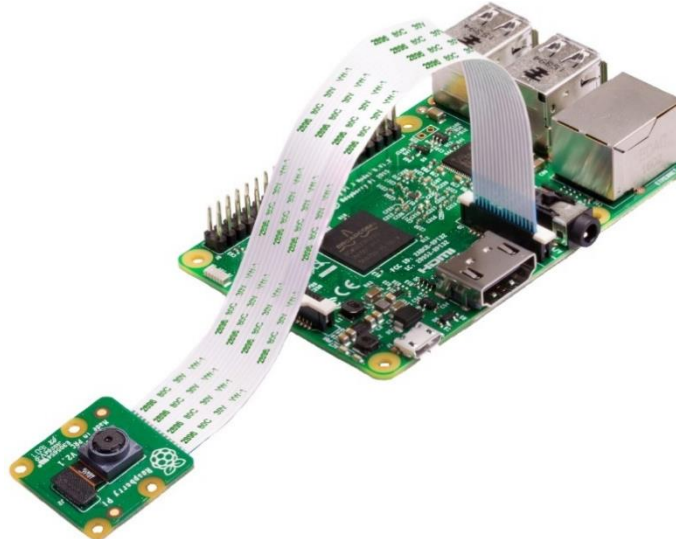


Figura 4.6. Conexión de la Cámara Pi v2 a la Raspberry Pi 3.

Una de las bibliotecas que permiten utilizar la Cámara Pi v2 en la Raspberry Pi bajo licencia BSD (Berkeley Software Distribution) es *Raspicam*. Esta biblioteca proporciona la clase

Raspicam_CV para un fácil control de la cámara con OpenCV; además, es de fácil compilación e instalación usando cmake. El proceso de instalación de la biblioteca Raspicam se encuentra en el Anexo 1.

4.3.3 Conexión entre Arduino DUE y Raspberry Pi 3

La conexión entre las placas electrónicas se realiza mediante cualquier puerto USB en la Raspberry Pi 3 y el puerto Micro USB en el Arduino DUE, para esto se requiere disponer de un cable de datos USB a Micro USB. En la Figura 4.7 se muestra el esquema de conexión entre Arduino DUE y la Raspberry Pi 3.

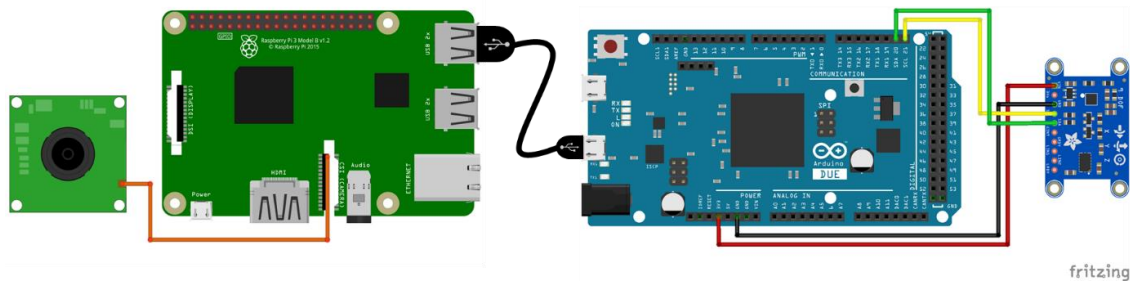


Figura 4.7. Esquema de conexión entre el Arduino DUE y la Raspberry Pi 3.

La comunicación entre ambas tarjetas se establece mediante el uso de *Libqt5serialport5*, módulo complementario para la biblioteca qt5 que proporciona las funcionalidades básicas que incluyen la configuración, operaciones entrada/salidas, obtener y establecer las señales de control de la interfaz RS-232. La clase base del módulo es *QSerialPort* y proporciona un conjunto de métodos y propiedades básicas para acceder a recursos de los puertos series. El proceso de instalación de *Libqt5serialport5* se encuentra en el Anexo 2.

4.3.4 Conexión entre la computadora y la Raspberry Pi 3

La conexión se realiza de forma remota a través de la red wifi mediante el módulo de conexión inalámbrica disponible en la Raspberry Pi 3. Para visualizar y controlar el escritorio de la Raspberry desde la computadora es necesario instalar *TightVNC*, paquete de software de control remoto gratuito que utiliza el protocolo VNC y permite ver y controlar el escritorio de una máquina remota desde otra computadora o dispositivo móvil. El proceso de instalación y configuración de *TightVNC* se encuentra en el Anexo 3.

4.4 Carcasa

La carcasa se diseñó de forma que el prototipo mantuviera el menor tamaño posible, por lo que, en base a las características de cada uno de los componentes necesarios para el procesamiento de la información se determinó que la carcasa asumiera una forma de prisma rectangular con dimensiones de 14 cm de largo, 11 cm de ancho y 5.03 cm de altura.

Tal como se muestra en la Figura 4.8, el diseño es simple y de fácil ensamble, cinco de sus caras están fijas, mientras que la sexta cara debe ser atornillada para cerrar completamente el prisma.

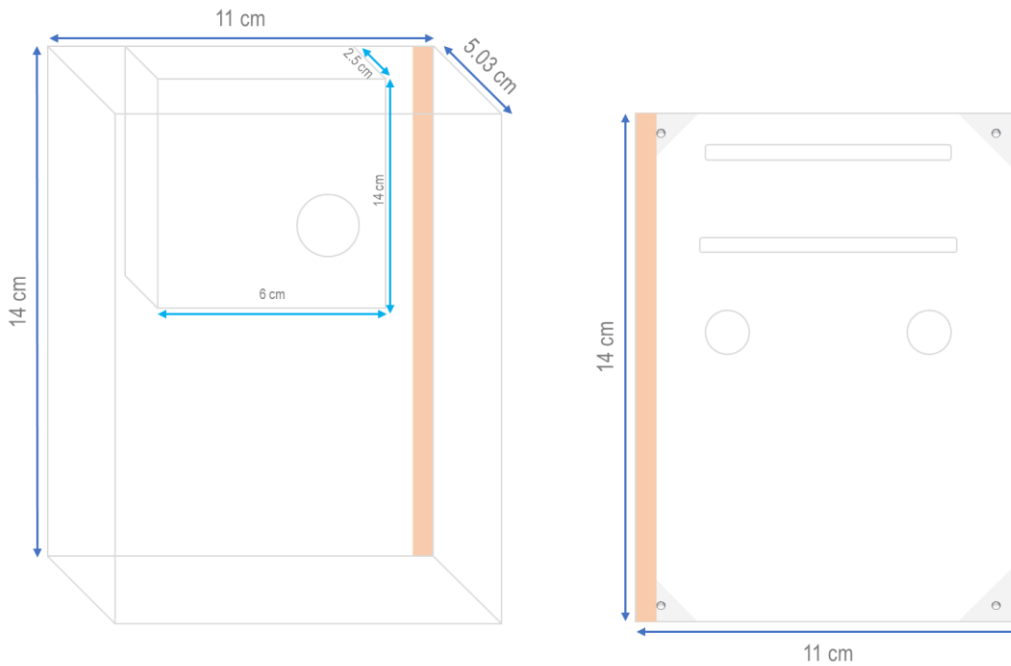


Figura 4.8. Diseño de la carcasa del prototipo.

4.5 Construcción

La carcasa del prototipo está construida a base de acrílico transparente de 2 cm de ancho, cuatro de sus caras tienen orificios que permiten la entrada y/o salida del aire para reducir el calentamiento de las tarjetas. Además, cada uno de los componentes que se encuentran en su interior son atornillados para evitar contratiempos durante la navegación realizada por el prototipo. En la Figura 4.9, se muestra el prototipo construido.

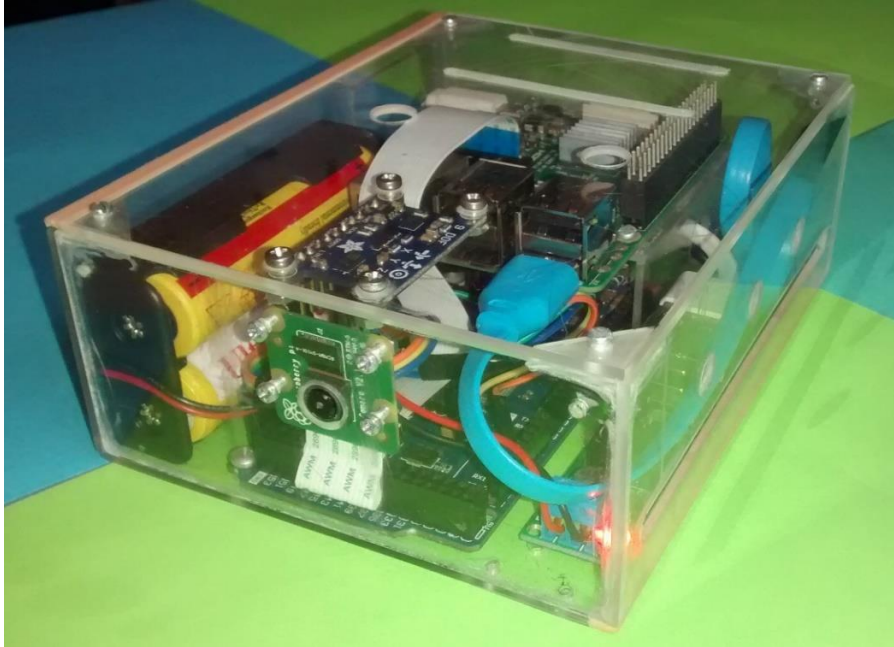


Figura 4.9. Prototipo construido.

En la Figura 4.10 en a) se muestra la IMU Adafruit 9 DoF colocada en la cara superior (tapa) y en b) la Cámara Pi vs instalada en la cara frontal.

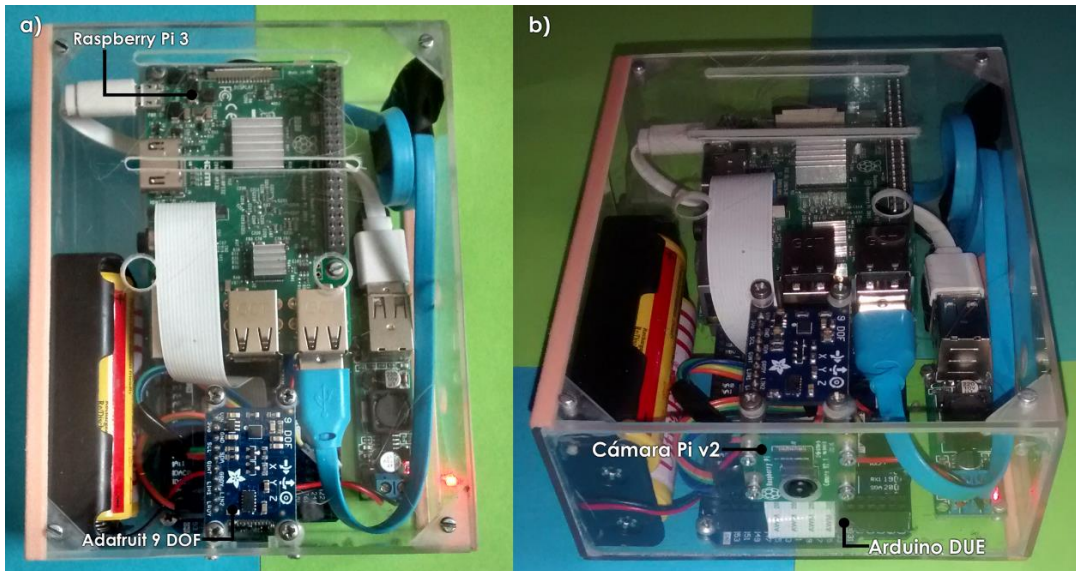


Figura 4.10. Ubicaciones de los sensores del prototipo. a) IMU 9 DoF. b) Cámara Pi v2.

4.6 Discusión

Los componentes principales del prototipo son la Raspberry Pi 3, la IMU Adafruit 9 DoF, el Arduino DUE y la Cámara Pi v2. Sin embargo, es necesario mencionar que también se utilizaron un par de baterías recargables BRC 18650 de 3.7 Volts, un módulo regulador y un porta pilas doble, componentes utilizados para alimentar el prototipo. Además, un cable USB a Micro USB para alimentar la Raspberry Pi desde el módulo de las baterías.

El prototipo construido puede ser montado sobre robots móviles pequeños sin contratiempos relacionados con deslizamientos por parte de los componentes en el interior de la carcasa, esto debido a que cada uno de ellos se encuentran atornillados a la carcasa.

Capítulo 5

Desarrollo del sistema

En este capítulo se presenta la arquitectura general del sistema de navegación desarrollado, se describen las técnicas de odometría inercial y detección de movimiento implementadas para obtener los datos de navegación del prototipo. Además, se explica el proceso de acoplamiento de datos inerciales con los visuales y se presenta la interfaz gráfica desarrollada para el control del sistema desde una conexión remota.

5.1 Arquitectura del sistema

El sistema de navegación inercial asistido por visión se desarrolló utilizando el IDE de Qt Creator 5.3.2 (Qt io (2018)) instalado sobre el sistema operativo Raspbian Stretch (Raspberry Pi Org (2018)) de la Raspberry pi 3. El sistema consiste en tres módulos: inercial, visual y fusión de datos, el primero desarrollado tanto en la Raspberry Pi como en el Arduino DUE y los dos últimos desarrollados únicamente en la Raspberry Pi.

La arquitectura del sistema de navegación se muestra en la Figura 5.1, en la cual se observan los tres módulos que integran al sistema. Por un lado, las funciones del módulo inercial son la estimación de la pose inicial y la estimación de la orientación y posición durante la navegación. Para esto, primeramente, se calibran y filtran las lecturas entregadas por la IMU, después las aceleraciones lineales se emplean para la estimación de la pose inicial del prototipo mediante el cálculo de los ángulos de inclinación. En seguida las velocidades angulares se integran para obtener los ángulos de rotación (orientación), estos ángulos son utilizados para eliminar la gravedad de las aceleraciones lineales que se integran doblemente para obtener el desplazamiento (posición) del prototipo desde el punto inicial hasta el punto final del recorrido. Por otro lado, las funciones del módulo visual son la detección del movimiento y su dirección. Para esto, se calcula el flujo óptico a partir de los puntos característicos detectados en las imágenes capturadas por la cámara. A partir del flujo óptico,

se obtienen vectores de desplazamiento, que permiten determinar si el prototipo se encuentra en estado de “reposo” o “movimiento”, así como estimar la dirección del movimiento. Y juntando los dos lados, inicia la función del módulo fusión de datos que es revisar las salidas del módulo visual y con base en ellas ejercer acciones sobre las velocidades tanto lineales como angulares empleadas para la estimación del orientación y posición del prototipo.

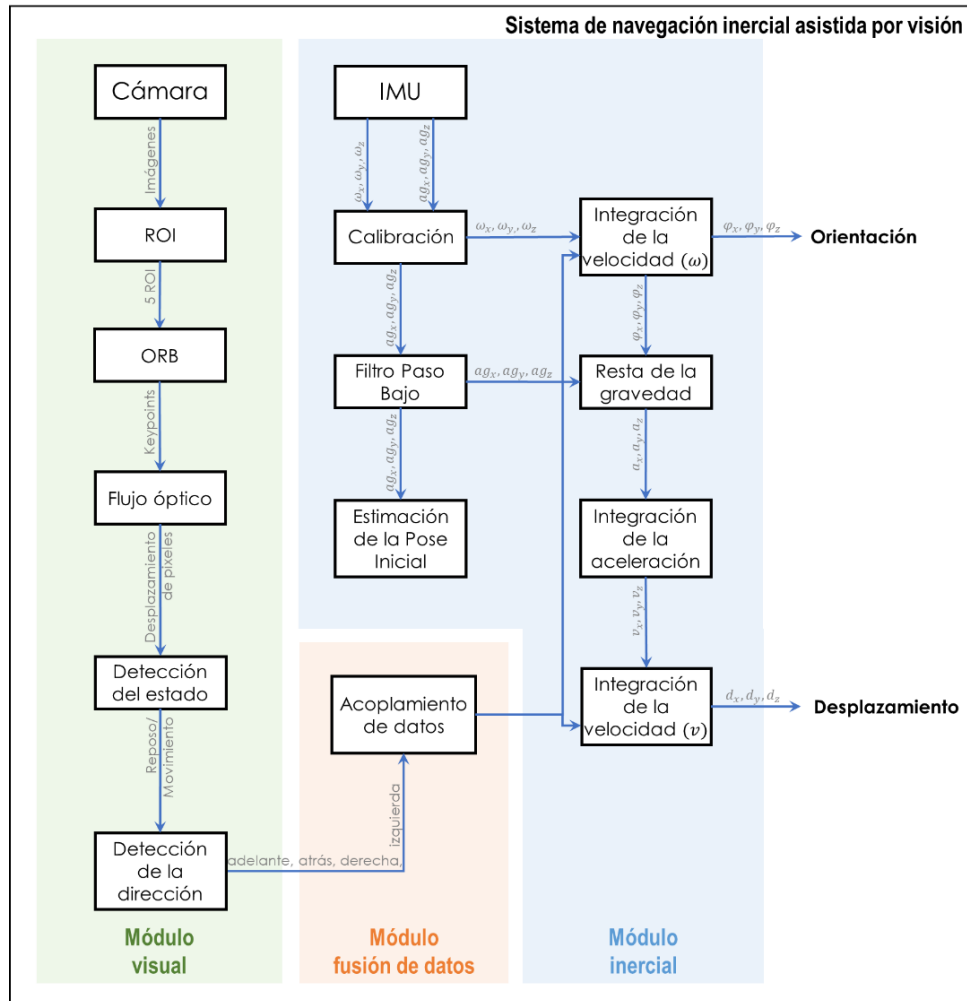


Figura 5.1. Sistema de navegación inercial asistido por visión.

5.2 Módulo inercial

El software del módulo inercial desarrollado está conformado por tres clases: calibración, pose inicial y odometría. Las clases calibración y pose inicial se desarrollaron utilizando el IDE de Arduino 1.8.5 (Arduino Org (2018)b) en el Arduino DUE, mientras que la clase odometría se desarrolló utilizando el IDE de Qt Creator 5.3.2 en la Raspberry Pi. El software desarrollado estima la orientación y posición del prototipo en dos dimensiones x y y , dichas dimensiones se pueden apreciar en la Figura 5.2, que muestra el sistema de coordenadas de la IMU construido de acuerdo con el trabajo de Navarro Tarín (2014).

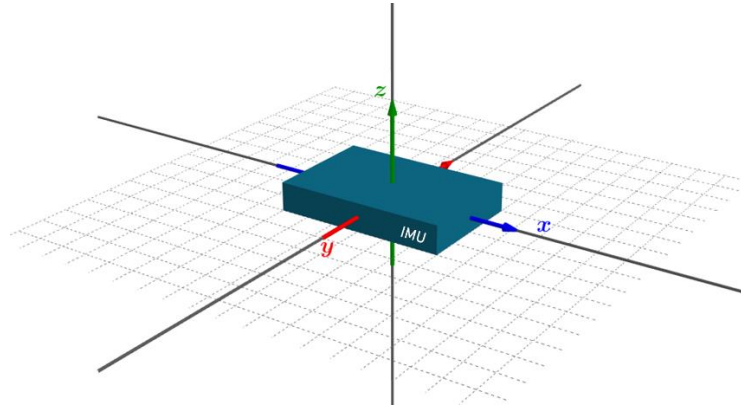


Figura 5.2. Sistema de coordenadas (x, y, z) del módulo inercial.

Si la IMU estuviera libre de errores el proceso de convertir los datos inerciales a datos de navegación sería tan fácil como restar los efectos de la gravedad e integrar las aceleraciones lineales y velocidades angulares en velocidad, posición y orientación (Barrett et al. (2013)). Sin embargo, la IMU no está libre de errores, por lo que sus datos deben ser calibrados y filtrados para después ser convertidos en datos de navegación. Para esto, se utilizó el algoritmo presentado en Woodman (2007). De acuerdo con Woodman, en un sistema en el cual los sensores de inercia están montados rígidamente en el dispositivo, se producen medidas en el marco del cuerpo en lugar del marco global. Por lo tanto, las señales del giroscopio se integran para realizar un seguimiento de la orientación y para el seguimiento de la posición, las señales del acelerómetro se resuelven en coordenadas globales utilizando la orientación determinada por la integración de las señales del giroscopio. En la Figura 5.3, se muestra el procedimiento de las integraciones de las señales del giroscopio y del acelerómetro desarrollado.

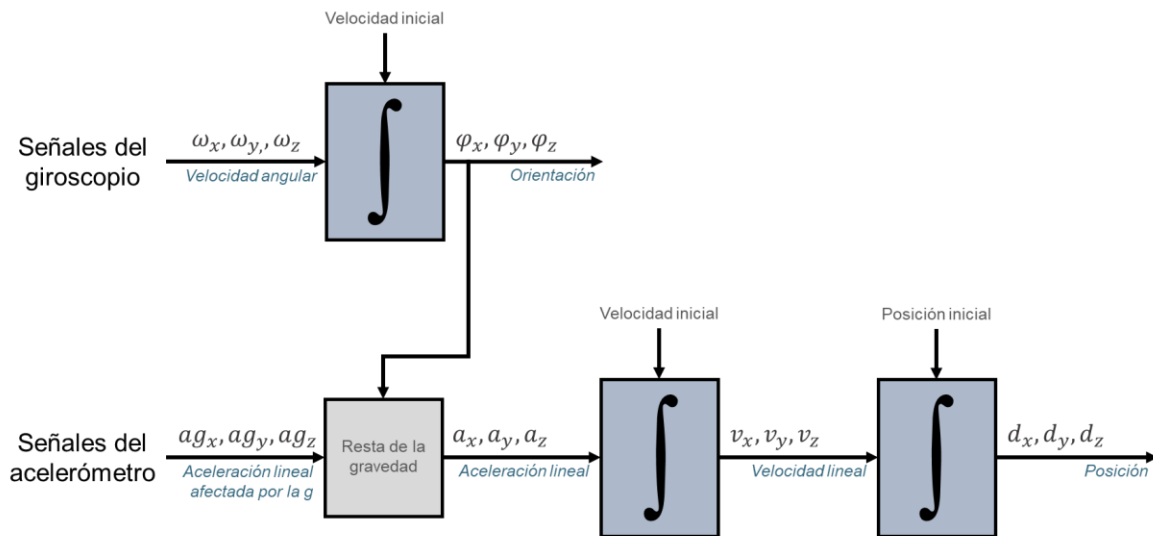


Figura 5.3. Algoritmo de navegación inercial (Woodman (2007)).

5.2.1 Calibración

Por un lado, el giroscopio entrega datos de velocidad angular muy estables, en ocasiones con errores en uno o dos de sus ejes. Sin embargo, dicho error es constante ante cualquier movimiento lo que permite eliminarlo mediante una suma o resta según su signo.

Por el otro, el acelerómetro entrega datos crudos que se convirtieron a m/s^2 para ser utilizados como aceleración lineal. Sin embargo, antes de la conversión los datos se calibraron para mejorar sus mediciones. El método de calibración utilizado consta de cuatro pasos y se explica a continuación:

1. Recolección y promediado de datos

Se recolectan un número considerado de lecturas por cada eje con y sin el efecto de la gravedad ($x+$, $x-$, $y+$, $y-$, $z+$, $z-$), con esto se tienen 6 colecciones de datos. Posteriormente, se promedia cada colección de datos teniendo como resultado seis valores correspondientes a la $x+$, $x-$, $y+$, $y-$, $z+$ y $z-$.

2. Cálculo del valor zero-g

Se calculan los valores zero-g para cada eje con las siguientes fórmulas:

$$x_{zero} = \frac{(promedio_{x_{positivo}} + promedio_{x_{negativo}})}{2} \quad (5.1)$$

$$y_{zero} = \frac{(promedio_{y_{positivo}} + promedio_{y_{negativo}})}{2} \quad (5.2)$$

$$z_{zero} = \frac{(promedio_{z_{positivo}} + promedio_{z_{negativo}})}{2} \quad (5.3)$$

3. Cálculo del valor escala-g

Se calculan los valores escala-g para cada eje con las siguientes fórmulas:

$$x_{escala} = \frac{(promedio_{x_{positivo}} - promedio_{x_{negativo}})}{2} \quad (5.4)$$

$$y_{escala} = \frac{(promedio_{y_{positivo}} - promedio_{y_{negativo}})}{2} \quad (5.5)$$

$$z_{escala} = \frac{(promedio_{z_{positivo}} - promedio_{z_{negativo}})}{2} \quad (5.6)$$

4. Obtención de los datos calibrados

Se obtienen los datos calibrados con las siguientes fórmulas:

$$x = \frac{(lectura_x - x_{zero})}{x_{escala}} \quad (5.7)$$

$$y = \frac{(lectura_{a_y} - y_{zero})}{y_{escala}} \quad (5.8)$$

$$z = \frac{(lectura_{a_z} - z_{zero})}{z_{escala}} \quad (5.9)$$

5. Conversión a m/s^2

Los datos calibrados se multiplican por el valor de la gravedad local (9.7810883404689 en Cuernavaca) para obtener la aceleración lineal.

$$x_g = x * 9.781088 \quad (5.10)$$

$$y_g = x * 9.781088 \quad (5.11)$$

$$z_g = x * 9.781088 \quad (5.12)$$

La implementación del método de calibración consiguió valores de aceleración entre -0.22356 y -0.19314 en el eje x , valores entre 0.07312 y 0.0777 en el eje y y valores entre 9.75817 y 9.74381 en el eje z , tal como se muestra en la Figura 5.4.

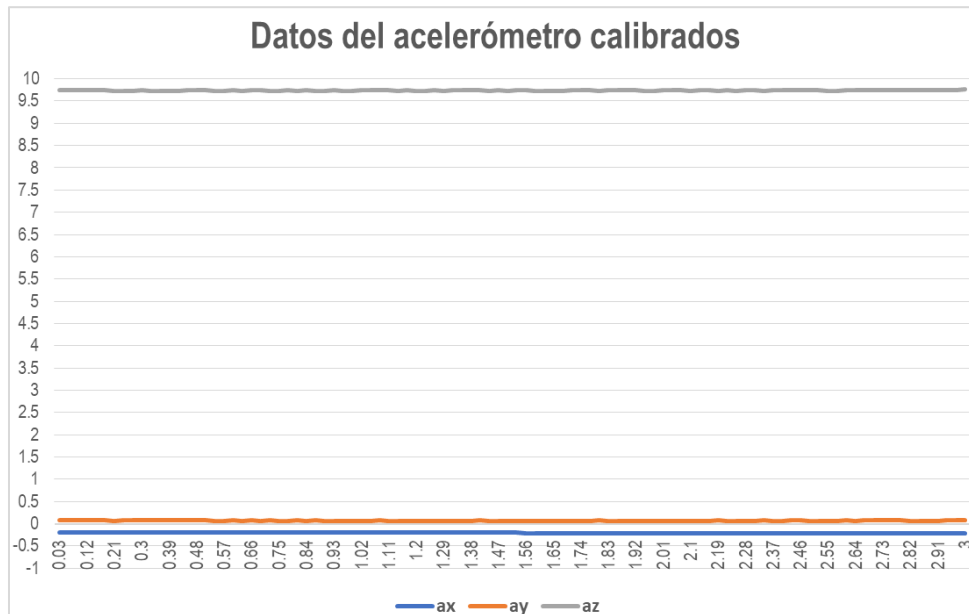


Figura 5.4. Resultado de la calibración del acelerómetro.

5.2.2 Filtro paso bajo

El acelerómetro es muy sensible a las vibraciones, lo que provoca que sus mediciones presenten ruido. Por lo tanto, es necesario aplicar un filtro paso bajo para disminuir el ruido. El filtro paso bajo es caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas. La ecuación del filtro paso bajo se define como:

$$ag_x = x_g * alpha + (ag_{x-1} * (1 - alpha)) \quad (5.13)$$

$$ag_y = y_g * alpha + (ag_{y-1} * (1 - alpha)) \quad (5.14)$$

$$ag_z = z_g * alpha + (ag_{z-1} * (1 - alpha)) \quad (5.15)$$

donde en las ecuaciones 5.13-15,

ag_x, ag_y, ag_z = aceleraciones lineales filtradas.

x_g, y_g, z_g = aceleraciones lineales entregada por el acelerómetro.

$alpha$ = valor que determina la frecuencia del corte del filtro.

$ag_{x-1}, ag_{y-1}, ag_{z-1}$ = aceleraciones lineales filtradas en el tiempo anterior.

El filtro paso bajo implementado utiliza un $alpha$ de 0.08, para evaluar su funcionamiento se realizaron movimientos con el prototipo sobre sus tres ejes. El resultado del filtro se puede observar en la Figura 5.5.

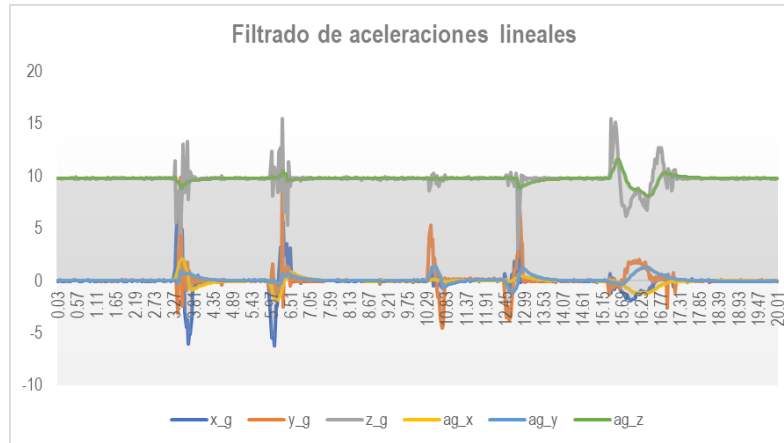


Figura 5.5. Filtrado de las aceleraciones lineales.

5.2.3 Pose inicial

La pose inicial se obtiene mediante el cálculo del valor de la gravedad y los ángulos de inclinación. Para esto se recolectan 500 lecturas, mientras el prototipo se encuentra ubicado en estado de “reposo” el punto de inicio. La fórmulas utilizadas son las presentadas en los trabajos de Y. Liu, Noguchi, et al. (2014) y Ferrer (2015).

- **Gravedad**

$$g = \sqrt{\overline{ag}_x^2 + \overline{ag}_y^2 + \overline{ag}_z^2} \quad (5.16)$$

donde en la fórmula 5.16,

g = valor de la gravedad.

\overline{ag}_x = promedio de las aceleraciones del eje x.

$\overline{a\bar{g}}_y = \text{promedio de las aceleraciones del eje } y.$

$\overline{a\bar{g}}_z = \text{promedio de las aceleraciones del eje } z.$

- **Ángulos de inclinación**

$$\theta_x = \cos^{-1}\left(\frac{\overline{a\bar{g}}_x}{g}\right) \quad (5.17)$$

$$\varphi_y = \cos^{-1}\left(\frac{\overline{a\bar{g}}_y}{g}\right) \quad (5.18)$$

$$\psi_z = \cos^{-1}\left(\frac{\overline{a\bar{g}}_z}{g}\right) \quad (5.19)$$

donde en las fórmulas 5.17-19,

$\theta_x = \text{ángulo del eje } x.$

$\varphi_y = \text{ángulo del eje } y.$

$\psi_z = \text{ángulo del eje } z.$

$\overline{a\bar{g}}_x, \overline{a\bar{g}}_y, \overline{a\bar{g}}_z = \text{promedios de las aceleraciones.}$

$g = \text{valor de la gravedad.}$

Durante el traslado del punto inicial al final se obtienen los ángulos de rotación con la integración de la velocidad angular. Enseguida dichos ángulos son utilizados para eliminar la gravedad de las aceleraciones lineales. Por último, las aceleraciones lineales se integran doblemente para obtener la posición.

5.2.4 Integración de la velocidad angular

Los ángulos de rotación se obtienen integrando una vez la velocidad angular mediante la regla del trapecio definiendo las ecuaciones como:

$$\varphi_{x_{t_1}} = \varphi_{x_{t_0}} + \frac{1}{2}(\omega_{x_{t_0}} + \omega_{x_{t_1}}) * (t_1 - t_0) \quad (5.20)$$

$$\varphi_{y_{t_1}} = \varphi_{y_{t_0}} + \frac{1}{2}(\omega_{y_{t_0}} + \omega_{y_{t_1}}) * (t_1 - t_0) \quad (5.21)$$

$$\varphi_{z_{t_1}} = \varphi_{z_{t_0}} + \frac{1}{2}(\omega_{z_{t_0}} + \omega_{z_{t_1}}) * (t_1 - t_0) \quad (5.22)$$

donde en las fórmulas 5.20-22

$\omega_x, \omega_y, \omega_z = \text{velocidades angulares.}$

$\varphi_x, \varphi_y, \varphi_z = \text{ángulos de orientación.}$

$t_1 = \text{tiempo actual.}$

$t_0 = \text{tiempo anterior.}$

5.2.5 Resta de la gravedad

Las fórmulas utilizadas para la resta de la gravedad de las aceleraciones lineales presentadas en Y. Liu et al. (2014) se definen como:

$$a_x = ag_x(t) - g * \cos \varphi_x \quad (5.23)$$

$$a_y = ag_y(t) - g * \cos \varphi_y \quad (5.24)$$

$$a_z = ag_z(t) - g * \cos \varphi_z \quad (5.25)$$

donde en las fórmulas 5.23-25

$a_x, a_y, a_z =$ aceleraciones lineales.

$ag_x, ag_y, ag_z =$ aceleraciones afectadas por g .

$\varphi_x, \varphi_y, \varphi_z =$ ángulos de orientación.

5.2.6 Doble integración de la aceleración lineal

El desplazamiento se obtiene integrando dos veces la aceleración lineal, con la primera integración se obtiene la velocidad y con la segunda el desplazamiento. Las ecuaciones de las integraciones utilizando la regla del trapecio se definen como:

- **Primera integración**

$$v_{x_{t_1}} = v_{x_{t_0}} + \frac{1}{2}(a_{x_{t_0}} + a_{x_{t_1}}) * (t_1 - t_0) \quad (5.26)$$

$$v_{y_{t_1}} = v_{y_{t_0}} + \frac{1}{2}(a_{y_{t_0}} + a_{y_{t_1}}) * (t_1 - t_0) \quad (5.27)$$

$$v_{z_{t_1}} = v_{z_{t_0}} + \frac{1}{2}(a_{z_{t_0}} + a_{z_{t_1}}) * (t_1 - t_0) \quad (5.28)$$

donde en las fórmulas 5.26-28

$a_x, a_y, a_z =$ aceleraciones lineales.

$v_x, v_y, v_z =$ velocidades lineales.

$t_1 =$ tiempo actual.

$t_0 =$ tiempo anterior.

- **Segunda integración**

$$d_{x_{t_1}} = d_{x_{t_0}} + \frac{1}{2}(v_{x_{t_0}} + v_{x_{t_1}}) * (t_1 - t_0) \quad (5.29)$$

$$d_{y_{t_1}} = d_{y_{t_0}} + \frac{1}{2}(v_{y_{t_0}} + v_{y_{t_1}}) * (t_1 - t_0) \quad (5.30)$$

$$d_{z_{t_1}} = d_{z_{t_0}} + \frac{1}{2}(v_{z_{t_0}} + v_{z_{t_1}}) * (t_1 - t_0) \quad (5.31)$$

donde en las fórmulas 5.29-31

$v_x, v_y, v_z =$ velocidades lineales.

$d_x, d_y, d_z =$ posición.

$t_1 = \text{tiempo actual.}$

$t_0 = \text{tiempo anterior.}$

5.3 Módulo visual

El software del módulo visual se desarrolló en la Raspberry Pi 3 bajo el lenguaje de programación C++, utilizando el IDE Qt Creator 5.3.2 y la biblioteca de OpenCV 3.3.0 (OpenCV Org (2018)). El software detecta si el prototipo se encuentra en estado de “reposo” o “movimiento” en dos dimensiones x y y . En caso de movimiento, determina sobre qué eje se ejecuta dicho movimiento, pudiendo ser sobre uno o ambos ejes. Además, determina la dirección del movimiento que podría ser adelante (z), atrás ($-z$), derecha (x) o izquierda ($-x$). Para un mejor entendimiento en la Figura 5.6, se muestra el sistema de coordenadas del módulo visual construido de acuerdo al trabajo de Navarro Tarín (2014).

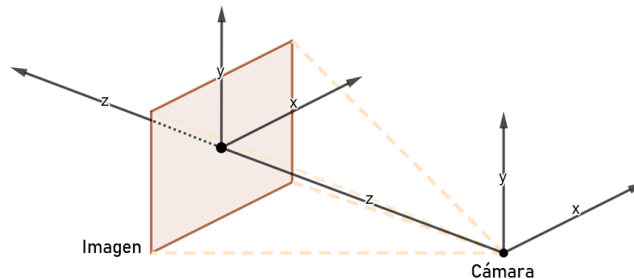


Figura 5.6. Sistema de coordenadas (x, y, z) del módulo visual.

El algoritmo desarrollado para la detección del estado (“reposo” o “movimiento”) del prototipo está basado en el trabajo de Aroca (2015). Dicho algoritmo consiste en los siguientes seis pasos:

1. Adquisición de dos fotogramas consecutivos en escala de grises: fotograma pre_actual y fotograma actual.
2. Detección de puntos característicos (keypoints) en el fotograma pre_actual mediante el detector ORB.
3. Búsqueda de las correspondencias de los keypoints del fotograma pre_actual en el fotograma actual y cálculo del flujo óptico entre ambos fotogramas mediante la función `calcOpticalFlowPyrLk`.
4. Cálculo del desplazamiento de píxeles mediante una diferencia de puntos, para esto el vector de coordenadas (x, y) de los keypoints del fotograma actual se le resta al vector de coordenadas (x, y) de los keypoints del fotograma pre_actual.
5. Se determina el estado de cada keypoint en los ejes x y y , para esto se establece un umbral, que en caso de ser superado por el valor de desplazamiento se determina “movimiento” y en caso de no ser superado se determina “reposo”.
6. Se determina el estado del prototipo mediante un conteo de keypoints en estado “reposo” y “movimiento”, el estado con más keypoints será el estado detectado.

El algoritmo era capaz de determinar si el prototipo se encontraba en reposo o movimiento, por lo que se trabajó en modificarlo y mejorarlo de tal manera que además de determinar si el prototipo se encontraba en reposo o movimiento, detectará sobre que eje se ejecuta el movimiento, pudiendo ser sobre uno o ambos ejes (x,y) . Además, que determinara la dirección del movimiento que podría ser adelante, atrás, derecha o izquierda. Para lograrlo, se crearon regiones de interés (roi) en las cuales se ejecuta el algoritmo de detección del estado, que mediante conteos determina el estado y la dirección de movimiento de cada eje. En conjunto las regiones determinan si el prototipo se encuentra en reposo o movimiento. Mientras que, para determinar la dirección del movimiento se utilizan dos grupos de regiones, de los cuales uno determina si el movimiento es hacia adelante o atrás y el otro determina si el movimiento es hacia la derecha o izquierda. A continuación, se detallan las funciones del módulo visual que hacen posible la detección.

5.3.1 Adquisición de la imagen

Las imágenes utilizadas en el método de detección de movimiento son capturadas por la cámara pi conectada a la Raspberry Pi. Para utilizar las imágenes es necesario configurar los espacios de colores de la cámara y de OpenCV de modo que coincida con el espacio de color utilizado por Qt Creator. Qt utiliza RGB, mientras que la cámara pi dispone de RGB, BRG y Grayscale con tres dimensiones por cada formato:

- 1280x960: 29.5fps, 640x480 : 29.5fps, 320x240 : 29.5fps RGB Mode
- 1280x960: 28 fps, 640x480 : 29.29fps, 320x240 : 29.24fps BGR Mode
- 1280x960: 14 fps, 640x480 : 29.29fps, 320x240 : 29.24fps Grayscale Mode

Por lo tanto, la cámara pi se configuró en modo RGB con una resolución de 330X240 y una velocidad de 29.5 fps. La configuración de la cámara en Qt Creator es la siguiente:

```
101 // Configuración de la cámara pi
102 // Modo RGB (640x480)→ 29.5 fps
103 camara.set(CV_CAP_PROP_FORMAT, CV_8UC3);
104 camara.set(CV_CAP_PROP_FRAME_WIDTH, 640);
105 camara.set(CV_CAP_PROP_FRAME_HEIGHT, 480);
```

Con la configuración del espacio de color de la cámara realizada, lo siguiente es adquirir la imagen para poder procesarla mediante OpenCV. El espacio de color predeterminado de OpenCV es BGR, por lo tanto, es necesario convertirlo a RGB utilizando la función OpenCv cvtColor con el parámetro CV_BGR2RGB. La implementación de la adquisición de la imagen en formato RGB en Qt Creator es la siguiente:

```
322 camara.grab(); // Captura del fotograma
323 camara.retrieve(frame); // Decodifica y devuelve el fotograma capturado
324 cvtColor(frame, frame, CV_BGR2RGB); // Convierte el fotograma de BGR a RGB
```

5.3.2 Regiones de interés (ROI)

En cada una de las imágenes adquiridas por la cámara se crearon cinco regiones de interés: ROI0, ROI1, ROI2, ROI3 y ROI4 (Figura 5.7). La distribución de las regiones se debe a que, durante varias pruebas se observó que ante movimientos con dirección adelante o atrás los keypoints pertenecientes a las regiones de las esquinas registraban mayor desplazamiento que los keypoints de la región centro, mientras que ante movimientos hacia la derecha o izquierda se registraban desplazamientos con valores semejantes en toda la imagen. Por lo tanto, ROI0 que pertenece a la región centro de la imagen se utiliza para la detección de movimientos con dirección hacia la derecha o izquierda. ROI1, ROI2, ROI3 y ROI4 que corresponden a las esquinas de la imagen se utilizan para la detección de movimientos con dirección hacia adelante o atrás. El tamaño de las regiones de las esquinas es de 220 x 160 píxeles un poco mayor a la región centro de 200 x 160 píxeles. La diferencia en el tamaño se debe a que la región centro se encuentran muchos más keypoints que en las regiones de las esquinas. Por lo tanto, se les da unos píxeles más para ampliar la búsqueda de keypoints.

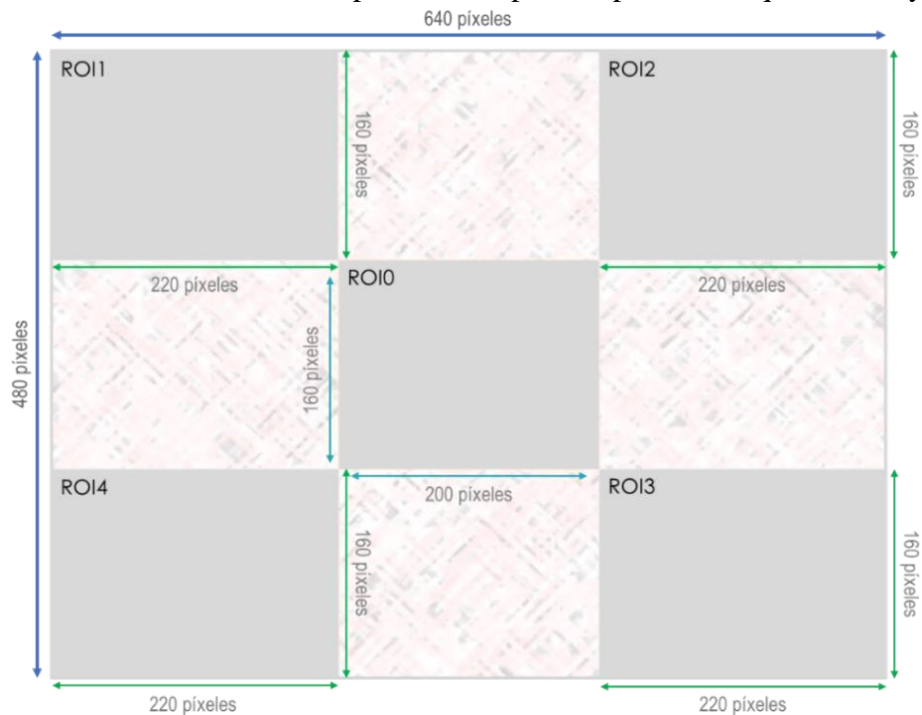


Figura 5.7 Regiones de interés de la imagen.

El proceso para crear regiones de interés se muestra en el Algoritmo 1, en el cual para crear regiones de interés se utiliza la función de OpenCV *Rect()* (línea 5), donde *Rect(x, y, w, h)* crea un rectángulo que representa la región deseada, es necesario indicar las coordenadas (x, y) además del ancho y el alto (w, h) (funvision.blogspot (2015)). Después de crear la región de interés, dicha región es convertida a escala de grises (línea 7).

Algoritmo 1: Crear regiones de interés (rois[]) en las imágenes de un vídeo.

```

1  procedimiento ROIs(video, posx[], posy[], fil[], col[])
2  mientras leer(video) hacer
3  definir tamaño de rois en 5
4  para i←0 hasta i←5 hacer
5      rect←(posx[i], posy[i], fil[i], col[i])
6      roi←(frame, rect) //
7      cvtColor(roi, roi, CV_RGB2GRAY); // conversión a escala de grises.
8      rois[i]←roi // inserción de roi en el vector de regiones de interés
9  fin para
10 regreser rois[5]
11 fin mientras
12 fin procedimiento

```

5.3.3 Flujo óptico

Con las regiones de interés en escala de grises, se procese a calcular el flujo óptico entre las regiones de dos fotogramas. En el Algoritmo 2, se muestra el proceso para calcular el flujo óptico entre dos regiones consecutivas, en el cual es necesario contar con las regiones de los fotogramas: *pre_actual* y *actual* (línea 5). A continuación, se ejecuta la función de OpenCV *ORB()* para obtener keypoints de las regiones pertenecientes al fotograma actual (línea 9). En la primera ejecución del algoritmo tanto el fotograma actual como el *pre_actual* contienen la misma información, por lo que a partir del segundo ciclo la función *calcOpticalFlowPyrLK()* de OpenCV obtiene el flujo óptico utilizando los keypoints del fotograma *pre_actual* (línea 17). Esta función devuelve las nuevas coordenadas de los keypoints en el fotograma actual, por lo tanto, el desplazamiento de píxeles entre fotogramas se obtiene mediante una diferencia entre las coordenadas de sus keypoints (líneas 23 y 24). Al final de cada ciclo del algoritmo, el fotograma actual se vuelve el *pre_actual* y los keypoints del fotograma actual pasan al vector *pre_actual* (líneas 32 y 33). Además, el algoritmo determina cuándo realizar una nueva búsqueda de keypoints y cuándo seguir trabajando con los keypoints disponibles. Esto se realiza en base al porcentaje de pérdidas de keypoints, si se pierde más del 50% se ejecuta una nueva búsqueda (línea 29).

Algoritmo 2: Cálculo del flujo óptico entre dos regiones consecutivas.

```

1  procedimiento FLUJO_OPTICO(video, keypoints_reset, keypoints)
2  mientras leer(video) hacer
3  rois_act[]←roi
4  si rois_pre[] vacío entonces
5  rois_pre[]←rois_act[]
6  fin si
7  si keypoints_reset==1 entonces // búsqueda de keypoints
8  limpiar keypoints_a[]
9  detectorORB(rois_act[], keypoints) // obtención de keypoints por ORB
10 para i←0 hasta tamaño(keypoints) hacer
11 keypoints_a[]←keypoints // keypoints al vector
12 fin para
13 regresar
14 fin si
15 si keypoints_a[]>0 entonces // condición para el cálculo del flujo óptico
16 keypoints_reset=0
17 calcOpticalFlowPyrLK(rois_pre, rois_act, keypoints_a, keypoints_b) // obtención del flujo óptico
18 contador_key=0
19 para j←0 hasta tamaño(keypoints_a) hacer
20 dx=0.0
21 dy=0.0
22 contador_key = contador_key+1; // contador de keypoints disponibles
23 dx = dx + (keypoints_act[].x - keypoints_b[].x) // desplazamiento de píxeles en x
24 dy = dy + (keypoints_act[].y - keypoints_b[].y) // desplazamiento de píxeles en y
25 fin para
26 regresar dx, dy
27 fin si
28 porcentaje_keypoints=contador_key/keypoints_a[] // Porcentaje de keypoints disponibles
29 si porcentaje_keypoints < 0.50 entonces // condición de búsqueda de nuevos keypoints
30 keypoints_reset=1
31 fin si
32 keypoints_b[]←keypoints_a // reasignación de los vectores Points2f
33 rois_pre[]←rois_act // reasignación de las regiones
34 fin mientras
35 fin procedimiento

```

5.3.4 Detección del estado

La detección del estado determina si existe movimiento o reposo sobre uno o dos ejes, para esto se establece un umbral, que en caso de ser superado por el valor absoluto de desplazamiento del píxel se determina “movimiento” y en caso de no ser superado se determina “reposo”. Esta detección se realiza por cada keypoints, al final si la mayoría de keypoints registraron movimiento, se determina “movimiento” y en caso contrario se determina “reposo”. El proceso para detectar si el prototipo está en reposo o movimiento se muestra en el Algoritmo 3.



Algoritmo 3: Detectar si el prototipo se encuentra en reposo o en movimiento.

```

1  procedimiento ESTADO(video, umbral)
2  mientras leer(video) hacer
3  para i←0 hasta tamaño(keypoints_a) hacer
4  si (abs(dx) > umbral | abs(dy) > umbral) entonces
5  movimiento = movimiento + 1;
6  sino
7  reposo = reposo + 1;
8  fin si
9  fin para
10 regresarse
11 si movimiento > reposo entonces
12 imprimir "movimiento"
13 sino
14 imprimir "reposo"
15 fin si
16 fin mientras
17 fin procedimiento
    
```

5.3.5 Detección de la dirección

La detección de la dirección del movimiento se logra mediante el análisis del comportamiento del flujo óptico. En la Figura 5.8, se muestra el comportamiento del flujo óptico ante movimientos con dirección adelante, atrás, derecha e izquierda.

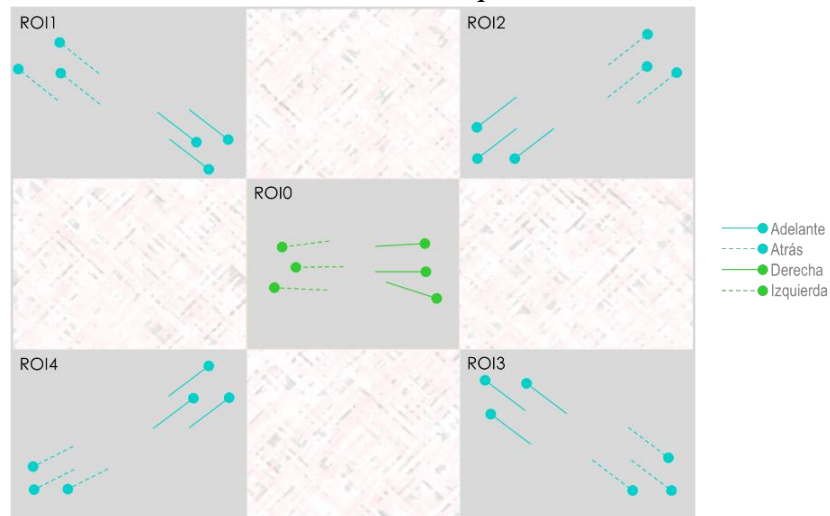


Figura 5.8. Comportamiento del flujo óptico ante diferentes movimientos.

Para la detección de la dirección del movimiento se utiliza el principio de la detección del estado, sólo que en caso de movimiento se detecta a qué dirección corresponde el comportamiento del flujo óptico. El umbral utilizado para detectar movimientos hacia la derecha o izquierda en ROI0 es de 1 píxel. Y el umbral utilizado para detectar movimientos hacia adelante o atrás en las regiones restantes es de 5 píxeles. La detección en el ROI0 es unánime, mientras que en las regiones 1,2,3 y 4 se determina mediante votación, por lo tanto, la dirección más votada entre las cuatro regiones es la dirección determinada. Además, en

caso de empate se toma la dirección detectada anteriormente. El proceso para detectar si el prototipo se encuentra en reposo o en movimiento se muestra en el Algoritmo 4.

Algoritmo 4: Detectar si el prototipo se encuentra en reposo o en movimiento, en caso de movimiento detectar la dirección

```

1  procedimiento ESTADO_DIRECCION(video)
2  mientras leer(video) hacer
3  para i←0 hasta tamaño(keypoints_a) hacer
4  si num_roi < 4 entonces
5  si abs(dx) > 1.0 entonces
6  si dx > 0.0 entonces
7  adelante = adelante + 1
8  sino
9  atras = atras + 1
10 fin si
11 sino
12 si dx < 0.0 entonces
13 adelante = adelante + 1
14 sino
15 atras = atras + 1
16 fin si
17 sino
18 reposo = reposo + 1
19 fin si
20 fin si
21 fin para
22 regresar
23 si m<4 entonces
24 si (adelante > atras) & (adelante > reposo) entonces
25 r_adelante = r_adelante + 1
26 fin si
27 si (atras > adelante) & (atras > reposo) entonces
28 r_atras = r_atras + 1
29 fin si
30 si (reposo > adelante) & (reposo > atras) entonces
31 r_reposo = r_reposo + 1
32 fin si
33 fin si
34 si (r_reposo > r_adelante) & (r_reposo > r_atras) entonces
35 imprimir "reposo"
36 sino
37 si (t_adelante > t_atras) entonces
38 imprimir "adelante"
39 fin si
40 si t_atras > t_adelante entonces
41 imprimir "atrás"
42 fin si
43 fin si
44 fin mientras
45 fin procedimiento

```

Las salidas del módulo de visión representan el estado de los ejes z y x del sistema de visión del prototipo, estados que pueden ser “reposo”, “adelante”, “atrás”, “izquierda” o “derecha”, tal como se muestra en la Tabla 5.1.

Tabla 5.1. Estados de los ejes z y x del sistema de visión detectados por el módulo de visión.

Salidas	Estados (z, x)
1	reposo, reposo
2	adelante, reposo
3	atrás, reposo
4	reposo, derecha
5	reposo, izquierda
6	adelante, derecha
7	adelante, izquierda
8	atrás, derecha
9	atrás, izquierda

5.4 Módulo fusión de datos

El software del módulo de fusión de datos se desarrolló en la Raspberry Pi 3 bajo el lenguaje de programación C++, utilizando el IDE Qt Creator 5.3.2. El software realiza el acoplamiento entre los datos del módulo inercial y visual para la estimación de la orientación y posición del prototipo. Para esto, se debe conocer qué coordenadas del módulo inercial corresponden a las coordenadas del módulo visual, es por ello por lo que en la Figura 5.9 se muestra el sistema de referencia del prototipo, con las coordenadas de los módulos inercial (x_i, y_i, z_i) y visual (x_c, y_c, z_c).

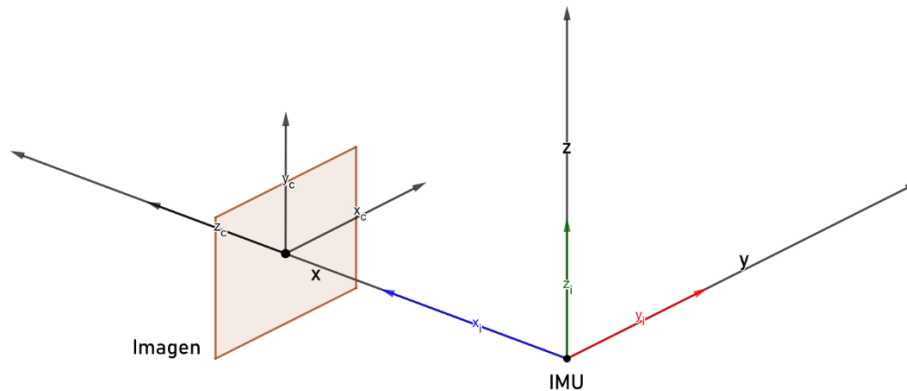


Figura 5.9. Sistema de referencia (x, y, z) del prototipo.

El proceso de acoplamiento consta de dos pasos: paso uno, revisar la salida del módulo visual; y paso dos, con base en la salida del módulo visual ejecutar acciones en el módulo inercial. Mediante la revisión del módulo visual se conocen los ejes que presentan reposo o movimiento, pudiendo ser reposo en ambos ejes lo que significa que el prototipo está en

estado de “reposo” o presentar movimiento en uno o ambos ejes lo que significaría que el prototipo está en estado de “movimiento”. De acuerdo con los estados detectados por el módulo visual el módulo inercial realiza acciones, si se detecta “reposo”, las velocidades lineales y angulares se mantienen en cero, por el contrario, si se detecta “movimiento”, las velocidades lineales y angulares se integran para estimar la rotación y el desplazamiento desplegados durante la navegación. El proceso para el acoplamiento de datos inerciales con visuales se muestra en el Algoritmo 5.

Algoritmo 5: Acoplamiento de datos inerciales con visuales

```

1  procedimiento ACOPLAMIENTO(video, dirx, diry, vel_angx, vel_angy, vel_linx, vel_liny)
2  mientras leer(video) hacer
3      si dirx == reposo entonces // Si el eje x se mantiene en reposo
4          va_x = 0.0 // velocidad angular en x a cero
5          vl_x = 0.0 // velocidad lineal en x a cero
6      sino
7          va_x = vel_angx // integración de la velocidad angular en x
8          vl_x = vel_linx // integración de la velocidad lineal en x
9      fin si
10     si diry == reposo entonces // Si el eje y se mantiene en reposo
11         va_y = 0.0 // velocidad angular en y a cero
12         vl_y = 0.0 // velocidad lineal en y a cero
13     sino
14         va_y = vel_angy // integración de la velocidad angular en y
15         vl_y = vel_liny // integración de la velocidad lineal en y
16     fin si
17     fin mientras
18     retorna va_x, va_y, vl_x, vl_y
19 fin procedimiento
    
```

En la Tabla 5.2 se muestran las acciones ejecutadas en el módulo inercial de acuerdo con los estados detectados por el módulo de visión.

Tabla 5.2. Acoplamiento de datos inerciales con visuales.

Movimiento	Estado (visión)	Acción (IMU)	
		x	y
1	reposo, reposo	velocidades a cero	Velocidades a cero
2	adelante, reposo	Integración	Velocidades a cero
3	atrás, reposo	Integración	Velocidades a cero
4	reposo, derecha	Velocidades a cero	Integración
5	reposo, izquierda	Velocidades a cero	Integración
6	adelante, derecha	Integración	Integración
7	adelante, izquierda	Integración	Integración
8	atrás, derecha	Integración	Integración
9	atrás, izquierda	Integración	Integración



5.5 Interfaz gráfica

La interfaz gráfica de usuario fue desarrollada bajo el lenguaje de programación de C++, mediante el uso del IDE Qt Creator 5.3.2. Mediante la interfaz se establece la comunicación entre los componentes y se controlan los procesos de cálculo de la pose inicial y estimación de la orientación y posición del prototipo durante la navegación.

Los objetos de la interfaz gráfica de usuario son: un `QVideoWidget`, cuatro `QPushButton`, un `QLabel` y diez `QTextEdit`, tal como se muestra en la Figura 5.10. A continuación, se mencionan las acciones de cada objeto de la interfaz.

- **QVideoWidget** permite visualizar el video capturado durante la navegación.
- **QPushButton Iniciar** ejecuta el preprocesamiento de los datos inerciales.
- **QPushButton Pose Inicial** ejecuta la estimación de la pose inicial del prototipo.
- **QPushButton Navegación** ejecuta la estimación de la orientación y posición del prototipo durante la navegación.
- **QPushButton Cerrar** ejecuta el paro de la estimación de la orientación y la posición del prototipo, así como el cierre del sistema.
- **QTextEdit Pose Inicial** muestra los ángulos de inclinación (x, y, z) y el valor de la gravedad calculados en la ejecución de la pose inicial.
- **QTextEdit Posición** muestra los desplazamientos (x, y, z) estimados en la ejecución de la navegación.
- **QTextEdit Orientación** muestra los ángulos de rotación (x, y, z) estimados en la ejecución de la navegación.
- **QLabel** muestra el estado actual del sistema, mostrando el inicio y fin de cada proceso ejecutado.



Figura 5.10. Interfaz gráfica del sistema de navegación inercial asistido por visión.

El procedimiento para ejecutar el sistema durante la navegación se muestra mediante un diagrama de flujo en la Figura 5.11. Una vez iniciado el sistema el procedimiento consta de cuatro actividades principales:

1. Pulse QPushButton Iniciar
Ejecuta el preprocesamiento de los datos inerciales con el objetivo de reducir el ruido en las lecturas. En el caso de las aceleraciones lineales se aplica un filtro paso bajo y en el de las velocidades angulares el ruido suele ser constante lo que permite reducirlo mediante una suma o resta según su signo.
2. Pulse QPushButton Pose Inicial
Ejecuta la estimación de la pose inicial mediante el cálculo de los ángulos de inclinación del prototipo. Además, apertura la escritura de un archivo con los datos de navegación.
3. Pulse QPushButton Navegación
Ejecuta la estimación de la orientación y posición del prototipo durante la navegación mediante la detección del movimiento y el cálculo de la odometría inercial. Además, inicia la grabación de un video con las escenas de la navegación.
4. Pulse QPushButton Cerrar
Ejecuta el paro de la estimación de la orientación y posición del prototipo y cierra el sistema. Además, detiene y guarda tanto el archivo como el video con los datos de navegación.

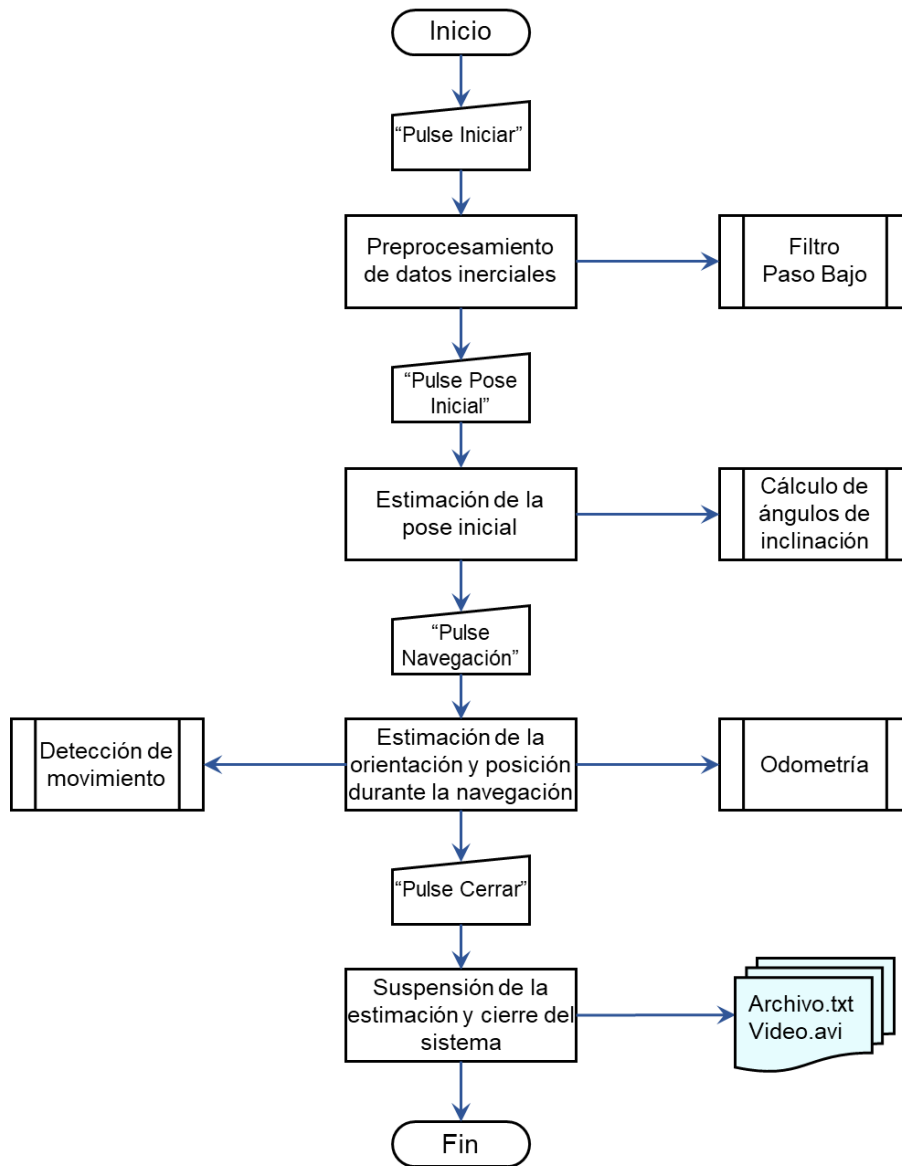


Figura 5.11. Diagrama de flujo del sistema de navegación inercial asistido por visión.

5.6 Discusión

El sistema de navegación inercial asistido por visión se desarrolló para entornos con condiciones controladas como puntos característicos, poca variación en el suelo y sin obstáculos. La interfaz del sistema es fácil de utilizar, muestra los datos de navegación como: orientación y posición y a la vez presenta un video del recorrido.

Capítulo 6

Experimentación y resultados

En este capítulo se presentan los experimentos realizados al sistema de odometría inercial asistido por visión (fusión de datos), así como a los módulos que lo integran: inercial y visual. Además, se detallan los diferentes escenarios empleados y se analizan los resultados obtenidos en cada uno de los experimentos ejecutados.

6.1 Descripción general de los experimentos

Los experimentos que se presentan son los más representativos del funcionamiento de cada módulo y del sistema desarrollado, siendo un total de 24 experimentos, 12 del módulo inercial, 6 del módulo visual y 6 más del sistema de odometría inercial asistido por visión (Figura 6.1). Los experimentos de los módulos inercial y visual se realizaron mediante movimientos ejecutados de forma manual, mientras que los experimentos del sistema de odometría inercial asistido por visión se ejecutaron de manera manual y utilizando un carro de control remoto.

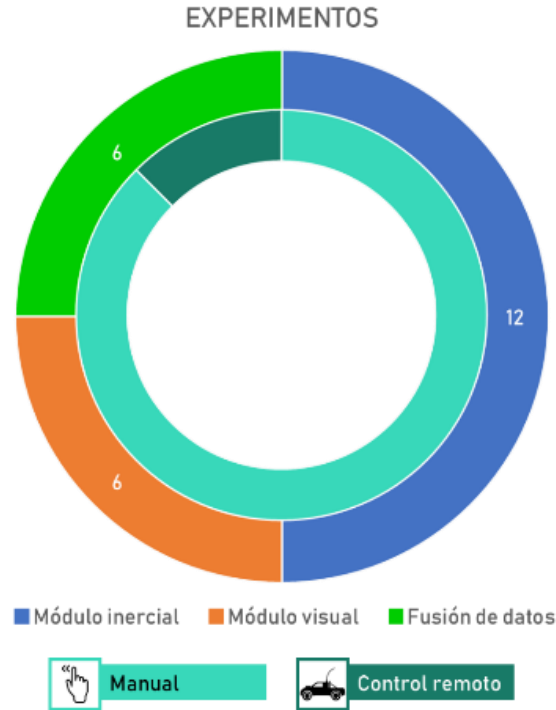


Figura 6.1. Experimentos ejecutados.

Los escenarios utilizados para la experimentación fueron dos: una mesa utilizada para los experimentos ejecutados de forma manual y el piso firme para los experimentos ejecutados por el carro de control. A continuación, se detallan estos escenarios.

6.1.1 Escenario utilizado para los experimentos ejecutados de forma manual

Mesa de madera de $1.50\text{ m} \times 0.70\text{ m}$, bajo condiciones incontroladas como la iluminación y controladas como puntos característicos, poca variación y sin obstáculos, tal como se muestra en la Figura 6.2.



Figura 6.2. Escenario utilizado para los experimentos hechos de forma manual.

6.1.2 Escenario utilizado para los experimentos ejecutados sobre un carro de control

Piso firme de 5 m x 3 m, bajo condiciones incontroladas como iluminación y controladas como poca variación y sin obstáculos, tal como se muestra en la Figura 6.3.



Figura 6.3. Escenario utilizado para los experimentos hechos sobre el carro de control.

Las trayectorias de los experimentos del módulo inercial, visual y fusión de datos se construyen a partir de los datos registrados en el archivo (Figura 6.4) creado por el sistema durante la navegación. En el archivo se registran datos del acelerómetro como aceleración (ax, ay, az), velocidad (vx, vy, vz) y desplazamiento (dx, dy, dz); datos del giroscopio como velocidad (va_x, va_y, va_z) y rotación (rx, ry, rz); y datos del sistema de visión como los estados de los ejes x y y ($visión_x, visión_y$). Las lecturas inerciales se toman cada 0.03 s (dt) y por cada lectura el sistema de visión detecta el estado de los ejes (x, y).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	dt	visión_x	visión_y	va_x	va_y	va_z	rx	ry	rz	axg	avg	azg	ax	ay	az	vx	vy	vz	dx	dy	dz
2	0.03	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.20545	0.067551	9.72668	-0.018601	-0.0375	-0.001797	0	0	0	0	0	0
3	0.06	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.208005	0.06986	9.73484	-0.021156	-0.035191	0.006364	0	0	0	0	0	0
4	0.09	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.210356	0.068876	9.73926	-0.023507	-0.036175	0.010785	0	0	0	0	0	0
5	0.12	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.212519	0.074188	9.73716	-0.02567	-0.030863	0.008679	0	0	0	0	0	0
6	0.15	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.211418	0.069749	9.72905	-0.024569	-0.035302	0.000569	0	0	0	0	0	0
7	0.18	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.213496	0.065664	9.73085	-0.026647	-0.039387	0.002368	0	0	0	0	0	0
8	0.21	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.209226	0.068125	9.73559	-0.022377	-0.036926	0.007109	0	0	0	0	0	0
9	0.24	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.211479	0.06417	9.73686	-0.02463	-0.040881	0.008384	0	0	0	0	0	0
10	0.27	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.210461	0.06675	9.73495	-0.023612	-0.038301	0.006471	0	0	0	0	0	0
11	0.3	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.209524	0.062905	9.73936	-0.022675	-0.042146	0.010883	0	0	0	0	0	0
12	0.33	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.211754	0.065586	9.74342	-0.024905	-0.039465	0.014943	0	0	0	0	0	0
13	0.36	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.210714	0.064944	9.74098	-0.023865	-0.040107	0.012505	0	0	0	0	0	0
14	0.39	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.209757	0.064353	9.74183	-0.022908	-0.040698	0.013348	0	0	0	0	0	0
15	0.42	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.208877	0.066918	9.73951	-0.022028	-0.038133	0.011037	0	0	0	0	0	0
16	0.45	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.204976	0.066169	9.7343	-0.018127	-0.038882	0.005826	0	0	0	0	0	0
17	0.48	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.201387	0.06548	9.73568	-0.014538	-0.039571	0.007203	0	0	0	0	0	0
18	0.51	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.204268	0.064846	9.74003	-0.017419	-0.040205	0.011557	0	0	0	0	0	0
19	0.54	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.206918	0.067372	9.73478	-0.020069	-0.037679	0.006303	0	0	0	0	0	0
20	0.57	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.206265	0.069696	9.73612	-0.019416	-0.035355	0.007643	0	0	0	0	0	0
21	0.6	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.208755	0.068724	9.73735	-0.021906	-0.036327	0.008875	0	0	0	0	0	0
22	0.63	reposo	reposo	0	0	0	1.59	1.56	0.02	-0.207955	0.067831	9.73849	-0.021106	-0.03722	0.010009	0	0	0	0	0	0

Figura 6.4. Información contenida en el archivo creado por el sistema desarrollado.

6.2 Módulo inercial

Los experimentos del módulo inercial del sistema de navegación inercial asistido por visión se realizaron con el objetivo de cuantificar el error y determinar la longitud de los desplazamientos a utilizar en los experimentos del módulo de fusión de datos. Por lo tanto, los experimentos consistieron en conducir el prototipo de forma manual por trayectorias predefinidas que consistían en desplazamientos en línea recta con longitudes de: -0.90, -0.60, -0.30, 0.30, 0.60 y 0.90 m con una velocidad de 0.69 m/s aproximadamente, tanto para el eje x como el y . En total se definieron 12 experimentos de odometría inercial con 10 repeticiones cada uno, obteniendo un total de 120 pruebas, tal como se muestra en la Tabla 6.1.

Tabla 6.1. Experimentos del módulo inercial.

Experimento	Eje	Longitud
1	x	-90
2		-60
3		-30
4		30
5		60
6		90
7	y	-90
8		-60
9		-30
10		30
11		60
12		90

Los resultados obtenidos en cada una de las pruebas realizadas se presentan en el Anexo 4 y en la Tabla 6.2 se muestran las pruebas con las mejores y peores estimaciones obtenidas en cada experimento ejecutado.

Tabla 6.2. Resultados de los experimentos del módulo inercial.

Experimento	Eje	Longitud	Resultado	Prueba	Estimación	Error
1	x	-0.90 m	Mejor	1.6	-0.87	0.03
			Peor	1.3	-1.06	0.16
2		-0.60 m	Mejor	2.9	-0.59	0.01
			Peor	2.2	-0.51	0.09
3		-0.30 m	Mejor	3.10	-0.30	0.00
			Peor	3.2	-0.41	0.11
4		0.30 m	Mejor	4.6	0.28	0.02
			Peor	4.5	0.16	0.14
5		0.60 m	Mejor	5.6	0.59	0.01
			Peor	5.3	0.80	0.20
6		0.90 m	Mejor	6.5	0.86	0.04
			Peor	6.4	0.64	0.26
7	y	-0.90 m	Mejor	7.4	-0.90	0.00
			Peor	7.7	-1.07	0.17
8		-0.60 m	Mejor	8.7	-0.61	0.01
			Peor	8.6	-0.45	0.15
9		-0.30 m	Mejor	9.7	-0.31	0.01
			Peor	9.2	-0.37	0.07
10		0.30 m	Mejor	10.5	0.34	0.04
			Peor	10.10	0.19	0.11
11		0.60 m	Mejor	11.2	0.59	0.01
			Peor	11.3	0.85	0.25
12		0.90 m	Mejor	12.8	0.88	0.02
			Peor	12.1	1.05	0.15

De acuerdo con la Tabla 6.2, el error acumulado durante los experimentos de 0.30 m (3, 4, 9 y 10) se registró entre 0.01 y 0.14 m. En total se realizaron 40 pruebas de 0.30 m, de las cuales el 87.5% (35 pruebas) registraron un error acumulado menor a 0.10 m y el 47.5% (19 pruebas) registran un error acumulado menor a 0.05 m. A continuación, se analizan los resultados de las pruebas registradas con la mejor y peor estimación en los desplazamientos con longitud absoluta de 0.30, 0.60 y 0.90.

En la Figura 6.5, se muestran las trayectorias generadas a partir de los datos obtenidos durante la prueba 4.5 registrada como la peor estimación con un error acumulado de 0.14 m y la prueba 3.10 registrada como la mejor estimación sin error acumulado.

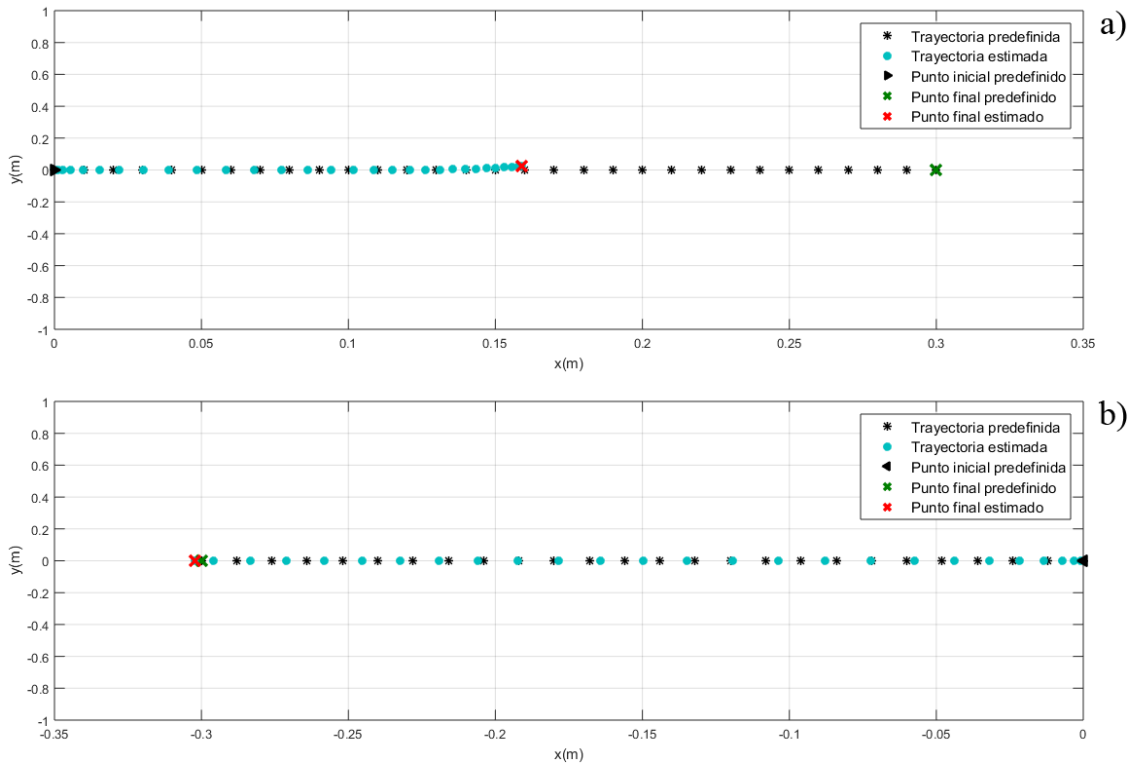


Figura 6.5. Trayectorias obtenidas por el sistema. a) Prueba 4.5 registrada como la peor estimación. b) Prueba 3.10 registrada como la mejor estimación.

En la Figura 6.6 se muestran las gráficas de desplazamiento vs tiempo de las pruebas registradas como la peor (prueba 4.5) y mejor estimación (prueba 3.10).

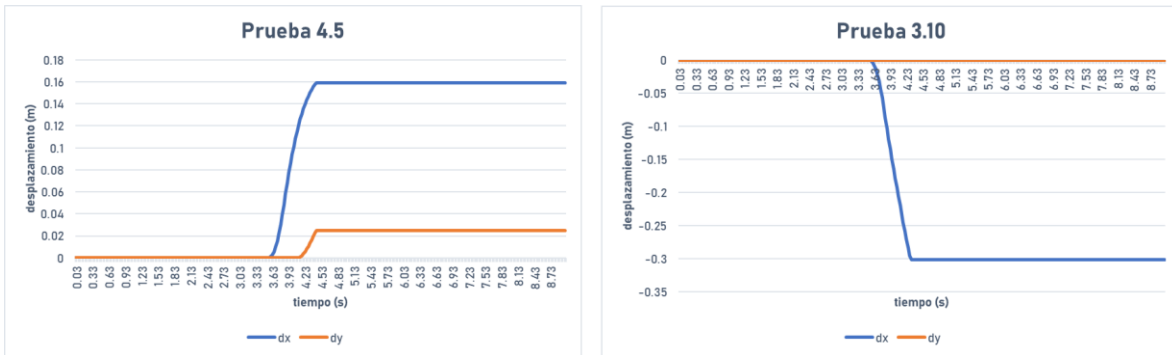


Figura 6.6. Gráficas de desplazamiento vs tiempo de las pruebas 4.5 y 3.10.

De acuerdo con la Tabla 6.2, el error acumulado durante los experimentos de 0.60 m (2, 5, 8 y 11) se registró entre 0.01 y 0.25 m. En total se realizaron 40 pruebas de 0.60 m, de las cuales el 80% (32 pruebas) registraron un error acumulado menor a 0.15 m y del 20% (8 pruebas) restante solo el 5% (2 pruebas) registran error acumulado mayor a 0.20 m.

En la Figura 6.7, se muestran las trayectorias generadas a partir de los datos obtenidos durante la prueba 11.3 registrada como la peor estimación con un error acumulado de 0.25 m y la prueba 2.9 registrada como la mejor estimación con un error acumulado de 0.01 m.

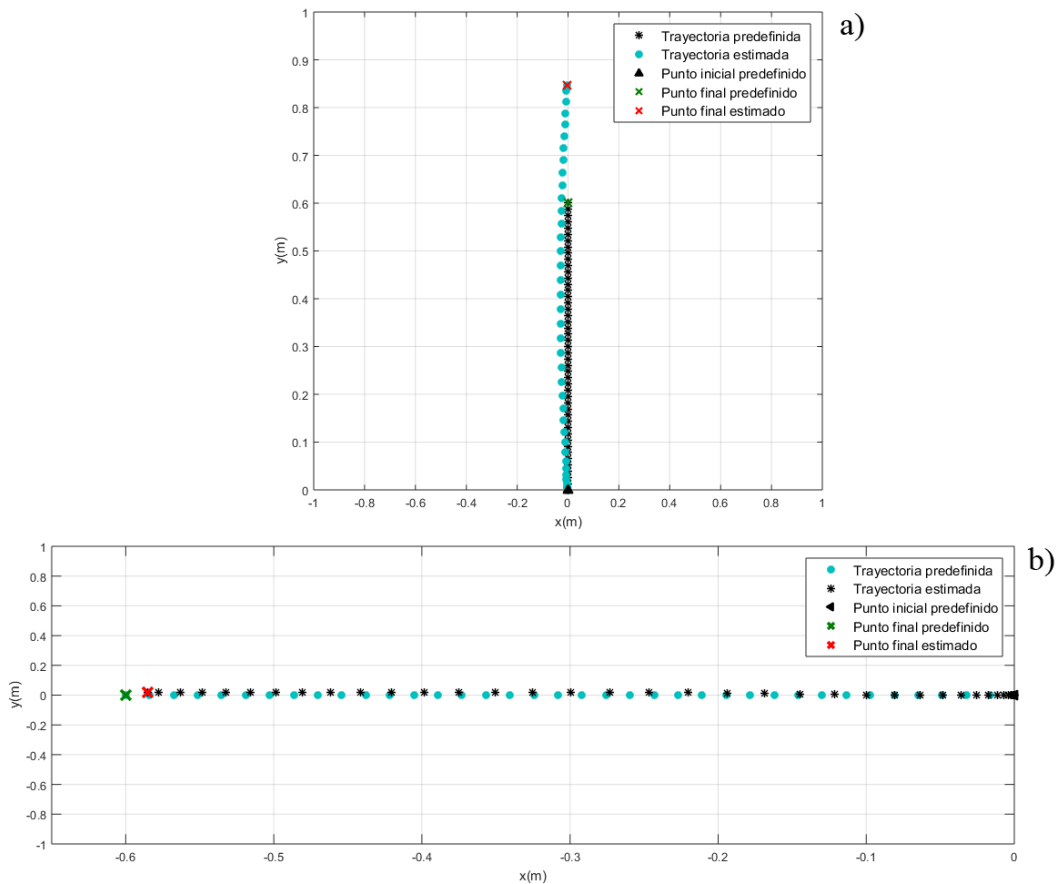


Figura 6.7. Trayectorias obtenidas por el sistema. a) Prueba 11.3 registrada como la peor estimación. b) Prueba 2.9 registrada como la mejor estimación.

En la Figura 6.8 se muestran las gráficas de desplazamiento vs tiempo de las pruebas registradas como la peor (prueba 11.3) y mejor estimación (prueba 2.9).



Figura 6.8. Gráficas de desplazamiento vs tiempo de las pruebas 11.3 y 2.9.

De acuerdo con la Tabla 6.2, el error acumulado durante los experimentos de 0.90 m (1, 6, 7 y 12) se registró entre los 0.02 y 0.26 m. En total se ejecutaron 40 pruebas de 0.90 m, de las

cuales el 77.5% (31 pruebas) registraron un error acumulado menor a 0.15 m y del 22.5% (9 pruebas) restante solo el 5% (2 pruebas) registran error acumulado mayor a 0.20 m.

En la Figura 6.9, se muestran las trayectorias generadas a partir de los datos obtenidos durante la prueba 6.4 registrada como la peor estimación con un error acumulado de 0.26 m y la prueba 7.4 registrada como la mejor estimación sin error acumulado.

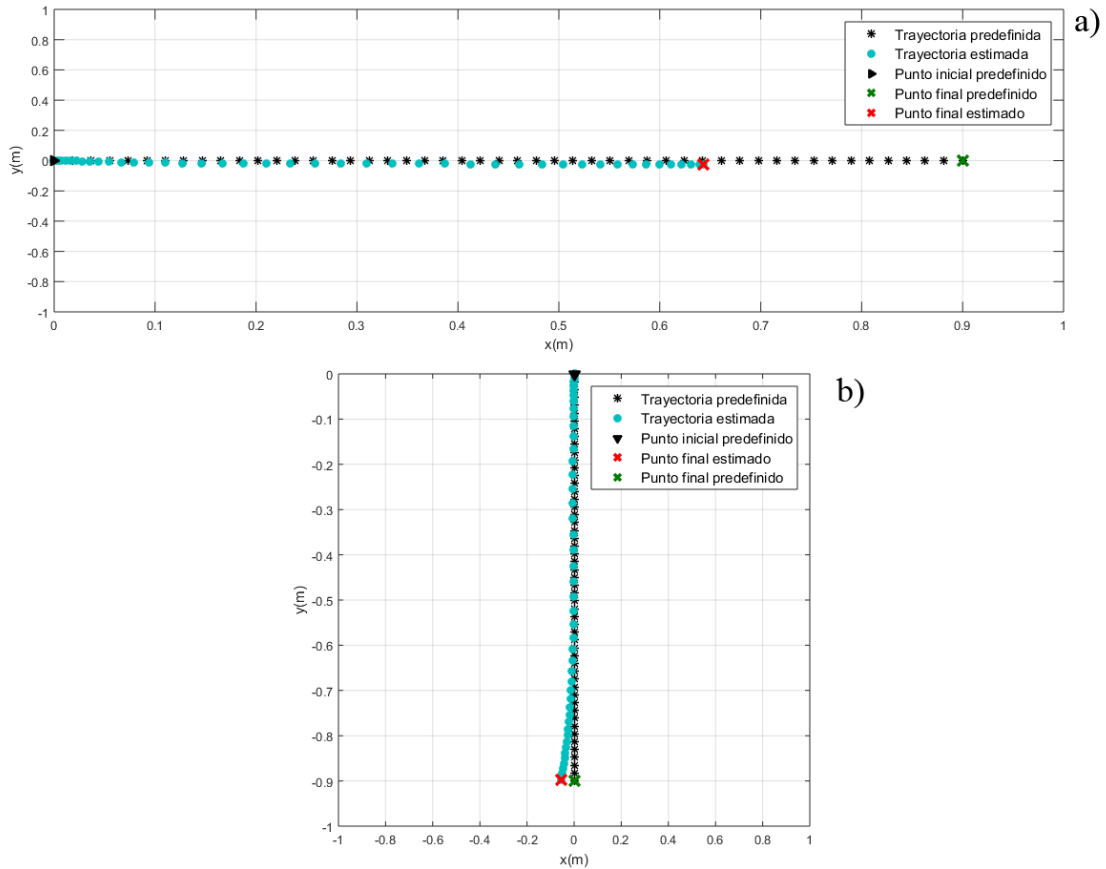


Figura 6.9. Trayectorias obtenidas por el sistema. a) Prueba 6.4 registrada como la peor estimación. b) Prueba 7.4 registrada como la mejor estimación.

En la Figura 6.10 se muestran las gráficas de desplazamiento vs tiempo de las pruebas registradas como la peor (prueba 6.4) y mejor estimación (prueba 7.4).

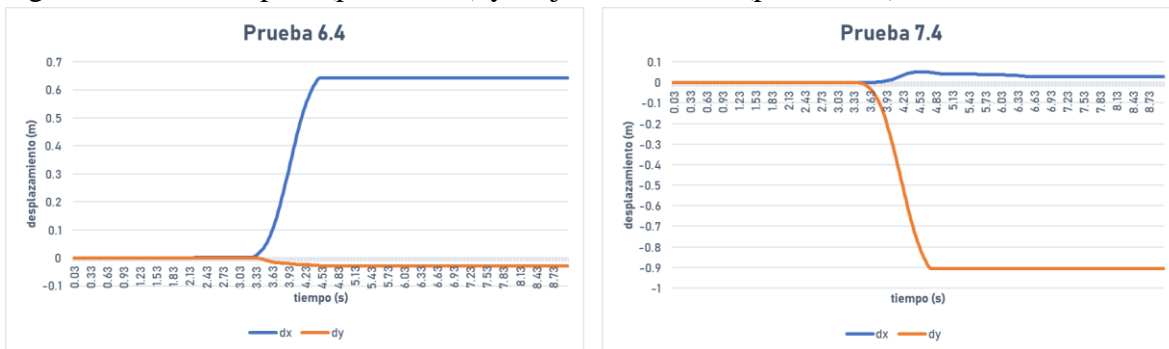


Figura 6.10. Trayectorias obtenidas por el sistema. a) Prueba 6.4. b) Prueba 7.4.

El análisis de los resultados obtenidos de los experimentos sobre los ejes x y y mostró que las estimaciones más precisas se registran ante desplazamientos de 0.30 m de longitud, mientras que las estimaciones menos precisas se registran ante movimientos de 0.90 m de longitud. Por lo que se decidió ejecutar movimientos entre 0.30 m de longitud para las pruebas de fusión, así como movimientos de 0.60 m debido a que son más fáciles de ejecutar de forma manual que los desplazamientos de 0.30 m y además presentaron un error acumulado menor a 0.15 m en el 80% de sus pruebas.

6.3 Módulo visual

Los experimentos del módulo visual del sistema de navegación inercial asistido por visión se realizaron con el objetivo de evaluar el algoritmo desarrollado para la detección de reposo y movimiento. Los experimentos consistieron en colocar el prototipo en un punto de inicio conocido, mantenerlo en reposo por unos segundos, moverlo en alguna dirección, detenerlo y mantenerlo en reposo por otros segundos, moverlo nuevamente en alguna dirección y por último detenerlo. Durante los experimentos se ejecutaron tres momentos de reposo y dos momentos de movimiento con dos direcciones diferentes y con tres velocidades diferentes. Las direcciones empleadas fueron adelante, atrás, derecha e izquierda, mientras que las velocidades ejecutadas fueron de 0.18m/s, 0.40m/s y 0.69m/s. En total se definieron 6 experimentos, tal como se muestra en la Tabla 6.3.

Tabla 6.3. Experimentos de detección de reposo y movimiento

Experimentos	Velocidad	Dirección
1	0.18m/s	reposo, adelante, reposo, atrás, reposo
2		reposo, derecha, reposo, izquierda, reposo
3	0.40m/s	reposo, adelante, reposo, atrás, reposo
4		reposo, derecha, reposo, izquierda, reposo
5	0.69m/s	reposo, adelante, reposo, atrás, reposo
6		reposo, derecha, reposo, izquierda, reposo

Los resultados obtenidos en cada una de las pruebas realizadas se presentan en el Anexo 5 y en la Tabla 6.4 se muestran los resultados obtenidos durante las series de pruebas de cada experimento ejecutado.

Tabla 6.4. Resultados de los experimentos de detección de reposo y movimiento.

Dirección	0.18 m/s	0.40 m/s	0.69 m/s
reposo, adelante, reposo, atrás, reposo	10/10	9/10	7/10
reposo, derecha, reposo, izquierda, reposo	10/10	8/10	8/10
Total	20/20	17/20	15/20
Porcentaje	100%	85%	75%

El análisis de los resultados obtenidos de los experimentos de detección de reposo y movimiento muestran buen rendimiento del algoritmo de visión ante movimientos con velocidad de 0.18 m/s, bueno ante 0.40 m/s y aceptable ante 0.69 m/s. El rendimiento del algoritmo se ve afectado por la pérdida de puntos característicos, debido a la altura de la cámara en el prototipo que es de 3 centímetros del suelo y a la alta velocidad (>0.69 m/s) durante la navegación, como se muestra en las Figuras 6.11 y 6.12.

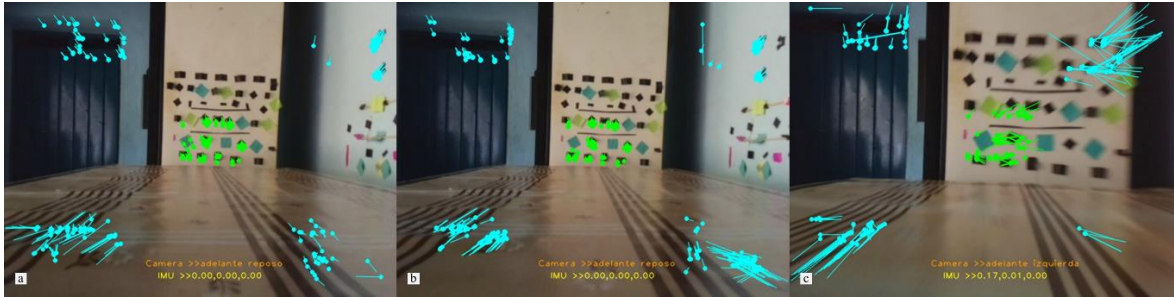


Figura 6.11. Algoritmo de detección del movimiento ante diferentes velocidades. a) 0.18 m/s. b) 0.40 m/s. c) 0.69 m/s.

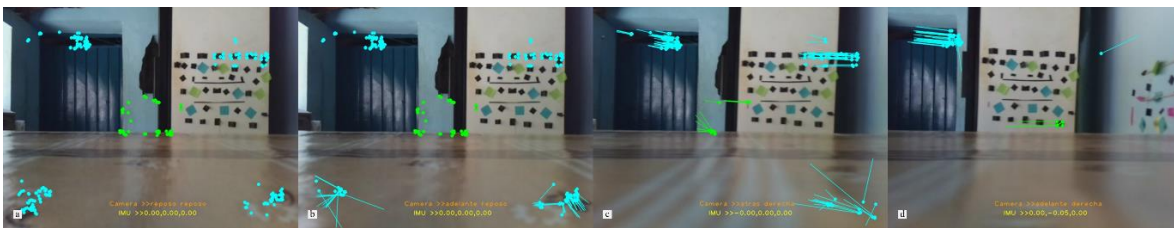


Figura 6.12. Algoritmo de detección de movimiento afectado por la pérdida de puntos característicos debido a la altura de la cámara y a la alta velocidad.

6.4 Módulo fusión de datos

Los experimentos del sistema de navegación inercial se realizaron con el objetivo de conocer los beneficios de apoyarse en datos visuales durante la navegación inercial. Para esto, se realizaron varios experimentos de forma manual que se vieron afectados por la pérdida de puntos característicos en las regiones de las esquinas bajas de la imagen, motivo por el cual se decidió utilizar un carro de control remoto de tamaño pequeño que permitiera mantener la cámara a mayor altura del suelo y así mejorar la búsqueda de puntos.

A continuación, se presentan los primeros tres experimentos que corresponden a movimientos ejecutados de forma manual. Durante los experimentos el prototipo es conducido por trayectorias predefinidas que consistieron en series de desplazamientos con longitud entre 0.30 y 0.60 m y con una velocidad de 0.69 m/s aproximadamente.

6.4.1 Experimento manual número 1

Las indicaciones de la trayectoria predefinida para el experimento, así como el resultado obtenido por el sistema se muestran en la Tabla 6.5.

Tabla 6.5. Mediciones predefinidas y estimadas del experimento manual núm. 1.

Trayectoria predefinida	Trayectoria estimada
0 m sobre el eje x 0.55 m sobre el eje y	0.03 m sobre el eje x 0.57 m sobre el eje y
Reposo	Reposo
0.60 m sobre el eje x. 0 m sobre el eje y	0.61 m sobre el eje x. -0.01 m sobre el eje y
Reposo	Reposo
Desplazamiento final	Desplazamiento final
0.55 m sobre el eje y 0.60 m sobre el eje x	0.56 m sobre el eje y 0.64 m sobre el eje x

En la Figura 6.13 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número 1. El resultado del experimento se considera bueno, debido a que la deriva es de unos cuantos centímetros.

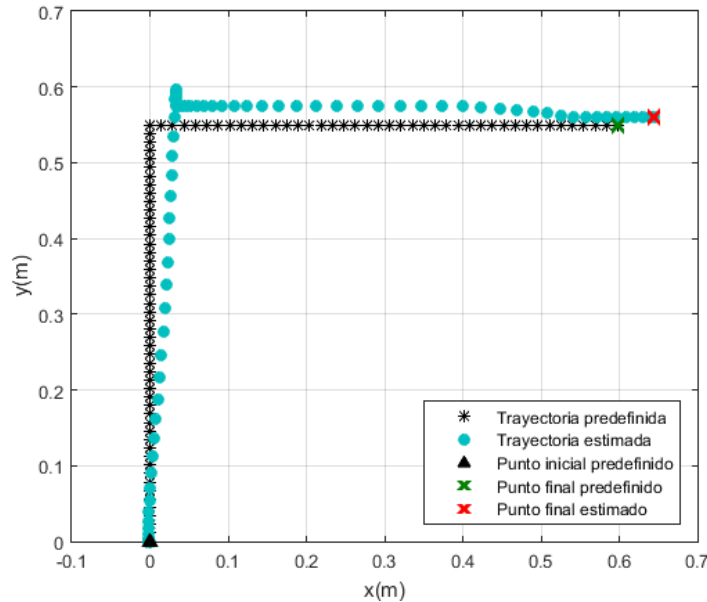


Figura 6.13. Resultado del experimento manual núm. 1.

Para una representación visual más completa, en la Figura 6.14 se muestra una gráfica de desplazamiento vs tiempo del experimento manual número 1.



Figura 6.14. Gráfica de desplazamiento vs tiempo del experimento manual número 1.

6.4.2 Experimento manual número 2

Las indicaciones de la trayectoria predefinida para el experimento, así como el resultado obtenido por el sistema se muestran en la Tabla 6.6.

Tabla 6.6. Mediciones predefinidas y estimadas del experimento manual número 2.

Trayectoria predefinida	Trayectoria estimada
0 m sobre el eje x 0.55 m sobre el eje y	0.05 m sobre el eje x 0.68 m sobre el eje y
Reposo	Reposo
0.60 m sobre el eje x. 0 m sobre el eje y	0.34 m sobre el eje x. -0.02 m sobre el eje y
Reposo	Reposo
0 m sobre el eje x -0.55 m sobre el eje y	-0.04 m sobre el eje x -0.55 m sobre el eje y
Reposo	Reposo
-0.60 m sobre el eje x. 0 m sobre el eje y	-0.49 m sobre el eje x. -0.05 m sobre el eje y
Reposo	Reposo
Desplazamiento final	Desplazamiento final
0 m sobre el eje x 0 m sobre el eje y	-0.14 m sobre el eje x 0.06 m sobre el eje y

En la Figura 6.15 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número dos. El resultado del experimento se considera no tan bueno, debido a que la trayectoria estimada se queda lejos de lo trayectoria predefinida. También se puede observar que los errores del primer y segundo movimiento afectan las mediciones del tercer y cuarto movimiento que eran buenas.

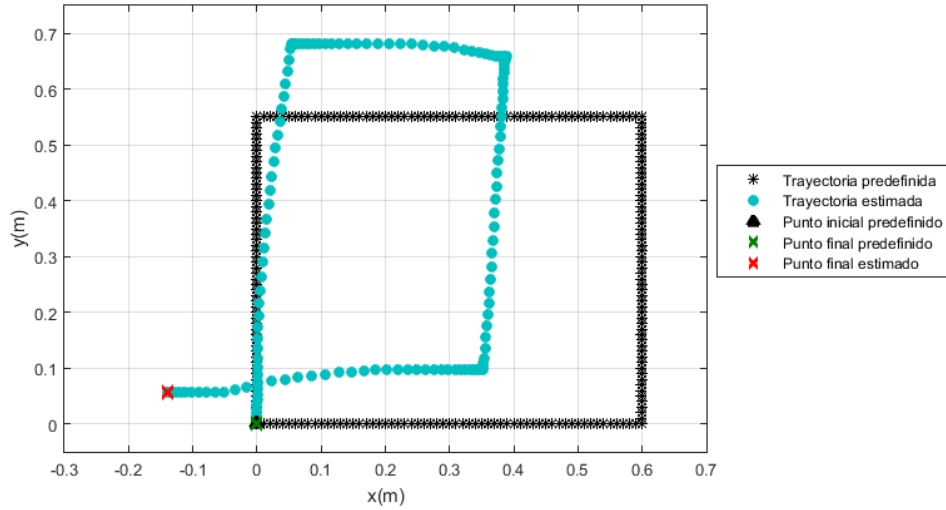


Figura 6.15. Resultado del experimento manual núm. 2.

Para una representación visual más completa, en la Figura 6.16 se muestra una gráfica de desplazamiento vs tiempo del experimento manual número 2.

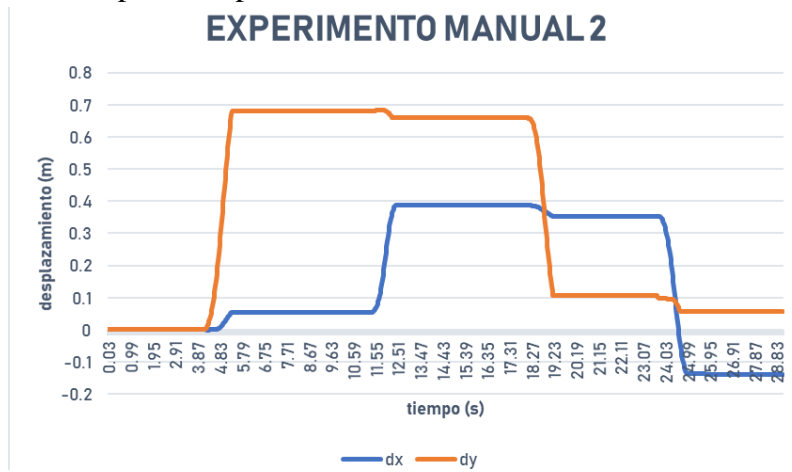


Figura 6.16. Gráfica de desplazamiento vs tiempo del experimento manual núm. 2.

6.4.3 Experimento manual número 3

Las indicaciones de la trayectoria predefinida para el experimento, así como el resultado obtenido por el sistema se muestran en la Tabla 6.7.

Tabla 6.7. Mediciones predefinidas y estimadas del experimento manual núm. 3.

Trayectoria predefinida	Trayectoria estimada
0 m sobre el eje x 0.40 m sobre el eje y	0.05 m sobre el eje x 0.68 m sobre el eje y
Reposo	Reposo
0.60 m sobre el eje x. 0 m sobre el eje y	0.34 m sobre el eje x. -0.02 m sobre el eje y
Reposo	Reposo
Regreso al punto inicial	Regreso al punto 0.04,-0.28
Reposo	Reposo
Desplazamiento final	Desplazamiento final
0 m sobre el eje x 0 m sobre el eje y	0.04 m sobre el eje x -0.28 m sobre el eje y

En la Figura 6.17 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número tres. El resultado del experimento se considera malo, debido a que la trayectoria estimada se queda lejos de lo trayectoria predefinida. Sin embargo, en este experimento se puede apreciar la detección de movimiento en dos dimensiones (x, y) indicando hacia atrás y hacia la derecha al mismo tiempo.

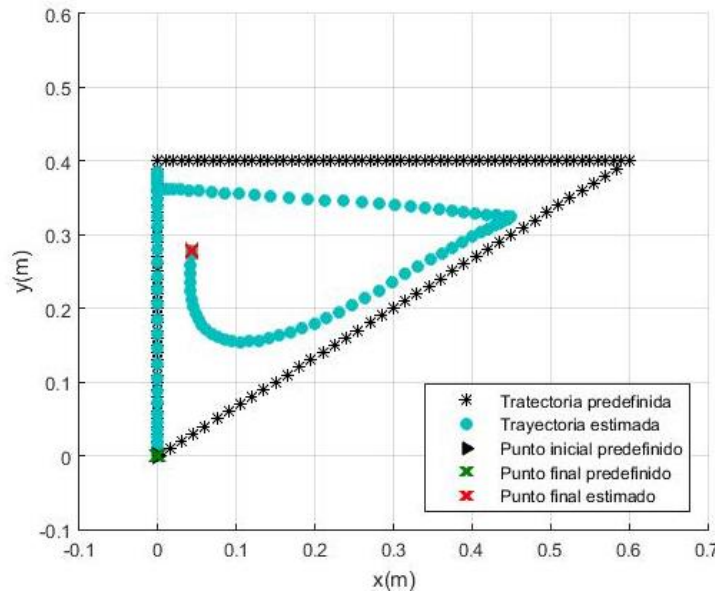


Figura 6.17. Resultado del experimento manual núm. 3.

Para una representación visual más completa, en la Figura 6.18 se muestra una gráfica de desplazamiento vs tiempo del experimento manual número 3.

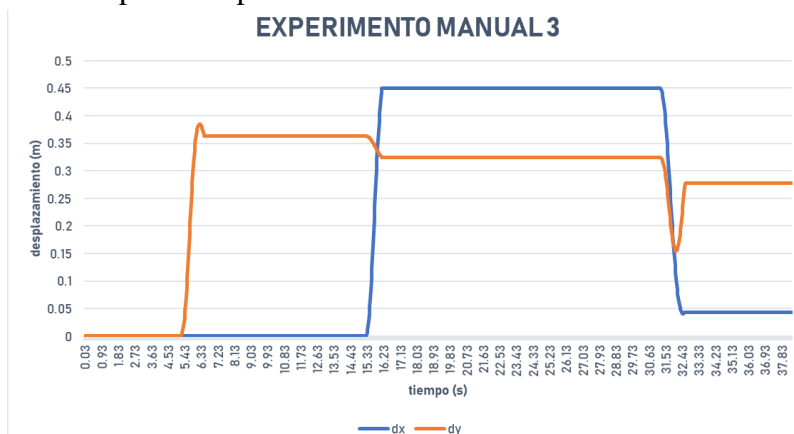


Figura 6.18. Gráfica de desplazamiento vs tiempo del experimento manual núm. 3.

Por último, se presentan los tres experimentos que corresponden a movimientos ejecutados por un carro de control. Durante los experimentos el prototipo fue montado en el carro de control y conducido por trayectorias que consistieron en series de desplazamientos con longitud entre 1 y 4 m con una velocidad un poco menor a los 0.69 m/s aproximadamente.

6.4.4 Experimento a control remoto número 1

El experimento uno consistió en conducir el carrito sobre una trayectoria recta de 4 m de longitud. Las maniobras realizadas durante el recorrido, así como el resultado obtenido por el sistema de navegación desarrollado se muestran en la Tabla 6.8.

Tabla 6.8. Trayectoria predefinida y estimada del experimento a control remoto número 1.

Trayectoria predefinida	Trayectoria estimada
1.5 m sobre el eje x 0 m sobre el eje y	1.22 m sobre el eje x -0.04 m sobre el eje y
Reposo	Reposo
1.5 m sobre el eje x. 0 m sobre el eje y	1.78 m sobre el eje x. -0.03 m sobre el eje y
Reposo	Reposo
1 m sobre el eje x. 0 m sobre el eje y	1.15 m sobre el eje x. -0.10 m sobre el eje y
Reposo	Reposo
Desplazamiento final	Desplazamiento final
4 m sobre el eje x 0 m sobre el eje y	4.15 m sobre el eje x -0.17 m sobre el eje y

En la Figura 6.19 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número uno. El resultado del experimento se considera bueno, debido a que la deriva es menor a 0.50 m.

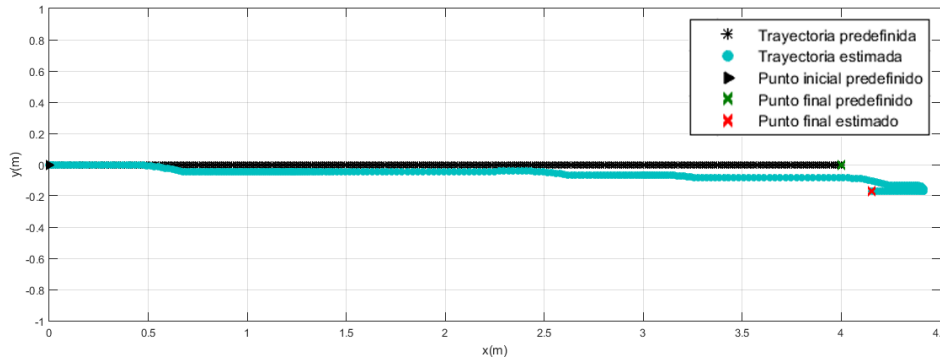


Figura 6.19. Resultado del experimento a control remoto núm. 1.

Para una representación visual más completa, en la Figura 6.20 se muestra una gráfica de desplazamiento vs tiempo del experimento a control remoto número 1.

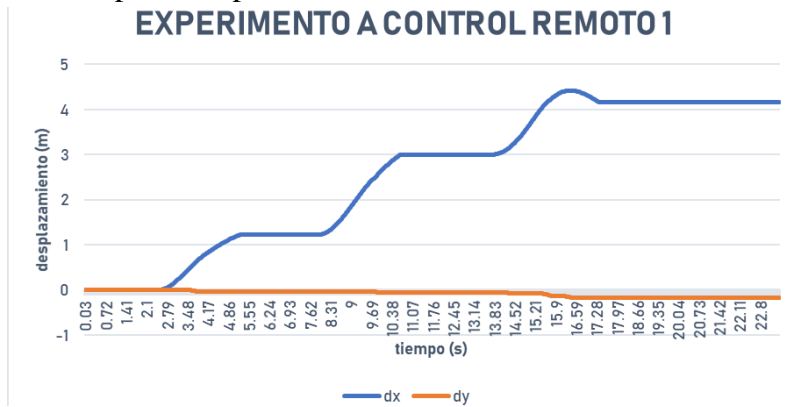


Figura 6.20. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 1.

6.4.5 Experimento a control remoto número 2

El experimento dos consistió en conducir el carrito sobre una trayectoria recta de 4 m de longitud. Las maniobras realizadas durante el recorrido, así como el resultado obtenido por el sistema de navegación desarrollado se muestran en la Tabla 6.9.

Tabla 6.9. Trayectoria predefinida y estimada del experimento a control remoto número 2.

Trayectoria predefinida	Trayectoria estimada
4 m sobre el eje x 0 m sobre el eje y	9.32 m sobre el eje x -0.36 m sobre el eje y
Reposo	Reposo
Desplazamiento final	Desplazamiento final
4 m sobre el eje x 0 m sobre el eje y	9.32 m sobre el eje x -0.36 m sobre el eje y

En la Figura 6.21 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número dos. El resultado del experimento muestra el error acumulado durante un recorrido de 4 metros sin momentos de reposo y pérdida continua de puntos característicos.

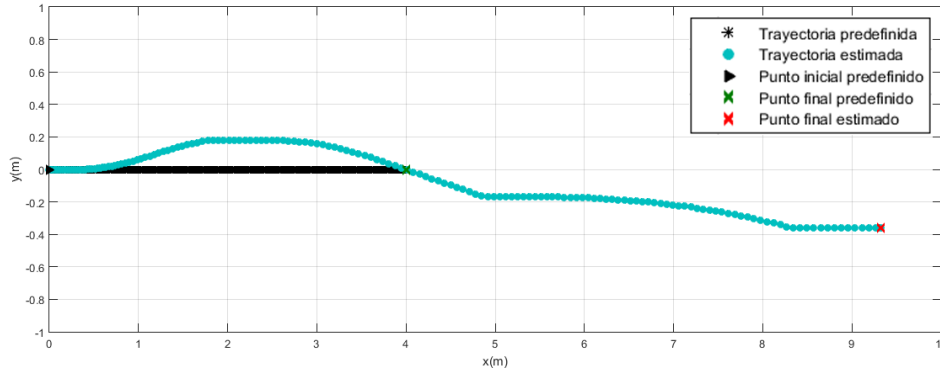


Figura 6.21. Resultado del experimento a control remoto núm. 2.

Para una representación visual más completa, en la Figura 6.22 se muestra una gráfica de desplazamiento vs tiempo del experimento a control remoto número 2.

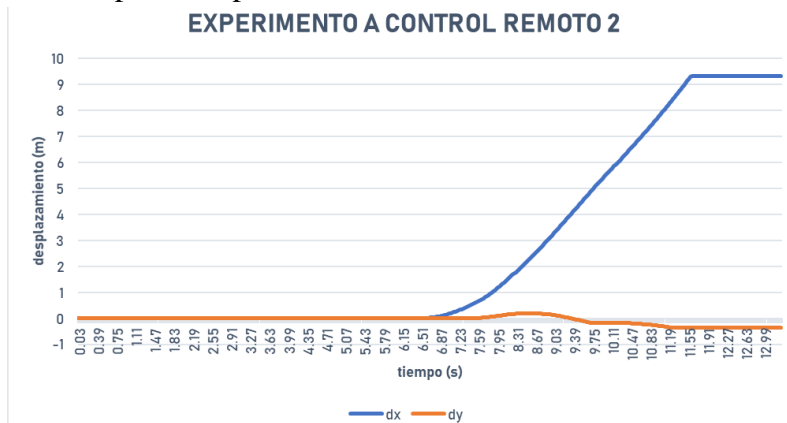


Figura 6.22. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 2.

6.4.6 Experimento a control remoto número 3

El experimento tres consistió en conducir el carrito sobre una trayectoria recta, detenerse y volver a avanzar en un ángulo de 45°. Las maniobras realizadas durante el recorrido, así como el resultado obtenido por el sistema de navegación desarrollado se muestran en la Tabla 6.10.

Tabla 6.10. Trayectoria predefinida y estimada del experimento a control remoto número 3.

Trayectoria predefinida	Trayectoria estimada
3 m sobre el eje x	2.84 m sobre el eje x
0 m sobre el eje y	0.01 m sobre el eje y
Reposo	Reposo

Trayectoria predefinida	Trayectoria estimada
1 m sobre el eje x -1.5 m sobre el eje y girando 45° sobre el eje z	1.01 m sobre el eje x -1.89 m sobre el eje y girando 43° sobre el eje z
Desplazamiento final	Desplazamiento final
4 m sobre el eje x -1.5 m sobre el eje y	3.85 m sobre el eje x -1.88 m sobre el eje y

En la Figura 6.23 se muestra la trayectoria predefinida y la trayectoria obtenida por el sistema de navegación desarrollado del experimento número tres. El resultado del experimento se considera bueno, es importante recalcar que durante el recorrido el algoritmo de visión no perdió puntos característicos.

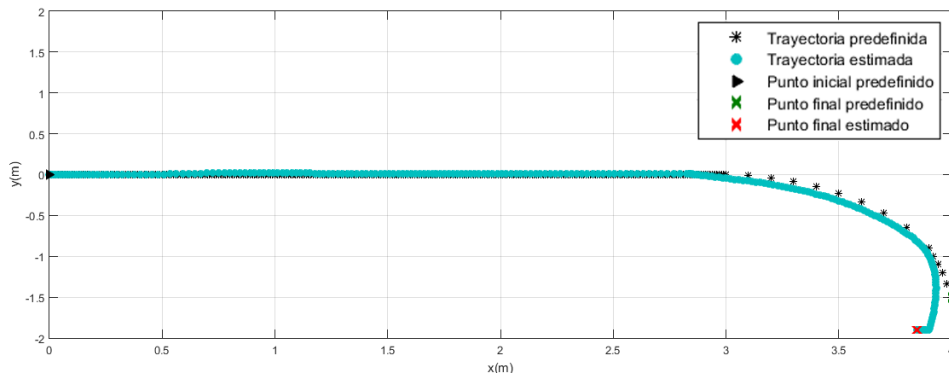


Figura 6.23. Resultado del experimento a control remoto núm. 3.

Para una representación visual más completa, en la Figura 6.24 se muestra una gráfica de desplazamiento vs tiempo del experimento a control remoto número 3.

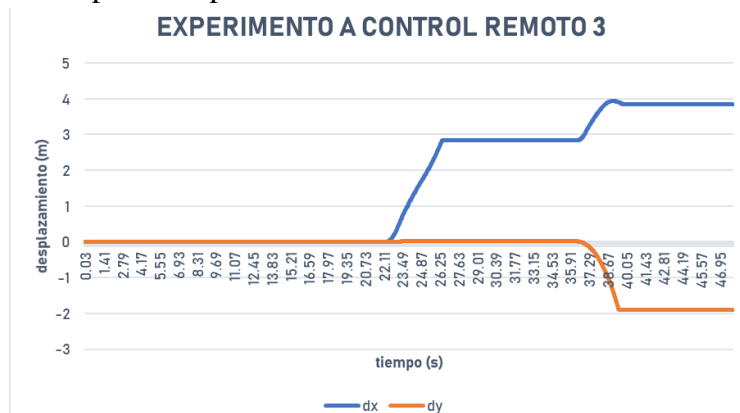


Figura 6.24. Gráfica de desplazamiento vs tiempo del experimento a control remoto núm. 3.

El análisis de los resultados obtenidos de los experimentos del sistema desarrollado muestra un bajo rendimiento en la estimación de la odometría inercial durante la conducción de prototipo de forma manual y un mejor rendimiento durante la conducción a control remoto. Una de las razones de la mejora del rendimiento es la posición de la cámara que se encontraba



a 13 centímetros del suelo, propiciando el resultado de la búsqueda y correspondencia de puntos característicos. En las Figuras 6.25, 6.26 y 6.27 se muestra el rendimiento en la detección de movimiento ante traslaciones en diferentes direcciones ejecutadas por el carrito de control remoto.

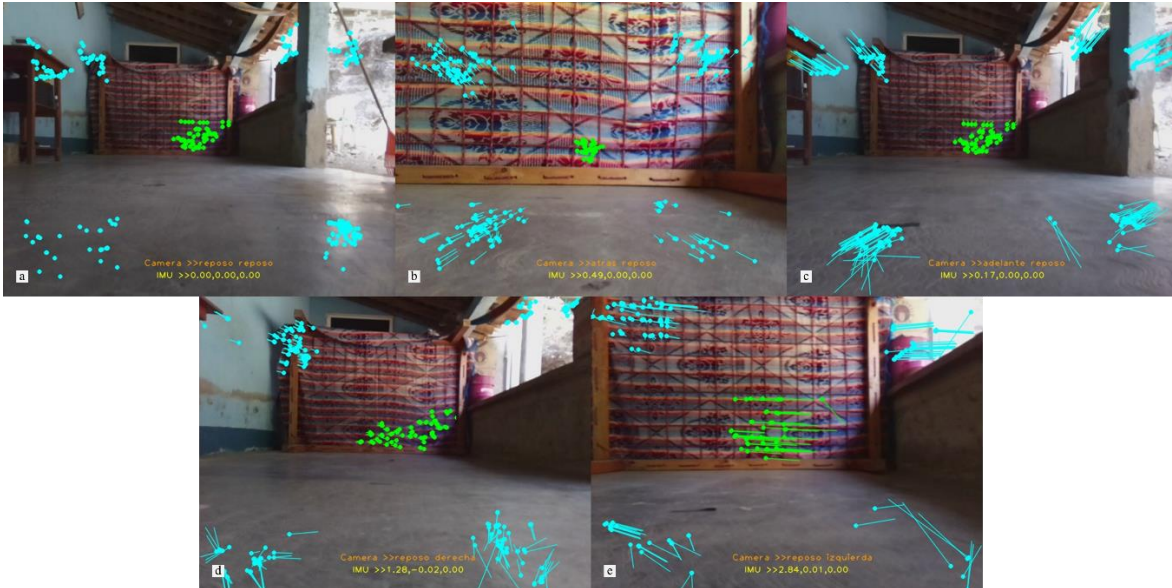


Figura 6.25. Resultados del algoritmo de detección de movimientos en una dimensión durante recorridos ejecutados por el carrito de control remoto. a) reposo. b) atrás. c) adelante. d) derecha. e) izquierda.



Figura 6.26. Resultados del algoritmo de detección de movimientos en dos dimensiones durante recorridos ejecutados por el carrito de control remoto. a) adelante/izquierda. b) adelante/derecha.



Figura 6.27. Resultados del algoritmo de detección de movimientos ante pérdidas de puntos característicos durante recorridos ejecutados por el carrito de control remoto.

6.5 Comparación

Al iniciar el proyecto se desarrolló un sistema de navegación de odometría inercial puro, en el IDE de Arduino 1.8.5 utilizando las mismas técnicas de odometría inercial presentes en el sistema de navegación inercial asistido por visión. El sistema se desarrolló para estudiar el error acumulado generado durante la navegación inercial, así como para ser comparado con el sistema desarrollado como producto de la investigación. Por lo tanto, se realizó una prueba que consistió en desplazar el prototipo 0.55 m en línea recta sobre el eje y , detenerlo, volver a desplazarlo 0.60 m en línea recta sobre el eje x y detenerlo para terminar el recorrido. En la Figura 6.28 se puede observar las trayectorias obtenidas por cada uno de los sistemas, permitiendo conocer el grado de mejora del sistema puramente inercial asistíéndose de datos visuales.

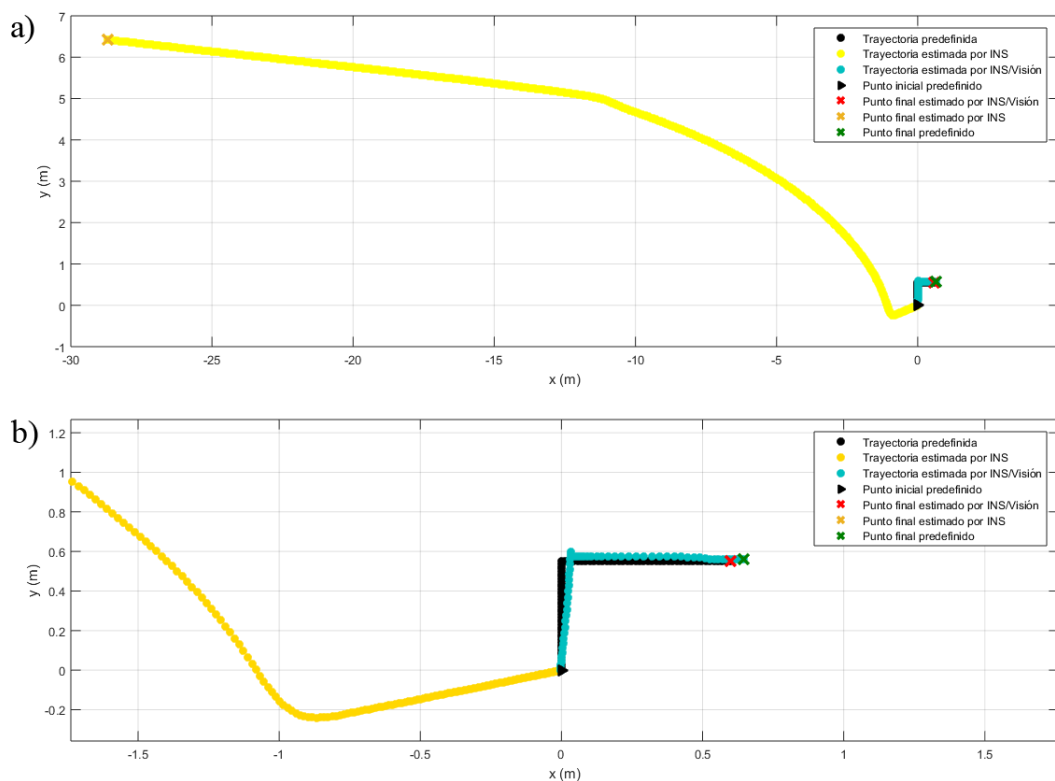


Figura 6.28. Resultados obtenidos por INS (línea amarilla) e INS/Visión (línea turquesa) a partir de la trayectoria predefinida (línea negra). a) trayectoria del recorrido. b) trayectoria del recorrido con acercamiento para mejorar su visualización.

En la Figura 6.29 se muestra una gráfica de desplazamiento vs tiempo de las mediciones obtenidas por el sistema de navegación inercial puro y el sistema de navegación inercial asistido por visión. En ella se puede observar el beneficio de poner la velocidad a cero durante el reposo, para reducir los errores ocasionados por el ruido del sensor y la doble integración durante el movimiento.

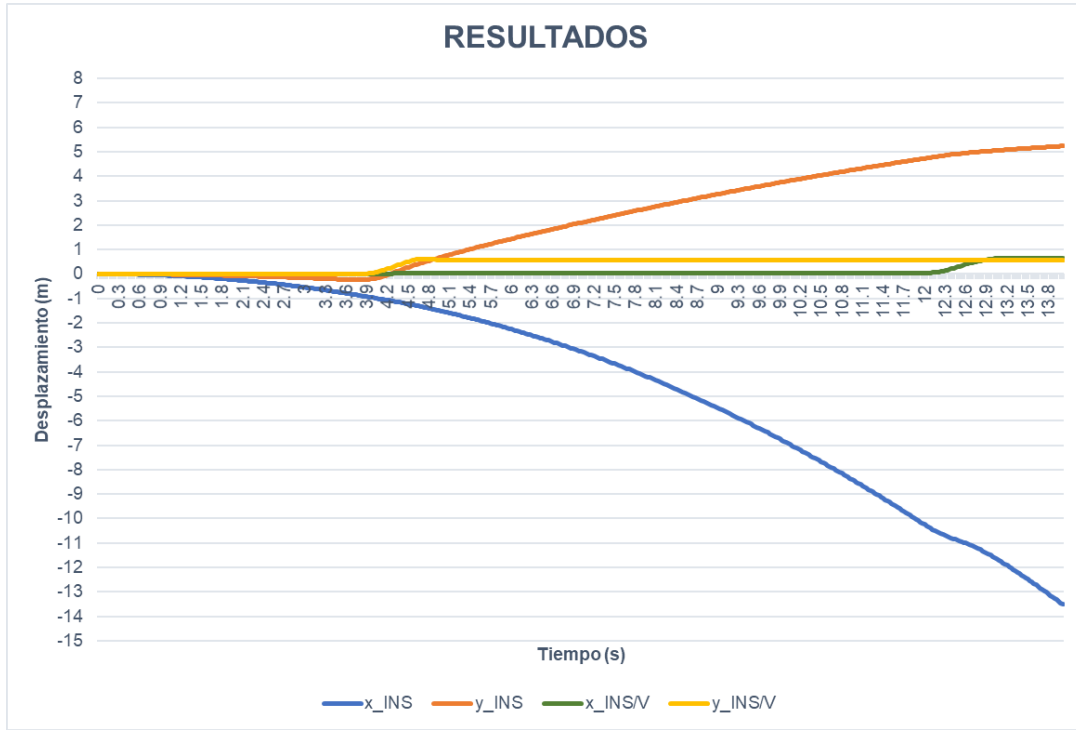


Figura 6.29. Gráfica de desplazamiento vs tiempo de las mediciones obtenidas por INS e INS/Visión.

En la Tabla 6.11 se muestran las mediciones obtenidas por los sistemas INS e INS_Visión durante la prueba de comparación. Los resultados muestran para el INS errores de 29.306 en x y 5.873 en y , mientras que para el INS_Visión se muestran errores de 0.044 en x y 0.060 en y . Estos resultados demuestran que un sistema de navegación inercial se puede apoyar de datos visuales para reducir la deriva en la estimación de la posición.

Tabla 6.11. Comparación entre los resultados obtenidos por INS e INS_Visión.

dt(s)	Trayectorias		
	Predefinida	INS	INS_Visión
5.04	(0, 0.55)	(-1.549, 0.746)	(0.0328, 0.574)
13.08	(0.60, 0.55)	(-11.709, 5.063)	(0.644, 0.560)
20.01	(0.60, 0.55)	(-28.706, 6.423)	(0.644, 0.560)
Error	(0,0)	(29.306, 5.873)	(0.044, 0.060)

6.6 Discusión

De acuerdo con los resultados de la experimentación, para obtener un buen rendimiento del sistema desarrollado es necesario que, durante la navegación, se realicen desplazamientos entre 0.50 y 1.0 m de longitud. Además, que entre cada desplazamiento realizado exista un momento de reposo, para no permitir el crecimiento del error.

Capítulo 7

Conclusiones y trabajos futuros

En este último capítulo se presentan las conclusiones, los productos, las aportaciones y los trabajos futuros emanados como resultado del trabajo de tesis.

7.1 Conclusiones

El sistema de navegación desarrollado permite estimar la posición del prototipo a través de las ecuaciones de navegación inercial empleadas para el cálculo de la odometría y el flujo óptico utilizado para la detección de movimiento.

Los resultados de la experimentación mostraron que la forma de conducción del prototipo produce resultados diferentes. En forma manual la estimación presentó mejor desempeño en desplazamientos entre 0.30 y 0.60 m de longitud, mientras que, a control remoto lograba estimar con buen rendimiento desplazamientos de hasta 1 m. Esto se debe al resultado obtenido durante la búsqueda y la correspondencia de puntos característicos utilizados para el cálculo del flujo óptico. Durante la conducción de forma manual la cámara estaba a 3 centímetros del suelo, lo que provocaba escasez de puntos característicos en dos de las regiones de interés (roi 3 y roi 4). Mientras que, durante la conducción a control remoto la cámara estaba a 13 centímetros del suelo, lo que propiciaba la búsqueda y correspondencia de puntos característicos en las dos regiones afectadas en la conducción realizada de forma manual. Además de la altura de la cámara, otro motivo que provocaba la pérdida de puntos característicos fueron las altas velocidades con las que se ejecutaban los movimientos.

El sistema depende de la búsqueda y correspondencia de puntos característicos, por lo que una mala detección de puntos puede producir grandes errores en la estimación de la odometría inercial del prototipo durante la navegación. Por lo tanto, se recomienda mejorar el algoritmo de detección de movimiento de modo que ante altas velocidades sea capaz de distinguir el movimiento ejecutado.

Gracias a los experimentos realizados se puede demostrar que un sistema de odometría inercial puro se puede apoyar de datos visuales para reducir la deriva durante la navegación.

En este proyecto se desarrolló e implementó un sistema de navegación inercial que reduce el error con la asistencia de datos visuales, para esto el sistema mantiene las velocidades lineales y angulares en cero mientras la visión detecte que el prototipo se encuentra en reposo y ejecuta las ecuaciones de navegación mientras el prototipo se encuentre en movimiento. Al igual que el método desarrollado existen muchos más, como calcular tanto odometría inercial como visual y después fusionarlas mediante un Filtro de Kalman o desarrollar un podómetro en caso de la navegación peatonal.

7.2 Objetivos alcanzados

El objetivo general de este trabajo definido como “estudio e implementación de técnicas de odometría para las señales de la unidad de medición inercial aplicables a robots móviles pequeños con visión”, se desarrolló de manera satisfactoria y cinco de los seis objetivos específicos se cumplieron de acuerdo con lo establecido. El sexto objetivo que se refiere a la heurística, también se cumplió, pero se modificó durante el desarrollo del proyecto. En la Tabla 7.1, se muestran los comentarios correspondientes a cada uno de los objetivos específicos.

Tabla 7.1. Comentarios de los objetivos específicos.

Objetivos específicos	Comentarios
1. Conocer y comprender el funcionamiento de la IMU.	Se estudiaron artículos, tesis, reportes y tutoriales para cumplir con el objetivo propuesto.
2. Estudiar técnicas de detección de movimiento a partir de imágenes.	Se estudiaron las técnicas más adecuadas con base en el estudio del estado del arte. De las cuales, el flujo óptico se adaptó mejor a las necesidades del proyecto.
3. Construir un prototipo integrado por una IMU y una cámara.	Se construyó un prisma rectangular de acrílico, integrado principalmente por una IMU Adafruit 9 DoF, un Arduino DUE, una Raspberry Pi 3 y una Cámara pi v2.

Objetivos específicos	Comentarios
4. Implementar técnicas de odometría inercial.	Las técnicas implementadas para obtener la odometría inercial del prototipo primeramente calibran y filtran las señales de la IMU; a continuación, calculan la orientación integrando las mediciones del giroscopio; enseguida la orientación se emplea para restar los efectos de la gravedad de las mediciones del acelerómetro; y, por último, se estima la posición integrando doblemente las aceleraciones.
5. Implementar técnicas de odometría inercial con visual	Se desarrolló un sistema de navegación ligeramente acoplado, en el cual se implementaron tanto las técnicas de odometría inercial, como las técnicas de detección de movimiento.
6. Proponer una heurística que permita trabajar, en diferentes momentos, con datos del IMU o visuales, dependiendo de los escenarios de navegación, y cuantificar la reducción del error de dicha heurística.	Se propuso una heurística que mediante la detección del estado (“movimiento” o “reposo”) por medio de datos visuales, determina en qué momento integrar las mediciones de la IMU. Dicha heurística obtiene un error menor a los 0.08 m en trayectorias de longitud total de 1.1 m.

7.3 Productos

- Se desarrolló una heurística que permite controlar las integraciones por componentes (x, y) del INS mediante la detección de movimiento utilizando la información del flujo óptico. Esta heurística, funciona detectando por medio de la visión si el prototipo se encuentra en reposo o movimiento. En caso de movimiento se determina el o los componentes con movimiento y se procesa a integrar sus velocidades correspondientes. En caso de reposo, las velocidades se mantienen en cero.

7.4 Aportaciones

- Se construyó un prototipo integrado principalmente por una IMU y una cámara, el cual puede utilizarse en trabajos futuros de visión robótica. Este prototipo es de tamaño pequeño y un peso menor a los 0.50 kg, por lo que puede ser montado en infinidad de robots móviles pequeños. Además, su arquitectura permite integrar nuevos componentes para enriquecer el rendimiento del sistema.

- Se desarrolló una interfaz gráfica para el control del sistema de navegación, dicha interfaz permite establecer la comunicación entre los componentes y controlar la ejecución de los procesos para la obtención de los datos de navegación.
- Se implementó una mejora al método de detección del estado del vehículo presentado en Aroca (2015), dicho método utilizaba toda la imagen para determinar si el vehículo se encontraba en reposo o movimiento. La mejora consistió en utilizar solamente 5/9 de la imagen y poder detectar las direcciones (adelante, atrás, derecha e izquierda) del movimiento sobre los ejes x y y además de la detección del reposo.
- Se desarrolló un sistema de navegación que mediante el acoplamiento de datos inerciales con visuales reduce el error acumulado en las mediciones del INS.

7.5 Trabajos futuros

- El sistema de navegación opera en dos dimensiones, por lo que es aplicable a robots móviles terrestres con ruedas. Sería interesante trabajar la tercera dimensión para conocer la altura a la que navega el robot, lo que permitiría al sistema ser aplicable a robots aéreos. Para esto, se puede incorporar un sensor de presión barométrica.
- Mejorar el algoritmo de búsqueda y correspondencia de puntos característicos, de modo que se encuentre mayor cantidad de puntos característicos propiciando un mejor rendimiento del algoritmo de detección de movimiento.
- Las pruebas se realizaron de forma manual y utilizando un carrito de control, sin embargo, sería interesante utilizar un robot móvil con su propio sistema de control. En este último resultaría interesante observar el funcionamiento del sistema, además información del sistema de control puede enriquecer la información de navegación del robot móvil.
- Dentro del estado del arte, se menciona al Filtro de Kalman Extendido (EKF) como la técnica más empleada y con mejores resultados en la fusión de datos inerciales con visuales, por lo que sería bastante bueno estudiarlo e implementarlo.

Referencias

- Adafruit Industries. (2018). Introduction | Adafruit 9-DOF IMU Breakout | Adafruit Learning System. Retrieved April 3, 2018, from <https://learn.adafruit.com/adafruit-9-dof-imu-breakout>
- Álvarez, C. (2016). *Navegación inercial en aeronaves* (1ª edición). Madrid, España: Ediciones Paraninfo, S.A.,2016.
- Amezcuca, R., & Pineda, A. (2012). *Sistema de referencia inercial: análisis de funcionamiento, fundamentos y evolución*. Tesina de ingeniería, Depto. Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional, México D.F, Enero 2012.
- Andrejašič, M. (2008). *MEMS ACCELEROMETERS. Report, Dept.Physics, University of Ljubljana*. Marec 2008. <https://doi.org/10.1049/ep.1965.0296>
- Arduino Org. (2018a). Arduino Due. Retrieved April 3, 2018, from <https://store.arduino.cc/usa/arduino-due>
- Arduino Org. (2018b). Download the Arduino IDE. Retrieved November 30, 2018, from <https://www.arduino.cc/en/main/software>
- Aroca, F. (2015). *Diseño de un control de posición X-Y de un quadrotor basado en un algoritmo de flujo óptico*. Trabajo de fin de grado, Depto. Ingeniería en Tecnologías Industriales, Universitat Politècnica De València, Valencia, 2015.
- Babu, B. P. W., Cyganski, D., & Duckworth, J. (2014). Gyroscope assisted scalable visual simultaneous localization and mapping. *IEEE Conference on Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), Corpus Christ, Texas, USA, Nov 20-21, 2014*, 220–227. <https://doi.org/10.1109/UPINLBS.2014.7033731>
- Barrett, J. M., Gennert, M. A., Michalson, W. R., Audi, M. D., Center, J. L., Kirk, J. F., & Welch, B. W. (2013). Development of a low-cost, self-contained, combined vision and inertial navigation system. *IEEE Conference on Technologies for Practical Robot Applications, TePRA*, 1–6. <https://doi.org/10.1109/TePRA.2013.6556351>
- Barrientos Sotelo, V. R., García Sánchez, J. R., & Silva Ortigoza, R. (2007). Robots Móviles : Evolución y Estado del Arte. *Polibits [En Linea] 2007*, 35, 12–17. Retrieved from <http://www.redalyc.org/articulo.oa?id=402640448003>
- Bastian, P., Eichler, W., Huber, F., Jaufmann, N., Manderla, J., Spielvogel, O., ... Tkotz, K. (2001). *Electrotecnia (vol. 1 de Ciclos formativos)*. España: Ediciones AKAL, 2001.
- Budiyono, A. (2008). Advances in Unmanned Aerial Vehicles Technologies. *International Symposium on Intelligent Unmanned System (ISIUS 2008-Nanjing)*, 1–13.
- Craig, J. J. (2004). *Introduction to Robotics: Mechanics and Control*. Prentice Hall (Third Edit, Vol. 1). <https://doi.org/10.1109/MEX.1986.4306961>
- Elarabi, T., & Suprem, A. (2015). Orientation and Displacement Detection for Smartphone Device Based Inertial Measurement Units. *IEEE Access*, 5, 122–126.

<https://doi.org/10.1109/ACCESS.2016.2631000>

- Elharrouss, O., Moujahid, D., & Tairi, H. (2015). Motion detection based on the combining of the background subtraction and the structure-texture decomposition. *Optik*, 126(24), 5992–5997. <https://doi.org/10.1016/j.ijleo.2015.08.084>
- Fang, W., & Zheng, L. (2018). Rapid and robust initialization for monocular visual inertial navigation within multi-state kalman filter. *Chinese Journal of Aeronautics*, 31(1), 148–160. <https://doi.org/10.1016/j.cja.2017.10.011>
- Ferrer, D. (2015). *Adquisición de datos IMU en un sistema embebido*. Trabajo de fin de grado, Depto. Ingeniería en Geomática y Topografía, Universitat Politècnica de València, Valencia, julio, 2015.
- Fuentes varias. (2018). Imágenes de Google. Retrieved April 20, 2018, from <https://www.google.com.mx/imghp?hl=es&tab=wi&authuser=0>
- funvision.blogspot. (2015). Opencv C++ Tutorial, Mat Roi, Region of interest - Fun Computer Vision opencv tutorials and .. Retrieved April 15, 2018, from <http://funvision.blogspot.com/2015/12/basic-opencv-3-mat-tutorial-part-2-roi.html>
- Gómez, C. L. (2013). *Centro Nacional de Investigación y Desarrollo Tecnológico*. tesis de maestría, Depto. Ciencias Computacionales, CENIDET, Cuernavaca, 2013.
- Grewal, M. S., & Andrews, A. P. (2008). *Kalman filtering - theory and practice using matlab* (3rd ed., Vol. 53). New Jersey: John Wiley & Sons, Inc., Hoboken, 2008. <https://doi.org/10.1017/CBO9781107415324.004>
- Heo, S., Cha, J., & Park, C. G. (2017). Monocular Visual Inertial Navigation for Mobile Robots using Uncertainty based Triangulation. *IFAC-PapersOnLine*, 50(1), 2217–2222. <https://doi.org/10.1016/j.ifacol.2017.08.928>
- Hesch, J. A., Kottas, D. G., Bowman, S. L., & Roumeliotis, S. I. (2014). Consistency analysis and improvement of vision-aided inertial navigation. *IEEE Transactions on Robotics*, 30(1), 158–176. <https://doi.org/10.1109/TRO.2013.2277549>
- i2c-bus.org. (2018a). I2C - What's That? - I2C Bus. Retrieved July 21, 2018, from <https://www.i2c-bus.org/>
- i2c-bus.org. (2018b). TWI Bus - I2C Bus. Retrieved July 21, 2018, from <https://www.i2c-bus.org/twi-bus/>
- Ibrahim, M., & Moselhi, O. (2016). Inertial measurement unit based indoor localization for construction applications. *Automation in Construction*, 71, 13–20. <https://doi.org/10.1016/j.autcon.2016.05.006>
- Kale, H., & Limaye, S. S. (2014). Autocorrelation of a Sound signal. *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)*, 4, 50–53. Retrieved from <http://iosrjournals.org/iosr-jeee/Papers/ICAET-2014/electronics/volume-2/12.pdf?id=7590>
- Karam, N., Hadj-Abdelkader, H., Deymier, C., & Ramadasan, D. (2010). Improved visual localization and navigation using proprioceptive sensors. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 4155–4160. <https://doi.org/10.1109/IROS.2010.5649978>

-
- Kim, J., & Sukkarieh, S. (2007). Real-time implementation of airborne inertial-SLAM. *Robotics and Autonomous Systems*, 55(1), 62–71. <https://doi.org/10.1016/j.robot.2006.06.006>
- Koundal, D., Vishraj, R., Gupta, S., & Singh, S. (2016). An automatic ROI extraction technique for Thyroid Ultrasound image. *2015 2nd International Conference on Recent Advances in Engineering and Computational Sciences, RAECS 2015*, (December). <https://doi.org/10.1109/RAECS.2015.7453309>
- Liu, X., Chen, Z., Chen, W., & Xing, X. (2016). Multiple optical flow sensors aiding inertial systems for UAV navigation. *2016 UKACC International Conference on Control, UKACC Control 2016*, (61403007). <https://doi.org/10.1109/CONTROL.2016.7737615>
- Liu, Y., Noguchi, N., & Ishii, K. (2014). *Development of a Low-cost IMU by using sensor fusion for attitude angle estimation. IFAC Proceedings Volumes (IFAC-PapersOnline)* (Vol. 19). IFAC. <https://doi.org/10.3182/20140824-6-ZA-1003.00610>
- Lye, Z., Nisar, H., Lai, K., & Yeap, K. (2014). Localization and feature recognition: Implementation on an indoor navigator robot. *10th France-Japan Congress, 8th Europe-Asia Congress on Mechatronics, MECATRONICS 2014*, 238–243. <https://doi.org/10.1109/MECATRONICS.2014.7018561>
- Mamilla, V. R., & Chakradhar, K. S. (2014). Micro Machining for Micro Electro Mechanical Systems (MEMS). *Procedia Materials Science*, 6(Icmcp), 1170–1177. <https://doi.org/10.1016/j.mspro.2014.07.190>
- Navarro Tarín, S. (2014). *Orientación espacial de múltiples sensores integrados con constreñimientos geométricos*. Tesis doctoral, Depto. Ingeniería Cartográfica, Geodesia y Fotogrametría, Universitat Politècnica De València, Valencia, febrero 2014.
- Nister, D., Naroditsky, O., & Bergen, J. (2004). Visual odometry. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference On, 1*, I-652-I-659 Vol.1. <https://doi.org/10.1109/CVPR.2004.1315094>
- OpenCV Org. (2018). OpenCV 3.3 - OpenCV library. Retrieved November 30, 2018, from <https://opencv.org/opencv-3-3.html>
- Ortega, F., Molina, M. A., López, E., & Palomo, E. J. (2016). Smart motion detection sensor based on video processing using self-organizing maps. *Expert Systems with Applications*, 64, 476–489. <https://doi.org/10.1016/j.eswa.2016.08.010>
- Qazizada, M. E., & Pivarčiová, E. (2016). Mobile robot controlling possibilities of inertial navigation system. *Procedia Engineering*, 149(June), 404–413. <https://doi.org/10.1016/j.proeng.2016.06.685>
- Qt io. (2018). Download Qt: Choose commercial or open source. Retrieved November 30, 2018, from <https://www.qt.io/download>
- Raspberry Pi Org. (2018a). Camera Module V2 - Raspberry Pi. Retrieved April 3, 2018, from <https://www.raspberrypi.org/products/camera-module-v2/>
- Raspberry Pi Org. (2018b). Raspberry Pi 3 Model B - Raspberry Pi. Retrieved April 3, 2018, from <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>



-
- Raspberry Pi Org. (2018). Download Raspbian for Raspberry Pi. Retrieved November 30, 2018, from <https://www.raspberrypi.org/downloads/raspbian/>
- Santana, J., Van Den Hoven, R., Van Liempd, C., Colin, M., Saillen, N., Zonta, D., ... Van Hoof, C. (2012). A 3-axis accelerometer and strain sensor system for building integrity monitoring. *Sensors and Actuators, A: Physical*, 188, 141–147. <https://doi.org/10.1016/j.sna.2011.11.017>
- Sengar, S. S., & Mukhopadhyay, S. (2017). Motion detection using block based bi-directional optical flow method. *Journal of Visual Communication and Image Representation*. Elsevier Inc. <https://doi.org/10.1016/j.jvcir.2017.08.007>
- Suprem, A., Deep, V., & Elarabi, T. (2017). Orientation and Displacement Detection for Smartphone Device Based IMUs. *IEEE Access*, 5, 987–997. <https://doi.org/10.1109/ACCESS.2016.2631000>
- Suwandi, B., Kitasuka, T., & Aritsugi, M. (2017). Low-cost IMU and GPS fusion strategy for apron vehicle positioning. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2017–Decem, 449–454. <https://doi.org/10.1109/TENCON.2017.8227906>
- Tazartes, D. (2014). An historical perspective on inertial navigation systems. *2014 International Symposium on Inertial Sensors and Systems (ISISS)*, 1–5. <https://doi.org/10.1109/ISISS.2014.6782505>
- Tessier, C., Cariou, C., Debain, C., Chausse, F., Chapuis, R., & Rousset, C. (2006). A real-time, multi-sensor architecture for fusion of delayed observations: application to vehicle localization. *2006 IEEE Intelligent Transportation Systems Conference*, 1316–1321. <https://doi.org/10.1109/ITSC.2006.1707405>
- Tomažič, S., & Škrjanc, I. (2015). Fusion of visual odometry and inertial navigation system on a smartphone. *Computers in Industry*, 74, 119–134. <https://doi.org/10.1016/j.compind.2015.05.003>
- Varesano, F. (2011). *Using arduino for tangible human computer interaction*. Università degli Studi di Torino. Thesis of master, Dept. Informatica, Università degli Studi di Torino, April 2011. Retrieved from <http://www.sciamannalucio.it/wp-content/uploads/2015/01/varesano-thesis.pdf>
- Vergara, A. (2015). *Centro Nacional de Investigación y Desarrollo Tecnológico*. tesis de maestría, Depto. Ciencias Computacionales, CENIDET, Cuernavaca, 2015.
- Vincenzo, C., Rosario, V., & Cicala, L. (2013). High Altitude UAV Navigation using IMU , GPS and Camera. *Information Fusion (FUSION)*, 2013, 647–654.
- Wang, H., Azaizia, D., Lu, C., Zhang, B., Zhao, X., & Liu, Y. (2018). Hardware in the loop based 6DoF test platform for multi-rotor UAV. *2017 4th International Conference on Systems and Informatics, ICSAI 2017, 2018–Janua(Icsai)*, 1693–1697. <https://doi.org/10.1109/ICSAI.2017.8248556>
- Wawrzyński, P., Mozaryn, J., & Klimaszewski, J. (2015). Robust estimation of walking robots velocity and tilt using proprioceptive sensors data fusion. *Robotics and Autonomous Systems*, 66, 44–54. <https://doi.org/10.1016/j.robot.2014.12.012>

-
- Woodman, O. J. (2007). An introduction to inertial navigation. *American Journal of Physics*, 77(9), 844–847. <https://doi.org/10.1119/1.3081061>
- Zaidner, G., & Shapiro, A. (2016). A novel data fusion algorithm for low-cost localisation and navigation of autonomous vineyard sprayer robots. *Biosystems Engineering*, 146, 133–148. <https://doi.org/10.1016/j.biosystemseng.2016.05.002>
- Zhang, J., Wang, W., Xie, G., & Shi, H. (2014). Camera-IMU-based underwater localization. *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, 8589–8594. <https://doi.org/10.1109/ChiCC.2014.6896442>
- Zhang, L., Xiong, Z., Lai, J., & Liu, J. (2016). Optical flow-aided navigation for UAV: A novel information fusion of integrated MEMS navigation system. *Optik*, 127(1), 447–451. <https://doi.org/10.1016/j.ijleo.2015.10.092>
- Zhang, Y., Tan, J., Zeng, Z., Liang, W., & Xia, Y. (2014). Monocular Camera and IMU Integration for Indoor Position Estimation. *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, 1198–1201.
- Zheng, Q., Li, B., & Xu, W. (2016). Unified discrete time optimal control for MEMS gyroscopes. *Proceedings of the American Control Conference, 2016–July*, 1548–1553. <https://doi.org/10.1109/ACC.2016.7525136>

1. Instalación de la biblioteca Raspicam

La instalación de la librería Raspicam se realiza sobre Raspbian, sistema operativo instalado previamente en la Raspberry Pi. Una vez iniciada la Raspberry Pi se ejecutan los siguientes pasos.

1. Descargar la librería Raspicam desde <https://github.com/cedricve/raspicam>.
2. Descomprimir el archivo zip.
3. Abrir la terminal.
4. Teclear `sudo rpi-update`, para actualizar el firmware.
5. Teclear `cd raspicamxx`, para entrar al archivo descomprimido.
6. Teclear `mkdir build`, para crear un nuevo directorio.
7. Teclear `cd build`, para abrir el directorio creado.
8. Teclear `cmake`, para preparar la compilación.
9. Teclear `make`, para compilar.
10. Teclear `sudo make install`, para instalar la librería.
11. Teclear `sudo ldconfig`, para actualizar la librería.
12. Cerrar terminal.

2. Instalación del módulo libqt5serialport5

La instalación del módulo libqt5serialport5 se realiza sobre Raspbian utilizando la terminal y su configuración se realiza en el proyecto creado en Qt Creator. Una vez iniciada la Raspberry se ejecutan los siguientes pasos:

Instalación

1. Abrir la terminal.
2. Teclear `sudo apt-get install libqt5serialport5`, para instalar la biblioteca.
3. Teclear `sudo apt-get install libqt5serialport5-dev`, para instalar los archivos de desarrollo de la biblioteca.
4. Cerrar la terminal.

Configuración

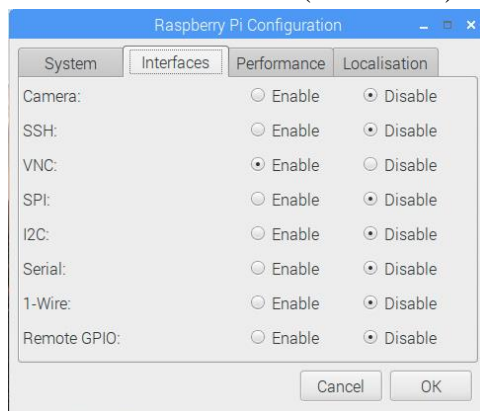
1. Abrir el proyecto Qt Creator.
2. En el archivo de configuración (.pro) en `Qt += core gui` añadir `Serialport`
3. En el archivo .h agregar `#include <QtSerialPort/QSerialPort>`
4. Añadir la información de comunicación serial.
5. Ejecutar proyecto.

3. Instalación y configuración de TightVNC

TightVNC consta de dos herramientas: VNC Viewer para controlar la Raspberry y VNC Server para el cliente, el cual viene preinstalado por defecto en Raspbian. Por lo tanto, se requiere de configurar VNC Server en la Raspberry Pi e instalar VNC Viewer en la computadora. Los pasos por seguir son los siguientes:

En el escritorio de la Raspberry Pi

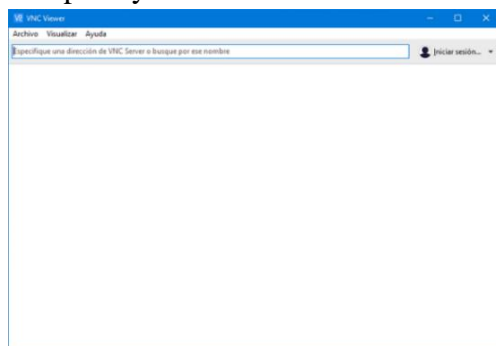
1. Seleccionar **Menu>Preferencias>Configuraciones de Raspberry Pi>Interfaces**.
2. Ir a VNC y seleccionar la casilla **Habilitar (Anexo 3.1)**.



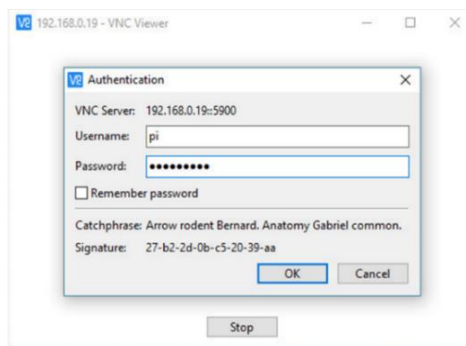
Anexo 3.1. Configuración de TightVNC en la Raspberry Pi.

En la computadora

1. Descargar VNC Viewer de acuerdo con las características de la computadora desde su web oficial <https://www.realvnc.com/es/connect/download/viewer/>
2. Instalar VNC Viewer ejecutando el archivo .exe.
3. Finalizando la instalación aparecerá la ventana principal (Anexo 3.2.a), desde el cual se accede al servidor de la Raspberry escribiendo su IP en el cuadro de búsqueda.
4. Si la IP es correcta, aparecerá la ventana de identificación (Anexo 3.2.b) en la que se deben llenar los campos de nombre de usuario y contraseña. Si los datos de la Raspberry no se han modificado el nombre de usuario es *pi* y la contraseña *raspberrypi*.
5. Si los datos son correctos, se abrirá una ventana con el entorno del escritorio de la Raspberry.



a



b

Anexo 3.2. VNC Viewer. a) Ventana principal. b) Ventana de identificación.

4. Experimentación del módulo inercial

A continuación, se muestran los resultados obtenidos en cada uno de los experimentos realizados en el módulo inercial del sistema desarrollado.

En el Anexo 4.1 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 1 (desplazamiento de -0.90 m sobre el eje x).

Anexo 4.1. Tabla de resultados de las pruebas del experimento número 1 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
1.1	x	-0.90	-0.82	0.08
1.2		-0.90	-0.84	0.06
1.3		-0.90	-1.06	0.16
1.4		-0.90	-1.03	0.13
1.5		-0.90	-0.84	0.06
1.6		-0.90	-0.87	0.03
1.7		-0.90	-0.77	0.13
1.8		-0.90	-0.82	0.08
1.9		-0.90	-0.81	0.09
1.10		-0.90	-0.98	0.08

En el Anexo 4.2 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 2 (desplazamiento de -0.60 m sobre el eje x).

Anexo 4.2. Tabla de resultados de las pruebas del experimento número 2 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
2.1	x	-0.60	-0.56	0.04
2.2		-0.60	-0.51	0.09
2.3		-0.60	-0.52	0.08
2.4		-0.60	-0.59	0.01
2.5		-0.60	-0.61	0.01
2.6		-0.60	-0.68	0.08
2.7		-0.60	-0.54	0.06
2.8		-0.60	-0.56	0.04
2.9		-0.60	-0.59	0.01
2.10		-0.60	-0.66	0.06

En el Anexo 4.3 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 3 (desplazamiento de -0.30 m sobre el eje x).

Anexo 4.3. Tabla de resultados de las pruebas del experimento número 3 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
3.1	x	-0.30	-0.30	0.00
3.2		-0.30	-0.41	0.11
3.3		-0.30	-0.20	0.10
3.4		-0.30	-0.24	0.06
3.5		-0.30	-0.22	0.08
3.6		-0.30	-0.22	0.08
3.7		-0.30	-0.32	0.02
3.8		-0.30	-0.29	0.01
3.9		-0.30	-0.28	0.02
3.10		-0.30	-0.30	0.00

En el Anexo 4.4 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 4 (desplazamiento de 0.30 m sobre el eje x).

Anexo 4.4. Tabla de resultados de las pruebas del experimento número 4 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
4.1	x	0.30	0.22	0.08
4.2		0.30	0.36	0.06
4.3		0.30	0.39	0.09
4.4		0.30	0.34	0.04
4.5		0.30	0.16	0.14
4.6		0.30	0.28	0.02
4.7		0.30	0.34	0.04
4.8		0.30	0.26	0.04
4.9		0.30	0.25	0.05
4.10		0.30	0.39	0.09

En el Anexo 4.5 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 5 (desplazamiento de 0.60 m sobre el eje x).

Anexo 4.5. Tabla de resultados de las pruebas del experimento número 5 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
5.1	x	0.60	0.65	0.05
5.2		0.60	0.45	0.15
5.3		0.60	0.80	0.20
5.4		0.60	0.59	0.01
5.5		0.60	0.57	0.03
5.6		0.60	0.59	0.01
5.7		0.60	0.57	0.03
5.8		0.60	0.61	0.01
5.9		0.60	0.77	0.17
5.10		0.60	0.64	0.04

En el Anexo 4.6 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 6 (desplazamiento de 0.90 m sobre el eje x).

Anexo 4.6. Tabla de resultados de las pruebas del experimento número 6 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
6.1	x	0.90	0.96	0.06
6.2		0.90	0.75	0.15
6.3		0.90	1.05	0.15
6.4		0.90	0.64	0.26
6.5		0.90	0.86	0.04
6.6		0.90	0.72	0.18
6.7		0.90	0.97	0.07
6.8		0.90	0.71	0.19
6.9		0.90	1.00	0.10
6.10		0.90	0.66	0.24

En el Anexo 4.7 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 7 (desplazamiento de -0.90 m sobre el eje y).

Anexo 4.7. Tabla de resultados de las pruebas del experimento número 7 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
7.1	y	-0.90	-0.90	0.00
7.2		-0.90	-0.91	0.01
7.3		-0.90	-0.92	0.02
7.4		-0.90	-0.90	0.00
7.5		-0.90	-0.92	0.02
7.6		-0.90	-0.98	0.08
7.7		-0.90	-1.07	0.17
7.8		-0.90	-0.80	0.10
7.9		-0.90	-0.77	0.13
7.10		-0.90	-0.90	0.00

En el Anexo 4.8 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 8 (desplazamiento de -0.60 m sobre el eje y).

Anexo 4.8. Tabla de resultados de las pruebas del experimento número 8 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
8.1	y	-0.60	-0.72	0.12
8.2		-0.60	-0.66	0.06
8.3		-0.60	-0.58	0.02
8.4		-0.60	-0.44	0.16
8.5		-0.60	-0.78	0.18
8.6		-0.60	-0.45	0.15
8.7		-0.60	-0.61	0.01
8.8		-0.60	-0.47	0.13
8.9		-0.60	-0.55	0.05
8.10		-0.60	-0.46	0.14

En el Anexo 4.9 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 9 (desplazamiento de -0.30 m sobre el eje y).

Anexo 4.9. Tabla de resultados de las pruebas del experimento número 9 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
9.1	y	-0.30	-0.27	0.03
9.2		-0.30	-0.37	0.07
9.3		-0.30	-0.37	0.07
9.4		-0.30	-0.26	0.04
9.5		-0.30	-0.32	0.02
9.6		-0.30	-0.35	0.05
9.7		-0.30	-0.31	0.01
9.8		-0.30	-0.34	0.04
9.9		-0.30	-0.27	0.03
9.10		-0.30	-0.29	0.01

En el Anexo 4.10 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 10 (desplazamiento de 0.30 m sobre el eje y).

Anexo 4.10. Tabla de resultados de las pruebas del experimento número 10 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
10.1	y	0.30	0.37	0.07
10.2		0.30	0.20	0.10
10.3		0.30	0.22	0.08
10.4		0.30	0.35	0.05
10.5		0.30	0.34	0.04
10.6		0.30	0.23	0.07
10.7		0.30	0.25	0.05
10.8		0.30	0.35	0.05
10.9		0.30	0.34	0.04
10.10		0.30	0.19	0.11

En el Anexo 4.11 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 11 (desplazamiento de 0.60 m sobre el eje y).

Anexo 4.11. Tabla de resultados de las pruebas del experimento número 11 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
11.1	y	0.60	0.71	0.11
11.2		0.60	0.59	0.01
11.3		0.60	0.85	0.25
11.4		0.60	0.44	0.16
11.5		0.60	0.57	0.03
11.6		0.60	0.46	0.14
11.7		0.60	0.52	0.08
11.8		0.60	0.56	0.04
11.9		0.60	0.62	0.02
11.10		0.60	0.56	0.04

En el Anexo 4.12 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 12 (desplazamiento de 0.90 m sobre el eje y).

Anexo 4.12. Tabla de resultados de las pruebas del experimento número 12 del módulo inercial.

Prueba	Eje	Longitud predefinida	Longitud estimada	Error
12.1	y	0.90	1.05	0.15
12.2		0.90	1.01	0.11
12.3		0.90	0.79	0.11
12.4		0.90	0.77	0.13
12.5		0.90	0.96	0.06
12.6		0.90	0.86	0.04
12.7		0.90	0.98	0.08
12.8		0.90	0.88	0.02
12.9		0.90	0.83	0.07
12.10		0.90	1.01	0.11

5. Experimentación del módulo visual

A continuación, se muestran los resultados obtenidos en cada uno de los experimentos realizados en el módulo de visión del sistema desarrollado.

En el Anexo 5.1 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 1 (movimientos con dirección hacia adelante y después hacia atrás con velocidad de aproximadamente 0.18 m/s).

Anexo 5.1. Tabla de resultados de las pruebas del experimento número 1 del módulo visual.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
2	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
3	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
4	reposo, adelante, reposo, atrás, reposo	reposo, atrás, reposo, atrás, reposo
5	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
6	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
7	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
8	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
9	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
10	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo

En el Anexo 5.2 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 2 (movimientos con dirección hacia la derecha (dcha.) y después hacia la izquierda (izq.) con velocidad de aproximadamente 0.18 m/s).

Anexo 5.2. Tabla de resultados de las pruebas del experimento número 2 del módulo visual.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
2	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
3	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
4	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
5	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
6	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
7	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
8	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
9	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
10	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo

En el Anexo 5.3 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 3 (movimientos con dirección hacia adelante y después hacia atrás con velocidad de aproximadamente 0.40 m/s).

Anexo 5.3. Tabla de resultados de las pruebas del experimento número 3 del módulo visual, donde las pruebas sombreadas no detectaron la dirección correcta.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
2	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
3	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
4	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
5	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
6	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, adelante, reposo
7	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
8	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
9	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
10	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo

En el Anexo 5.4 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 4 (movimientos con dirección hacia la derecha y después hacia la izquierda con velocidad de aproximadamente 0.40 m/s).

Anexo 5.4. Tabla de resultados de las pruebas del experimento número 4 del módulo visual, donde las pruebas sombreadas no detectaron la dirección correcta.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
2	reposo, dcha., reposo, izq., reposo	reposo, izq., reposo, izq., reposo
3	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
4	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
5	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
6	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
7	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
8	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
9	reposo, dcha., reposo, izq., reposo	reposo, reposo, reposo, izq., reposo
10	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo

En el Anexo 5.5 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 5 (movimientos con dirección hacia adelante y después hacia atrás con velocidad de aproximadamente 0.69 m/s).

Anexo 5.5. Tabla de resultados de las pruebas del experimento número 5 del módulo visual, donde las pruebas sombreadas no detectaron la dirección correcta.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
2	reposo, adelante, reposo, atrás, reposo	reposo, atrás, reposo, atrás, reposo
3	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
4	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
5	reposo, adelante, reposo, atrás, reposo	reposo, atrás, reposo, atrás, reposo
6	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, reposo, reposo
7	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
8	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
9	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo
10	reposo, adelante, reposo, atrás, reposo	reposo, adelante, reposo, atrás, reposo

En el Anexo 5.6 se muestran los resultados obtenidos en cada una de las diez pruebas realizadas del experimento número 6 (movimientos con dirección hacia la derecha y después hacia la izquierda con velocidad de aproximadamente 0.69 m/s).

Anexo 5.6. Tabla de resultados de las pruebas del experimento número 6 del módulo visual, donde las pruebas sombreadas no detectaron la dirección correcta.

Prueba	Dirección predefinida	Dirección detectada
1	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
2	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
3	reposo, dcha., reposo, izq., reposo	reposo, izq., reposo, izq.
4	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
5	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, reposo, reposo
6	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
7	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
8	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
9	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo
10	reposo, dcha., reposo, izq., reposo	reposo, dcha., reposo, izq., reposo