ESTIMACIÓN DE LA LOCALIZACIÓN DE UN VEHÍCULO USANDO UN SISTEMA DE VISIÓN POR COMPUTADOR.

DIEGO ALEJANDRO AGUDELO ESPAÑA

JUAN SEBASTIÁN SILVA LÓPEZ

JUAN DAVID GIL LÓPEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA, RISARALDA

OCTUBRE 2013

ESTIMACIÓN DE LA LOCALIZACIÓN DE UN VEHÍCULO USANDO UN SISTEMA DE VISIÓN POR COMPUTADOR.

DIEGO ALEJANDRO AGUDELO ESPAÑA
1.088.286.391

JUAN SEBASTIÁN SILVA LÓPEZ
1.088.285.268

JUAN DAVID GIL LÓPEZ
1.088.283.920

proyecto de grado para optar por el título de:
INGENIERO DE SISTEMAS Y COMPUTACIÓN.

Director
JOHN HAIBER OSORIO.

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA, RISARALDA

OCTUBRE 2013

Índice general

Índice de figuras

Capítulo 1

Presentación del proyecto

1.1. Planteamiento del problema

Hoy en día el tráfico en las avenidas, calles y autopistas en Colombia se ha vuelto un problema crítico, debido a la gran cantidad de vehículos que transitan, al afán de los peatones por cruzar las calles, a los embotellamientos en los centros de las grandes ciudades y al índice de accidentalidad que aún se presenta en casi la totalidad de las vías del país. Tan solo entre 2010 y 2011, según estadísticas del fondo de prevención vial[1], en el periodo de vacaciones del 1 de diciembre del 2010 al 16 de enero del 2011, se presentaron 2774 accidentes de tránsito, lo que dejó una cifra de 551 muertos y 3773 personas lesionadas, que si bien representó una disminución del 36.2 % con respecto al periodo 2009-2010, sigue reflejando la realidad sobre los problemas de tránsito y las causas de dichos problemas. Entre las causas se encuentran: exceso de velocidad (392 accidentes), cruzar sin observar (298 accidentes), desobedecer señales de tránsito (206 accidentes), embriaguez (197 accidentes), no respetar prelación (142 accidentes), no mantener distancia de seguridad (121 accidentes), invasión de carril (105 accidentes), otras causas (1313 accidentes).

Según la fuente anterior, se evidencia que gran cantidad de los accidentes de tránsito pueden atribuirse a fallas humanas frente al volante ocasionadas por distintos motivos como: estrés, ansiedad de llegar al lugar de trabajo o residencia, distracción con *smartphones* o llamadas y un sin fin de causas que pueden llevar a que un piloto pierda el control del automóvil ocasionando un desastre. A pesar de los esfuerzos del gobierno nacional por hacer

[1]FONDO DE PREVENCIÓN VIAL. *Publimotos [en línea]*. [citado en 22 de julio de 2012]. URL: `http://www.publimotos.com/nacionales/accidentes-de-transito-en-la-temporada-de-vacaciones-diciembre-y-enero-en-colombia/?id=3077`.

frente a este problema[2], no se han encontrado soluciones que puedan reducir considerablemente las muertes por accidentes de tránsito como lo evidencian las cifras anteriormente descritas.

En Colombia también es un tema muy importante el de las discapacidades que algunos ciudadanos pueden presentar y les impide desenvolverse normalmente dentro de una comunidad. Según el DANE que desarrolló un censo en 2005[3], en donde uno de los objetivos fue el de recopilar información sobre personas discapacitadas, se encontró que de 41.242.948 habitantes, 2.632.255 tiene por lo menos una limitación, dentro de las más comunes se encuentran: limitaciones visuales, auditivas y motrices. Estas limitaciones imposibilitan a estas personas de poder tener un vehículo exclusivo para ellos que los pueda transportar de forma segura de un punto a otro.

Teniendo en cuenta las causas de los accidentes de tránsito anteriormente descritas y la imposibilidad de algunas personas para conducir automóviles, por su estado de discapacidad, se ha propuesto como una solución a esta situación el desarrollo de vehículos autónomos[4], es decir, vehículos que puedan conducirse sin intervención humana.

El proyecto presentado hace parte de un proyecto más grande conocido como el Proyecto "Optimus" del Grupo de Investigación Sirius de la Universidad Tecnológica de Pereira. "Optimus" busca darle a un automóvil completa autonomía para que se desplace por ambientes rurales o urbanos sin ayuda humana. En este momento el proyecto "Optimus" no cuenta con un mecanismo de localización válido que le permita conocer su ubicación y orientación en un marco de referencia determinado con suficiente confiabilidad[5], debido a que dispone únicamente de la estimación de la localización provista por un *GPS*y una unidad de medida inercial, la cual no es lo suficientemente adecuada para un vehículo autónomo.

[2]FONDO DE PREVENCIÓN VIAL. *Fonprevial [en línea].* [citado en 22 de julio de 2012]. URL: `http://www.fonprevial.org.co/quienes_somos`.

[3]DANE. *Estadísticas [en línea].* [citado en 22 de julio de 2012]. URL: `http://www.discapacidadcolombia.com/Estadisticas.htm`.

[4]MUSTAFA CONKA ALEX FORREST. «Autonomous Cars and Society». En: *Department of Social Science and Policy Studie* (May 1, 2007), pág. 29.

[5]CRISTIAN CAMILO PERILLA RESTREPO. «*Generación de un mapa de entorno tridimensional a partir de la integración entre un escáner láser y una unidad de medida inercial.* Trabajo de grado (Ingeniero Electrónico)». Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2011.

1.2. Justificación

Los vehículos autónomos han ganado un alto protagonismo como tema de investigación interdisciplinario, debido al alto impacto que conlleva su implantación en los sistemas de tráfico de las ciudades modernas y las grandes ventajas que representaría para algunas poblaciones discapacitadas. Entre las consecuencias más notables de su implantación se encuentran: mejora de la seguridad de las personas que usan un vehículo, flujo de tráfico más eficiente, ahorro en el tiempo de las personas, uso más eficiente del combustible, etc[6]. Este campo ha estado capturando la atención de los investigadores alrededor del mundo por los retos y desafíos que representa la construcción de un vehículo autónomo.

Un vehículo autónomo se puede modelar como sistema de la siguiente forma[7]:

- El subsistema de percepción se encarga de representar el ambiente a través de la medición de variables exógenas y endógenas por medio de sensores. Esta etapa también tiene la función de minimizar el error en el modelo construido y presentar la información de forma adecuada, de tal forma que sea un insumo valioso para el subsistema de Control y Planificación. Es importante mencionar que el problema de la localización hace parte del subsistema de percepción.
- En control y planificación se toma el modelo del ambiente obtenido en el subsistema de percepción con el fin de tomar las decisiones adecuadas sobre el ambiente donde se encuentra el vehículo autónomo. Tales decisiones se relacionan con: cálculo y corrección de la trayectoria, evasión de obstáculos, identificación de elementos en el sistema de tráfico (e.g. semáforos, señales de tránsito, peatones, otros vehículos, líneas de la carretera, etc.) y control sobre las variables de velocidad y aceleración del vehículo autónomo.
- La actuación es el subsistema encargado de ejecutar las decisiones tomadas por Control y Planificación que permiten que el vehículo autónomo efectivamente se conduzca sin intervención humana. Este subsistema se relaciona inherentemente con la parte mecánica del vehículo autónomo y las características propias del modelo del vehículo.

El ambiente no representa un componente del sistema autónomo, representa gran parte del estado en el que se encuentra el vehículo autónomo en un momento determinado. Este

[6] ALEX FORREST, óp. cit.

[7] CARNEGIE MELLON UNIVERSITY. «*Autonomous driving in urban environments: boss and the urban challenge.*» En: (19 de junio de 2008), pág. 2.

ambiente es modificado continuamente por las acciones que ejecuta el vehículo autónomo.

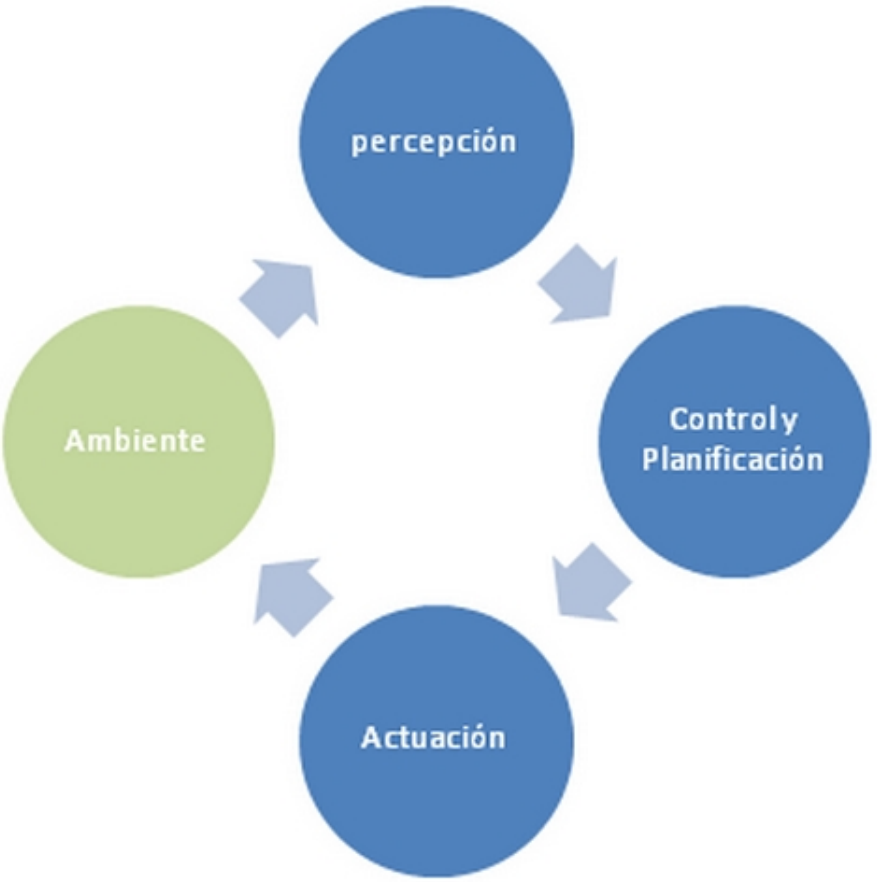


Figura 1.1: Modelo como sistema para un vehículo autónomo [8].

En trabajos anteriores[9] se hace referencia a la percepción como un problema fundamental en el desarrollo de vehículos autónomos y se menciona la necesidad de tener un sistema de percepción robusto antes de desarrollar los sistemas de control y planificación.

Es importante mencionar que el módulo de la percepción no solamente se ocupa de hacer el levantamiento del terreno o del ambiente que rodea el vehículo, sino que también debe proveer información relacionada con el movimiento del vehículo, como la posición, la velocidad y en general la localización del mismo[10].

Para lograr la autonomía de un vehículo, este debe tener la capacidad de conocer su posición dentro de su ambiente. Un vehículo que no pueda localizarse a sí mismo corre el

[8] Fuente:Autores

[9] PERILLA RESTREPO, óp. cit.

[10] J IBAÑEZ GUZMÁN. JIUN KEAT ONG. «*Perception Management for the Guidance of Unmanned vehicles.Cybernetics and Intelligent Systems*». En: (Singapore 2004).

riesgo de chocar contra obstáculos, elegir las rutas inadecuadas o no poder evitar las áreas peligrosas. Estas son algunas de las razones por las cuales el problema de la localización es importante[11].

La localización de un vehículo es un problema que en principio puede ser resuelto desde dos perspectivas. La primera consiste en usar un sistema de posicionamiento global (*GPS*) para georeferenciar el vehículo y obtener la ubicación de este en la tierra, esta solución se caracteriza porque tiene un error inherente de varios metros (de 5 a 10 metros)[12], además las mediciones proporcionadas por el sistema *GPS* se ven afectadas por diversas variables y por el tipo de entorno que rodean el vehículo (e.g. urbano, rural, presencia de edificaciones altas, túneles etc.). Este primer enfoque tiene la característica de que el error no es acumulativo. Por otro lado, el otro enfoque usado para la localización de vehículos se conoce como *Dead-reckoning* (navegación por estima), el cual consiste en estimar la posición actual tomando como referencia posiciones determinadas previamente, teniendo en cuenta estimaciones de velocidad, orientación y posición en un intervalo de tiempo determinado. Este último enfoque tiene la característica de manejar un error acumulativo, lo cual se torna bastante inmanejable a largo plazo[13] .

Teniendo en cuenta lo anterior, es evidente que la solución al problema de la localización no es trivial y amerita estudiar cada una de las ventajas y desventajas de cada enfoque de solución, buscando usarlos conjuntamente de tal manera que la estrategia final de solución presente mejores resultados que las estrategias previas por separado[14].

El proyecto "Optimus" cuenta con un *GPS* integrado a una unidad de medida inercial para conocer la posición del vehículo. Por otro lado, no se dispone de un sistema alterno de localización relativa *(Dead-reckoning)* que permita corregir las mediciones entregadas por el *GPS*, más específicamente se carece de un sistema de odometría.

El beneficio que se obtendrá desde la perspectiva académica e investigativa es la contribución a la construcción de un sistema de localización robusto, exacto y preciso para el proyecto Optimus. Además, se obtendrá la base de conocimiento que se generará en el tema, las implementaciones realizadas y la documentación necesaria para recrear el proyecto con fines varios, no sólo vehículos autónomos, también como un sistema de localización

[11]GUTMANN JENS Et. Al. «*An Experimental Comparison of Localization Methods*». En: (Germany), pág. 1.

[12]TOOMO INOUE Et. Al. «*Use of human geographic recognition to reduce GPS error in mobile mapmaking learning. Faculty of Science and Technology*». En: *Faculty of Science and Technology, Keio University.* (Japan 2006), pág. 2.

[13]JONATHAN MICHAEL WEBSTER. «*A Localization Solution for an Autonomous Vehicle in an Urban Environment*». Tesis de lic. Virginia Polytechnic Institute y State University, December 3, 2007.

[14]Ibíd.

para apoyar el levantamiento de un entorno 3D[15], entre otros. En la parte económica, se establecería una forma de construir un sistema de odometría con elementos de bajo costo (cámaras), esto implica un aporte muy valioso y una de las principales justificaciones del actual proyecto si se compara con proyectos realizados anteriormente[16]. Es conocido que tales proyectos cuentan con gran cantidad y variedad de sensores, algunos de ellos con un costo bastante elevado[17].

[15]PETER K. ALLEN Et. Al. «*New Methods for Digital Modeling of Historic Sites*». En: *Columbia University.* (November 2012).
[16]CARNEGIE MELLON UNIVERSITY, óp. cit.
[17]Ibíd.

1.3. Objetivos general y específicos

1.3.1. Objetivo general:

- Desarrollar e implementar un sistema que permita estimar la localización de un vehículo usando técnicas de odometría visual.

1.3.2. Objetivos específicos:

1. Elaborar un informe acerca del estado del arte en lo referente a sistemas de localización para *UGV's* (*Unmanned Ground Vehicles*).
2. Elaborar un informe acerca del estado del arte relacionado con técnicas de visión por computador para estimación de variables odométricas de un vehículo.
3. Realizar un estudio comparativo de los distintos enfoques para resolver el problema de la localización de un vehículo usando técnicas de visión por computador.
4. Diseñar una estrategia de visión artificial para lograr por medio de cámaras una estimación de la posición y orientación de un vehículo.
5. Implementar dicha estrategia de visión artificial para lograr por medio de cámaras una estimación de la posición y orientación de un vehículo.
6. Realizar una evaluación del funcionamiento de la estrategia implementada.

1.4. Marco Referencial

1.4.1. Marco de antecedentes

El término odometría visual fue acuñado en el 2004 en un artículo publicado por *Nister* et al.[18], se usó ese término por la relación que tiene con la odometría de las ruedas (*wheel odometry*), esta última forma de hacer odometría busca estimar la localización del vehículo/robot integrando el número de revoluciones de dichas llantas en el tiempo. Por otro lado, la odometría visual busca hacer la misma estimación de la localización, pero examinando los cambios inducidos en las imágenes de las cámaras debido el movimiento del vehículo.

Los orígenes de la odometría visual se remontan a la década de los 80, por lo tanto ya se completan tres décadas de trabajos en esta área. Es importante mencionar que las dos primeras décadas han servido para la generación de trabajos teóricos y la creación de prototipos y aproximaciones a sistemas funcionales, pero sólo durante la última década se han desarrollado implementaciones que funcionan en tiempo real de manera adecuada[19].

El problema particular de estimar el movimiento de un vehículo (*ego-motion*) a partir de imágenes únicamente, comenzó a inicios de 1980 y fue descrito por Moravec. Matthies y Shafer (1987) basados en el trabajo de Moravec, usaron un sistema binocular logrando un error relativo del 2 % en un recorrido de 6 metros para un *rover* planetario. Olson (2000) extendió dichos trabajos agregando un sensor de orientación. Este último demostró que el error en la localización crece con la distancia recorrida y agregando dicho sensor logró reducir el error de la posición relativa al 1.2 % en un recorrido de 20 metros. Cheng, Milella y Siegwart (2006) lograron un desempeño superior mejorando los algoritmos de detección de bordes. Estos últimos usaron un escáner láser para refinar la posición relativa.

Se debe recalcar que los últimos desarrollos sobre odometría visual se han obtenido gracias al desarrollo y construcción de robots de exploración planetaria. Estos robots, específicamente del programa de exploración de Marte desarrollado por la NASA[20], han contribuido a este desarrollo con la búsqueda de un mecanismo de estimación del movimiento en 6 grados de libertad, en condiciones de terrenos inestables y de deslizamiento de ruedas, situación en la cual la odometría tradicional falla[21].

[18] NISTER, D. Et. Al. «*Visual Odometry*». En: *Sarnoff Corporation.* (Princeton USA. 2004.).

[19] D. SCARAMUZZA y F. FRAUNDORFER. «*Visual Odometry [Tutorial]*». En: *Robotics Automation Magazine, IEEE* 18.4 (Dec.).

[20] M. MAIMONE, Y. CHENG y L. MATTHIES. «*Two years of visual odometry on the mars exploration rovers: Field reports*». En: *J. Field Robot, vol 24 no. 3* (2007.), págs. 169-186.

[21] SCARAMUZZA y FRAUNDORFER, óp. cit.

Algunos de los últimos avances en la odometría visual y en sistemas de localización provienen del surgimiento de la competencia *DARPA Grand Challenge*. En el año 2003, la Agencia de Investigación de Proyectos Avanzados de Defensa *DARPA* anunció el primer *Grand Challenge*. El objetivo era construir un vehículo autónomo que se condujera a través del desierto a altas velocidades. Lamentablemente, se reconoció que aunque existían programas de investigación en tales temas, no existía la tecnología adecuada. En el año 2004 se realizó el primer *Grand Challenge* y a pesar que ningún vehículo alcanzó la meta, esto significó un impulso para aumentar la investigación alrededor de los vehículos autónomos. El año siguiente cinco vehículos terminaron la competencia, uno de ellos fue Stanley, el cual fue desarrollado por la Universidad de Stanford y fue el automóvil que alcanzó la meta en la primera posición. En el 2006, *DARPA* planteó un desafío similar, esta vez en un ambiente urbano, esta competencia implicaba que los vehículos tuvieran la capacidad de evadir obstáculos y respetar las señales de tránsito, el ganador en esta ocasión fue Boss, desarrollado por estudiantes e investigadores de *Carnegie Mellon, General motors, Caterpillar* e *Intel*[22].

Figura 1.2: Boss, el vehículo autónomo de Carnegie Mellon [23].

Boss fue dotado con una variedad de sensores láser, cámaras, sistemas de navegación inercial e incluso radares que le permiten reconocer y procesar los elementos necesarios para la navegación a través del tráfico de una ciudad. La solución de localización se logró por medio de una estrategia híbrida donde se combinaban las mediciones hechas por el *GPS* con otras estimaciones de sensores comerciales y la información proveniente de un mapa del entorno pregenerado.

[22] CARNEGIE MELLON UNIVERSITY, óp. cit.
[23] (ibíd.)

En el año 2009, los estudiantes de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de pereira René Gómez y Esteban Correa plantearon como proyecto de grado la medición de variables de tráfico usando visión por computador para el proyecto del Observatorio de movilidad vial[24]. Como resultado se obtuvo la estimación de la velocidad de los vehículos que utilizaban cierta intersección vial, usando un algoritmo desarrollado con la librería OpenCV. Gracias a este montaje, el Grupo de Investigación Sirius cuenta con los recursos necesarios para realizar desarrollos de visión por computador usando cámaras.

En el año 2011, el estudiante de Ingeniería Electrónica de la Universidad Tecnológica de Pereira, Cristian Camilo Perilla, formuló como proyecto de grado la generación de un mapa de entorno tridimensional con la integración de un escáner láser y una *IMU* para el proyecto "Optimus". Como resultado, se pudo realizar la lectura del terreno en condiciones muy estrictas y limitadas, pero cumpliendo a de manera satisfactoria los términos propuestos en el proyecto de grado. Se encontraron dificultades en la localización del vehículo, debido a los errores instrumentales y a la configuración y calibración del instrumento usado para tal fin; un sensor de medida inercial.

[24]RENÉ. GOMEZ y ESTEBAN. CORREA. «*Uso de técnicas de visión por computador para la medición de variables del tráfico en el proyecto Observatorio de movilidad vial de Pereira.* Trabajo de grado (Ingeniero de Sistemas.)» Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2010.

1.4.2. Marco teórico

Visión por computador y visión de máquinas

En ciertas ocasiones suele confundirse visión por computadora con visión de máquinas, se debe aclarar que aunque abarcan conceptos y temáticas similares, la problemática abordada es completamente diferente. Mientras las técnicas de visión por computadora se encargan de la automatización de procesos para el análisis de imágenes extraídas del entorno por medio de cámaras, la visión de máquinas se encarga de usar estas técnicas de visión por computadora para ayudar a que un robot o máquina tome decisiones y desarrolle determinadas tareas.

A pesar de este hecho, para efectos prácticos el término visión por computadora (*computer vision*) y el término visión de máquinas (*machine vision*) se usarán de manera indistinguible.

Un sistema de visión está formado por los siguientes componentes[25]:

- **Fuente de radiación:** Para que algunos sistemas puedan percibir un fenómeno debe haber una fuente de radiación, la cual puede ser electromagnética, nuclear o en el caso de un sistema de visión, radiación lumínica que permita que los objetos se hagan visibles a un observador.
- **Sensor:** Un sensor es esencial para poder apreciar la radiación emitida por los objetos, esta radiación llega en forma de luz. Normalmente este sensor es una cámara que puede ser de tecnología CMOS (*Complementary Metal Oxid Semiconductor*) o CCD (*Charge Coupled Device*).
- **Unidad de procesamiento:** Todo sistema de visión debe disponer de un núcleo que se encargue de hacer todo el análisis y procesamiento de la información que llega a la cámara en forma de señales, de tal manera que puedan tomarse decisiones acerca de la escena en cuestión (e.g. caracterizaciones, reconocimiento de patrones, reconstrucción, etc.).

Existen varias ramas de la computación y de la ciencia que están íntimamente relacionadas con la visión por computador, por ejemplo: inteligencia artificial, procesamiento digital de imágenes, óptica, biología, estadística, geometría y computación gráfica, entre otras.

[25]BERND JHÄNE Et. Al. «*Computer vision and applications: A guide for students and practitioners*». En: (2000.).

La inteligencia artificial es útil para la visión por computador porque brinda herramientas muy valiosas como los algoritmos para el reconocimiento de patrones y para el aprendizaje de máquina. Estos algoritmos permiten crear sistemas de visión inteligentes que hagan inferencias y extraigan información valiosa de las escenas percibidas.

El procesamiento digital de imágenes es importante debido a que ofrece algoritmos que permiten extraer la información más valiosa de una imagen a pesar del ruido que pueda presentarse en la misma. Entre las utilidades que nos brinda el procesamiento digital de imágenes se encuentran: clasificación, extracción de características, proyección, etc.

La biología también se ha constituido como otro insumo fundamental para la visión por computador debido a que en muchas ocasiones se ha buscado representar los sistemas de visión biológicos en sistemas de visión por computador, tratando de lograr la efectividad y desempeño de los sistema de visión biológicos en los sistemas artificiales.

Otra área de estudio útil para la visión por computador es la computación gráfica, la cual se ocupa básicamente de la representación y la manipulación de imágenes de modelos abstractos tridimensionales, por otro lado, la visión efectúa un proceso en cierto modo inverso, debido a que busca obtener modelos abstractos e información a partir de imágenes. Por lo anterior se consideran la computación gráfica y la visión por computador ramas complementarias.

Las aplicaciones de la visión por computador son grandes y abren un mundo de posibilidades, existen aplicaciones en la medicina en el área de diagnósticos, haciendo procesamiento sobre imágenes extraídas del paciente. Se utiliza también para hacer reconocimiento de patrones en escenas. Una de las aplicaciones más conocidas es la de reconocimiento de rostros[26], que se puede aplicar en sistemas de seguridad biométricos.

Para terminar, un sistema de visión debe estar estructurado como un proceso que se guía por el siguiente conjunto de actividades[27]:

- **Entrada de imagen por cámara:** En esta etapa se recibe la información del ambiente por medio de un sensor, generalmente es una cámara.
- **Preprocesamiento:** Se realiza con el fin de eliminar el ruido de la imagen o para filtrar ciertas características según las necesidades que se tengan.
- **Extracción de características:** A veces el tamaño de la imagen de entrada a analizar es demasiado grande como para poder ser procesada completamente en

[26]LI. STAN Z y JAIN. ANIL K. *Handbook of face recognition*. Springer-Verlag, 2011.
[27]E.R. DAVIES. *Computer and Machine Vision: Theory, Algorithms, Practicalities*. Elsevier Science, 2012. ISBN: 9780123869913.

un tiempo determinado, por lo tanto, la imagen de entrada debe ser representada en un conjunto de pequeñas características principales. Ese conjunto de pequeñas características debe contener los aspectos más relevantes que se quieran analizar de la imagen, para que en otras etapas del proceso pueda procesarse, evitando procesar la imagen completa.

- **Segmentación:** La segmentación es un proceso por el cual también se busca reducir la complejidad del análisis sobre la imagen de entrada, y a su vez analizar una región de interés. La diferencia de esta actividad con la extracción de características, radica en que en la segunda se busca definir unas características esenciales que se quieran tener en cuenta sobre la imagen, mientras que en la primera se buscan todos los elementos que tengan esas características dentro de la imagen en un conjunto de segmentos, para analizar dichos segmentos aislados de otros.

- **Procesamiento:** Esta etapa es donde se analiza la imagen y los elementos de la misma después de haber ejecutado todas las actividades anteriores. Este análisis varía de acuerdo de la aplicación específica. Algunos de los análisis que se podrían hacer sobre una escena son: contar la cantidad de personas o de vehículos que hayan en una imagen, estimar la velocidad o posición de un objeto de acuerdo a una imagen o una secuencia de estas, analizar la forma de un objeto de análisis particular, etc.

- **Toma de decisiones:** Esta etapa se refiere a las decisiones que son tomadas por el sistema de visión por computador de acuerdo con la información extraída de la escena y que son ejecutadas por los mecanismos actuadores con el fin de que se realice alguna acción determinada, por ejemplo: mover el volante hacia la derecha o izquierda para evadir un obstáculo, en el caso de un vehículo autónomo.

Visión por computador para odometría

Como se mencionó anteriormente, la visión por computador es la disciplina que estudia los métodos para adquirir, procesar, analizar y entender imágenes, con el objeto de obtener modelos abstractos de la realidad circundante y tomar decisiones basadas en esa información[28].

La odometría consiste en el uso de información proveniente de sensores de movimiento para estimar cambios en la posición durante un tiempo determinado, hace parte del conjunto de técnicas de localización relativas, lo que significa que sus estimaciones dependen de las mediciones anteriores. Normalmente, la odometría se ha llevado a cabo con el uso de

[28] M. SONKA, V. HLAVAC y R. BOYLE. *Image Processing, Analysis, and Machine Vision.* Nelson Education Limited, 2008. ISBN: 9780495082521.

información de los sensores de movimiento sobre las ruedas para estimar el cambio en la posición del robot o vehículo autónomo. Este último enfoque tiene algunas dificultades inherentes, debido a que en muchas ocasiones depende del medio sobre el que se desplaza el vehículo[29], por esta razón se ha buscado un enfoque alternativo tratando de minimizar la dependencia sobre factores externos.

La unión de la visión por computador con la estimación odométrica se conoce como odometría visual. La odometría visual se define como el proceso de determinar la posición y la orientación de un robot analizando los cambios inducidos por el movimiento en las imágenes de una cámara[30]. Esta forma de resolver el problema de la odometría ha tenido aplicaciones importantes, como por ejemplo, la misión de robots de exploración a Marte denominada "*Mars Exploration Rover*"[31] y que consistía de dos robots; el *Spirit* y el *Opportunity*. Debido a que estos dos robots se construyeron para desplazarse sobre un entorno desconocido y bajo condiciones no previstas, se usó la odometría visual para mejorar la exactitud de la estimación de la posición.

El proceso de estimación odométrica a partir de imágenes se fundamenta en el cálculo del flujo óptico de una secuencia de imágenes, debido a que este permite conocer parámetros de *ego-motion* de la cámara que se encuentra capturando las imágenes. Teniendo en cuenta que esta cámara se encontraría instalada sobre el vehículo, el conocer los parámetros de *ego-motion* de la cámara implicaría conocer los parámetros de *ego-motion* del vehículo. En términos generales, la estimación odométrica visual se puede representar por el siguiente algoritmo[32]:

1. Adquisición de imágenes de entrada.
2. Corrección de imágenes.
3. Detección de características para estimación del flujo óptico.
4. Revisión de vectores de flujo óptico para detección y corrección de errores.
5. Estimación del movimiento de la cámara a partir del flujo óptico.

[29] B. SICILIANO y O. KHATIB. *Springer Handbook of Robotics*. Gale virtual reference library. Springer, 2008.
[30] NISTER, D. Et. Al. Óp. cit.
[31] CALIFORNIA INSTITUTE OF TECHNOLOGY. *Mars exploration Rover, [en línea]*. [citado en 22 de julio de 2012]. URL: `http://marsrovers.nasa.gov/overview`.
[32] NISTER, D. Et. Al. Óp. cit.

Localización

La localización consiste en ubicar el vehículo en un marco de referencia absoluto o relativo. Este es un problema que puede parecer sencillo, pero a medida que se piensa en todos los factores externos al vehículo que pueden afectar esta medición (e.g. condiciones climáticas, tipo de suelo, tipo de ambiente, vibraciones, etc.), se convierte en un problema de elevada complejidad.

Normalmente han existido dos enfoques para resolver el problema de la localización.[33] El primer enfoque consiste en posicionar el vehículo de manera absoluta con ayuda de *GPS* y el segundo consiste en posicionar el vehículo en un marco de referencia relativo sólo con la información de los sensores odométricos, este último enfoque también es conocido como *Dead Reckoning*. Ambos enfoques tienen sus respectivos inconvenientes, el enfoque de posicionamiento absoluto es muy sensible a las condiciones del entorno (e.g. clima, tipo de edificaciones cercanas, etc.) además de tener un error inherente considerable. Por otro lado, el *Dead Reckoning* tiene la desventaja de que los errores se van acumulando a medida que el vehículo se desplaza y después de cierto tiempo la posición estimada puede variar considerablemente de la real.

Una solución natural al problema de la localización puede ser el uso conjunto de la información provista por los dos enfoques mencionados, de esta manera se compensan las debilidades de un enfoque con las fortalezas del otro. Generalmente se usa un filtro de *Kalman* para fusionar la información global y la relativa[34], este filtro es un algoritmo que usa una serie de medidas observadas en el tiempo que contienen ruido para producir estimaciones de variables desconocidas, dichas estimaciones tienden a ser más precisas que las basadas en una única medición.

[33] WEBSTER, óp. cit.

[34] R. E. KALMAN. «*A New Approach to Linear Filtering and Prediction Problems.*» En: *Research Institute for Advanced Study, Baltimore* (USA. 1960.).

1.4.3. Marco conceptual

Incertidumbre[35]

Parámetro asociado con el resultado de una medición, que caracteriza a la dispersión de los valores que en forma razonable se le podrían atribuir a la magnitud por medir. El parámetro puede ser, por ejemplo, una desviación estándar (o un múltiplo dado de ella), o la semilongitud de un intervalo que tenga un nivel de confianza determinado.

En general, la incertidumbre de la medición comprende muchos componentes. La distribución estadística de los resultados de series de mediciones se puede usar para evaluar algunos de estos componentes, que se pueden caracterizar mediante desviaciones estándar experimentales. Los otros componentes, que también se pueden caracterizar mediante desviaciones estándar, se evalúan a partir de distribuciones de probabilidad supuestas, basadas en la experiencia o en otra información.

Se entiende que el resultado de la medición es la mejor estimación del valor de la magnitud por medir, y que todos los componentes de la incertidumbre, incluyendo los ocasionados por efectos sistemáticos, tales como los componentes asociados con correcciones y con patrones de referencia, contribuyen a la dispersión.

Medición de la incertidumbre[36]

Es necesario para presentar un resultado de una medición de forma correcta expresar la incertidumbre relacionada con dicha medición. Para hallar la incertidumbre combinada se requiere cuantificar las fuentes de la incertidumbre relacionadas con la medición, como lo son las especificaciones de exactitud, la resolución del instrumento de medida, la repetición de las lecturas en el método de medición, entre otras.

Se presentan dos tipos en la estimación de la incertidumbre:

- TIPO A: Es el método de evaluación de la incertidumbre por medio del análisis estadístico de una serie de observaciones o mediciones y se estima con la dispersión de los datos individuales, tal dispersión es mejor conocida como la desviación estándar.

[35]GUÍA TÉCNICA COLOMBIANA GTC 51. «*Guía para la expresión de la incertidumbre en las mediciones.*» En: *Colombia* (2000).

[36]LLAMOSA, LUIS ENRIQUE. GOMEZ, JOSÉ. RAMÍREZ, ANDRÉS FELIPE. «*Metodología para la estimación de la incertidumbre en mediciones directas.*» En: *Ciencia y técnica. año XV no. 41. Universidad tecnológica de pereira.* (Mayo del 2009.).

La incertidumbre estándar se denota como la razón de la desviación estándar y la raíz cuadrada del total de observaciones o mediciones.

- TIPO B[37]: Las fuentes de incertidumbre tipo B son tomadas de datos externos como lo son:

  - Certificados de calibración.
  - Manuales del instrumento de medición.
  - Valores de mediciones anteriores.
  - Conocimiento previo del funcionamiento del sistema de medición.

Error de medida[38]

Es común en la literatura científica o técnica usar el término de error en la medida para denotar la diferencia entre el resultado de una medida y el valor considerado "correcto" de la misma, en otras ocasiones para denotar la imperfección del instrumento o método de medida empleado. El error de medida se define como la diferencia entre un valor medido de una magnitud y un valor de referencia (convencional o verdadero), no confundir con incertidumbre, referente a la dispersión del mensurando.

El error de medida posee dos componentes, el error sistemático y el error aleatorio de medida.

Supóngase que se ha medido con una incertidumbre despreciable cierta distancia, la cual se tomará como valor de referencia y es igual a 317,518 metros, al tomar 200 mediciones de la misma distancia con otro instrumento, se obtiene un promedio de 317,516, siendo el error en la medida de -0,002 m. Este valor es denominado error sistemático de medida, y se define como el error que en mediciones repetidas permanece constante o es predecible.

Tomando una medición individual cualquiera con respecto al promedio (317,515), el error es de -0,001. Luego, si se tomara otra medición (317,514) el error es de -0,002. Este error se denomina error aleatorio de medida, definido como la componente del error de medida que en mediciones repetidas varía de manera impredecible.

Por lo tanto, el error de medida para el resultado de una medición individual es igual a la suma del error sistemático y el error aleatorio.

[37]W. SCHMIDT, and R. LAZOS. «*Guía para estimar la incertidumbre en la medición. Centro Nacional de Metrología.*» En: (México, Mayo 2000.).

[38]RUIZ A, MIGUEL Et. Al. «*Error, incertidumbre, precisión y exactitud, términos asociados ala calidad espacial del dato geográfico.*» En: *Departamento de Ingeniería Cartográfica, Geodésica y Fotogrametría. Universidad de Jaén.* (Febrero 2010.).

Odometría[39]

Odometría viene del griego *hodos* que significa viaje y *metron* que significa medición. La odometría se ocupa de estimar el cambio en la posición y orientación en el tiempo de un robot o vehículo usando la información proporcionada por sensores de movimiento. En particular, para este proyecto la información que se usará para hacer dicha estimación será proporcionada por una o varias cámaras.

Es importante mencionar que la odometría es sensible a múltiples tipos de errores y tiene una característica que sugiere que no debe utilizarse como único método de localización, pues el error generado por la inexactitud de las medidas es acumulativo, por lo tanto, después de cierta cantidad de tiempo sin realizar ningún tipo de corrección, la estimación del movimiento de la odometría puede variar considerablemente del movimiento real. Por esta última razón la odometría se ha utilizado como un método complementario para corregir otras estimaciones de posición como las brindadas por el *GPS*.

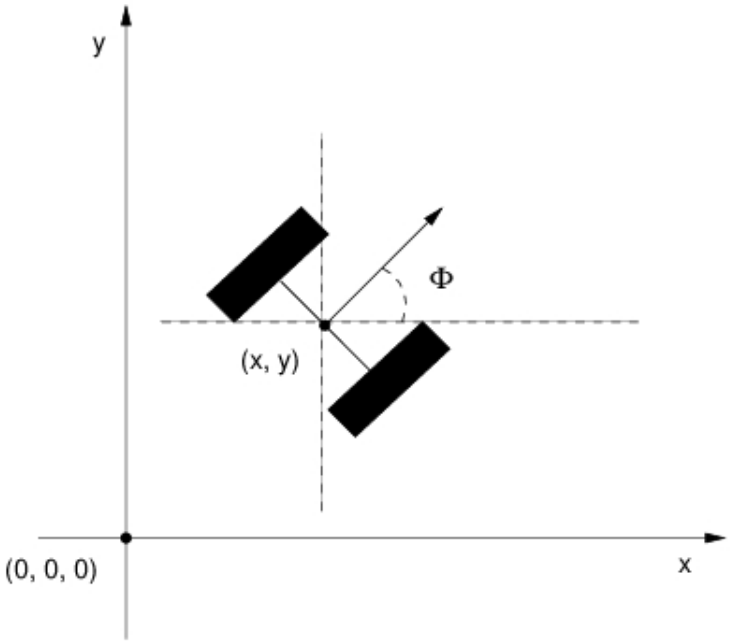


Figura 1.3: Parámetros de Odometría [40].

La imagen 1.3 es un simple ejemplo de cómo puede representarse la estimación de la posición de un vehículo autónomo[41]. Para este caso se tiene que el objeto está sobre un

[39]ODOMETRIA VISUAL. *[en línea]*. [citado en 22 de julio de 2012]. URL: `http://simreal.com/content/Odometry`.

[40] (PARAMETROS DE ODOMETRÍA. *[en línea]*. [citado en 22 de julio de 2012]. URL: `http://optimus.meleeisland.net/images/parametros_odometria.gif`)

[41]Ibíd.

plano cartesiano en 2 dimensiones, donde su posición inicial esta dada por una tripleta de la forma $(0, 0, 0)$ donde el primer elemento representa la coordenada x, el segundo elemento corresponde a la coordenada en y y el tercer elemento indica el ángulo Φ que representa la dirección que el vehículo o robot está siguiendo (orientación).

Existen diferentes dispositivos para hacer odometría, uno de los más comunes y más conocidos son los odómetros de ruedas (*wheel odometer*), capaces de medir las revoluciones de las llantas del vehículo, lo que le permite tener una estimación de cuánto se ha movido el mismo. Este odómetro a pesar de que ha sido útil en muchos aplicativos, posee una debilidad inherente que tiene que ver con que su exactitud se ve afectada tremendamente por situaciones como deslizamiento de ruedas, terrenos inestables y cambios de velocidad súbitos, por lo tanto, se han buscado otros enfoques para realizar odometría como el que utiliza visión por computador.

GPS[42]

Un *GPS*, Sistema de Posicionamiento Global (*Global Position System*), integrado en algún dispositivo, calcula la posición de dicho dispositivo tomando el tiempo de las señales enviadas por un conjunto de satélites que encuentran orbitando alrededor del planeta. Los satélites envían continuamente mensajes que los dispositivos receptor *GPS* reciben, estos mensajes contienen datos sobre el tiempo en el cual el mensaje fue transmitido y la posición del satélite al momento de transmitir el mensaje.

El dispositivo receptor *GPS* recibe los mensajes que los satélites están enviando constantemente, calcula cuánto fue el tiempo que tardó en llegar el mensaje y luego calcula la distancia del dispositivo a cada satélite. Posteriormente, con la información de las posiciones de los satélites, las distancias del dispositivo hacia ellos y con la ayuda de la trilateración, el dispositivo puede computar la posición aproximada en donde se encuentra. La posición se puede desplegar en el formato de latitud o longitud, o en un mapa digital que referencie la ubicación del dispositivo.

La trilateración es un método matemático usado para determinar la posición de un objeto en un espacio, es similar a la triangulación, pero a diferencia de usar ángulos y lados conocidos, este método usa las posiciones conocidas de unos puntos de referencia y la distancia de los puntos de referencia al punto de interés.

Para conocer la posición exacta de un punto en un plano bidimensional, por lo general,

[42]DAVID. TABORDA ALVAREZ. «*levantamiento de mapas de entorno por medio de sensores láser.* Trabajo de grado (Ingeniero de Sistemas.)» Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2009.

se requiere conocer las distancias desde el punto de interés a 3 puntos de referencia. En tres dimensiones, si se toma solo la distancia desde el punto de interés a uno de los tres puntos de referencia, no se puede ubicar de manera exacta el punto de interés, sólo se puede saber que dicho punto se encuentra en la superficie de una esfera con centro en el punto de referencia y con radio igual a la distancia entre el punto de referencia y el punto de interés, así que es necesario tener en cuenta las distancias a los otros puntos de referencia. El objetivo de la trilateración es encontrar la intersección de tres esferas (una esfera por punto de referencia), para esto se plantea un sistema de tres ecuaciones con tres incógnitas, la solución a este sistema representa el punto (x, y, z) que corresponde a la posición que se desea estimar.

El siguiente es el sistema de ecuaciones a resolver para hallar la posición de un punto en un espacio de 3 dimensiones[43]:

$$\begin{aligned} r_1^2 &= x^2 + y^2 + z^2 \\ r_2^2 &= (x-d)^2 + y^2 + z^2 \\ r_3^2 &= (x-i)^2 + (y-j)^2 + z^2 \end{aligned}$$

Medición

El proyecto consiste en un sistema de localización para un vehículo autónomo, en este sentido se puede concluir que el diseño del sistema de localización representa en sí mismo un instrumento de medición de la posición y orientación en un momento determinado. Se hace pertinente aclarar algunos conceptos

Definiciones de medición:

- Reglas para asignar números a objetos, de tal manera que puedan representarse sus atributos por medio de cantidades[44].
- La medición consiste en reglas para añadir símbolos a objetos, permite representar cantidades de algún atributo numéricamente o definir si un objeto cae o no en cierta categorización con respecto a un atributo[45].

[43] MAURICIO. GENDE. «*Trilateración.*» En: *facultad de ciencias astronómicas y geofísicas, La plata, Argentina.* (2008).
[44] J.C. NUNNALLY e I.H. BERNSTEIN. *Psychometric theory.* McGraw-Hill series in psychology no. 972. McGraw-Hill, 1994.
[45] Ibíd.

En las dos definiciones dadas se habla de reglas, estas reglas son importantes para tener en cuenta, ya que hacen parte de todo el proceso de medición y regulan cosas como la actitud o disposición del observador y cada una de las cuestiones específicas que deben de tenerse en cuenta para medir algo dependiendo del objeto de estudio. No para todos los objetos de estudio existen las mismas reglas de medición, en algunos casos pueden ser más estrictas o más laxas, por ejemplo, si se pretende medir el nivel de combustible que un avión necesita para ir de un lugar a otro, se requiere ser rígidos y estrictos con el método, por otro lado, si se quiere medir la estatura de alguien, se permite mayor laxitud y se pueden pasar por alto muchos detalles que pueden llevar a errores y que están inmersos tanto en el instrumento de medición como en el sujeto que hace la medición.

El ser humano está sujeto a cometer errores que pueden hacer que el proceso de medición se vea afectado por los mismos instrumentos que él diseña para realizar este propósito, por eso es importante tener en cuenta ciertos conceptos como incertidumbre o error que permiten tener una apreciación de que tan acertada es la medida.

La medición es un proceso que puede clasificarse en varias categorías, dependiendo de las formas de medir, la naturaleza de los instrumentos de medición, la naturaleza del fenómeno, y hasta la influencia del observador mismo.

Medición directa[46]

Las mediciones directas son las más comunes, son las mediciones donde se usa un instrumento de medición que está diseñado para recoger datos sobre un atributo específico de un objeto o fenómeno, por ejemplo, cuando se usa un metro para medir la longitud de un recorte de tela, la relación entre lo que quiere medir el instrumento de medición, y lo que se quiere medir es de la misma naturaleza, distancia en el metro y en el recorte de tela.

Mediciónes reproducibles[47]

Son un conjunto de mediciones sobre un mismo atributo de un mismo objeto que, al realizar comparaciones entre dichas mediciones, arrojan los mismos resultados o con un error muy pequeño, un ejemplo de esto es cuando se mide la longitud de una mesa circular.

[46]LLAMOSA, LUIS ENRIQUE. GOMEZ, JOSÉ. RAMÍREZ, ANDRÉS FELIPE. Óp. cit.

[47]T.G. BECKWITH, R.D. MARAGONI y J.H. LIENHARD. *Mechanical Measurements*. Pearson Prentice Hall, 2007.

Mediciones estadísticas[48]

Mediciones sobre atributo de un mismo objeto que al realizar comparaciones entre dichas mediciones, arrojan resultados diferentes, por ejemplo, cuando se quiere medir el radio de una haciendo uso de cámaras estrella distante, la cual crece y no mantiene siempre un tamaño constante.

Mediciones Indirectas[49]

A veces es complicado realizar mediciones directas con plena seguridad, debido a la naturaleza del objeto, el cual puede ser muy grande o muy pequeño, o porque no se posee un instrumento de medición adecuado para realizar dicha tarea.

Es posible hacer una medición indirecta cuando la variable que se quiere medir incide en otras variables propias del mismo fenómeno físico, lo cual se expresa en una relación matemática que permite ver la interacción entre las variables y así llegar al valor del atributo que se está buscando, por ejemplo, se puede medir la caída de tensión en un resistor midiendo la corriente que pasa por él y la resistencia del mismo, de esa manera se usa la relación de $V = R * I$ para calcular el voltaje aplicado al resistor.

Precisión y exactitud

Los términos de precisión y exactitud no son términos que representen cantidades cuantitativamente medibles, son más bien características cualitativas que pueden concluirse a partir de ciertas variables estadísticas correspondientes al fenómeno u objeto en estudio.

La precisión de una medición se define como la proximidad entre los valores medidos obtenidos en mediciones repetidas de un mismo objeto o fenómeno, bajo ciertas condiciones. Como este concepto se asocia con la repetibilidad, se relaciona a menudo con medidas de dispersión como la desviación típica o la varianza, para ser más específicos, la precisión se enuncia como una cualidad haciendo referencia a estas variables estadísticas, por lo tanto, no indica que la medida sea correcta o se asemeje a una medida que ya se conoce para ese fenómeno u objeto, simplemente, denota un carácter repetitivo en donde cada repetición de la medida no se aleja mucho de la anterior.

Por otro lado, la exactitud de una medición es también otro concepto cualitativo más no cuantitativo, por lo tanto se deduce de otras variables, más no representa una cantidad

[48] Ibíd.
[49] Ibíd.

real expresada en números. La exactitud representa qué tan cerca esta una medición de un valor real o de referencia ya medido para un fenómeno o atributo de un objeto, por ejemplo, se tiene en la mitad de una plaza una placa que indica la posición de ese punto en el planeta tierra, ahora, un *GPS* que quiera ser exacto debe dar una medición cuyo valor numérico sea muy cercano al impreso en la placa, en cada repetición que se haga de la medición.

1.5. Método o estructura de la unidad de análisis, criterios de validez y confiabilidad

La unidad de análisis será un experimento realizado bajo circunstancias controladas. Para este experimento se registrarán y procesarán los resultados obtenidos por el sistema de localización propuesto en este proyecto.

Para demostrar la validez de los datos obtenidos por el sistema de localización y que representan la estimación de la posición del vehículo en el ambiente, se comparará la posición real del vehículo con la posición estimada por el sistema de localización. Esta comparación se realizará en una serie de puntos de referencia (*landmarks*) de los cuales se conoce la posición real. La comparación entre las medidas se repetirá sistemáticamente para un conjunto de puntos en el entorno de pruebas, después de obtener los datos se establecerá una cota para la diferencia entre las medidas entregadas por el sistema de localización y las medidas esperadas, esta cota proporcionará información acerca del nivel de validez de las estimaciones hechas por el sistema de localización descrito.

Para demostrar la confiabilidad de las mediciones obtenidas por el sistema de percepción, se usará la siguiente estrategia: se repetirá la simulación en el mismo ambiente controlado una cantidad determinada de veces (estadísticamente aceptable), en cada repetición y para cada estimación, se determinará el error entre la medida del sistema de localización y la medida esperada, posteriormente se sacará el error promedio para cada estimación y la desviación estándar asociada. De esta manera se determinará el grado de confiabilidad del sistema de localización del vehículo en un ambiente determinado.

1.6. Diseño metodológico

- Los primeros tres objetivos tratan sobre la investigación del estado del arte de las técnicas de visión por computador para hacer odometría, por lo tanto, se usarán

las bases de datos con las cuales la Universidad Tecnológica de Pereira está relacionada para poder consultar trabajos de otros investigadores que han aportado al tema. Algunas bases de datos que podrían ser útiles son: *IEEE explore, ACM digital library*, entre otras. Esta investigación debe arrojar como resultado dos informes, uno relacionado con las técnicas de localización usadas para *UGV's* (*Unmmaned Ground Vehicles*) en la actualidad, el otro informe debe referenciar las principales técnicas y algoritmos de visión por computador utilizados para hacer odometría visual, además, debe realizarse un resumen comparativo entre diferentes técnicas para resolver el problema de la localización usando visión por computadora.

- Para el objetivo número 4 deben tenerse claras las ventajas y desventajas de los distintos enfoques existentes para hacer odometría visual, de tal manera que se pueda escoger una técnica de las que se investigaron en los incisos anteriores o un conjunto de éstas. La técnica que ha de escogerse debe ser la que más se acomode a las necesidades del proyecto en concreto, y que la instrumentación usada en dichos desarrollos sea similar a la que existe actualmente al alcance del Grupo de Investigación Sirius, adscrito al Programa de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira, el cuál será el facilitador de los equipos y del apoyo tecnológico y científico para el desarrollo del proyecto.

- Para la implementación del algoritmo debe escogerse un lenguaje de programación que posea librerías con las herramientas que faciliten el desarrollo del proyecto, además, se buscará que el lenguaje permita hacer un uso eficiente de los recursos de procesamiento buscando velocidad en el cómputo. Aunque para el proyecto se disponen de recursos de un buen rendimiento, la filosofía de trabajo siempre se basará en buscar una manera eficiente de hacer las cosas desde el desarrollo algorítmico.

- Para la evaluación del desempeño del sistema de visión se usará un entorno de pruebas, donde se definirá un sistema de coordenadas para definir puntos de referencia (*landmarks*), en donde se conozca su valor de posición real con respecto a ese sistema de coordenadas. Luego, se compararán los valores reales con las posiciones que el sistema de visión arroje a medida que el vehículo se mueve, la explicación de dicha evaluación se detalla en el siguiente inciso.

1.6.1. Hipótesis

¿Es posible hacer una estimación de la localización de un vehículo en un ambiente y trayectoria controlada, usando técnicas de visión por computadora

1.6.2. Tipo de investigación

Investigación cuantitativa

1.6.3. Variables

Posición del vehículo

1.6.4. Población

Todas las simulaciones posibles (teniendo en cuenta que una simulación es un recorrido por una trayectoria preestablecida donde se probará el desempeño del sistema de odometría visual) de las estimaciones de la localización que se hagan con el vehículo.

1.6.5. muestra

Un número específico de simulaciones que hagan parte de la población dentro de un ambiente controlado, que permitan realizar las mediciones de las variables.

1.6.6. Instrumentos de medición

Se dispondrá de un entorno de simulación controlado, dicho entorno estará referenciado por medio de un sistema de coordenadas, dentro de ese entorno se definirán unos puntos de referencia según el sistema de coordenadas definido, se compararán las mediciones hechas por el sistema de visión propuesto con las coordenadas establecidas para los puntos de referencia, de esta manera se podrá validar si la posición del vehículo dada por el sistema de visión diseñado tiene un error significativo o no significativo.

La información entregada por el sistema de localización basado en técnicas de visión por computador se codificará de la siguiente manera para que sea comparable con los puntos de referencia:

- La posición se tendrá en cuenta como un vector de dos dimensiones, en donde la primera componente determina el desplazamiento en metros del vehículo sobre el eje coordenado horizontal, la segunda componente de este vector determina el desplazamiento del vehículo sobre el eje coordenado vertical.

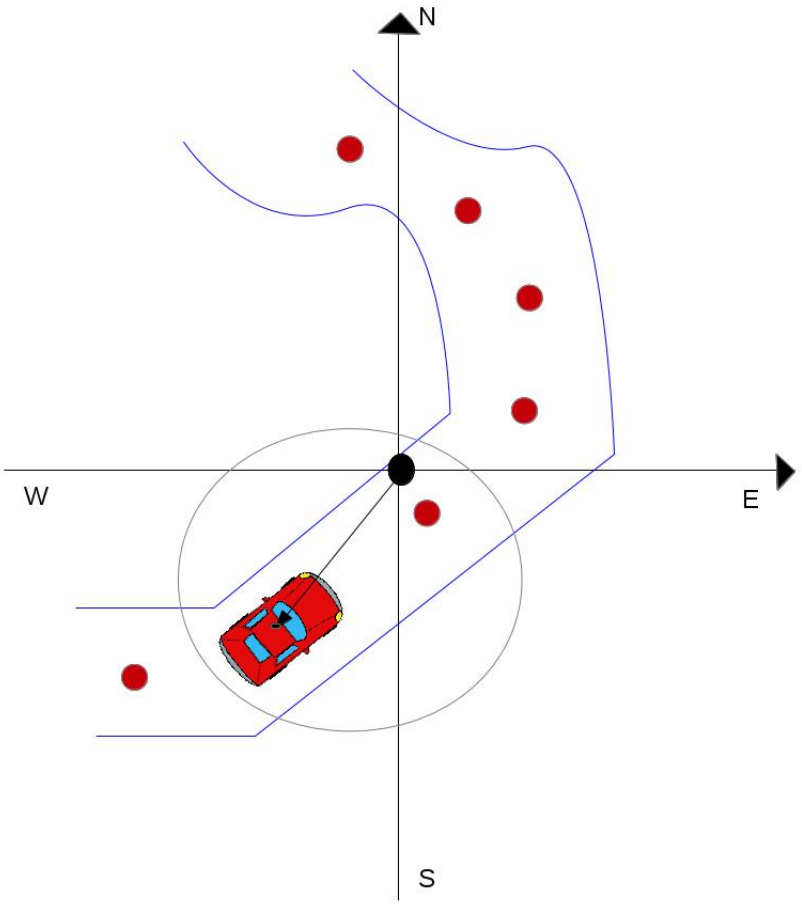


Figura 1.4: Posición de un vehículo sobre un par de ejes[50].

En la figura 1.4 puede verse que el sistema de coordenadas se asemeja mucho a un plano cartesiano, cada dirección de un eje coordenado representa un punto cardinal específico, ya sea norte, sur, oriente o occidente, el punto negro representa el origen. Cada punto rojo representa una de las marcas de referencia en donde se tomarán las mediciones de la posición entregadas por el sistema de visión por computadora. Las líneas de color azul representan la trayectoria del vehículo, o más específicamente, las líneas de la vía por donde el vehículo va a transportarse.

[50] Autores
[51] Autores

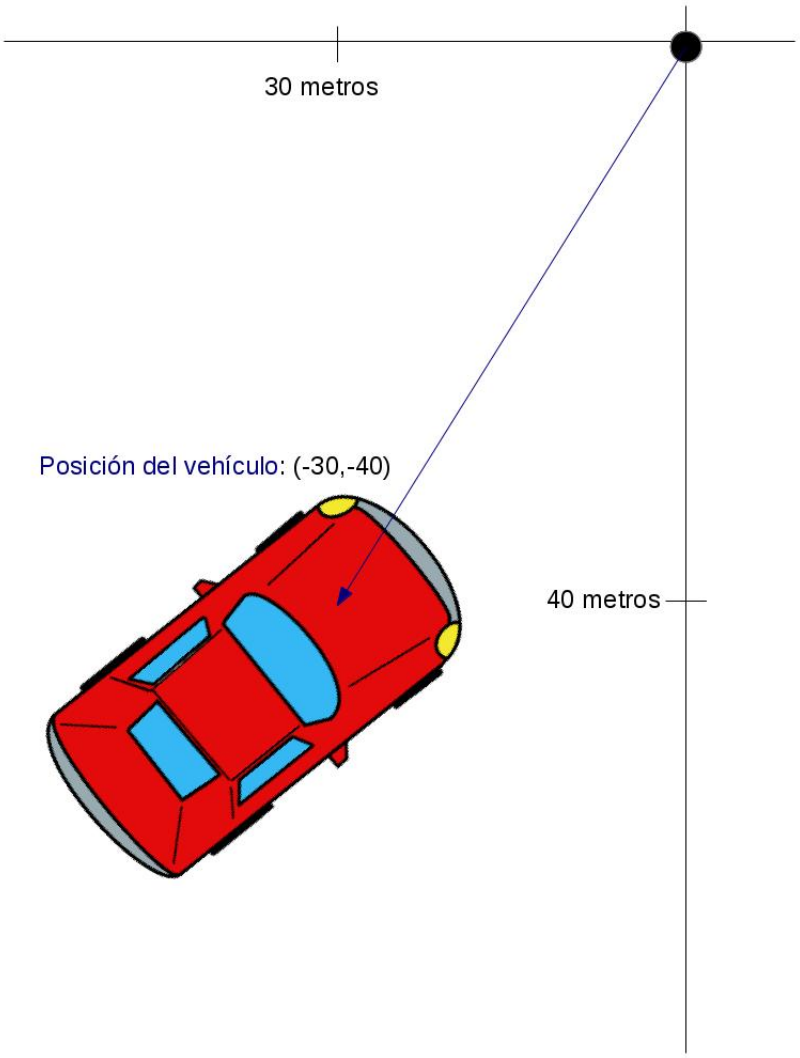


Figura 1.5: Detalle de la posición del vehículo[51].

El punto hacia donde el vector señala es el punto donde está el vehículo, dicho punto se representará como una pareja ordenada, para el ejemplo, la posición que reportara el vehículo será de $(-30, 40)$, esto significa que el vehículo se encuentra a 30 metros hacia el oeste y 40 metros hacia a el sur según este sistema de coordenadas. Este sistema será usado para reportar la posición del vehículo en cada una de las marcas de referencia, en donde se conoce la posición exacta de dicho punto y donde se espera que el vehículo reporte dicha posición con cierto margen de error.

Un detalle importante para mencionar en el experimento tiene que ver con la orientación del vehículo, este concepto se refiere a la dirección en que está alineado el automóvil, es evidente

que para definir completamente el estado de un vehículo en un sistema de referencia como el descrito, además de su posición (x, y), es necesario conocer su orientación, la cual puede ser medida como un ángulo. La orientación no será validada en el experimento de manera explícita pero sí de manera implícita, dado que si no se conoce la orientación es imposible localizar el vehículo con los métodos de localización relativos (*dead reckoning*), debido a que el vehículo o robot no conoce la dirección en la que está avanzando, por lo tanto, si se valida la localización (posición (x, y)) de manera adecuada se tiene como consecuencia que la estimación de la orientación es correcta.

Se hará una prueba piloto para comprobar que el entorno de simulación funciona de manera adecuada, de acuerdo a los resultados obtenidos en esta prueba piloto se modificarán las condiciones y circunstancias del entorno de simulación, con el objetivo de aumentar la calidad del experimento y obtener mediciones adecuadas.

Se espera que la implementación del sistema de odometría visual funcione de manera adecuada en tiempo real, en caso de que el tiempo de respuesta del sistema no sea suficientemente rápido para hacer la localización del vehículo eficazmente o que existan problemas con la adquisición de equipos para hacer el montaje físico, se procederá a validar el sistema de localización con secuencias de vídeo pregrabadas en el mismo entorno de simulación descrito.

1.6.7. Cronograma

Objetivo específico	Descripción	Dependencias	Inicio	Final
Objetivo 1	Informe del estado del arte acerca de sistemas de localización de *UGV's*	ninguna	semana 1	semana 4
Objetivo 2	Informe del estado del arte acerca de técnicas de visión por computador para la estimación de variables odométricas	ninguna	semana 1	semana 8
Objetivo 3	Estudio comparativo de los distintos enfoques para resolver el problema de la localización de un vehículo	objetivo 1 y 2	semana 8	semana 12
Objetivo 4	Diseñar una estrategia de visión artificial para estimar la posición de un vehículo	objetivo 3	semana 12	semana 18
Objetivo 5	Implementar dicha estrategia de visión artificial para estimar la posición de un vehículo	objetivo 4	semana 18	semana 30
Objetivo 6	Realización de pruebas a la estrategia implementada	objetivo 5	semana 18	semana 30

Capítulo 2

Sistemas de localización para vehículos

Desde el nacimiento del proyecto *DARPA Grand Challenge* y el éxito de implementaciones como *Boss*[1], se ha extendido el interés por la investigación de los vehículos autónomos y los sistemas de localización. La localización de *UGV's* o vehículos autónomos es uno de los puntos fundamentales, por no decir el más importante, para permitir la navegación del mismo. Para manejar un vehículo de forma autónoma, la pregunta más importante es ¿donde se encuentra este vehículo en el espacio?. Comúnmente, los vehiculos actuales vienen enquipados con sensores *GPS*, bastante usados por su bajo precio en el mercado, pero no lo suficientemente exactos y completamente confiables para solucionar el problema de la localización para un vehículo autónomo. Generalmente se toman dichas mediciones y se fusionan con mediciones entregadas por otros dispositivos como I*IMU's* o cámaras, usando filtros de *Kalman* para obtener una respuesta sólida.

Los sensores de localización en general, pueden separarse en dos categorías[2], los que entregan mediciones relativas (cámaras con Odometría visual, *IMU's*) y los que entregan mediciones absolutas (*GPS* o una brújula magnética). Como se mencionó anteriormente, enfoques como el *Dead-reckoning* son bastante populares en ambientes cerrados (donde la medición de los sensores absolutos pueda tener problemas en términos de recepción de la señal por parte de los satélites), pero su error es acumulativo. La mayoría de las implementaciones usan la fusión de un sensor *GPS* con otros sensores como cámaras, *Lidars*, ultrasonido o *IMU's*.

[1] CARNEGIE MELLON UNIVERSITY, óp. cit.

[2] V. Malyavej y P. Torteeka. «Unmanned ground vehicle localization by dead-reckoning/GPS sensor fusion». En: *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on*. 2010, págs. 508-512.

En el año 2010, *Malyavej y Torteeka*[3] plantean un algoritmo para la estimación de la posición de un vehículo autónomo usando un sensor *GPS*, una brújula electrónica y un codificador óptico para la distancia recorrida. Usan el tradicional *unscented Kalman Filter* (UKF), ya que ha demostrado tener mejores resultados que el *extended Kalman Filter* (EKF). La implementación se divide en tres modelos, el modelo cinemático, el modelo de mediciones y el modelo de ruido.

El modelo cinemático del vehículo incluye la posición y la velocidad en los ejes cartesianos. La dirección se obtiene a partir de la componente de la velocidad.

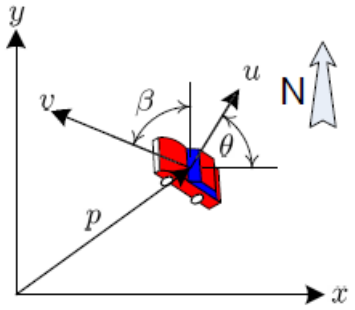


Figura 2.1: Modelo cinemático de la implementación[4].

Para el modelo de mediciones, se toman la latitud y la longitud obtenida por el *GPS*, se multiplican por algunos factores escalares para hacer la conversión en metros dentro del modelo cartesiano, $k_1 = 109369{,}3$ para la latitud y $k_2 = 106080$ para la longitud. Las anteriores constantes representan una aproximación para un grado en latitud que es equivalente a 106080 metros de oriente a occidente alrededor del Ecuador. La medida del codificador óptico arroja el acumulado de la distancia cada tiempo k, la cual es obtenida por medio de las componentes en x y en y de la velocidad, como una sumatoria de k de la raíz de los componentes de la velocidad al cuadrado.

Usando información adquirida por los sensores relativos, por ejemplo un odómetro o una *IMU's*, es posible conocer la posición relativa de un vehículo. Como se dijo anteriormente, este enfoque es conocido como *Dead-reckoning*, teniendo como ventaja la independencia de referencias externas, pero acumulando un error a medida que aumenta el recorrido. Para la implementación de *Malyavej y Torteeka*, el *Dead-reckoning* tiene como base el rodamiento de la brújula electrónica y la distancia adquirida por el codificador óptico.

La imagen 2.2 muestra con una línea continua la trayectoria recuperada de los sensores relativos, mientras la línea punteada muestra la trayectoria corregida. Estos errores pro-

[3] Ibíd.
[4] (ibíd.)

vienen tanto de la parte relativa como de la absoluta. La fuente principal de los errores vienen de las variaciones no uniformes del campo electromagnético terrestre, sin embargo, pueden ser reducidos con una calibración en los instrumentos de medida.

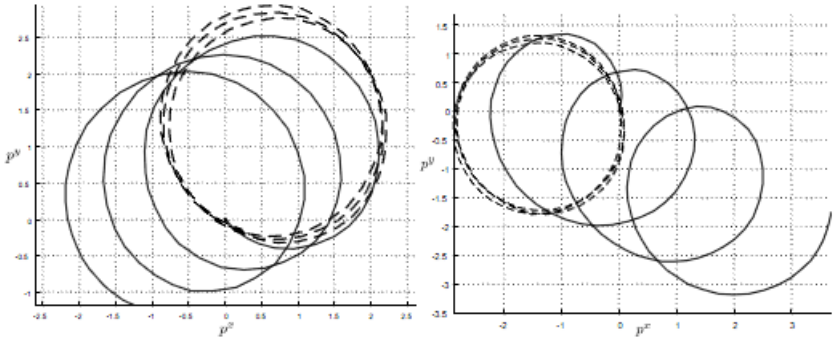


Figura 2.2: Línea punteada: trayectoria corregida. Línea continua: trayectoria sin corregir[5].

Para fusionar la información adquirida por el *GPS*, la brújula electrónica y el codificador óptico, se usa un *unscented Kalman Filter(UKF)*, pues presenta mejores resultados que el ampliamente usado *extended Kalman Filter(EKF)*. La principal diferencia entre estos dos algoritmos es el método usado para calcular la covarianza del ruido. Con el UKF los puntos cercanos al promedio llamados sigma, son escogidos de manera determinística para capturar la covarianza del ruido y junto con el promedio, propagar el ajuste para la siguiente iteración de la función dinámica no lineal. Por otro lado, el EKF usa una propagación basada en un modelo linealizado, calculado mediante aproximaciones de series de Taylor, el cual es menos robusto pero más sencillo de calcular. En términos computacionales, la única dificultad del UKF es realizar el cálculo de las raíces cuadradas, para la obtención de la posición tomando como base la componente de velocidad.

Los resultados obtenidos con el planteamiento propuesto por *Malyavej y Torteeka* son comparados con otra implementación anterior de los mismos autores, donde se calibran previamente los dispositivos con el fin de minimizar el error, mientras que en la actual implementación los errores son corregidos por el mismo modelo. Se realiza la prueba en un tramo del recorrido sin datos del *GPS* para comprobar el funcionamiento de los sensores relativos. En la imagen 2.3 se aprecia en línea verde el recorrido realizado por la implementación presentada, el cual posee la característica de calibrar automáticamente los parámetros con el fin de presentar mejores resultados, la línea rosada representa el resultado donde los instrumentos de medición son calibrados previamente. Es notoria la mejoría con la implementación calibrada automáticamente, en especial bajo la ausencia de *GPS*. Lo anterior demuestra la efectividad de la estrategia de fusión de sensores utilizando

[5] (ibíd.)

filtros de *Kalman* para dar una respuesta más precisa que utilizando sólo un enfoque, el relativo o el absoluto.

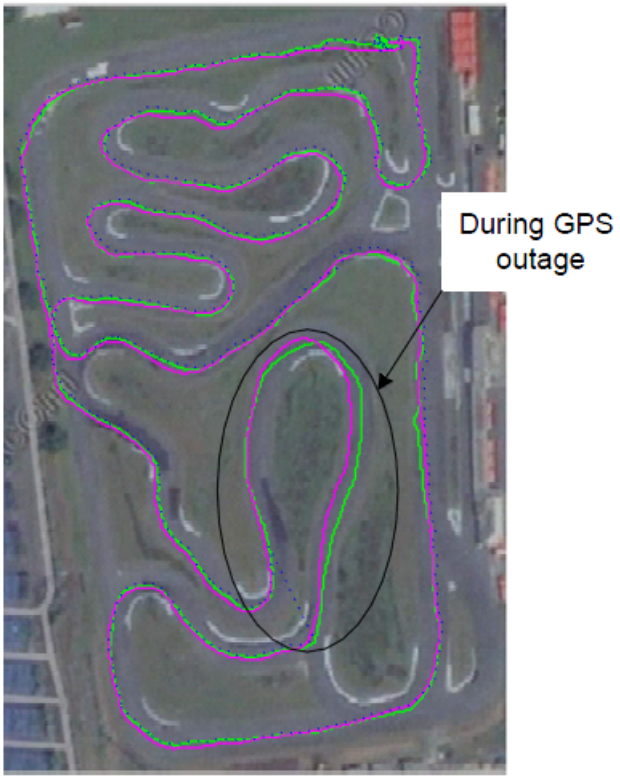


Figura 2.3: Línea verde: trayectoria calibrada automáticamente. Línea rosada: trayectoria calibrada previamente. El círculo muestra la parte del recorrido donde no se contó con los datos de *GPS*[6].

Otra propuesta para solucionar el problema de la ubicación de *UGV's* y que es ampliamente usada, es descrita por *Soon-Yong Park y Sung-In Choi*[7], el cuál usa las mediciones entregadas por un escáner láser para construir un modelo de superficie digital(*DSM*).

La novedad de esta implementación con respecto a otras que usan escáner láser es que la medición se realiza en 360° y no sólo al frente del vehículo, además de construir el modelo de superficie con refinamientos entre *frames* y realizar el emparejamiento de puntos cercanos a la elevación actual del vehículo, reduciendo el número de *Outliers*[8].

La obtención de los puntos 3D se realiza con un escáner *velodyne* de 64 lasers, de los cuales sólo son usados 48 para reducir el número de puntos, los datos se obtienen a partir

[6] (ibíd.)

[7] Soon-Yong Park y col. «Localization of an unmanned ground vehicle using 3D registration of laser range data and DSM». En: *Applications of Computer Vision (WACV), 2009 Workshop on.* 2009, págs. 1-6.

[8] Ibíd.

de paquetes Ethernet, donde cada paquete obtiene la lectura de 32 sensores. Los datos obtenidos se presentan en una tupla de 3 coordenadas: $(X, Y, Z)^T$. Cada punto incluye el número de línea y el ángulo horizontal. Como el láser arroja una medición donde encuentra una superficie para reflejarse, el formato de los puntos en 3D puede ser tratado como una nube de puntos en vez de arreglos. Cada giro del escáner se convierte en un *frame* de tamaño 48x1800, tomando 2083 puntos por línea.

Para la creación del modelo de superficie digital, se usa el algoritmo *Iterative Closest point (IPC)*, el cual reduce iterativamente el error entre dos nubes de puntos. Este algoritmo es usado generalmente para reconstruir superficies a partir de lecturas que generan nubes de puntos o similares. En el modelo son ignorados los árboles y los arbustos, pues su geometría irregular se presentan como *Outliers*. En algunas ocasiones, pedazos de edificios o baches en el piso lejanos al sensor aparecían, causando problemas al momento del emparejamiento. Con el fin de solucionar lo anterior, se asume que la lectura sólo se realiza en terreno plano. Se divide el modelo en capas, donde cada punto pertenece a como mínimo 2 capas, con el fin de mantener la posición del vehículo en la misma elevación. Mientras las diferencias entre *frames* no sean muy significativas con respecto a la altura, el vehículo conservará su posición.

La localización del vehículo se logra por medio de la transformación con respecto a la posición inicial en el modelo de superficie digital; para hallar la posición del vehículo en el *frame* n, es necesario derivar entre el *frame* actual y el anterior. Existen dos formas de realizar la derivación; entre 2 *frames* y entre un *frame* y el DSM. Cuando se presentan estas situaciones, se puede usar un simple promedio ponderado por transformación:

$$T_{n,n-1} = k_{wD} T_{wD} + k_{wp} T_{wp},$$

Donde k_{wD} y k_{wp} son ponderaciones para cada transformación. T_{wD} representa la transformación entre un *frame* y el DSM, T_{wp} es una pareja de datos entre el *frame* actual y el anterior. Lamentablemente, esta transformación no asegura el correcto cálculo de la posición, pues cualquier error significativo o *outlier* muy alejado del modelo afecta directamente el promedio ponderado. Para esto se propone realizar primero una transformación a un punto intermedio y luego a la posición actual:

$$T_{n,n-1} = T'_{wD} + T_{wp},$$

[9] (ibíd.)

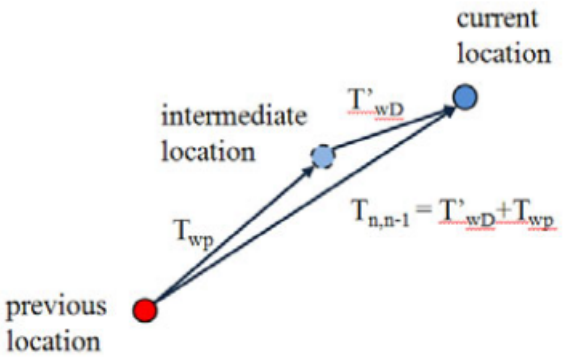


Figura 2.4: Transformación de dos pasos[9].

Los resultados obtenidos por esta implementación se ilustran bajo 4 recorridos efectuados, donde la posición hallada por las transformaciones son casi equivalentes al camino verdadero recorrido (*ground-truth*). Los errores se ilustran en la imagen 2.5

Path	Average (m)	Max(m)	Time/Frame (sec)
P1	7.2	20.6	0.36
P2	2.5	8.3	0.18
P3	6.6	23.2	0.27
P4	3.2	18.4	0.33

Figura 2.5: Errores promedio y máximo de la implementación presentada[10].

Las anteriores implementaciones son ejemplo de las estrategias más usadas para localizar un vehículo. Cada día aparecen nuevas técnicas de visión, mejores sensores y equipos de cómputo más rápidos que permiten la mejora permanente de estas estrategias.

[10] (ibíd.)

Capítulo 3

Odometría Visual

3.1. Introducción

Con anterioridad se ha mencionado la importancia de que un vehículo autónomo tenga la capacidad de localizarse con exactitud en un entorno (capítulo 1). Tradicionalmente, se han planteado dos enfoques para la solución a este problema, el enfoque absoluto y el enfoque relativo. El enfoque absoluto se caracteriza por usar medidas directas de la posición en un marco de referencia global[1]. El enfoque relativo se caracteriza por estimar la localización con respecto a un punto inicial de referencia usando las estimaciones anteriores e información acerca del cambio en la posición. La odometría hace parte del enfoque relativo; básicamente usa la información proveniente de sensores de movimiento para estimar el cambio en la posición. El tipo de sensores que se han usado comúnmente para hacer odometría son *rotary encoders*, pero han demostrado ser vulnerables a varias condiciones del entorno[2]. Por lo anterior han surgido diferentes formas de hacer odometría, las cuales se diferencian principalmente por el tipo de información que usan para hacer la estimación del movimiento (odometría láser, odometría visual, etc.). En este capítulo se pretende describir de manera detallada la odometría visual, primero se presenta una definición somera de la odometría visual, segundo una breve reseña histórica, en tercer lugar se muestran los avances investigativos hechos durante los últimos años en el área y por último se da una explicación detallada del funcionamiento de un sistema de odometría visual.

[1] WEBSTER, óp. cit.
[2] SCARAMUZZA y FRAUNDORFER, óp. cit.

3.2. Definición de la odometría visual

La odometría visual se refiere al proceso de estimar el movimiento de un agente (robot, vehículo, persona, etc.) a partir de secuencias de imágenes capturadas por una o varias cámaras ubicadas sobre dicho agente[3]. Este tipo de odometría funciona estimando de manera incremental la posición del agente al examinar los cambios en las imágenes inducidos por el movimiento del mismo. La odometría visual ha demostrado ser robusta incluso cuando el agente se mueve por terrenos inestables, circunstancia en la cual normalmente fallaba la odometría tradicional, este hecho sugiere que la odometría visual puede ser un buen complemento para otros sistemas de navegación (*GPS*, *IMU's*, etc.). La odometría en general no debe ser usada como único método de localización debido a que su estimación es relativa(*Dead Reckoning*), lo que significa que el error de las estimaciones se va acumulando durante el tiempo, de no tener una medición absoluta como referencia, se tendrá una estimación que podría variar considerablemente de la localización real. Por lo general, un sistema de odometría visual en ejecución tiene las siguientes etapas:

- **Adquisición y Preprocesamiento de las imágenes**: El sistema de odometría visual debe obtener las imágenes de un sensor (cámara monocular, cámara *stereo*, cámara omnidireccional, etc.). Las imágenes deben ser procesadas y corregidas de tal manera que las etapas posteriores puedan extraer la información de la manera más transparente posible.
- **Detección y Extracción de Características**: En esta etapa se ubican y definen unos puntos de interés para cada *frame*. Estos puntos de interés deben tener la característica de ser reconocibles en *frames* posteriores para que puedan brindar información acerca del movimiento.
- **Rastreo, emparejamiento de Características**: Esta etapa está relacionada con el establecimiento de correspondencias entre el mismo punto de interés en diferentes *frames*.
- **Estimación del movimiento**: A partir de las correspondencias establecidas en la etapa anterior se hace una estimación geométrica del movimiento.
- **Optimización y eliminación de *outliers***: Esta etapa tiene como objetivo mejorar las estimaciones del movimiento al intentar reducir el impacto que tiene el ruido sobre dichas estimaciones con el uso de técnicas más robustas (*bundle adjustment*, *RANSAC*, etc.).

[3]Ibíd.

3.3. Breve reseña histórica

La odometría visual es un campo de investigación que tiene aproximadamente 30 años de desarrollo hasta la fecha. Durante las dos primeras décadas se han hecho grandes contribuciones desde el punto de vista teórico y se han construido varios prototipos de sistemas de odometría visual, pero durante la última década se ha evidenciado un gran avance en el área, el cual ha permitido la creación de sistemas robustos de odometría visual en tiempo real que han sido implantados de manera exitosa.

El problema de estimar los parámetros de movimiento de un vehículo usando únicamente información visual fue trabajado por Moravec[4] en la década de los 80, su trabajo es de gran importancia para la odometría visual debido a que algunos de los conceptos que ideó relacionados con los bloques lógicos de un sistema de odometría visual, son usados en la actualidad[5], además, su trabajo dejó como resultado un famoso detector de esquinas conocido como el detector de esquinas de Moravec y que sirvió para el desarrollo de otros detectores de esquinas como el de Harris[6]. El trabajo de Moravec buscaba dotar a un robot planetario de un sistema de odometría visual, usando una *slider stereo camera* que se deslizaba por un riel instalado sobre el robot, el cual tenía un esquema de movimiento de avanzar y parar, en cada parada la cámara tomaba nueve fotos en distintas posiciones sobre el riel. Moravec propuso un método para encontrar la transformación de cuerpo rígido que mejor se ajustaba a dos conjuntos de nueve imágenes tomadas en posiciones consecutivas y para estimar el movimiento del robot a partir de dicha transformación.

A partir del trabajo de Moravec, la investigación en el área continuó pero fue evolucionando en dos subáreas de investigación: la odometría visual monocular y la odometría visual estereoscópica. En el enfoque monocular, las imágenes de cada cámara (si existen varias) son usadas de manera independiente. En contraste, en el enfoque estereoscópico, las imágenes de varias cámaras son usadas conjuntamente para extraer la información necesaria, es decir, en el enfoque monocular la relación entre las diferentes cámaras es tratada de forma individual y en el enfoque estereoscópico no. A continuación se presenta de manera resumida el desarrollo que estos dos enfoques han tenido.

[4]HANS MORAVEC. «*Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*». Tesis doct. Robotics Institute, Carnegie Mellon University y Stanford University, 1980.

[5]SCARAMUZZA y FRAUNDORFER, óp. cit.

[6]CHRIS HARRIS y MIKE STEPHENS. «*A combined corner and edge detector*». En: *In Proc. of Fourth Alvey Vision Conference*. 1988, págs. 147-151.

3.3.1. Enfoque Monocular

Se han obtenido resultados satisfactorios para sistemas de odometría visual que usan el enfoque monocular para recorridos de grandes distancias[7] -[8], estos sistemas serán descritos con más detalle en la siguiente sección. Estos avances se han dado tanto con el uso de cámaras tradicionales (de perspectiva), como también con el uso de cámaras omnidireccionales, las cuales tiene un campo de visión de 360° en el plano horizontal.

Uno de los primeros sistemas monoculares que presentó buenos resultados y tolerancia al ruido fue el propuesto por *Nister* et al. en el 2004[9], este sistema usaba un algoritmo 3D-2D para la estimación de la posición en cada instante y una de sus características mas innovadoras está relacionada con el uso del algoritmo *5-point RANSAC* para la estimación geométrica del movimiento bajo la presencia de *outliers* (datos que están fuera del modelo o patrón que mejor se ajusta a la mayoría de los otros datos). Este algoritmo obtuvo gran popularidad a partir del trabajo de *Nister* et al. y ha sido utilizado en varias implementaciones de sistemas monoculares subsiguientes[10].

En el 2005 Corke et al.[11] plantearon un sistema de odometría visual monocular que utilizaba una cámara omnidireccional para capturar las imágenes.

En el 2008 Tardif et al. presentaron un sistema de odometría visual con cámara omnidireccional[12]. Una de las cosas novedosas en su trabajo es que estimaron por separado la traslación y la rotación del agente (normalmente esta estimación se hacía de manera conjunta). En su desarrollo los *outliers* fueron removidos con *5-point RANSAC*.

En el 2009, Scaramuzza et al.[13] propusieron un sistema de odometría que también usaba una cámara omnidireccional e hicieron un gran aporte al proponer un algoritmo que denominaron *1-point RANSAC*, el cual permite hacer una estimación más rápida y eficiente de la posición del agente o vehículo al tener en cuenta un modelo de movimiento restringido basado en las restricciones no holonómicas de un vehículo en movimiento.

[7] NISTER, D. Et. Al. Óp. cit.

[8] SCARAMUZZA y FRAUNDORFER, óp. cit.

[9] NISTER, D. Et. Al. Óp. cit.

[10] J.-P. TARDIF, Y. PAVLIDIS y K. DANIILIDIS. «*Monocular visual odometry in urban environments using an omnidirectional camera*». En: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Sept. Págs. 2531-2538.

[11] P. CORKE, D. STRELOW y S. SINGH. «*Omnidirectional visual odometry for a planetary rover*». En: *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 4. Sept.-2 Oct. Págs. 4007-4012.

[12] TARDIF, PAVLIDIS y DANIILIDIS, óp. cit.

[13] D. SCARAMUZZA, F. FRAUNDORFER y R. SIEGWART. «*Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC*». En: *Robotics and Automation, ICRA '09. IEEE International Conference on*. 2009, págs. 4293-4299.

Existen otro conjunto de avances que se han hecho en la odometría visual monocular que se diferencian de los anteriormente descritos en cuanto que no usan un método basado en características para estimar el movimiento. Los métodos basados en apariencia no extraen puntos de interés de los fotogramas sino que usan la información de intensidad de todos los pixeles para estimar el movimiento. Algunos ejemplos de sistemas monoculares que usan métodos basados en apariencia son: Goeck et al. en el 2007 usaron la transformada de Fourier-Mellin para hacer odometría visual[14], Milford and Wyeth presentaron un método para estimar la velocidad traslacional y rotacional en el 2008 usando *template tracking*. Los métodos basados en apariencia tienen una desventaja y es la sensibilidad a las oclusiones. Por este motivo se han planteado sistemas híbridos que usan tanto el enfoque basado en apariencia, como el basado en características. Scaramuzza y Siegwart[15] describieron un sistema que usa el enfoque basado en apariencia para estimar rotación y el enfoque basado en características para calcular la traslación y corregir la estimación de la rotación.

3.3.2. Enfoque Estereoscópico

Muchas de las investigaciones desarrolladas en el campo de la odometría visual han sido desarrolladas haciendo uso de cámaras estereoscópicas debido a la simplicidad que se muestra a la hora de implementar los algoritmos para la estimación del movimiento. Mathies y Shaffer usaron un sistema binocular[16] -[17] y el detector de Moravec para hacer detección y rastreo de esquinas en una imagen dinámica. En contraste con los resultados obtenidos por Moravec, Mathies y shaffer demostraron un error del 2 % recuperando la trayectoria de un *rover* planetario en un recorrido de 5,5 metros. En trabajos posteriores, Olson et al.[18] -[19], desarrollaron un sistema estereoscópico usando un sensor de orientación, mejor conocido como cámara omnidireccional.En este trabajo demostraron que el error de un sistema de odometría visual estereoscópico crecía acumulativamente conforme la distancia recorrida aumentaba. Con el método propuesto Olson et al. se pudo reducir el error acumulativo a una función lineal que depende de la distancia total recorrida. Con esta mejora alcanzaron una estimación del movimiento con un error relativo del 1,2 % en un recorrido de 20 metros.

[14]R. Goecke y col. «*Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform*». En: *Intelligent Vehicles Symposium, 2007 IEEE*. 2007, págs. 450-455.

[15]D. SCARAMUZZA y R. SIEGWART. «*Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles*». En: *Robotics, IEEE Transactions on* 24.5 (), págs. 1015-1026.

[16]L. Matthies y S.A. Shafer. «Error modeling in stereo navigation». En: *Robotics and Automation, IEEE Journal of* 3.3 (June), págs. 239-248.

[17]Larry Henry Matthies. «Dynamic stereo vision». AAI9023429. Tesis doct. Pittsburgh, PA, USA, 1989.

[18]C.F. Olson y col. «Robust stereo ego-motion for long distance navigation». En: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. 2000, 453-458 vol.2.

[19]Olson C.F. y col. «Rover navigation using stereo ego-motion». En: *Robotics and Autonomous Systems* 43.4 (2003), págs. 215-229.

Los métodos hasta aquí descritos y otros más que se han desarrollado, tienen en común que los puntos son triangulados y que la estimación del movimiento se desarrolla usando el esquema 3D-3D(ver sección 3.4.4), que consiste en detectar los puntos característicos y triangularlos en 3 dimensiones, para luego rastrear dichos puntos en el siguiente fotograma y estimar su movimiento en un espacio de 3 dimensiones.

En *Nister* et al.[20] es donde se acuña el término *Visual Odometry* y se da un enfoque de implementación completamente diferente. En el trabajo de *Nister* et al. se desarrolló una metodología para hacer la estimación de movimiento conocida como 3D-2D(ver sección 3.4.4). En contraste con las metodologías estereoscópicas anteriores, la metodología 3D-2D de *Nister* et al. necesita de 3 fotogramas para hacer la estimación del movimiento por medio de un algoritmo conocido como (PnP, *Perspective From n Points*).

En Comport et al.[21] no se usó ninguna de las metodologías anteriormente mencionadas, 3D-3D o 3D-2D, se uso una estrategia basada en un tensor cuadrifocal (una matriz de vectores que describe las relaciones geométricas entre cuatro vistas distintas de una escena). Este tensor permite tener la estimación del movimiento calculada con emparejamientos de características en 2D-2D(ver sección 3.4.4) sin tener que triangularlos a un espacio de 3 dimensiones. La ventaja de este enfoque es la precisión con la que se hace la estimación del movimiento.

3.4. Descripción del problema de la odometría visual

3.4.1. Introducción

Como ya se mencionó anteriormente, un sistema de localización basado en odometría visual presenta las siguientes etapas:[22]:

1. Extracción de características.
2. Emparejamiento de características.
3. Estimación del movimiento.
4. Estimación Robusta.

[20] NISTER, D. Et. Al. Óp. cit.

[21] A.I. Comport, E. Malis y P. Rives. «Accurate Quadrifocal Tracking for Robust 3D Visual Odometry». En: *Robotics and Automation, 2007 IEEE International Conference on*. April, págs. 40-45.

[22] SCARAMUZZA y FRAUNDORFER, óp. cit.

Se plantearán las distintas alternativas en cada una de estas etapas, estas variaciones en conjunto dan lugar a distintos sistemas de odometría visual con diferentes características.

3.4.2. Extracción de características

La extracción de características consiste en extraer puntos característicos de una imagen, con el fin de realizar su correspondiente rastreo entre imágenes posteriores y estimar así el movimiento. Una característica puede ser, por ejemplo, un borde o una esquina. En los ambientes urbanos hay grandes cantidades de estos bordes y por lo tanto son escenarios ideales para la extracción de características.

Una esquina se define como un punto en la intersección de dos o más ejes. Un buen detector de características debe ser preciso con la localización de puntos de interés, además, debe ser eficiente computacionalmente, robusto e invariable ante rotaciones o cambios de perspectiva o iluminación dentro de la imagen.

Otro tipo de característica que puede detectarse dentro de un fotograma son los *blobs*, los cuales son patrones en las imágenes que difieren de sus vecinos en términos de intensidad, color y textura, pero no son bordes ni esquinas, por lo tanto, se pueden dividir los clasificadores en dos grupos, los *corners detectors* y los *blobs detectors*[23].

Dentro de los *corners detectors*, los más usados para la odometría visual son *Harris corners, Shi-Tomasi* y *FAST*. Dentro de los detectores de *blobs* se encuentran *SIFT, SURF* y *CenSurE.*

Algunos detectores de características

A continuación se mostrará una revisión de dos detectores de puntos de interés sobre una imagen. Inicialmente se detallará el algoritmo de *Harris* para detectar esquinas y luego se describirá un detector de *blobs* conocido como *SURF*.

Descripción del detector de esquinas de Harris

El detector de Harris[24] descrito en el algoritmo 3.1 proviene del detector de características de Moravec[25]. El detector de características de Moravec calcula el cambio en la intensidad

[23]Ibíd.
[24]HARRIS y STEPHENS, óp. cit.
[25]MORAVEC, óp. cit.

cuando una ventana determinada es desplazada en múltiples direcciones. Dado un pixel (x, y), una dirección (u, v) y una ventana de tamaño $2w + 1$, el cambio en la intensidad del pixel (x, y) en la dirección (u, v) se puede definir como:

$$E_{u,v}(x, y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} (I(x + i, y + j) - I(x + i + u, y + j + v))^2 \tag{3.1}$$

Una medida de que tan semejante es el pixel (x, y) a una esquina o característica esta dada por el valor mínimo de $E_{u,v}(x, y)$ en las 4 direcciones $(0, 1)$, $(0, -1)$, $(1, 0)$ y $(-1, 0)$. Esto se justifica al hacer las siguientes observaciones:

- Si el pixel que se está examinando hace parte de una línea en la imagen, el cambio en la intensidad del pixel es grande si se toma una dirección (u, v) perpendicular a la línea, pero si se toma una dirección (u, v) paralela a la imagen este cambio en la intensidad es pequeño, por lo tanto el mínimo valor de E para diversas direcciones es pequeño.
- Si el pixel que se está examinando hace parte de una región de intensidad uniforme, evidentemente el cambio en cualquier dirección (u, v) será pequeño y por lo tanto el mínimo valor de E también será pequeño.
- Si el pixel que se está examinando representa una esquina, el cambio en cualquier dirección (u, v) desde el pixel será grande y por lo tanto el valor mínimo de E para distintas direcciones también será grande.

El gran problema del detector de Moravec es que solo tiene en cuenta un número finito de direcciones para estimar $E_{u,v}(x, y)$. Este problema es resuelto por el detector de Harris[26] al realizar aproximaciones locales a los cambios de intensidad[27] por medio de una expansión en series de Taylor para $I(x + i + u, y + j + v)$ la cual se muestra a continuación:

$$I(x + i + u, y + j + v) \approx I(x + i, y + j) + I_x(x + i, y + j)u + I_y(x + i, y + j)v \tag{3.2}$$

Al reescribir 3.1 Teniendo en cuenta 3.2 se obtiene:

$$E_{u,v}(x, y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} (I_x(x + i, y + j)u + I_y(x + i, y + j)v)^2 \tag{3.3}$$

[26]HARRIS y STEPHENS, óp. cit.

[27]Mark Nixon y Alberto S. Aguado. *Feature Extraction & Image Processing, Second Edition.* 2nd. Academic Press, 2008. ISBN: 0123725380, 9780123725387.

La ecuación 3.3 se puede reescribir de manera equivalente como:

$$E_{u,v}(x,y) = A(x,y)u^2 + 2C(x,y)uv + B(x,y)v^2 \tag{3.4}$$

$$E_{u,v}(x,y) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.5}$$

Donde:

$$A(x,y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_x(x+i, y+j)^2 \tag{3.6}$$

$$B(x,y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_y(x+i, y+j)^2 \tag{3.7}$$

$$C(x,y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_x(x+i, y+j) I_y(x+i, y+j) \tag{3.8}$$

La ecuación 3.4 tiene dos ejes principales por ser una función cuadrática y por lo tanto puede ser rotada para que los ejes principales coincidan con los ejes coordenados y ser expresada de la siguiente manera:

$$F_{u,v}(x,y) = \alpha(x,y)u^2 + \beta(x,y)v^2 \tag{3.9}$$

$$F_{u,v}(x,y) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \alpha(x,y) & 0 \\ 0 & \beta(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.10}$$

Los valores de $\alpha(x,y)$ y $\beta(x,y)$ son proporcionales a la función de auto-correlación (ecuación 3.1) a lo largo de los ejes principales, por lo tanto, si ambos valores son grandes entonces se tiene una esquina en el punto (x,y), si uno de los valores es grande y el otro pequeño entonces en el punto (x,y) hay un borde y si ambos valores son pequeños entonces el punto (x,y) está ubicado en una región de intensidad uniforme.

En la práctica no hace falta calcular específicamente los valores de $\alpha(x,y)$ y $\beta(x,y)$, en lugar de eso se puede obtener de una medida de la semejanza del punto (x,y) a una esquina por medio de la siguiente expresión:

$$z(x,y) = \alpha(x,y)\beta(x,y) - k(\alpha(x,y) + \beta(x,y))^2 \tag{3.11}$$

Donde k es un parámetro del detector de Harris que determina la sensibilidad del detector a cambios en la imagen. La ecuación 3.11 evita calcular explícitamente los valores para $\alpha(x,y)$ y $\beta(x,y)$ dado que el producto $\alpha(x,y)\beta(x,y)$ y la suma $\alpha(x,y)+\beta(x,y)$ se puede calcular usando las siguientes igualdades:

$$M(x,y) = \begin{bmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{bmatrix} \tag{3.12}$$

$$\alpha(x,y)\beta(x,y) = Det(M(x,y)) = A(x,y) * B(x,y) - C(x,y)^2 \tag{3.13}$$

$$\alpha(x,y) + \beta(x,y) = Traza(M(x,y)) = A(x,y) + B(x,y) \tag{3.14}$$

Las ecuaciones 3.13 y 3.14 se obtienen a partir de las siguientes igualdades:

$$D(x,y) = \begin{bmatrix} \alpha(x,y) & 0 \\ 0 & \beta(x,y) \end{bmatrix} \tag{3.15}$$

$$\begin{aligned} M(x,y) &= R^{-T} D(x,y) R^{-1} \\ &= RD(x,y)R^T \end{aligned} \tag{3.16}$$

$$\begin{aligned} Det(M(x,y)) &= Det(RD(x,y)R^T) \\ &= Det(R)Det(D(x,y))Det(R^T) \\ &= Det(D(x,y)) \\ &= \alpha(x,y)\beta(x,y) \end{aligned} \tag{3.17}$$

$$\begin{aligned} Traza(M(x,y)) &= Traza(RD(x,y)R^T) \\ &= Traza(R^T RD(x,y)) \\ &= Traza(D(x,y)) \\ &= \alpha(x,y) + \beta(x,y) \end{aligned} \tag{3.18}$$

A continuación se presenta el algoritmo de Harris usando la respuesta de la función z definida en la ecuación 3.11.

Los resultados obtenidos en la extracción de características usando *Harris corners* son bastante satisfactorios, resaltando su simplicidad, siendo superado únicamente por *CenSurE*

Algoritmo 3.1 :Algoritmo de detección de esquinas de Harris

Entrada: Una imagen I.

1: Calcular las derivadas en x y en y de la imagen I.

$$\begin{aligned} I_x &= G_\sigma^x * I \\ I_y &= G_\sigma^y * I \end{aligned} \tag{3.19}$$

Donde G_σ^x es un kernel que brinda una estimación de la derivada horizontal de I y G_σ^y es el kernel que brinda la estimación de la derivada vertical. El operador $*$ es el operador de convolución.

2: Calcular cada uno de los siguientes productos entre las derivadas de la imagen para cada pixel.

$$\begin{aligned} I_{xx} &= I_x \cdot I_x \\ I_{yy} &= I_y \cdot I_y \\ I_{xy} &= I_x \cdot I_y \end{aligned} \tag{3.20}$$

3: Para cada pixel en la imagen I calcular los parámetros A, B y C detallados a continuación, estos parámetros dependen de las estimaciones de las derivadas I_x e I_y calculadas en el punto anterior. Para determinar estos parámetros se usa una vecindad alrededor del pixel (x, y) determinada por w.

$$\begin{aligned} A(x,y) &= \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_{xx}(x+i, y+j) \\ B(x,y) &= \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_{yy}(x+i, y+j) \\ C(x,y) &= \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_{xy}(x+i, y+j) \end{aligned} \tag{3.21}$$

4: Calcule el valor de respuesta del detector de características $z(x, y)$ para cada pixel en la imagen I.

$$z(x,y) = A(x,y)B(x,y) - C(x,y)^2 - k(A(x,y) + B(x,y))^2 \tag{3.22}$$

5: Un pixel será declarado como característica si el valor $z(x, y)$ es mayor que un umbral t. Otra alternativa es realizar $non - maximal\ supression$ sobre los valores de $z(x, y)$.

Salida: una lista de puntos característicos, cada uno representado por un par (x, y), la posición de dicho punto en la imagen I.

en términos de precisión, como se puede observar en la figura 3.18.

El uso de *SIFT*,*SURF* y *CenSurE* (*Blob detectors*) es bastante útil en escenarios donde las características analizadas no cambian a menudo, o cuando se desea realizar el rastreo de un conjunto en particular de características durante un periodo de tiempo considerable, pues los detectores de *Blobs* distinguen mejor un conjunto de características que los detectores de esquinas, pero lo hacen de forma más lenta[28]. Una característica importante de los detectores de *blobs* es que por lo general la descripción que hacen de un *blob* es invariante ante la escala y la rotación, esta característica hace preferible estos detectores para ciertos escenarios.

la Repetibilidad es la característica más importante de un detector de características, pues garantiza que se encontrarán los mismos puntos (o por lo menos una cantidad considerable de ellos) en un fotograma posterior, esto se busca con el objetivo de permitir el emparejamiento entre puntos de interés. Scaramuzza et al[29] presentan un cuadro comparativo con los principales métodos para la extracción de características, considerando como criterios de comparación la repetibilidad, precisión en la localización, robustez y eficiencia (ver sección 4.2).

SURF

A diferencia del detector de *Harris*, *SURF* no detecta puntos de interés como esquinas, en lugar de eso analiza regiones de la imagen (*blobs*) con el objetivo de buscar más información por medio de la cuál se pueda dar una mejor descripción de los puntos de interés de una imagen, dando lugar, a un detector más robusto, más repetitivo (capacidad de hallar las mismas características en diferentes imágenes de la misma escena) e invariante a circunstancias como la rotación o el escalamiento de las imágenes.

SURF nace en el año 2006 gracias a *Bay, Tuytelaars y Gool*[30], quienes lo diseñaron con el objetivo de crear un detector más robusto y que pudiese dar más información sobre los puntos de interés en una imagen I.

Como se mencionó, una de las características importantes de *SURF* es su invarianza a la escala. Esta característica es de vital importancia para la etapa de emparejamiento de características (sección 3.4.3) dado que para emparejar características comúnmente se examinan los vecindarios de los puntos de interés. Si se toma un vecindario de tamaño

[28]SCARAMUZZA y FRAUNDORFER, óp. cit.

[29]D. SCARAMUZZA y F. FRAUNDORFER. «*Visual Odometry [Tutorial part II]*». En: *Robotics Automation Magazine, IEEE* 19.2 (June.).

[30]Herbert Bay, Tinne Tuytelaars y Luc Van Gool. «SURF: Speeded Up Robust Features». En: *Proceedings of the ninth European Conference on Computer Vision*. 2006.

fijo de dos puntos de interés correspondientes que tienen una escala distinta entonces los patrones de intensidad no serán iguales, pero si se toma un vecindario proporcional al factor de escala asociado al punto de interés entonces se podrá hacer una mejor comparación que se ajuste al cambio de escala. SURF asocia este factor de escala descrito a cada una de los puntos de interés detectados.

SURF usa dos elementos para poder representar la respuesta de un punto con la cuál podrá clasificar si un punto $\mathbf{x} = (x, y)$ en alguna región de la imagen puede clasificarse como un punto de interés. En primer lugar *SURF* usa el determinante de una matriz Hessiana para asignar un valor de respuesta a un punto (x, y) sobre una imagen I, si el determinante de esta matriz Hessiana(ver ecuación 3.23) es alto entonces se espera que el punto examinado sea un punto donde hay grandes cambios de intensidad en múltiples direcciones y por lo tanto se espera que sea un punto de interés.

$$H = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{3.23}$$

donde:

$$L_{xx} = I(\mathbf{x}) * \frac{\partial^2 g(\sigma)}{\partial^2 x}$$

$$L_{xy} = I(\mathbf{x}) * \frac{\partial^2 g(\sigma)}{\partial x \partial y}$$

$$L_{yy} = I(\mathbf{x}) * \frac{\partial^2 g(\sigma)}{\partial^2 y}$$

El operador $*$ representa una convolución de la imagen con una ventana gaussiana para obtener las derivadas de la imagen de segundo orden.

El segundo elemento importante del detector *SURF* está relacionado en el parámetro σ (ecuación 3.23) que denota de manera implícita la escala a la cuál se aplicará el operador gaussiano (derivada). Si se examina la respuesta del filtro gaussiano a diferentes escalas en algún momento se alcanzará un valor máximo determinado y esto da lugar al principio de los detectores invariantes a la escala: si se se toma este valor máximo para dos objetos correspondientes que se encuentran en escalas diferentes, la razón entre los dos valores σ que produjeron ese máximo es igual a la razón entre las escalas de los objetos.[31].

De esa forma se declarará como un punto de interés a un punto sobre la imagen en donde

[31] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

se alcance un máximo local tanto en el espacio de las intensidades como en el espacio de la escala σ.

Un aspecto muy importante de SURF es su eficiencia. El cálculo de la convolución de la imagen con un operador gaussiano es costoso computacionalmente, en ese sentido se desarrolló una estrategia para hacerlo más eficiente. Esta estrategia consiste en usar varios kernel que aproximan los filtros gaussianos (Figura 3.1).

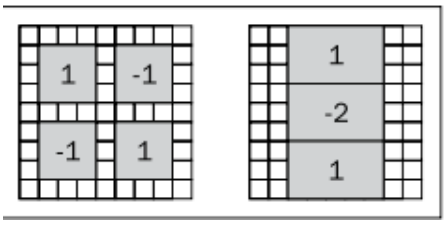


Figura 3.1: Aproximaciones a filtros gaussianos [32].

Los kernel son escalados de acuerdo al valor de σ y son aplicados eficientemente gracias al uso de una imagen integral que representa una matriz que para cada pixel guarda el acumulado de la intensidad de la imagen $I(\mathbf{x})$ hasta dicho pixel $\mathbf{x}$.

$$I(\mathbf{x}_k) = \sum_{i=0}^{i \leq x_k} \sum_{j=0}^{j \leq y_k} I(i,j) \tag{3.24}$$

De esta forma para cualquier región cuadrada sobre la imagen se puede calcular la suma de las intensidades usando la función definida en 3.24 y el principio de inclusión-exclusión.

El detector de *SURF* también define una manera de describir los puntos de interés. Esta descripción es invariante ante pequeños cambios de iluminación, rotaciones y escalamientos. El detector de *SURF* se basa en las diferencias locales de intensidades y se obtiene como sigue. Inicialmente se toma un vecindario de 20 veces el tamaño de la escala y se divide en un *grid* de $4x4$, en cada una de las celdas del *grid* los siguientes kernel son aplicados (varias veces en cada celda):

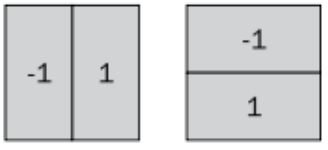


Figura 3.2: (izquierda) *Kernel dx*. (derecha) Kernel *dy* [33].

[32] (ibíd.)
[33] (ibíd.)

En la figura 3.2 el kernel de la izquierda aproxima los cambios de intensidad horizontales (dx) y el kernel de la derecha aproxima los cambios verticales de intensidad (dy). Para cada una de las celdas del *grid* se calcula lo siguiente:

$$\left[\sum dx \quad \sum dy \quad \sum |dx| \quad \sum |dy|\right] \tag{3.25}$$

Por lo tanto el descriptor de $SURF$ es un vector de 64 dimensiones dado que hay 16 celdas y 4 datos por celda. Para cada uno de los puntos de interés encontrados se tiene un descriptor asociado lo que permite hacer comparaciones entre estos puntos de interés. A continuación se muestra un conjunto de características identificadas por el detector $SURF$.

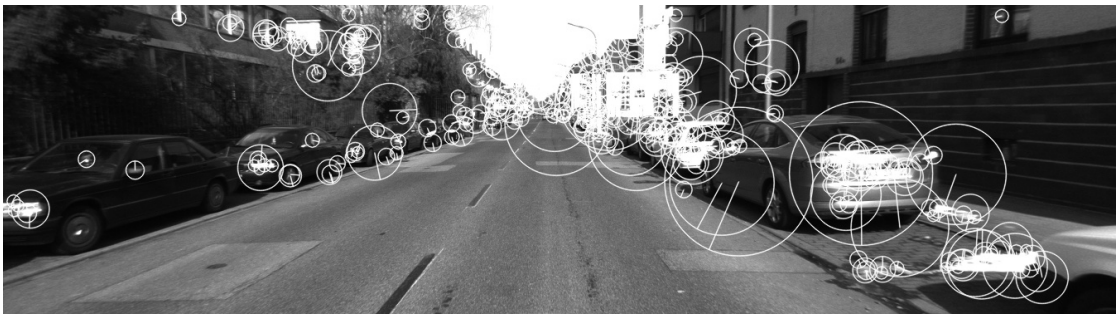


Figura 3.3: Características detectadas por $SURF$ con su respectiva escala y rotación.[34].

En la figura 3.3 se muestran las características detectadas por $SURF$ como el centro de cada uno de los círculos. El radio del círculo es proporcional a la escala calculada para cada característica, también se muestra la orientación asociada con cada característica lo cual permite que $SURF$ sea invariante a la rotación.

No es el objetivo de este trabajo profundizar en el estudio de este detector, una explicación más detallada de este algoritmo es dada por Bay, Ess, Tuytelaars y Gool[35], donde también se muestra el desempeño de $SURF$ con respecto a otros detectores.

[34] Autores

[35] Herbert Bay y col. «Speeded-Up Robust Features (SURF)». En: *Comput. Vis. Image Underst.* 110.3 (jun. de 2008), págs. 346-359. ISSN: 1077-3142.

3.4.3. Emparejamiento de características

Para que un sistema de odometría visual pueda hacer la estimación del movimiento de la cámara, debe identificar para un fotograma I_k sus correspondientes características en un fotograma I_{k-1}. A este proceso se le llama emparejamiento de características o *Feature Matching*. Inicialmente la manera más simple de hacer una comparación entre características de distintos fotogramas es comparando todos los puntos característicos de una imagen con los de la otra.

En primera instancia debe definirse una medida de similitud entre una característica F_k^i y F_{k-1}^j de tal manera que pueda establecerse una afinidad entre ambas, para luego determinar si dicho par de características puede formar una correspondencia, es decir, si pueden emparejarse. Entre algunas de estas medidas de similitud se encuentran *SSD o SAD*(*Sum of Square Differences*), *NCC*(*Normal Cross Correlation*)[36] o *NNDR*(*Nearest Neighbor distance Radio*)[37]. Estas medidas de similitud hacen parte de la primera etapa del *Feature Matching* y son usadas por muchas implementaciones.

A continuación se desarrolla una breve explicación de cada medida de similitud:

Dadas dos imágenes I_k e I_{k-1}, dos características F_k^i posicionada en (u', v') y F_{k-1}^j posicionada en (u, v) correspondientes a la imagen k y a la imagen $k-1$ respectivamente y un parámetro w que hace referencia al tamaño de una ventana cuadrada centrada en (u', v') y (u, v) , cada medida de similitud $\phi(F_k^i, F_{k-1}^j)$ entre ambas características se describe como:

- ***SSD***:

$$\phi(F_k^i, F_{k-1}^j) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} (I_k(i+u', j+v') - I_{k-1}(i+u, j+v))^2 \tag{3.26}$$

- ***NCC***:se precalculan lo siguientes valores para cada punto característico F_k^i y F_{k-1}^j en sus respectivas imágenes:

$$A = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I(i+u, j+v) \tag{3.27}$$

$$B = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I(i+u, j+v)^2 \tag{3.28}$$

[36] NISTER, D. Et. Al. Óp. cit.

[37] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN: 1848829345, 9781848829343.

$$C = \frac{1}{\sqrt{nB - A^2}} \tag{3.29}$$

Donde cada una de estas variables representa la intensidad en color para cada ventana centrada en (u, v) y (u', v'). Luego para cada par de características se calcula el siguiente valor de correlación:

$$D = \sum_{i=-w}^{w} \sum_{j=-w}^{w} I_{k-1}(i+u, j+v) I_k(i+u', j+v') \tag{3.30}$$

la medida de similitud finalmente viene dada por:

$$\phi(F_k^i, F_{k-1}^j) = (nD - A_K A_{k-1}) C_k C_{k-1} \tag{3.31}$$

donde $n = w * w$.

- ***NNDR***: Como su nombre lo indica, lo que se trata de buscar con esta medida de similitud es, en un espacio arbitrario n-dimensional, el vecino más cercano de una característica F_{k-1}^i. Esta medida de similitud se usa generalmente cuando se tienen descriptores de características como *SURF* y *SIFT*, donde a cada punto característico F_{k-1}^i y F_k^j se le asigna un descriptor que describe a ese característica por medio de una tupla $P(u) = (p1, p2, \ldots, pn)$, donde u representa un punto característico. Esta tupla describe un vector n-dimensional que representa a una característica en el espacio del descriptor. En ese sentido una medida de similitud *NNDR* se describe como:

  $$\phi(F_k^i, F_{k-1}^j) = D(P(F_{k-1}^i), P(F_k^j)) \tag{3.32}$$

  donde D es la distancia euclidiana en un espacio n-dimensional, o cualquier otra medida de distancia (distancia $Manhattan$, etc.).

Algunas metodologías para el emparejamiento de características:

- **Emparejamiento NO restringido(*Unconstrained Matching*)** Se usa una medida de similitud de las anteriormente mencionadas, de aquí en adelante el proceso es intuitivo, se busca comparar cada característica F_{k-1}^ien un fotograma I_{k-1} con cada una de las características F_k^i del fotograma actual I_k y se escoge la característica cuya medida de similitud sea máxima. Sin embargo este proceso esta sujeto a errores que pueden viciar las correspondencias hechas entre dos fotogramas, por tal razón algunos autores como *Nister* et al.[38] utilizaron un truco conocido como *Mutual*

[38]NISTER, D. Et. Al. Óp. cit.

Consistency Check MCC, que consiste en hacer dos medidas de similitud, una del fotograma I_{k-1} al fotograma I_k y otra del fotograma I_k al fotograma I_{k-1} y solo se generara una correspondencia entre dos características si la medida de similitud de la característica F_{k-1}^i con la característica F_k^i es la máxima dentro de las restantes y si la medida de similitud de la característica F_k^i con la característica F_{k-1}^i es la máxima también, en otras palabras, si la característica favorita de F_{k-1}^i es F_k^j y la característica favorita de F_k^j es F_{k-1}^i.

- **Emparejamiento restringido(*Constrained Matching*)** Un problema evidente en el enfoque anterior es que su complejidad es de $O(n^2)$ donde n es el número de puntos característicos en cada fotograma, por esta razón a medida que el conjunto de puntos característicos aumente el tiempo de ejecución del algoritmo será mucho más elevado. Un enfoque mucho más eficiente es el de usar estructuras de datos eficientes como árboles de búsqueda multidimensionales o tablas *hash* que permitan buscar eficientemente los puntos característicos en una región determinada, de esa manera se minimiza el espacio de búsqueda y se reduce la complejidad del algoritmo de emparejamiento. Las regiones de búsqueda pueden delimitarse usando una estimación del movimiento dada por otros dispositivos como *IMU's* o *GPS's*, por otro lado estas regiones también pueden encontrarse modelando regiones gaussianas dadas por un modelo de movimiento previo. Puede ser el caso de que las regiones de búsqueda de puntos característicos sean definidas *a priori* simplemente usando la intuición y las restricciones del problema a solucionar.

  Un enfoque planteado por Davidson[39] tiene la característica de usar un modelo de movimiento previamente conocido, asumiendo que la velocidad del vehículo sea constante, supone que cada punto característico debe estar en una posición (u, v) según el modelo planteado. Luego se calcula la incertidumbre dada por este modelo la cuál describe una forma elíptica gaussiana y representa la región en la cuál debe buscarse ese punto característico para hacer el emparejamiento usando las medidas de similitud previamente definidas. Para poder usar este enfoque, la posición en 3D de los puntos característicos debe ser conocida.

  Existe otro enfoque en donde no se necesita la posición en 3D de los puntos característicos, solo conociendo un modelo de movimiento como en el caso anterior basta para conocer la región en donde debe buscarse una característica. Más específicamente esta metodología trata de buscar en un fotograma I_k un punto característico ubicado en un fotograma I_{k-1} teniendo en cuenta que este punto se va encontrar sobre una linea epipolar formada por el centro de la proyección c_k y la proyección

[39] A.J. Davison. «Real-time simultaneous localisation and mapping with a single camera». En: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on.* 2003, 1403-1410 vol.2. DOI: 10.1109/ICCV.2003.1238654.

esperada del punto característico sobre el fotograma I_k. Está linea epipolar puede ser fácilmente calculada por medio de un modelo de movimiento ya previamente conocido como se especificó anteriormente. Esta metodología es ventajosa para enfoques monoculares o estereoscópicos que no hagan una triangulación de los puntos característicos.

Conocida entonces la región en donde deben encontrarse las posibles correspondencias M_k en I_k para una característica F^i_{k-1}, se procede entonces a establecer la afinidad entre esta última y cada una de las características de M_k(que es un subconjunto de las características de I_k), pero, ¿cómo saber cuáles características componen las posibles correspondencias M_k?, un enfoque intuitivo sería el de buscar por cada característica de I_k si se encuentra en la región especificada, sin embargo este enfoque tiene una complejidad cuadrática en el número de características y se tendría de nuevo el problema inicial del emparejamiento no-restringido(*Unconstrained matching*), por ese motivo, existen diferentes soluciones eficientes a este problema[40], pero para las necesidades del sistema de localización se tienen en cuenta dos ampliamente usadas, tablas hash multidimensionales(*multidimensional hashing (MH)*) y arboles de búsqueda multidimensionales(*kd trees*).

En resumen una tabla *hash* multidimensional es una estructura de datos que mapea una llave a un conjunto de datos, para el caso de un imagen cuyos puntos característicos han sido identificados, la llave sería una región específica que estaría mapeada a los puntos características que en ella contenga.

Por otro lado un árbol de búsqueda multidimensional o *kd-tree* es una estructura de datos que permite hacer consultas eficientes de datos que poseen alguna dimensionalidad no mayor que 20. Para el caso, se tiene una dimensionalidad de 2, ya que cada característica se establece por medio de una dupla, en donde el primer elemento es la posición en x y el segundo elemento es la posición en y. La primera particularidad para el caso del *kd-tree* es que divide el espacio de la imagen I_k por medio de rectas paralelas a los ejes coordenados como se muestra en el lado izquierdo de la figura 3.4[41], creando una representación del espacio de características en forma de árbol de búsqueda binario balanceado, lo que supone una altura de árbol mínima. Un *kd-tree* puede ser construido en $O(n \log n)$, y una consulta específica para una subregión cuadrada cualquiera sobre la imagen puede llegar a tardar $O(\sqrt{n} + k)$, donde n es el número de puntos característicos y k es el número de puntos que se reportan para una consulta específica.

[40] Szeliski, óp. cit.

[41] UNIVERSITY OF FLORIDA Alper Üngör. *Computational Geometry: Kd-Trees and Range Trees.* [citado en 24 de junio de 2013]. URL: `http://www.cise.ufl.edu/class/cot5520fa09`.

[42] (ibíd.)

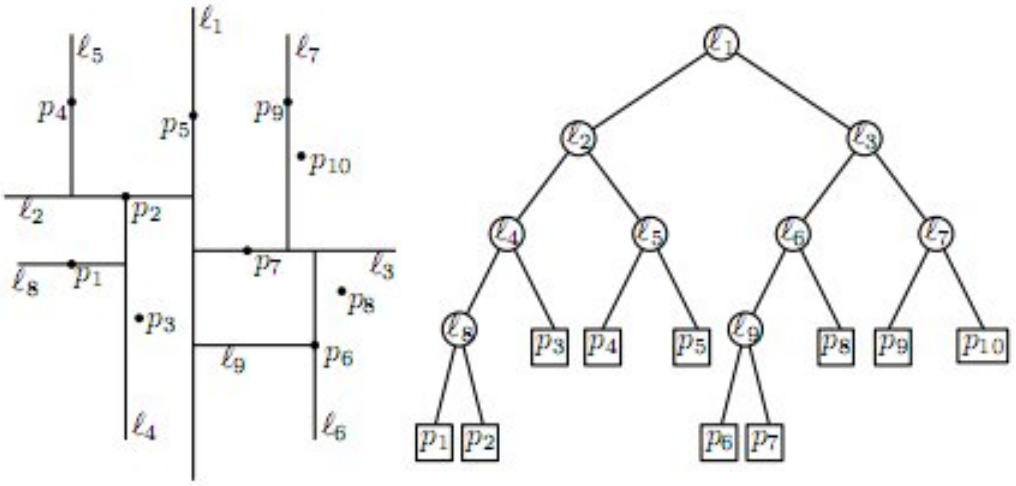


Figura 3.4: Representación gráfica de los árboles de búsqueda multidimensionales[42].

Es evidente entonces que una metodología de emparejamiento de características para el caso de una estimación de movimiento, donde el tiempo de respuesta es un cuello de botella, debe ser lo más eficiente posible en términos de tiempo de ejecución, debido a esta razón es recomendable usar un enfoque emparejamiento restringido(*Constrained Matching*) si se busca un implementar un sistema que funcione en tiempo real.

Otra alternativa que surge para el emparejamiento de características, es el rastreo de características o *Feature Tracking*. Mientras el emparejamiento de características toma dos fotogramas y genera unas correspondencias entre las características detectadas en ambos fotogramas, el rastreo de características toma los puntos característicos ubicados en el fotograma inicial I_0 y trata de rastrearlos en los fotogramas siguientes. El problema con el rastreo de características radica en el hecho de que solo funciona bien cuando los movimientos son pequeños y el cambio de la imagen de video entre cada fotograma es pequeña. Esta última restricción en la velocidad del movimiento puede ser evadida usando un rastreador *KLT* llamado así por sus inventores, *Kanade, Lucas, Tomasi*[43]. Sin embargo su implementación es más compleja que el emparejamiento de características, aunque los resultados obtenidos son buenos[44].

[43]J. Shi y C. Tomasi. «Good features to track». En: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*. 1994, págs. 593-600.

[44]SCARAMUZZA y FRAUNDORFER, «*Visual Odometry [Tutorial]*».

3.4.4. Estimación del movimiento

Introducción

Esta etapa de un sistema de odometría visual busca estimar el movimiento (traslación y rotación) a partir de los conjuntos de características F_{k-1} y F_k que han sido extraídos a partir de dos fotogramas y cuyas características han sido emparejadas (etapa de emparejamiento de características 3.4.3). El conjunto F_{k-1} representa las características extraídas de la imagen del instante anterior y el conjunto F_k representa las mismas características correspondientes en la imagen actual. Esta etapa es fundamental en un sistema de odometría visual debido a que la trayectoria total del vehículo o robot se puede determinar concatenando las estimaciones de movimiento entre imágenes.

Formalmente, esta etapa busca estimar la transformación de cuerpo rígido $T_{k,k-1}$ que representa el movimiento relativo de la cámara entre los instantes de tiempo $k-1$ y k a partir de los conjuntos de características F_k y F_{k-1}. Teniendo en cuenta lo anterior el método particular de estimación de movimiento se puede representar a través de una función G que debe satisfacer la ecuación 3.33.

$$T_{k,k-1} = G(F_k, F_{k-1}) \tag{3.33}$$

La transformación $T_{k,k-1}$ tiene una componente de traslación $t_{k,k-1}$ y una componente de rotación $R_{k,k-1}$, de tal manera que $T_{k,k-1}$ puede ser representada en coordenadas homogéneas así[45]:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{3.34}$$

Para estimar la posición de la cámara C_n en el instante n, y por lo tanto del agente en movimiento, se pueden concatenar todas las transformaciones $T_{k,k-1}$ para $k = (1, 2, \ldots, n)$ o equivalentemente usar la ecuación 3.35. Nótese que la estimación de la posición de la cámara se realiza de manera incremental.

$$C_n = T_{n,n-1} C_{n-1} \tag{3.35}$$

Existen diferentes maneras de realizar la estimación del movimiento a partir de las imáge-

[45] Ibíd.

nes, es decir, existen varias elecciones para la función G que se muestra en la ecuación 3.33. El tipo de método a utilizar generalmente está determinado por la manera en que se especifican las características (2D, 3D). Principalmente hay tres formas de estimar el movimiento[46]:

- 3D - 3D: Bajo esta perspectiva ambos conjuntos de características (F_{k-1}, F_k) están especificados en 3 dimensiones.
- 2D - 2D: En este enfoque ambos conjuntos de características (F_{k-1}, F_k) están especificados en coordenadas 2D.
- 3D - 2D: En este caso el conjunto de características F_{k-1} está especificado en 3D mientras que el conjunto F_k está especificado en 2D (proyección en la imagen actual).

A continuación se presentan cada una de los algoritmos para estimar el movimiento de acuerdo a la manera como están especificados los conjuntos de características (F_{k-1}, F_k).

Método 3D - 3D

Para hacer estimación 3D - 3D es necesario tener los conjuntos de características F_k y F_{k-1} especificados en el espacio tridimensional. Lo anterior implica usar visión estereoscópica para tener la capacidad de triangular las características en el espacio en cualquier instante de tiempo k.

Teniendo en cuenta que la triangulación de los puntos está sujeta a errores inherentes y que por lo tanto en la práctica es muy difícil tener una estimación exacta del movimiento, el problema bajo esta perspectiva de estimación de movimiento se formula como sigue: Encontrar la transformación de cuerpo rígido T_k que minimiza la distancia euclidiana entre los conjuntos de características F_k y ${F_{k-1}}$[47], esto se muestra en la ecuación 3.36, en esta ecuación el índice i representa la característica i-ésima en ambos conjuntos F_k y F_{k-1}

$$\arg\min_{T_k} \sum_i \|F_k^i - T_k F_{k-1}^i\|^2 \tag{3.36}$$

La solución al problema planteado en la ecuación 3.36 se puede encontrar usando cuaterniones* o descomposición de valores singulares[48].

[46] T.S. Huang y A.N. Netravali. «Motion and structure from feature correspondences: a review». En: *Proceedings of the IEEE* 82.2 (1994), págs. 252-268.

[47] SCARAMUZZA y FRAUNDORFER, óp. cit.

[48] K.S. Arun, T. S. Huang y S. D. Blostein. «Least-Squares Fitting of Two 3-D Point Sets». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-9.5 (1987), págs. 698-700.

Un hecho fundamental en la estimación 3D - 3D es que para determinar de manera única la transformación T_k solo son necesarias 3 correspondencias de puntos no colineales[49], es decir, 3 características emparejadas entre los conjuntos F_k y F_{k-1}. Este hecho es de suma importancia para la etapa de estimación robusta descrita en la sección 3.4.5 debido a que la complejidad computacional de esta etapa aumenta en la medida en que se necesitan más puntos para determinar de manera única la transformación T_k.

Scaramuzza et al. proponen el siguiente algoritmo para la estimación del movimiento 3D - 3D:

Algoritmo 3.2 :Método de estimación 3D-3D

1: **loop**
2: Capturar dos pares de imágenes estereoscópicas, I_{k-1} e I_k.
3: Extraer y Emparejar características a partir de I_{k-1} e I_k.
4: Triangular las características emparejadas para cada par, obteniendo F_{k-1} y F_k.
5: Computar T_k a partir de los conjuntos de características F_k y F_{k-1}.
6: $C_k = T_k C_{k-1}$ ▷ Concatenar T_k a la estimación de la trayectoria de la cámara
7: **end loop**

A continuación se presentan algunas de las características relevantes de la estimación 3D-3D:

- La escala del movimiento es absoluta con este método. Lo cual permite que la trayectoria de la cámara sea determinada concatenando de manera directa las transformaciones estimadas.
- A medida que la distancia de la cámara a la escena va superando el tamaño del *baseline* de la cámara estereoscópica se van produciendo errores en la triangulación cada vez mayores. Esto exige que se usen mecanismos de corrección, lo que representa un incremento en el tiempo de procesamiento, o esquemas diferentes de estimación de movimiento.
- En algunos trabajos[50] se reporta que la calidad de los resultados del método 3D-3D son inferiores con relación al enfoque 3D-2D y 2D-2D. Esto generalmente ocurre porque la triangulación usada en el enfoque 3D-3D puede introducir grandes errores en la estimación del movimiento.

[49] puntos colineales: puntos que se encuentran sobre una misma línea recta
[50] NISTER, D. Et. Al. Óp. cit.

Método 2D - 2D

Cuando se usa el enfoque 2D-2D tanto el conjunto de características F_{k-1} y F_k (ecuación 3.33) son especificados en 2D. Estos conjuntos se construyen a partir de los fotogramas I_{k-1} e I_k y el objetivo es estimar la transformación de cuerpo rígido T_k que experimentó la cámara entre el instante $k-1$ y el instante k. En la estimación 2D-2D se busca estimar el movimiento (T_k) a partir de la matriz esencial E, a continuación se define la matriz esencial, la manera de determinarla a partir de correspondencias 2D-2D y la relación de esta con el movimiento (T_k) de la cámara.

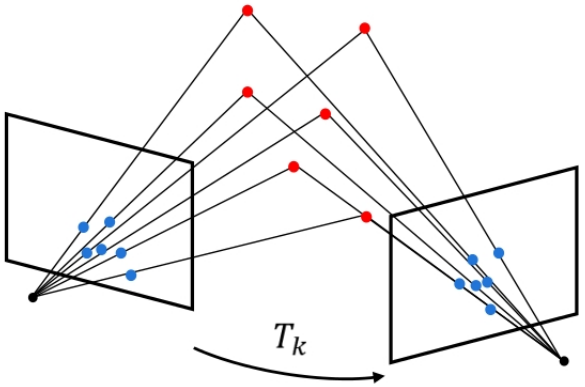


Figura 3.5: Estimación del movimiento (T_k) a partir de correspondencias 2D-2D[51].

La matriz esencial E es una matriz 3x3 que relaciona puntos correspondientes en un par de fotogramas asumiendo que las cámaras están calibradas y pueden ser modeladas como cámaras *pinhole*. La matriz E contiene los parámetros de movimiento pero el factor de escala es desconocido para la traslación, esto se muestra en la ecuación 3.37, en esta ecuación t y R representan la traslación y la rotación experimentada por la cámara entre los fotogramas I_{k-1} y I_k.

$$E \simeq [t]_x R \tag{3.37}$$

Se usa el símbolo $\simeq$ en 3.37 debido a que la equivalencia es válida incluso si se multiplica $[t]_x R$ por un escalar, dado que no se conoce el factor de escala de la traslación. La razón por la cual es imposible determinar la escala de la traslación a partir de 2 fotogramas se ilustra en el siguiente ejemplo: suponga que una cámara en movimiento captura dos fotogramas de un cuerpo rígido en dos instantes de tiempo distintos , si existiera una cámara nueva

[51] (UNIVERSITY OF ZURICH Davide Scaramuzza. *A tutorial on visual odometry.* [citado en junio de 2013]. URL: `http://sites.google.com/site/scarabotix/`)

que se moviera al doble de la velocidad de la cámara anterior y observara el mismo cuerpo rígido ampliado en un factor de 2 y al doble de la distancia y capturara dos fotogramas en los mismos instantes de tiempo que la cámara anterior, ambas cámaras obtendrían las mismas imágenes y es evidente que la traslación es distinta en ambos casos.

La notación $[t]_x$ (donde t es el vector columna $[t_x, t_y, t_z]$) se usa para representar la matriz antisimétrica siguiente:

$$[t]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{3.38}$$

La notación $[t]_x$ se usa comúnmente para representar el producto cruz como un producto matricial. Por lo tanto, se tiene que $[t]_x W = t \times W$.

De acuerdo a la equivalencia mostrada en 3.37 es natural pensar en estimar la matriz esencial E que relaciona dos fotogramas para extraer los parámetros de movimiento (R, t) a partir de dicha estimación. Para estimar la matriz esencial E se pueden usar las correspondencias 2D-2D entre las características de los fotogramas I_{k-1} e I_k usando la restricción epipolar.

Estimación de la matriz esencial por medio de la restricción epipolar

La restricción epipolar restringe la línea en la que puede aparecer la característica x' del fotograma I_k correspondiente a la característica x del fotograma I_{k-1}. En la figura 3.6 se muestra una ilustración de esta restricción.

La restricción epipolar se puede expresar matemáticamente como se muestra en la ecuación 3.39.

$$(x')^T E x = 0 \tag{3.39}$$

donde x' y x son puntos en el plano de la imagen representados en coordenadas homogéneas, además, x' y x son ambos vectores columnas con componentes $[u', v', 1]$ y $[u, v, 1]$ respectivamente .

Esta restricción epipolar se deriva del hecho de que los vectores $\vec{C_{k-1}}x$ que puede llamarse x, el vector $\vec{C_k}x'$ que se llamará x' y el vector de traslación t entre C_{k-1} y C_k son coplanares en el plano epipolar que se ilustra en la figura 3.6. Para poder derivar la restricción epipolar

[52] Autores

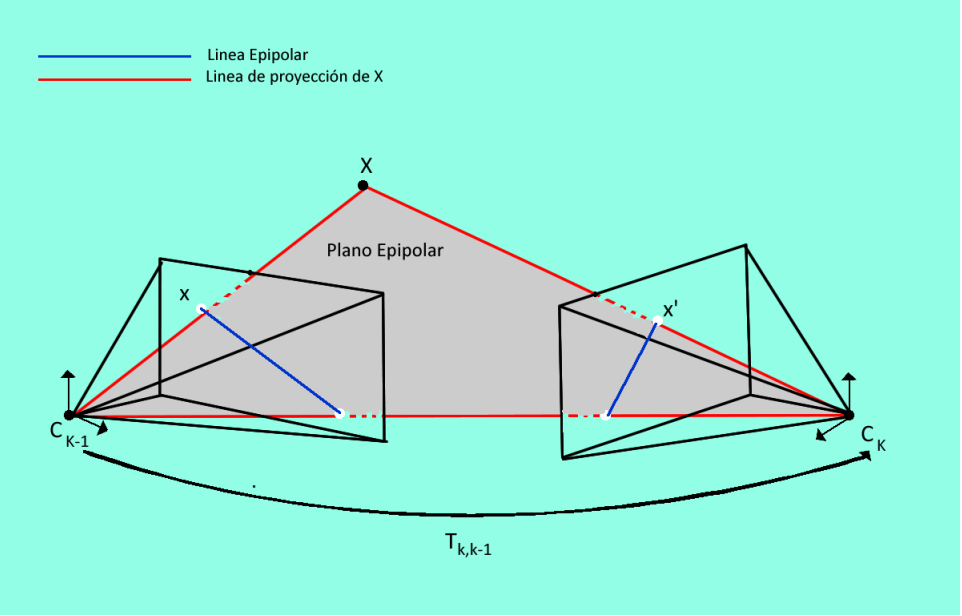


Figura 3.6: Restricción epipolar, fotograma I_{k-1} (izquierda), fotograma I_k (derecha)[52].

se deben llevar dichos vectores mencionados anteriormente a un sistema de coordenadas común, en este caso al sistema de coordenadas de la cámara C_k como se muestra en la figura 3.7.

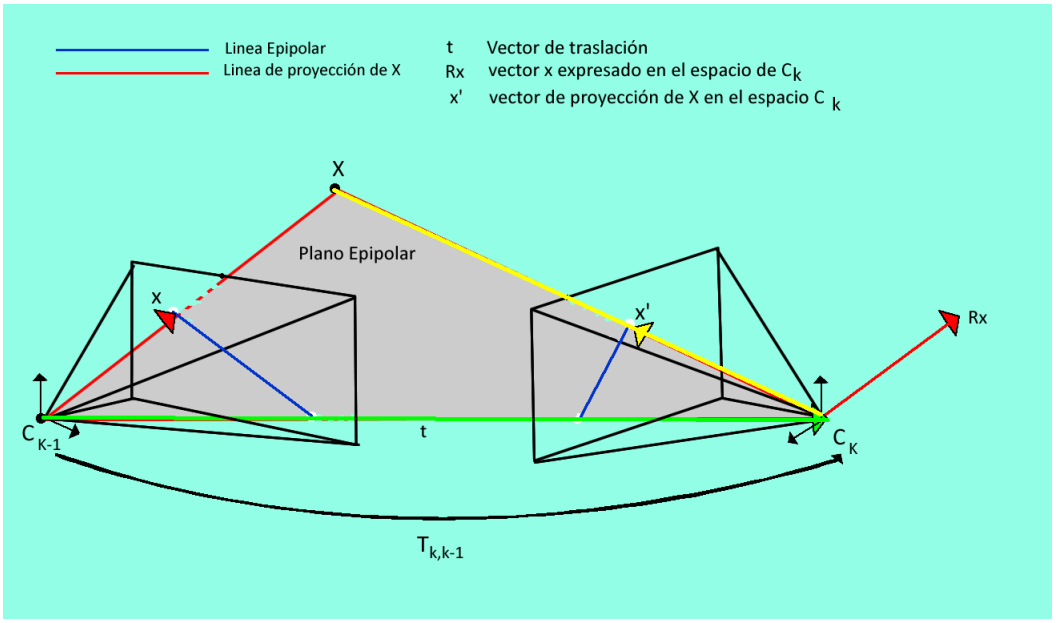


Figura 3.7: Vectores $C_{k-1}\vec{x}$(rojo), t(verde) y $C_k\vec{x'}$(amarillo) expresados en el sistema de coordenadas C_k. fotograma I_{k-1} (izquierda), fotograma I_k (derecha).[53].

[53]Autores

Dado esto y sabiendo ya que los vectores mencionados anteriormente son coplanares, se cumple que el producto mixto entre dichos vectores debe ser igual a cero. El producto mixto entre tres vectores $\vec{v_1}, \vec{v_2}$ y $\vec{v_3}$ en tres dimensiones se define de la siguiente manera:

$$\vec{v_1} \cdot (\vec{v_2} \times \vec{v_3}) = 0 \tag{3.40}$$

Este producto representa el volumen del paralelepípedo formado por tres vectores y puede representarse como un producto matricial de la siguiente manera:

$$v_1^T [v_2]_x v_3 = 0 \tag{3.41}$$

Ahora reemplazando en la expresión 3.41 v_1^T por x'^T, $[v_2]_x$ por $[t]_x$ y v_3 por Rx(Rx es el vector que se obtiene al rotar x con transformación $T_{k,k-1}$ entre las dos imágenes) se obtiene la siguiente ecuación:

$$x'^T [t]_x R x = 0 \tag{3.42}$$

Es evidente entonces que el elemento $[t]_x R$ es igual a la matriz esencial de la ecuación 3.39. La matriz esencial es entonces la multiplicación de una matriz de rotación y de traslación. E define una matriz de 3x3 cuyas entradas se listan como sigue:

$$E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} \tag{3.43}$$

Teniendo en cuenta la restricción epipolar (ecuación 3.39) cada correspondencia de características 2D-2D impone una restricción de la siguiente forma:

$$W D^T = 0 \tag{3.44}$$

$$W = \begin{bmatrix} uu' & u'v & u' & uv' & vv' & v' & u & v & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{bmatrix}$$

Los escalares e_i en D representan cada una de las componentes de la matriz esencial E (matriz 3×3). Si el número de correspondencias 2D-2D que se tienen son al menos 8

($N \geq 8$), entonces pueden hallarse todos los valores de la matriz esencial formulando un sistema de ecuaciones lineales de la siguiente manera:

$$AD^T = 0 \tag{3.45}$$

Donde:

$$A = \begin{bmatrix} u_1u'_1 & u'_1v_1 & u'_1 & u_1v'_1 & v_1v'_1 & v'_1 & u_1 & v_1 & 1 \\ u_2u'_2 & u'_2v_2 & u'_2 & u_2v'_2 & v_2v'_2 & v'_2 & u_2 & v_2 & 1 \\ \dots & & & & & & & & \\ \dots & & & & & & & & \\ u_nu'_n & u'_nv_n & u'_n & u_nv'_n & v_nv'_n & v'_n & u_n & v_n & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{bmatrix}$$

Existe un método lineal de solución para el problema de la estimación de la matriz esencial E conocido como el algoritmo de $Longuet - Higgins$[54] y que consiste básicamente en resolver el sistema de ecuaciones lineales homogéneas en las variables $e_1, e_2, \dots, e_9$ que se obtienen al reemplazar cada una de las correspondencias 2D-2D como se ilustra en la ecuación 3.45. La razón por lo cual solo se necesitan 8 correspondencias 2D-2D es que el sistema de ecuaciones que se desea resolver tiene 8 variables e_i, a excepción de una variable e_9 que se asume desconocida y representa el factor de escala mencionado. Si ($N > 8$) entonces se tiene un sistema de ecuaciones sobre determinado y se puede tener una estimación más robusta de E por mínimos cuadrados que se explica a continuación.

Una solución D para la ecuación 3.45, no es la única solución para este sistema, de hecho, este sistema de ecuaciones tiene infinitas soluciones como múltiplos escalares kD existan. Por tal motivo el problema a solucionar debe formularse desde otra perspectiva.

El problema de solucionar $AD^T = 0$ es análogo a encontrar un D tal que minimice $\|AD^T\|$ con la restricción de que $\|D\| = 1$(asegurando así una solución no trivial). Este es un problema de estimación de mínimos cuadrados.

Para resolver este problema se usa una herramienta conocida como descomposición en valores singulares(SVD por sus siglas en inglés). Esta descomposición se aplica a la matriz

[54]HC Longuet-Higgins. «A computer algorithm for reconstructing a scene from two projections». En: *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds* (1987), págs. 61-62.

de $n \times 9$ A de tal forma que:

$$SVD(A) = USV^T \tag{3.46}$$

Donde U es una matriz unitaria de $n \times n$, S es una matriz diagonal de $n \times 9$ y V es una matriz unitaria de 9×9.

Luego, la solución D que minimiza a $\|AD^T\|$ con la restricción anteriormente planteada es la última columna de V(Hartley & zisserman[55]). De esta manera se obtiene la solución de la matriz esencial dadas 8 o más correspondencias, sin embargo la matriz esencial de manera directa no representa rotación y la traslación que hubo entre dos instantes. Para poder obtener R y $\vec{t}$ se debe hacer otro procedimiento.

Reforzando la restricción singular

Una característica importante que debe resaltarse de la matriz esencial es que debe ser de rango 2 y cuyo determinante es cero. Cabe resaltar que el rango de una matriz se puede definir como el número de valores singulares de la misma que sean diferentes de 0. Para el caso de la matriz esencial E se esperaría que dos de sus valores singulares sean diferente de cero para que sea de rango 2. Los valores singulares de una matriz son los valores de $diag(S)$ luego de aplicar el SVD de la ecuación 3.46, pero esta vez para la matriz E.

En la práctica, la matriz esencial E que se encuentra luego de resolver 3.45 no es de rango 2, por tal motivo se debe "forzar" a esta matriz para que sea del rango deseado. Para lograr esto se declara una matriz esencial E' de rango 2 que minimice la norma de *frobenius* $\|E - E'\|$ con la restricción de que $det(E') = 0$. Para poder lograr lo anterior, la descomposición $SVD(E) = USV^T$ con $diag(S) = (r, s, t)$ debe cumplir que $r \geq s \geq t$, de esta manera se deja $E' = US'V^T$ donde S' es una matriz diagonal con el último valor singular igual a cero, $diag(S) = (r, s, 0)$, de esa manera se asegura que la matriz esencial obtenida sea de rango 2. Este método fue sugerido por *hartley*[56] y planteado por *Tsai y Huang*[57].

Extracción de R y $\vec{t}$ de la matriz esencial

Como se ha puntualizado anteriormente, no es suficiente con la matriz esencial para conocer la posición de la cámara en un instante dado, para esto se debe conocer cuál fue la rotación

[55] R. I. Hartley y A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. 2004.

[56] R.I. Hartley. «In defense of the eight-point algorithm». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.6 (1997), págs. 580-593.

[57] R. Tsai y T.S. Huang. «Uniqueness and estimation of three-dimensional motion parameters of a rigid planar patch from three perspective views». En: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82*. Vol. 7. 1982, págs. 834-838.

R y la traslación $\vec{t}$. Para poder hallar estos valores se usa de nuevo SVD para este propósito, pero esta vez sobre la matriz E' que se halla luego de asegurar la restricción singular.

Cabe resaltar que luego de este procedimiento existen cuatro posibles soluciones para R y $[t]_x$ que satisfacen la restricción epipolar, por tal motivo debe realizarse una desambiguación que se detallará posteriormente en la sección 5.6. Las cuatro soluciones se listan como sigue:

dado que

$$SVD(E') = USV^T$$

Se establece que las cuatro posibles soluciones son[58]:

$$R = U(\pm W)V^T$$
$$[t]_x = U(\pm W)SV^T$$

donde:

$$W = \begin{bmatrix} 0 & \pm 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

Un detalle importante radica en que la matriz de rotación y de traslación representa en sí la matriz de proyección $[R|t]$ de la cámara en la posición C_k. La importancia de este detalle indica que conociendo esta matriz de proyección se puede entonces triangular las correspondencias que están sobre las imágenes I_k e $I_k - 1$, asumiendo que la matriz de proyección de la cámara en la posición C_{k-1} es igual a $[I|0]$, donde I es la matriz de identidad.

Entonces ¿cuál elección de R y t se debe tomar para asegurar de que se tiene una solución correcta?, la respuesta a esta pregunta subyace en la triangulación descrita anteriormente. Una matriz $[R|t]$ correcta, con la que se triangulen las correspondencias de las dos imágenes, deberá ubicar estos puntos triangulados al frente de ambos fotogramas I_{k-1} e I_k. De esa manera solo queda probar que combinación de R y t cumple con esa restricción planteada anteriormente para la mayor o la totalidad de las correspondencias detectadas entre las dos imágenes.

Tres características importantes del algoritmo de $Longuet - Higgins$ (también conocido

[58]SCARAMUZZA y FRAUNDORFER, óp. cit.

como el algoritmo de los 8 puntos) son las siguientes:

1. Para obtener una estimación apropiada de la matriz esencial E los puntos 3D asociados a las correspondencias 2D-2D que se tienen no pueden ser coplanares.

2. Como el sistema de coordenadas usado para representar las imágenes tiene como origen el pixel ubicado en la esquina superior izquierda de la imagen, a la hora de computar la solución a la ecuación 3.45 existen problemas de estabilidad numérica descubiertos y expuestos por *hartley*[59], este propone una normalización de los puntos característicos de ambas imágenes I_{k-1} e I_k antes de resolver el sistema de ecuaciones. Esta normalización se detallará más adelante en la sección de diseño e implementación (ver capítulo 5).

3. El proceso de estimación robusta (ver 3.4.5) tiene una complejidad exponencial sobre el mínimo número de correspondencias, para este caso el algoritmo de ocho puntos puede ser menos eficiente que otros enfoques.

Si el número de correspondencias 2D-2D que se tienen son menos de 8 ($N < 8$) el problema de la estimación de la matriz esencial E se convierte en un problema no lineal. El mínimo número de correspondencias 2D-2D que se necesitan para estimar la matriz esencial son 5 correspondencias 2D-2D. Lo anterior se deriva del hecho de que los parámetros de movimiento de la cámara desde el instante $k-1$ hasta el instante k tienen 6 grados de libertad (3 grados de libertad para la rotación y 3 grados de libertad para la traslación) pero dado que el factor de escala de la traslación es desconocido a partir de únicamente información 2D-2D, se tienen realmente solo 5 grados de libertad. La solución a partir de 5 correspondencias 2D-2D ($N = 5$) resulta ser más compleja que para $N \geq 8$ debido a que se requiere resolver un conjunto se ecuaciones no lineales.

Nister et al.[60] propusieron un algoritmo eficiente y robusto para la estimación de la matriz esencial a partir de cinco ($N = 5$) correspondencias 2D-2D. Este algoritmo se ha convertido en el algoritmo estándar para la estimación 2D-2D en presencia de *outliers*[61]. Un interrogante que surge de manera natural es: ¿Porque preferir el algoritmo de los cinco puntos de *Nister* sobre el algoritmo de los ocho puntos de Longuet-Higgins, teniendo en cuenta que es un algoritmo más complicado? lo anterior se puede explicar al tener en cuenta que el tiempo de ejecución de una de las etapas de la odometría visual conocida como estimación robusta (sección 3.4.5) es dependiente del número de correspondencias que se tenga, por

[59]Hartley, óp. cit.
[60]NISTER, D. Et. Al. Óp. cit.
[61]SCARAMUZZA y FRAUNDORFER, óp. cit.

lo tanto, si se desea tener un desempeño eficiente en tiempo real es necesario reducir el número de características a las mínimas posibles.

Como se explicó con anterioridad no se puede establecer el factor de escala de la traslación a partir de únicamente información 2D-2D, pero un sistema de odometría visual requiere establecer dicho factor para lograr determinar la trayectoria real de la cámara. La estimación de este factor de escalar se puede realizar principalmente de tres maneras[62]:

- Por medio de la restricción trifocal entre tres imágenes de perspectiva.
- Con el uso de la triangulación se pueden mantener dos conjuntos de puntos 3D emparejados entre sí (W_{k-1} y W_k), para lo anterior es necesario emparejar características en al menos tres fotogramas 2D, usando un par de puntos de W_{k-1} y el par correspondiente en W_k se puede establecer el factor de escala de la traslación como:

$$r = \frac{\|W_{k-1,i} - W_{k-1,j}\|}{\|W_{k,i} - W_{k,j}\|} \tag{3.48}$$

- Usando un método estadístico explicado en la sección 5.6.1

Debido a la presencia de *outliers* es necesario realizar varias estimaciones del factor de escala de la traslación (r) y tomar la media (o la mediana) de todas las estimaciones hechas de r.

El algoritmo sugerido por Scaramuzza et al.[63] para realizar odometría visual a partir de correspondencias 2D-2D se presenta a continuación:

Algoritmo 3.3 :Método de estimación 2D-2D

1: **loop**
2: Capturar el nuevo fotograma I_k.
3: Extraer y Emparejar características a partir de I_{k-1} e I_k.
4: Calcular la matriz esencial E a partir de los conjuntos de características calculados(F_{k-1}, F_k).
5: Extraer la matriz de rotación R y el vector de traslación t a partir de la matriz esencial E .
6: Computar el factor de escala para la traslación y escalar t como corresponde.
7: Estimar el movimiento (T_k) entre el instante $k-1$ y el instante k a partir de R y t.
8: $C_k = T_k C_{k-1}$ ▷ Concatenar T_k a la estimación de la trayectoria de la cámara
9: **end loop**

A continuación se presentan algunas de las características más importantes de la estimación del movimiento por medio de correspondencias 2D-2D:

[62] Ibíd.
[63] Ibíd.

- El método 2D-2D se reconoce como el método que brinda estimaciones de movimiento más exactas debido principalmente a que evita el uso de la triangulación[64], la cual es utilizada por los métodos 3D-2D y 3D-3D.
- El método 2D-2D tiene un tiempo de respuesta más alto que los métodos 3D-2D y 3D-3D. Lo anterior ocurre principalmente porque el número de correspondencias mínimas que necesita el método 2D-2D para estimar el movimiento es mayor que el número de correspondencias mínimas que requieren los métodos 3D-2D y 3D-3D. Esta situación puede limitar el uso de la estimación 2D-2D en algunos escenarios.
- El factor de escala de la traslación es desconocido en la estimación 2D-2D.
- El algoritmo estándar para la estimación $2D-2D$ es el propuesto por *Nister* et al.[65] Este algoritmo es conocido como el algoritmo de los cinco puntos.

Método 3D - 2D

Bajo este enfoque el conjunto de características F_{k-1} (i.e. el conjunto de características determinado en el instante de tiempo $k-1$) debe estar especificado en 3D y el conjunto F_k (i.e. el conjunto de características correspondientes en el instante de tiempo k) debe estar en 2D. A diferencia del enfoque 3D-3D que se puede implementar únicamente con visión estereoscópica, este método se puede implementar con visión estereoscópica y con visión monocular. Si se quiere implementar con visión estereoscópica se usa la triangulación para especificar el conjunto F_{k-1} en 3D y para el conjunto F_k se utiliza únicamente la información proveniente de un fotograma de una de las cámaras. Para usar este método de estimación en visión monocular se requiere hacer correspondencia entre tres fotogramas (I_{k-2}, I_{k-1}, I_k). Con los fotogramas I_{k-2} e I_{k-1} se triangula el conjunto de características F_{k-1} y con la información del fotograma I_k se determina el conjunto de características F_k.

El problema bajo esta perspectiva se define de manera similar a los otros métodos de estimación de movimiento, es decir, se define como un problema de mínimos cuadrados. La estimación 3D-2D busca minimizar el error de reproyección en la imagen, en otros términos, busca encontrar la transformación de cuerpo rígido T_k que minimiza la distancia euclidiana entre F_k^i y P_{k-1}^i, donde F_k^i representa la posición de la característica i-ésima en el fotograma actual (I_k) y P_{k-1}^i representa la reproyección del punto F_{k-1}^i, especificado en 3 dimensiones, en el fotograma I_k de acuerdo a la transformación T_k[66]. Nótese que tanto

[64] Ibíd.
[65] NISTER, D. Et. Al. Óp. cit.
[66] SCARAMUZZA y FRAUNDORFER, óp. cit.

F_k^i como P_{k-1}^i están especificados en 2D. La fórmula 3.49 representa la descripción dada del problema de la estimación 3D-2D.

$$\arg\min_{T_k} \sum_i \|F_k^i - P_{k-1}^i\|^2 \tag{3.49}$$

La minimización planteada en la ecuación 3.49 es equivalente a un problema bien conocido en el campo de la visión por computador y que es conocido como el problema PnP (*perspective from n points*). El problema PnP se define de la siguiente manera: Dado un conjunto de puntos 3D de interés expresados en un marco de referencia y sus respectivas proyecciones 2D en un fotograma capturado por una cámara, encontrar la transformación relativa entre la cámara y el origen del marco de referencia[67], es decir, encontrar la traslación y la rotación que llevo a la cámara desde el origen del marco de referencia hasta la posición donde capturó el fotograma en cuestión. Este problema se ilustra en la figura 3.8[68].

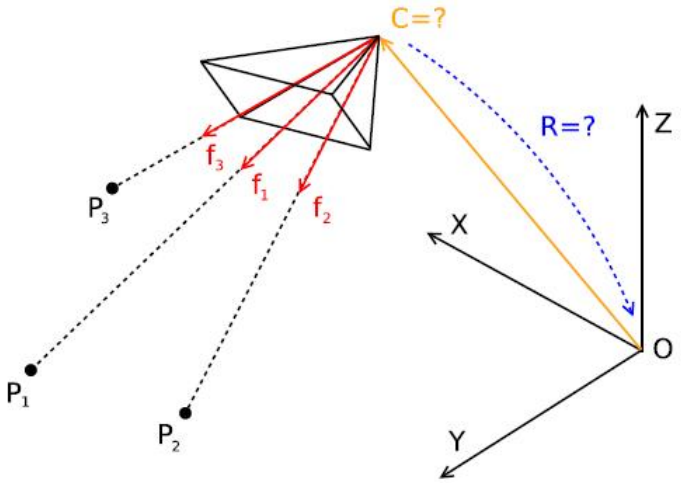


Figura 3.8: problema PnP [69].

El problema PnP es más interesante cuando $n < 6$, principalmente porque cuando $n \geq 6$ el problema descrito en la fórmula 3.49 se convierte en un problema de minimización lineal y tiene una solución única usando el algoritmo "transformación lineal directa"[70].

[67] Dawei Leng y Weidong Sun. «Finding all the solutions of PnP problem». En: *Imaging Systems and Techniques, 2009. IST '09. IEEE International Workshop on.* 2009, págs. 348-352.

[68] L. Kneip, D. Scaramuzza y R. Siegwart. «A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation». En: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* 2011, págs. 2969-2976.

[69] (ibíd.)

[70] Hartley y Zisserman, óp. cit.

Cuando $n < 6$ el problema PnP se convierte en un problema no lineal[71] y su solución es más compleja debido a que normalmente se determinan múltiples soluciones. El número de correspondencia 3D-2D mínimas para tener un número finito de soluciones son 3 correspondencias[72]. Con 3 correspondencias se pueden obtener hasta 4 estimaciones de la transformación T_k en 3.49, esta situación puede ser resuelta usando una o más correspondencias adicionales. El problema PnP cuando $n = 3$ se conoce como P3P y es muy relevante para la odometría visual debido a que entre menos correspondencias se necesiten para determinar una solución, más rápida y eficiente será la ejecución de la etapa de eliminación de *outliers* descrita en la sección 3.4.5. Utilizar un mínimo número de correspondencias permite un desempeño más eficiente en tiempo real.

Las soluciones al problema P3P son esencialmente de dos tipos[73]:

- Métodos de forma cerrada: Bajo esta perspectiva generalmente el problema se reduce a una ecuación polinómica. Las soluciones a esta ecuación polinómica se usan para encontrar todas las estimaciones de movimiento válidas.
- Métodos iterativos: Estos métodos funcionan minimizando de manera iterativa una función de costo definida.

El inconveniente de los métodos de forma cerrada es que generalmente son numéricamente inestables[74], debido a que se generan polinomios de altos grados. Por otro lado, los métodos iterativos tiene el inconveniente de que solo pueden encontrar una solución al tiempo y también que dependen de una aproximación inicial que debe ser provista como entrada del método, en muchas ocasiones, la calidad de la estimación depende de que tan buena es la aproximación inicial.

Algunos ejemplos de soluciones de forma cerrada para el problema P3P son: Kneip et al.[75], Fischler y Bolles[76] y Quan y Lan[77]. Algunos ejemplos de soluciones que usan métodos iterativos son: Chien-Ping Lu et al.[78] y Dementhon and Davis[79].

[71] Leng y Sun, óp. cit.

[72] Kneip, Scaramuzza y Siegwart, óp. cit.

[73] Leng y Sun, óp. cit.

[74] Algoritmo numéricamente inestable: algoritmo en el cual cualquier error en el procesamiento aumenta a medida que el algoritmo se ejecuta.

[75] Kneip, Scaramuzza y Siegwart, óp. cit.

[76] Martin A. Fischler y Robert C. Bolles. «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography». En: *Commun. ACM* 24.6 (), págs. 381-395.

[77] Long Quan y Zhongdan Lan. «Linear N-point camera pose determination». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21.8 (1999), págs. 774-780.

[78] C-P Lu, Gregory D. Hager y Eric Mjolsness. «Fast and globally convergent pose estimation from video images». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.6 (2000), págs. 610-622.

[79] Daniel F Dementhon y Larry S Davis. «Model-based object pose in 25 lines of code». En: *International journal of computer vision* 15.1-2 (1995), págs. 123-141.

Scaramuzza et al. proponen el algoritmo 3.4 para la estimación del movimiento 3D - 2D. Este algoritmo es independientemente del método que se escoja para resolver el problema PnP.

Algoritmo 3.4 :Método de estimación 3D-2D

1: Capturar dos fotogramas I_{k-2},I_{k-1}.
2: Extraer y emparejar características entre los fotogramas.
3: Triangular las características a partir de I_{k-2} y I_{k-1} para determinar el conjunto F_{k-1} inicial.
4: **loop**
5: Capturar un nuevo fotograma I_k.
6: Extraer las características de I_k (F_k) y emparejarlas con respecto a las características de I_{k-1}.
7: Calcular la posición de la cámara (PnP) a partir de correspondencias 3D-2D, es decir, a partir de F_{k-1} y F_k.
8: Triangular todas las nuevas características entre I_k y I_{k-1}.
9: **end loop**

A continuación se presentan algunas características importantes de la estimación 3D-2D:

- De acuerdo a *Nister* et al.[80], la estimación de movimiento usando el método 3D-2D es más adecuada que usando el método 3D-3D debido a que en 3D-2D (fórmula 3.49) se minimiza el error de reproyección y en 3.36 se minimiza el error de posición de las características en 3D (fórmula: 3.36).
- El enfoque 2D-2D brinda estimaciones más exactas que el enfoque 3D-2D debido principalmente a que el método 2D-2D evita la triangulación de las características.
- De acuerdo a Scaramuzza et al.[81], en la práctica es más utilizado el enfoque 3D-2D que el 2D-2D debido a su veloz desempeño. La velocidad del método 3D-2D se relaciona con que este método solo requiere 3 correspondencias para determinar el movimiento, mientras que el método 2D-2D requiere 5 correspondencias y como se verá en la sección 3.4.5 el número de correspondencia necesarias para estimar el movimiento determina el tiempo de respuesta de la etapa de eliminación de *outliers*.

3.4.5. Estimación robusta

La eliminación de estimaciones erróneas (*outliers*) es un paso crítico en el desarrollo de sistemas de odometría visual, debido a el hecho de que muchas veces el emparejamiento o

[80]NISTER, D. Et. Al. Óp. cit.
[81]SCARAMUZZA y FRAUNDORFER, óp. cit.

rastreo de características presenta errores. Varias causas se han atribuido a este fenómeno, entre ellas se encuentran movimientos bruscos, ruido dentro de la imagen, etc. Se han desarrollado algunas metodologías para hacerle frente a este problema.

RANSAC

RANSAC(*Random Sample Consensus*) es la metodología mas usada para la eliminación de observaciones erróneas(*outliers*) para el caso de una estimación de movimiento. *RANSAC* representa una alternativa esencialmente diferente a otras alternativas (e.g. Estimación por mínimos cuadrados) para encontrar un modelo descriptivo que se ajuste de manera óptima a un conjunto de datos. *RANSAC* se caracteriza por buscar estimar el modelo óptimo a partir de únicamente datos correctos (*inliers*), así se busca el modelo descriptivo que sea consistente con la mayor cantidad de datos posibles, en contraste, otros algoritmos de estimación parten de la idea de que todos sus datos son correctos y que deben encontrar un modelo que tenga en cuenta a todos los datos.[82].

Algoritmo 3.5 :Algoritmo de *RANSAC* estándar

1: Sea A el conjunto de correspondencias resultado de la extracción y emparejamiento de características de las imágenes I_{k-1} e I_k.
2: **loop**
3: Escoger aleatoriamente un subconjunto de A con nombre A' y con cardinalidad s.
4: Estimar un modelo de movimiento con A'.
5: Calcular la medida de error E de cada una de las correspondencias en $A - A'$(correspondencias restantes) asociada al modelo de movimiento calculado a partir de A'.
6: Sea ND el número de correspondencias en $A - A'$ tal que $|E| \leq d$.
7: **loop** hasta el máximo número de iteraciones N.
8: **end loop**
9: El subconjunto A' cuyo ND sea el mayor de todos, se escogerá como el modelo de movimiento correcto.

s representa el tamaño del mínimo subconjunto de puntos característicos con el que se puede construir un modelo de movimiento, en algunos casos con ocho puntos y en otros con cinco. N representa el máximo número de iteraciones del algoritmo de *RANSAC* que deben ejecutarse para encontrar una estimación robusta del movimiento. El número de

[82]Martin A. Fischler y Robert C. Bolles. «Random Sample Consensus: A Paradigm for Model Fitting with Applicationsto Image Analysis and Automated Cartography». En: *Communications of the ACM* 24.6 (1981), págs. 381-395.

iteraciones N puede calcularse con bastante exactitud usando la siguiente expresión:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{3.50}$$

Donde ϵ es el porcentaje de puntos característicos mal detectados o mal emparejados que darán paso a una estimación errónea(*outliers*) y p es el porcentaje de encontrar puntos característicos bien detectados y bien emparejados(*inliers*) con los que se calculará una estimación de movimiento correcta.

El tamaño s del conjunto A' se convierte en un cuello de botella para el *RANSAC* ya que el número de iteraciones N del algoritmo es exponencial con respecto a s. El escoger un valor de s puede afectar tanto el tiempo de ejecución como la calidad de la estimación del algoritmo. Tanto el nivel de calidad de la estimación que haga *RANSAC* como el tiempo de ejecución es directamente proporcional al tamaño de s.

En algunos trabajos[83] se han mostrado modelos veloces en donde el tamaño s es de 1, es decir, con un solo punto característico bastaría para hacer una estimación del movimiento, lo cual representa una ganancia en tiempo de ejecución grande, sin embargo las estimaciones de movimiento realizadas por este tipo de sistemas han resultado ser pobres[84]. De esa manera algunos autores recomiendan que si el tiempo de respuesta no es un componente crítico en la implementación, puede usarse un sistema cuyo tamaño s sea grande, dando así una estimación de movimiento más acertada[85]. Las conclusiones de *Nister* et al.[86] muestran que con $s = 5$ se han obtenido buenos resultados en el caso de que un sistema de odometría visual se componga solo de cámaras monoculares. También con un valor de $s = 3$ pueden obtenerse buenos resultados si se dispone de visión estereoscópica.

Otra metodología alterna al *RANSAC* se conoce como el *Histogram Voting* que puede aplicarse en metodologías en donde el movimiento puede estimarse con un solo parámetro[87], este modelo ha resultado ser eficiente, intuitivo y simple, además de que solo requiere de una iteración del algoritmo de estimación del movimiento.

Como se dijo anteriormente el número de iteraciones de *RANSAC* incrementa exponencialmente con respecto al número de puntos s que se tengan en cuenta para la estimación de movimiento. En diferentes trabajos se ha buscado solucionar este problema, buscando la optimización del *RANSAC* por diferentes metodologías.

[83]SCARAMUZZA, FRAUNDORFER y SIEGWART, óp. cit.
[84]Ibíd.
[85]SCARAMUZZA y FRAUNDORFER, «*Visual Odometry [Tutorial part II]*».
[86]NISTER, D. Et. Al. Óp. cit.
[87]SCARAMUZZA, FRAUNDORFER y SIEGWART, óp. cit.

RANSAC Preventivo,*Preemptive RANSAC*

Preemptive RANSAC[88] es una metodología basada en *RANSAC* cuyo objetivo es hacer de este último un proceso muchísimo mas eficiente y que tenga un tiempo de respuesta menor, dadas las exigencias de un sistema de odometría visual que funciona en tiempo real. Esta metodología desarrollada por *Nister* et al. busca entonces no usar el *RANSAC* como un simple contador de puntos característicos que NO se ajustan o que SI se ajustan a un modelo de movimiento, si no dejarlo en términos de una función de probabilidad o una función de costo bayesiana, para lograr más eficiencia.

En este trabajo[89] *Nister* et al. plantea que el *RANSAC* estándar (ver 3.4.5) se puede ver como una metodología en profundidad, es decir, primero se calculan todas las correspondencias que se ajusten a la hipótesis de movimiento actual antes de seguir con la siguiente hipótesis. *Nister* en dicho trabajo propone un esquema diferente al mencionado anteriormente; este nuevo esquema es llamado esquema en anchura, donde primero se generan todas las hipótesis de movimiento con subconjuntos de s puntos del conjunto de todas las correspondencias. s representa la mínima cantidad de correspondencias con las que se puede construir un modelo de movimiento como se explicó anteriormente. Luego de que se tienen todas las posibles hipótesis se procede entonces a probar cada una de las hipótesis, hasta encontrar la mejor hipótesis de movimiento. Esta demostrado en el trabajo de *Nister* que esta metodología es muy superior a otras propuestas en otros trabajos.

El esquema de *RANSAC* propuesto por *Nister* et al. esta compuesto por los siguientes elementos:

- Asumir que se tiene un numero de observaciones $o = 1, ..., N$, un número de hipótesis $h = 1, ..., M$. Las observaciones son emparejamientos de un par de puntos entre dos fotogramas, por otro lado las hipótesis son estimaciones del movimiento usando algún algoritmo específico.
- Una función $\rho(o, h)$ que da como resultado un escalar que indica que tan correcta es una hipótesis de movimiento dada una observación o, este escalar se conoce como probabilidad logarítmica o *log-likelihood* y se denota como $L(h)$.

$$L(h) = \sum_{o=1}^{N} \rho(o, h) \tag{3.51}$$

[88] David Nistér. «Preemptive RANSAC for live structure and motion estimation». En: *Machine Vision and Applications* 16.5 (2005), págs. 321-329.
[89] Ibíd.

$$\rho(o,h) = \prod_{i=1}^{10}(1+u_i) \tag{3.52}$$

Donde: u_i representa el error de reproyección(ver 3.49) para una correspondencia establecida de dos puntos entre dos fotogramas.

El esquema de *RANSAC* preventivo toma un numero determinado de observaciones al azar, luego para todas las hipótesis de h se calcula que tan correcta es cada hipótesis dada la observación i-ésima por medio de $L_i(h)$. En un siguiente paso se observan las hipótesis que tuvieron un valor de $L_i(h)$ mayor y luego se toman estas hipótesis para usarlas en una siguiente etapa. La etapa siguiente consta en mirar la $i+1$ observación y repetir el cálculo anterior, descartando en cada etapa a las hipótesis con menor valor de $L(h)$ hasta que se agoten el número de observaciones. Para el momento que se agoten las observaciones debe existir por lo menos una hipótesis que haya tenido un valor grande de $L(h)$ para con todas las observaciones. Esta última hipótesis se escoge como la ganadora y como la hipótesis de movimiento correcta. Una abstracción de este proceso se muestra en la imagen 3.9, cada una de las flechas indica una transición entre los pasos del algoritmo. el eje coordenado vertical representa la secuencia de observaciones, mientras el horizontal representa las hipótesis. Nótese que a medida que el algoritmo avanza, se van descartando hipótesis(representadas en cuadrados pequeños grises) hasta llegar al número final de observaciones.

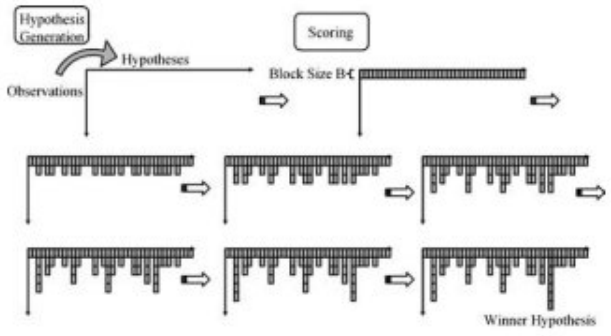


Figura 3.9: abstracción del proceso de *RANSAC* preventivo [90]

A continuación se presenta un esquema del algoritmo planteado por *Nister* et al. :

donde $f(i)$ es una función que calcula para el instante i el número de hipótesis de movimiento que deben tenerse para dicho momento, las hipótesis que se escogen para seguir en el momento i deben de ser las que hasta el momento tenga un valor grande de similitud

[90] (ibíd.)

Algoritmo 3.6 :Algoritmo de *RANSAC* preventivo

1: Escoger aleatoriamente un conjunto de observaciones.
2: Generar las hipótesis indexadas por $h = 1, ..., f(1)$
3: Calcular las probabilidades logarítmicas $L_1(h) = \rho(1, h)$ para $h = 1, ..., f(1)$. asignar a $i = 2$.
4: **loop**
5: Reconsiderar las hipótesis para que el nuevo rango de hipótesis $h = 1, ..., f(i)$ contenga las mejores $f(i)$ hipótesis restantes de acuerdo a $L_{i-1}(h)$.
6: Si $i > N$(N es el número de observaciones), extraer la mejor hipótesis restante que se encuentre y seleccionarla como la hipótesis correcta. si $i < N$, calcular $L_i(h) = \rho(i, h) + L_{i-1}(h)$ para $h = 1, ..., f(i)$, aumente i y vaya al paso 4.
7: **end loop**

con las observaciones o más específicamente un valor grande de $L_{i-1}(h)$ por encima de un umbral determinado.

De esta manera se tiene un algoritmo para la remoción de *outliers* muchísimo mas eficiente ya que el número de hipótesis y de observaciones a analizar puede ser determinado por los desarrolladores permitiendo una manipulación más directa sobre los parámetros del algoritmo. A pesar de que existe el peligro de que el hecho de escoger observaciones aleatorias pueda resultar en la elección observaciones afectadas con ruido, *Nister* plantea que dado el número tan grande de observaciones entre dos fotogramas la probabilidad de que la mayoría de observaciones consideradas aleatoriamente sean observaciones correctas(sin ruido) es muy alta. Mientras que en el *RANSAC* estándar (ver 3.4.5), para cada hipótesis se calculaba su similitud con las N observaciones antes de pasar a la siguiente, con el esquema de *Nister* et al. se puede hacer este proceso para todas las hipótesis en una sola etapa, mejorando por mucho el desempeño de un sistema de odometría visual en tiempo real, sin embargo debe dejarse claro que el *RANSAC* estándar es mucho más robusto en el sentido que tiene en consideración todas las observaciones posibles.

3.5. Estado del arte

El trabajo desarrollado por *David Nister, Oleg Naroditsky y James Bergen*[91] muestra un enfoque para hacer odometría visual el cual usa como imagen de entrada una cámara monocular o una cámara estereoscópica. El trabajo de *Nister* et al. marca un precedente en el estudio de la odometría visual ya que reactiva la investigación en esta área luego de un largo periodo sin hitos importantes[92].

[91] NISTER, D. Et. Al. Óp. cit.
[92] SCARAMUZZA y FRAUNDORFER, «*Visual Odometry [Tutorial]*».

En resumen, el sistema desarrollado por *Nister et al.* se puede describir bajo la siguiente linea de procedimientos. Primero, se recibe la imagen, ya sea por medio de un sistema estereoscópico o monocular, segundo, sobre la imagen recibida se aplica una extracción de características(*feature extraction*), en tercer lugar, se aplica un emparejamiento de características(*feature matching*) entre cada fotograma o *frame* de la imagen y por último, se hace una estimación del movimiento(*robust estimation*) de los puntos característicos entre cada fotograma.

Feature Extraction

Este procedimiento consiste en tomar cada fotograma de la imagen de video, luego, para cada fotograma se detectan unos puntos característicos (*feature points*) los cuales el sistema de odometría visual usará para hacer la estimación del movimiento de la cámara empotrada en algún lugar del vehículo en marcha.

Estos puntos característicos se denominan *Harris Corners* (detallado en la sección 3.4.2, más específicamente se busca detectar las esquinas en cada fotograma, una esquina puede definirse como el lugar o pixel de una imagen en donde se intersectan un par de bordes. Las esquinas detectadas han demostrado ser buenos puntos característicos, ya que estos se mantienen estables bajo circunstancias en las que el movimiento de la cámara no varía mucho entre cada fotograma.

Esta etapa de extracción de características se implementa con algunas optimizaciones desarrolladas en *MMX*, el cual es un conjunto de instrucciones para linea *pentium* de los microprocesadores *intel*.

Feature Matching

En el paso anterior, para cada fotograma de la imagen de video se encontraron las esquinas que representan unos puntos característicos sobre cada fotograma. Para que el sistema de odometría visual pueda hacer una estimación del movimiento de la cámara, debe conocer para cada punto característico(x_k, y_k) ubicado en un fotograma F_k en un instante k su correspondiente posición(x_{k-1}, y_{k-1}) en el fotograma F_{k-1} para el instante $k-1$, este proceso se llama emparejamiento. Conociendo estas posiciones el sistema de odometría visual podrá estimar el movimiento de dicho punto en la parte de *Robust Estimation*.

El proceso de emparejamiento parte del principio que el movimiento del vehículo va a ser puramente traslacional y que los cambios entre un *frame* F_{k-1} y F_k van a ser casi despreciables. Para que el emparejamiento sea posible debe encontrarse para un punto característico (x_k, y_k) su símil en el fotograma anterior. Esto último se logra de la siguiente usando *NCC(Normal Cross Correlation) y MCC(Mutual Consistency Check)* ambos

especificados en la sección 3.4.3:

1. Para cada punto característico (x_k, y_k) se define un panel de 11*11 pixeles, este panel representará el espacio de búsqueda en el fotograma F_{k-1} , asumiendo como centro de ese panel dentro de la imagen el punto (x_k, y_k).
2. Para cada punto característico en el fotograma F_k se definirá un panel del mismo tamaño y para cada panel se precalculan los valores de NCC especificados en 3.4.3.
3. Se aplica MCC para establecer correspondencias entre dos fotogramas I_k e I_{k-1}.

Cada panel de 11*11 se representa como un vector de tamaño de 128 bytes y se hacen algunas optimizaciones en MMX para hacer los cálculos respectivos.

Robust Estimation

Dependiendo del tipo de cámara o cámaras que el vehículo posea para el sistema de odometría visual, se puede usar dos enfoques para estimar el movimiento de la cámara luego de que se tienen emparejados los puntos característicos de un fotograma actual con los de un fotograma anterior. El primer enfoque es el enfoque monocular y el segundo enfoque es el estereoscópico explicados en secciones anteriores.

Para el trabajo de *Nister et al.* el enfoque monocular se resume en:

1. Rastrear puntos característicos en un cierto número de fotogramas, usando *Preemptive RANSAC*(ver sección 3.4.5) para obtener grupos de características bien emparejados y usar el algoritmo de los cinco puntos(*5-points algorithm*3.4.4) para estimar el movimiento relativo.
2. Construir puntos en 3D para cada rastreo con la primera y última observación.
3. Rastrear otro número de fotogramas. Calcular el movimiento de la cámara con los puntos en 3D conocidos.
4. Volver al paso 1.

Por otro lado el enfoque estereoscópico se resume en:

1. Triangular los puntos característicos emparejados en puntos 3D con la imagen de la izquierda y de la derecha.

2. Rastrear estos puntos característicos en un número determinado de fotogramas y estimar el movimiento relativo.

3. Reconstruir los puntos 3D.

4. volver al paso 2.

La ventaja del sistema estereoscópico es que permite calcular el movimiento de la cámara con una escala conocida, en contraste con el sistema monocular. El sistema monocular usa el algoritmo de los cinco puntos para estimar el movimiento del vehículo el cuál es un enfoque complejo en su implementación.

Experimentos y resultados de *Nister et al.*

En La imagen 3.10 se muestra del montaje que fue desarrollado para la implementación del sistema de odometría visual.

Figura 3.10: Montaje de la implementación de *Nister et al.*[93].

- Fue usado un montaje de cámaras estereoscópica, con distancias entre puntos epipolares (*baseline*) de 28 cm.
- Se uso una resolución de 720*420.

[93] (NISTER, D. Et. Al. Óp. cit.)

- La tasa de fotogramas por segundo usada fue de 13 Hz.
- Cada cámara estereoscópica fue equipada un campo de visión horizontal de 50°.

Para poder hacer una validación del sistema y observar si la respuesta del sistema era la adecuada y si las estimaciones del movimiento eran correctas, se tomaron como medidas de referencia las mediciones entregadas por una unidad de medida inercial y un *GPS* diferencial.

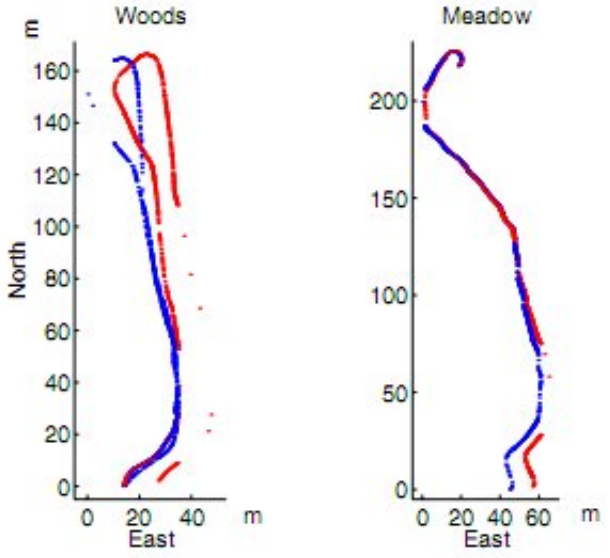


Figura 3.11: Resultados encontrados por *Nister et al.*(1) [94].

En La imagen 3.11 se encuentra la trayectoria de un recorrido calculada por el sistema de odometría visual(en rojo) y la trayectoria calculada por el *GPS* diferencial[95]. Cada una de las gráficas corresponde a un ambiente diferente, en la gráfica de la izquierda se desarrolla la trayectoria en un bosque mientras que al gráfica de la derecha se desarrolla sobre un prado. Puede notarse que la diferencia entre los dos enfoques es poca.

En la imagen 3.12 se aprecia otro de los experimentos desarrollados para probar el funcionamiento del sistema de odometría visual, a simple vista se puede ver que el camino reconstruido por el sistema de odometría visual fue altamente similar al reconstruido por el *GPS* diferencial[97]. Para esta prueba el vehículo hizo un recorrido de tres circuitos de aproximadamente 20 metros cada uno. El error en la distancia entre los puntos finales de cada recorrida fue de 4.1 metros.

[94] (ibíd.)
[95] Ibíd.
[96] (ibíd.)
[97] Ibíd.

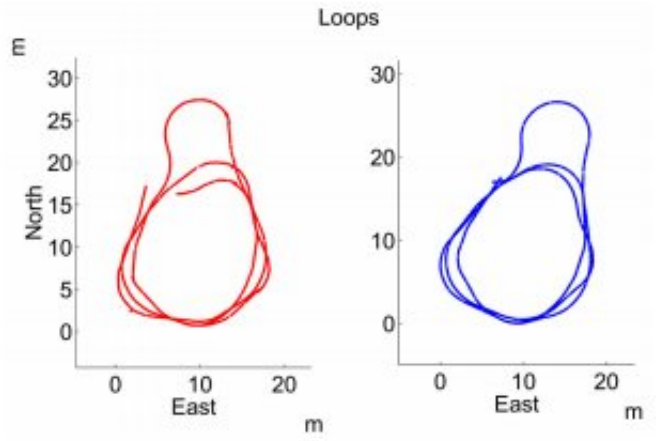


Figura 3.12: Resultados encontrados por *Nister et al.*(2) [96].

Run	Frames	DGPS(m)	VisOdo(m)	% error
Loops	1602	185.88	183.90	1.07
Meadow	2263	266.16	269.77	1.36
Woods	2944	365.96	372.02	1.63

Figura 3.13: Resultados encontrados por *Nister* (3)[98].

En la imagen 3.13 se puede observar el cálculo del recorrido total hecho por el *GPS* diferencial y por el sistema de odometría visual, en la última columna se puede ver el error del segundo sistema con respecto al primero. Cada fila de la tabla representa los diferentes ambientes por los cuales se experimentaron[99].

La publicación de este trabajo generó en la comunidad científica y de ingeniería un interés por abordar este tema de manera más profunda y más seria a medida que los vehículos autónomos comenzaban a tomar fuerza gracias a las competencias organizadas por *DARPA*. Diferentes investigadores han tratado el tema de la odometría visual desde diferentes perspectivas, usando incluso sensores que si bien hacen recolección de imágenes, fueron diseñados para otros propósitos distintos de la odometría visual, dispositivos como por ejemplo *Kinect* de la plataforma de videojuegos *X-box 360*.

El trabajo de *Fiala y Ufkes*[100] describe una implementación de un sistema de odometría

[98] (ibíd.)
[99] Ibíd.
[100] M FIALA y A. UFKES. «*Visual Odometry Using 3-Dimensional Video Input*». En: *Computer and*

visual usando un sensor *Kinect*, hecho que difiere de las implementaciones tradicionales que usan sensores de imágenes comunes, además de usar un sensor de profundidad junto con imagen de video. Se evita el uso de visión estereoscópica con el fin de ganar robustez y se aprovecha el video monocular para asociarlo con las mediciones obtenidas del sensor de profundidad. A diferencia de la mayoría de los trabajos realizados sobre el Kinect, Fiala y Ufkes no realizan *SLAM (simultaneous localization and mapping)*, que consiste en una técnica de localización usando las características de las zonas previamente visitadas, sino que usan la información en 3D con el fin de estimar la posición entre *frames*. Conocer un lugar en un mapa es un problema distinto al de la odometría visual, requiere de una gran base de datos de características para poder encontrar las que caracterizan a un lugar en específico.

El sensor Kinect usado en el trabajo de *Fiala y Ufkes*

El sensor Kinect posee dos dispositivos de recepción de imágenes: una cámara que capta colores y un sensor infrarrojo de profundidad, ambos captando a $640 * 480$ y con una frecuencia de $30Hz$. Debido a la distancia entre los lentes de los sensores, se hace necesario una calibración con el fin de poder mapear un punto hallado por un sensor en la imagen captada por el otro, a pesar que dicho mapeo genera un error, no es significativo a la hora de correr el algoritmo de extracción de características.

$$u_{ir} = A.u_{vis} + B$$
$$v_{ir} = C.v_{vis} + D$$

Ecuación que permite mapear puntos de la cámara con puntos del sensor de profundidad, donde A, B, C y D son parámetros escalares de calibración.

La profundidad de un punto de interés o característica encontrada por la cámara es hallada de la siguiente manera:

- Se localiza la característica en la imagen de la cámara (u_{vis}, v_{vis}).
- Convertir dicha posición en una posición en la imagen de profundidad del sensor con la ecuación anterior.
- Tomar el valor de la profundidad de las coordenadas halladas $d(u_{ir}, v_{ir})$ y multiplicarlo por el inverso de la matriz K_{ir} (donde K_{ir} es una matriz que representa el modelo del sensor ajustando su distancia focal)

Robot Vision (CRV), 2011 Canadian Conference on. 2011, págs. 86-93.

$$K_{ir} = \begin{bmatrix} 450 & 0 & 320 \\ 0 & 475 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = K_{ir}^{-1} d(u_{ir}, v_{ir})$$

el resultado representa las coordenadas en 3D en coordenadas de la cámara del punto de interés. (X_c, Y_c, Z_c)

Algoritmo implementado de *Fiala y Ufkes*

SIFT (Scale-invariant feature transform) es un algoritmo para detectar y describir características locales en un imágenes. Fue publicado por David Lowe en 1999[101].

Para cualquier objeto de una imagen, se extrae un conjunto de características de una imagen de entrenamiento. Dichas características permiten identificar al objeto en otras imágenes que pueden contener o no muchos otros objetos, independientemente de los cambios en la escala de la imagen, orientación, ruido o iluminación.

El primer paso consiste en la extracción de los puntos de interés o características de un conjunto de imágenes de referencia, los cuales son almacenados en una base de datos. Cuando se rastrean dichas características en una nueva imagen, se busca en la base de datos con el fin de emparejarlo con una entrada existente basados en el calculo de la distancia euclidiana de cada característica. Los subconjuntos de características que correspondan al patrón buscado en términos de posición, escala y orientación son filtrados para identificar buenas parejas. Un grupo de 3 o mas características que concuerden con un objeto de la base de datos es sujeto a verificaciones del modelo del objeto y a remoción de *outliers*. Se logra gran robustez y eficiencia implementando una tabla *hash* al momento de realizar la transformada de Hough (algoritmo de detección de características). Los emparejamientos que pasen todas las pruebas anteriores pueden ser categorizadas como correctos, con alto grado de seguridad.

Teniendo el conjunto de correspondencias de características en 3D para cada par de *frames* sucesivos, se calcula una matriz de rotación R y de translación T usando pruebas aleatorias de *RANSAC*, donde se usan cuatro características para probar las hipótesis R_h y T_h. El conjunto completo de pares de características en 3D entre *frames* sucesivos se prueban con las hipótesis con el fin de analizar la linealidad de los puntos, con un umbral o *threshold* de 1,5cm para clasificar dicho punto como un *inlier*. La prueba aleatoria de *RANSAC* con

[101] D.G. LOWE. «*Object recognition from local scale-invariant features*». En: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on.* Vol. 2, 1150-1157 vol.2.

el mayor número de *inliers* es escogida para el cálculo de la posición. A pesar de que la implementación permite 6 grados de libertad, se recomienda el uso en ambientes cerrados y planos, para los cuales sólo requieren de 3 grados de libertad.

Experimentos y resultados con el *kinect*

La imagen 3.14 muestra un error en la rotación de uno de los recorridos debido a un ajuste en el umbral establecido para el *RANSAC*. Una prueba en línea recta demostró algunos errores en la orientación (figura 3.15).

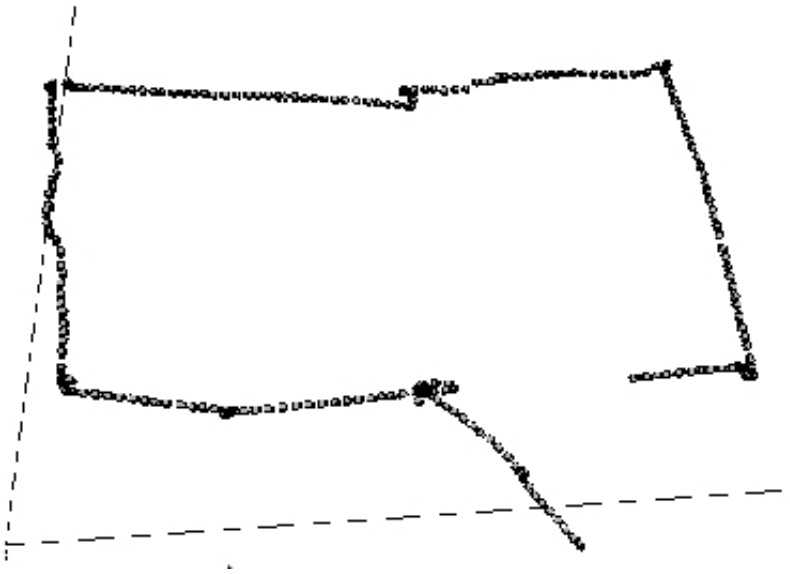


Figura 3.14: Robot en recorrido de 21 metros con error rotacional[102].

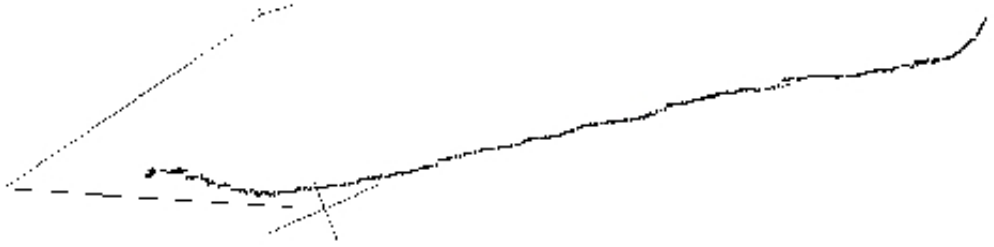


Figura 3.15: Robot en recorrido en línea recta (50 m.)[103].

La mayoría de los errores encontrados en la implementación estuvieron relacionados con la rotación en grados de libertad. En algunas ocasiones, las mediciones reportaban que el robot se encontraba 80 cm por debajo del suelo.

[102] (FIALA y UFKES, óp. cit.)
[103] (ibíd.)

El prototipo demostró un desempeño satisfactorio pese a varias imperfecciones en su sistema. Se propone para futuros trabajos realizar los cálculos del algoritmo *SIFT* en una *GPU* con el fin de acelerar la implementación y permitir odometría visual en tiempo real.

El sensor *Kinect* ayuda a resolver problemas en el cálculo de la profundidad en la escena evitando el uso de cámaras haciendo uso de cámaras estereoscópicas, sin embargo, la odometría visual encara otros desafíos que tienen que ver con la naturaleza del terreno presente en la escena que la cámara registra para hacer odometría.

El documento escrito por *Kurt Konolige, Motilal Agrawal y Joan Sola*[104] presenta los resultados de la implementación realizada; un sistema integrado para localizar un robot móvil en terrenos irregulares, usando odometría visual y sensores de medida inercial.

La tarea de estimar el movimiento se apoya en diferentes sensores, *GPS* e *IMU* como principales, Lamentablemente son muy sensibles a errores; el *GPS* no funciona en entornos cerrados o con muchos árboles y la *IMU* tiende a degradarse si no se hace una pronta corrección. Por tal motivo, la estimación visual del movimiento es un buen método para realizar dicha labor y para complementar los métodos tradicionales.

Teniendo en cuenta el pobre desempeño de los algoritmos tradicionales para la detección de características (*Harris, FAST, SIFT*), el documento presenta un nuevo algoritmo: *CenSurE*, mejorando la estabilidad en entornos tanto interiores como exteriores a un bajo costo computacional. También se realiza la integración con un sensor *IMU* con el fin de reducir el crecimiento de la medición del error angular de la odometría visual.

Existen dos clases principales de técnicas para examinar los cambios inducidos por el movimiento: *optical flow* y seguimiento de características. Mientras *optical flow* analiza el movimiento de los patrones de brillo sobre la imagen entera, el seguimiento de características rastrea una pequeña cantidad de características de imagen a imagen. Los sistemas causales asocian *landmarks* con características usando filtros de Kalman[105], lastimosamente, estos métodos no son viables para largas distancias, pues se hace demandante y poco óptimo la manipulación de gran cantidad de características. *Structure from Motion* es un método de seguimiento de características basado en la correspondencia de las mismas. se requieren 5 puntos para dos *frames* de una cámara y 3 puntos para cámaras estéreo.

Usando una *IMU* con acelerómetro de 3 ejes y una cámara estereoscópica, se pretende determinar la orientación y la posición global de un vehículo para cada par de *frames*. De manera individual para cada par:

[104]KURT KONOLIGE, MOTILAL AGRAWAL y JOAN SOLA. «*Large scale visual odometry for rough terrain*». En: *In Proc. International Symposium on Robotics Research*. 2007.
[105]R. E. KALMAN, óp. cit.

1. Extraer las características de la imagen izquierda.
2. Realizar la correspondencia de las características con la imagen de la derecha usando Estéreo denso.
3. Realizar la correspondencia de las imágenes previas de la izquierda con *ZNCC (zero-mean normalized cross correlation).*
4. Consenso del movimiento estimado usando *RANSAC.*
5. Refinamiento de las coordenadas 3D de los últimos N *frames (Bundle adjustment)*
6. Fusión de resultados con la *IMU*

Es posible reducir el área de búsqueda para la correspondencia de los *frames* tratando de predecir donde buscarlas, con la ayuda de sensores de movimiento del vehículo o robot.

CenSurE detector usado por *Konolige et al.*

Una de las principales dificultades en la odometría visual es la asociación de características, es necesario que permanezcan estables bajo cambios de luz y puntos de vista, adicional a que sean rápidas de calcular. Comúnmente se usa Harris[106], *FAST* o *SIFT*[107]. En ambientes externos, las esquinas son difíciles de detectar, pues tienden a desvanecerse debido a variaciones en la distancia o en la textura. Con el fin de evitar dichos inconvenientes, la implementación relatada usa *center-surround feature*, un área oscura rodeada por una clara o viceversa. Dicha característica es dada por el Laplaciano normalizado de la función Gaussiana (LOG):

$$\sigma^2 \nabla^2 G(\sigma)$$

donde $G(\sigma)$ es el Gaussiano de la imagen a una escala de σ.

El cálculo de LOG se aproxima usando *center-surround Haar wavelets.* La imagen 3.16 muestra un *center-surround* genérico que se aproxima a LOG. El valor $H(x, y)$ es 1 en las zonas blancas y -8 en las oscuras. se realiza una convolución y luego la normalización del área de la *wavelet.*

$$(3n)^{-2} \times \sum_{x,y} H(x, y) I(x, y).$$

[106]HARRIS y STEPHENS, óp. cit.
[107]LOWE, óp. cit.
[108] (KONOLIGE, AGRAWAL y SOLA, óp. cit.)

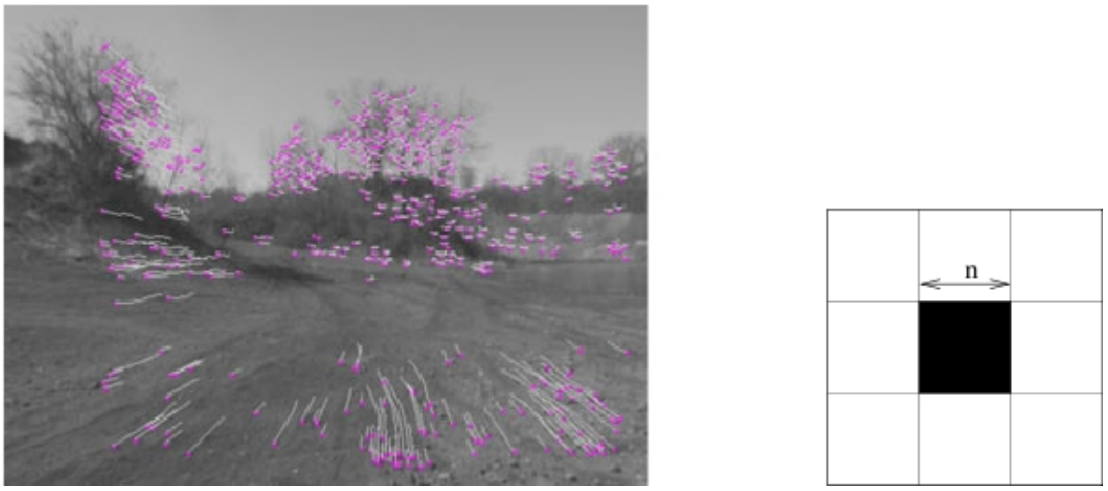


Figura 3.16: Resultado de aplicar *CenSurE*; ejemplo de un *center-surround*[108].

La implementación usa 6 escalas con un tamaño de bloque de $n = [1, 3, 5, 7, 9, 11]$. Luego de realizar dicho cómputo para cada posición y escala, se encuentran los valores extremos comparando cada punto en el espacio de la imagen en 3D con sus 26 vecinos en escala y posición. Su eficiencia es comparable con *FAST* o Harris.

Pruebas del trabajo de *Konolige et al.*

Se realizaron pruebas de los algoritmos más usados para la detección de características con el fin de contrastarlos con *CenSurE*. Las pruebas se realizaron sobre un conjunto de 47.000 imágenes en un trayecto de 150 metros. La imagen 3.17 ilustra el resultado de *CenSurE*, detectando 94 características y realizando un consenso de 44 correspondencias *(inliers)*. La imagen 3.18 muestra los resultados de las pruebas realizadas de dos maneras: la tabla de la izquierda muestra la cantidad de fallos de correspondencia entre *frames* junto con el promedio de distancia de una característica rastreada y la tabla de la derecha muestra el error en metros y en porcentaje al cerrar una trayectoria.

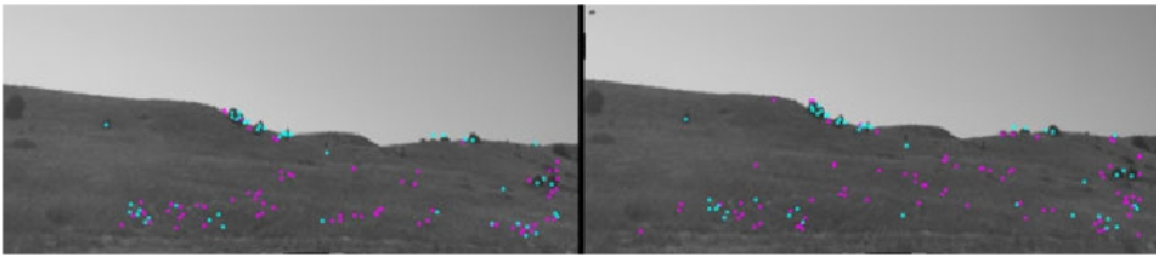


Figura 3.17: Puntos fucsia: características. Puntos azules, correspondencias consideradas correctas[109].

[109] (ibíd.)
[110] (ibíd.)

	Harris	FAST	SIFT	CenSurE
Fail	0.53%	2.3%	2.6%	0.17%
Length	3.0	3.1	3.4	3.8

	Harris	FAST	SIFT	CenSurE
Err	4.65	12.75	14.77	2.92
%	1.55%	4.25%	4.92%	0.97%

Figura 3.18: Tablas comparativas con los diferentes métodos[110].

Integración con una *IMU* y resultados del trabajo de *Konolige et al.*

Se realizó estimación incremental de la pose y un sensor *IMU* como inclinómetro y como apoyo para estimar la precisión angular, usando filtros de *Kalman*[111]. existen sesgos inherentes al sistema en el momento de realizar la medición de las rotaciones, por lo cual se realiza un *Sparse Bundle Adjustment(SBA)*.

La imagen 3.19 ilustra los resultados del recorrido realizado usando el conjunto de imágenes de prueba (izquierda) y las imágenes tomadas en Ft. Carson (Derecha). Tómese como punto de referencia la línea roja, medición realizada con un *RTK GPS* con un error de 10 cm.

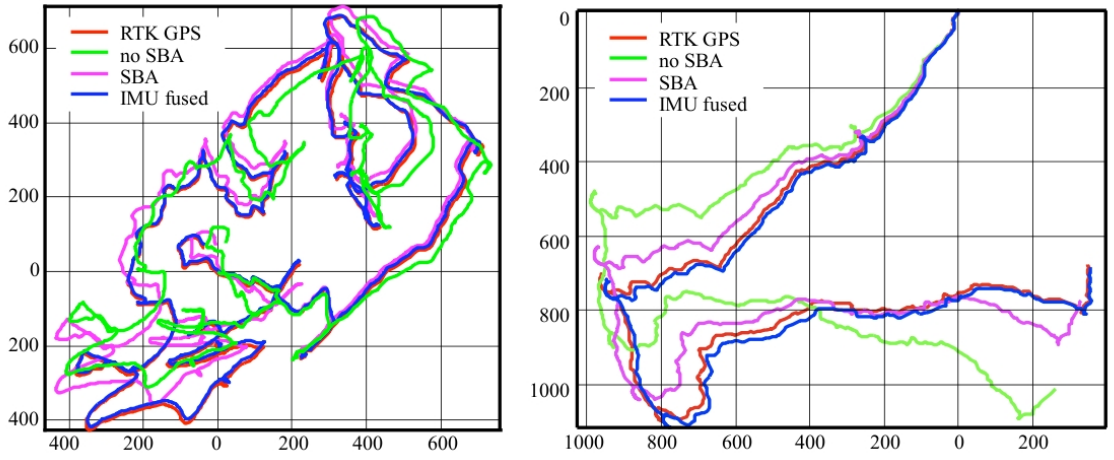


Figura 3.19: Resultados obtenidos en 2 recorridos de 9 Km, 47K *frames* [112].

El error estadístico de la implementación se encuentra por debajo del error en implementaciones tradicionales que usan *IMU/GPS* en rangos hasta los 10 KM. El costo computacional es bastante reducido, realizando todas las operaciones en un procesador Pentium de 2 GHz.

EL problema de la odometría visual cuando la naturaleza del terreno es rural y se está en campo abierto es una dificultad que distintos investigadores han tratado de resolver y en

[111] R. E. KALMAN, óp. cit.
[112] (KONOLIGE, AGRAWAL y SOLA, óp. cit.)

competencias como *DARPA Grand Challenge* se tratan de poner a prueba dichas soluciones. Esta subárea de la odometría visual requiere de diferentes estrategias para poder recuperar la posición del vehículo ya que en ambientes rurales no abundan los puntos de interés como lo hacen en los urbanos, además la distancia a la escena en un ambiente rural o externo(diferente al urbano)puede volverse un problema.

El trabajo de *Joern Rehder, Kamal Gupta, Stephen Nuske, and Sanjiv Singh*[113]describe un sistema de odometría visual que pretender ser utilizado en un vehículo cuyo ambiente de operación es externo. Los principales desafíos que se plantean en esta investigación son:

- Construir un sistema de odometría visual estereoscópico que funcione de manera adecuada cuando la distancia a la escena de interés es grande.
- Localizar el vehículo en un marco de referencia global usando con poca frecuencia las medidas entregadas por el *GPS*.

A continuación se plantean y describen las soluciones planteadas en la investigación para estos dos desafíos.

Odometría visual de largo alcance

La razón por la cual es necesario replantear los algoritmos de odometría visual cuando los puntos de interés se encuentran alejados de la cámara está relacionada con el hecho de que estos puntos distantes inducen errores importantes en la estimación del movimiento. De acuerdo a experimentos realizados por Rehder et al. cuando se usan características alejadas a la cámara el enfoque estereoscópico tiende a subestimar el movimiento, es decir, generalmente se estima un movimiento menor al real. Por tal motivo se desarrolló un mecanismo de corrección que se describe a continuación.

Una de las etapas más importantes en el *pipeline* de un sistema de odometría visual estereoscópico es la etapa de triangulación, esta etapa buscar estimar la posición de un punto en el espacio dadas sus proyecciones en dos o más imágenes. En la práctica la triangulación induce errores en la estimación del movimiento, este error no es gaussiano y ha demostrado tener un *bias* que está relacionado con la distancia de los puntos de interés a la cámara. En este trabajo se describe como calcular el *bias* del sistema de odometría visual en tiempo real (para cada fotograma) y sin utilizar otras fuentes de información (*GPS* , *IMU*, etc.).

[113]JOERN REHDER y col. «*Global pose estimation with limited GPS and long range visual odometry*». En: *ICRA'12*. 2012, págs. 627-633.

Los autores exponen que todos los enfoques de estimación de movimiento para odometría visual estereoscópica sufren del problema de la subestimación del movimiento por cuenta de los puntos de interés alejados. En este trabajo se usó el método de minimización iterativa del error de reproyección como método de la estimación del movimiento, este algoritmo se denotará por la función $g(.)$. La estimación del movimiento se puede representar de la siguiente manera:

$$[R_0, T_0] = g(F^{i-1}, F^i) \tag{3.53}$$

Donde F^i representa el conjunto de características extraídas en el fotograma actual y F^{i-1} representa el conjunto de características extraídas en el fotograma anterior. La estimación del movimiento está representada por $[R_0, T_0]$, donde R_0 representa la rotación estimada y T_0 representa la traslación estimada.

En el artículo se plantea que la estimación $[R_0, T_0]$ tiene un *bias* k que se relaciona con el movimiento verdadero $[R_w, T_w]$ de la siguiente forma:

$$[R_w, T_w] = [R_0, kT_0] \tag{3.54}$$

Para estimar el *bias* se plantea una nueva estimación $[\bar{R}, \bar{T}]$ simulando una cámara estereoscópica en la posición de la estimación inicial $[R_0, T_0]$ y proyectando los puntos triangulados de F^{i-1} sobre el plano de la cámara simulada, obteniendo de esta manera un nuevo conjunto de características simuladas $\bar{F}^i$. Es importante mencionar que durante la proyección sobre el plano de la cámara simulada se agrega un ruido aleatorio γ de una distribución gaussiana. Nótese que a partir de F^{i-1} y de $\bar{F}^i$ se puede generar una nueva estimación $[\bar{R}, \bar{T}]$ así:

$$[\bar{R}, \bar{T}] = g(F^{i-1}, \bar{F}^i) \tag{3.55}$$

El proceso de simular la cámara estereoscópica se repite una cantidad determinada de veces y se calcula la estimación promedio para $[\bar{R}, \bar{T}]$ y a partir de esta se determina el factor de *bias* de la siguiente manera:

$$k = \frac{\|T_0\|}{\|\bar{T}\|} \tag{3.56}$$

Esta estimación del factor de *bias* se considera correcta siempre y cuando se den dos

condiciones:

- El modelo de ruido para los pixeles γ es el adecuado.
- El factor de *bias* k se pueda aproximar por el *bias* de la estimación original $[R_0, T_0]$.

Estimación óptima de la posición con escasa información del *GPS*

Para el problema de la intermitencia en la disponibilidad del *GPS* se plantean los siguientes requerimientos:

- Fusionar las medidas entregadas por el sistema de odometría visual, por una unidad de medida inercial y por el *GPS* (cuando esté disponible) para entregar una estimación de la posición en tiempo real.
- Evitar las discontinuidades en la estimación de la trayectoria del vehículo por cuenta de las intermitencias en el *GPS*.

Para cumplir con estos requerimientos se utilizó un enfoque de estimación global de la posición que usa optimización no lineal sobre un histórico de posiciones del vehículo las cuales están representadas en un grafo. En este grafo los nodos representan posiciones del vehículo en ciertos instantes de tiempo y las aristas entre los nodos representan restricciones impuestas por las mediciones de los sensores.

Las mediciones de los sensores que se convierten en entradas para el algoritmo de estimación de la posición son de los siguientes tipos:

- Mediciones Locales (e.g. odometría visual, giróscopos, etc.)
- Mediciones Globales (e.g. lecturas de *GPS*, acelerómetros, etc.)

Ambos tipos de mediciones son incorporados en la estimación de una manera común.

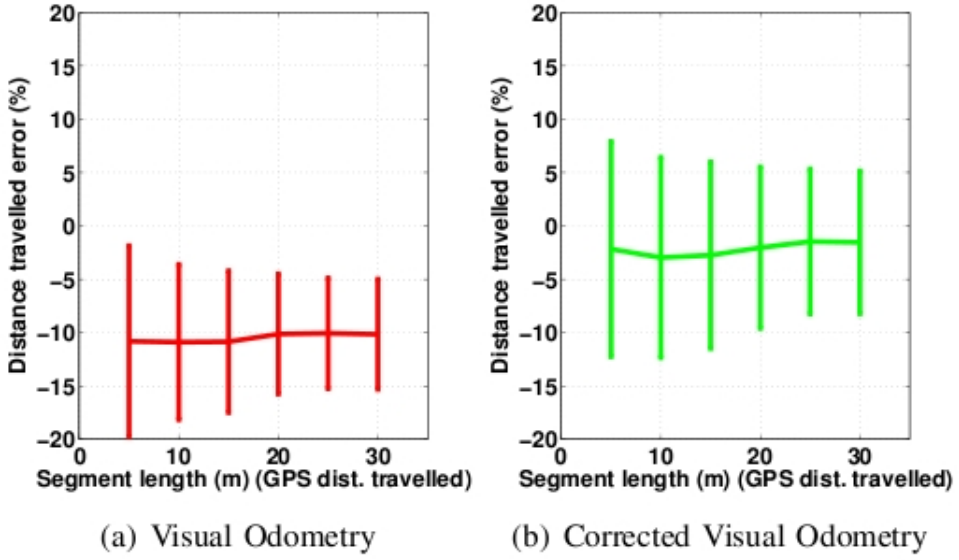
Para la optimización se usó una librería que implementa el algoritmo de *Levenberg–Marquardt*. Con el objetivo de lograr un desempeño en tiempo real se debe limitar el número de posiciones (nodos) que se consideran en el proceso de optimización. La descripción detallada de la formulación del problema de optimización y de la estrategia de reducción de nodos se puede encontrar en la publicación de Rehder et al.[114]

[114] Ibíd.

Resultados de *Rehder et al.* para la odometría visual corregida

Se reporta como resultado la construcción de un sistema estereoscópico adecuado para odometría de largo alcance. Este sistema presenta mejoras sobre el enfoque estereoscópico estándar(las estimaciones son aproximadamente 5 veces más precisas con respecto al enfoque tradicional).

Para evaluar el sistema de odometría corregido se usó la información de una cámara estereoscópica empotrada sobre una bote que navegó el río *Allegheny* y que hizo un recorrido en *loop* de aproximadamente 2 Km. Como *ground truth* se utilizó un sistema *GPS Trimble L1/L2*. La comparación se hizo con respecto al sistema de Geiger et al.[115], el resultado se muestra en la Figura 3.20 (En la imagen se muestra la desviación estándar y la media del error para diferentes distancias recorridas).



(a) Visual Odometry (b) Corrected Visual Odometry

Figura 3.20: Comparación odometría de largo alcance con odometría estándar [116].

En el la figura 3.20 se evidencia que el error promedio del enfoque estándar de odometría visual tiene un error promedio de 10 % mientras que el error de la solución propuesta tiene un error de aproximadamente el 2 %.

Resultados de *Rehder et al.* para la estimación global de la posición

Para evaluar la estimación global de la posición se usó la misma secuencia de video capturada para evaluar la estrategia de odometría visual corregida (un recorrido en bote de

[115] Andreas Geiger, Julius Ziegler y Christoph Stiller. «Stereoscan: Dense 3d reconstruction in real-time». En: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, págs. 963-968.
[116] rehder

2 Km. sobre el río *Allegheny*).

En la figura 3.21 (a) se muestra el error en la estimación absoluta de la posición usando el sistema de odometría visual corregido + GPS + IMU (línea verde) y el mismo error para el sistema de odometría no corregido + GPS + IMU (línea roja). El experimento se llevó a cabo usando pocas mediciones de GPS (6 mediciones durante el recorrido). El error promedio usando el sistema de odometría visual no corregido es de aproximadamente 10 m. mientras que el error del sistema de odometría visual corregido es de aproximadamente 5 m, lo que representa una mejora por un factor de 2 en la estimación de la posición global.

En la figura 3.21 (b) se muestra el efecto que tiene en la exactitud de ambos sistemas el usar un determinado número de mediciones GPS. El error en la odometría visual permanece por debajo de los 6 metros usando incluso solo 5 mediciones de GPS en un recorrido de 2 km. Por otro lado, el sistema odométrico no corregido tiene un error dos veces mayor que el corregido para un número de 5 mediciones de GPS.

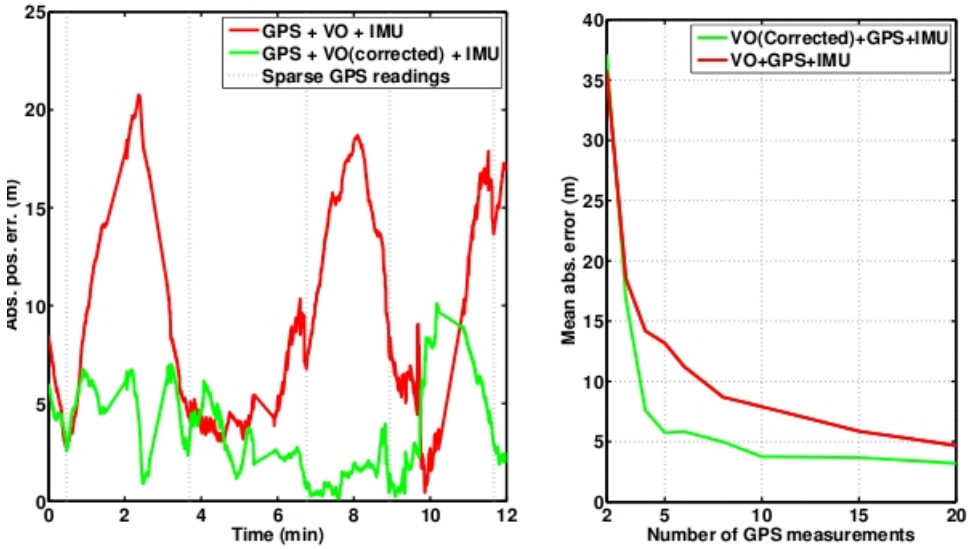


Figura 3.21: Evaluación de la estimación global de la posición con odometría visual corregida y no corregida [117].

En el trabajo en *Rheder et al.* se puede ver que surge una estrategia que incluye la fusión de las estimaciones del sistema de odometría visual con las estimaciones de un GPS para mejorar la precisión y exactitud del sistema. En ese sentido algunos investigadores han aunado esfuerzos por hacer sistemas de localización basados en odometría visual usando

[117] (REHDER y col., óp. cit.)

la información de múltiples sensores para corregir la medición dada por el sistema.

El trabajo desarrollado por *Netramai, hubert Roth y Anatoly Sachencko*[118], esta motivado por el hecho de que en los sistemas de odometría visual basados en una cámara estereoscópica poseen problemas a la hora de estimar los parámetros de *ego motion* debido a la ambigüedad que subyace en el movimiento. Es por esta razón que en el trabajo se plantean la necesidad de usar sistemas de odometría visual basados en varias cámaras que permitan hacer una estimación del movimiento más acertada.

Tipos de ambigüedades en la estimación del movimiento en sistemas de odometría visual basados en una sola cámara

- **Ambigüedad traslacional**: Es bien conocido que la estimación de la profundidad que hace una cámara estereoscópica tiene una incertidumbre que va creciendo a medida que la distancia entre la cámara y la escena aumenta. también puede mostrarse que esta incertidumbre es mas marcada cuando se tienen puntos sobre el eje óptico de la cámara. La ambigüedad traslacional sucede entonces cuando se tiene que la estimación del movimiento se hace sobre un punto que este sobre el eje óptico de la cámara, por lo general, sucede cuando se tienen movimientos pequeños.

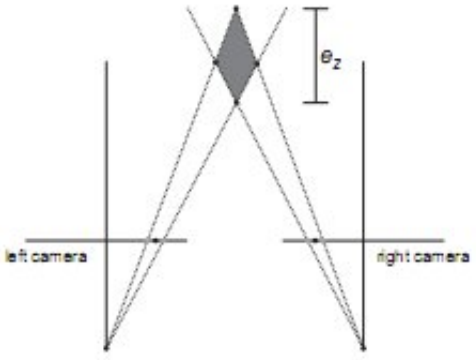


Figura 3.22: Ejemplo de ambigüedad traslacional[119]

Al observar la figura 3.22 se puede apreciar con mejor detalle en que consiste la ambigüedad traslacional. Un sistema estereoscópico estima una posición en 3D para un punto característico con un error e_z en comparación a la posición real del punto característico en el espacio, esto debido a factores intrínsecos de la cámara. La am-

[118] C. NETRAMAI, H. ROTH y A. SACHENCKO. «*High accuracy visual odometry using multi-camera systems*». En: *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*. Vol. 1. 2011, págs. 263-268.
[119] (ibíd.)

bigüedad traslacional sucede específicamente cuando la cantidad de movimiento es menor o igual a e_z, lo que causa que no se pueda estimar el movimiento con certeza.

- **Ambigüedad rotacional**: La ambigüedad rotacional ocurre cuando hay una rotación sobre un eje que es perpendicular al eje óptico de la cámara. si se observa la siguiente imagen:

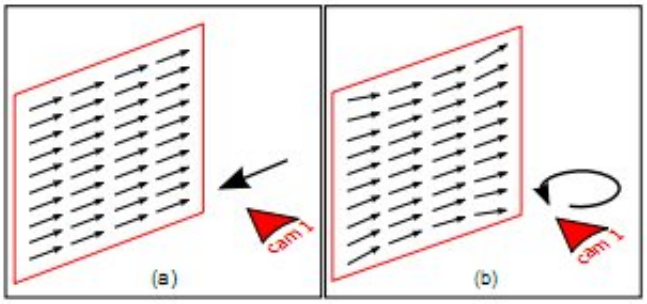


Figura 3.23: Ejemplo de ambigüedad rotacional[120].

Al observar la figura puede verse más explícitamente en que consiste la ambigüedad rotacional. Cuando la cámara se mueve hacia la izquierda 3.23, puede verse que el campo vectorial que representa el flujo óptico va en dirección contraria. Por otro lado cuando la cámara hace una pequeña rotación cuya dirección es perpendicular al eje óptico de la cámara, el campo vectorial que representa el flujo óptico tiene una forma similar al de la traslación; por lo tanto una pequeña rotación puede ser confundida con una traslación.

Eliminación de la ambigüedad en la estimación del movimiento usando sistemas de múltiples cámaras

- **Eliminación de la ambigüedad traslacional usando sistemas de 3 cámaras estereoscópicas**: Un sistema de odometría visual basado en tres cámaras estereoscópicas, se implementaría ubicando cada cámara mirando hacia una dirección perpendicular a las otras dos. Por lo tanto cuando haya un pequeño movimiento, una de las cámaras, la que esta apuntando a la dirección del movimiento, estimará un movimiento incorrecto, pero las otras dos cámaras apuntando en direcciones perpendiculares, harán una estimación correcta del movimiento. Esto se ilustra en la figura 3.24.

[120] (ibíd.)
[121] (ibíd.)

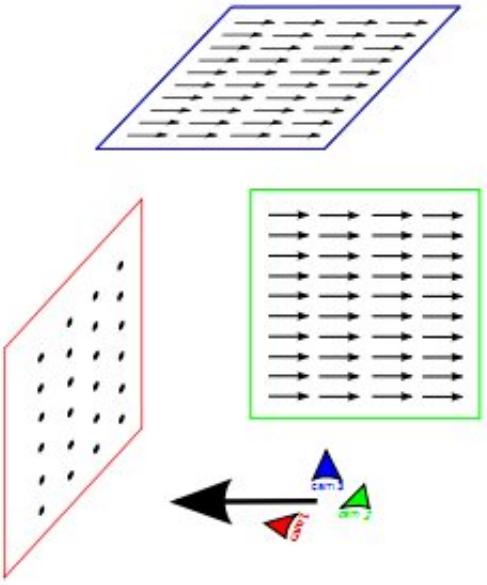


Figura 3.24: Eliminación de ambigüedad traslacional usando tres cámaras[121]

- **Eliminación de la ambigüedad traslacional usando sistemas de 2 cámaras estereoscópicas**: Si lo que se desea es eliminar la ambigüedad rotacional especificada anteriormente, puede hacerse usando un sistema de dos cámaras estereoscópicas. las dos cámaras estereoscópicas deben estar ubicadas cada una con su eje óptico en dirección perpendicular a la otra cámara como se muestra en la figura 3.25. Por lo tanto cuando el vehículo haga una rotación el flujo óptico de ambas cámaras apuntaran hacia la misma dirección, lo que permite que con certeza el sistema de odometría visual pueda determinar ese movimiento como una rotación y no como una traslación.

El proceso de adquisición de características y de estimación del movimiento sigue una línea de procedimientos similar al propuesto por *Nister* et al.[123]. Por otro lado la estimación del movimiento de un vehículo cuyo sistema de odometría este basado en 3 cámaras puede hacerse teniendo la estimación del movimiento en dos dimensiones de cada una de las cámaras del montaje 3.26. Para este objetivo se define una tabla en donde se tienen todas las posibles combinaciones de movimiento entre cada una de las cámaras 3.27 y a su lado una columna (*motion*)en donde para cada combinación de movimientos relativos de cada cámara, se estima el movimiento total del sistema.

[122] (ibíd.)
[123]NISTER, D. Et. Al. Óp. cit.
[124] (NETRAMAI, ROTH y SACHENCKO, óp. cit.)
[125] (ibíd.)

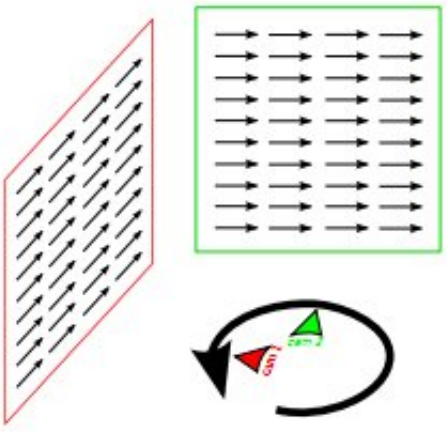


Figura 3.25: Eliminación de ambigüedad rotacional usando tres cámaras[122].

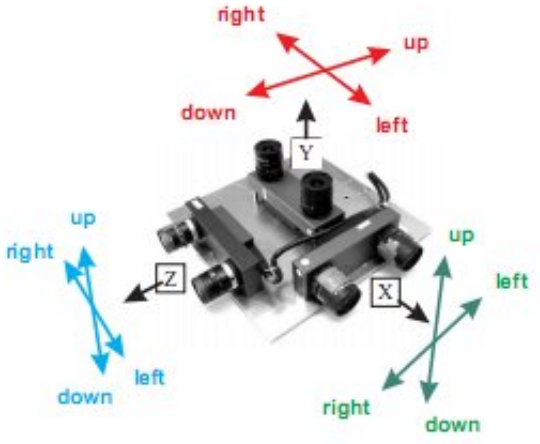


Figura 3.26: Dirección en 2D de un sistema de odometría visual usando tres cámaras estereoscópicas[124]

a manera de ejemplo, en la 5 fila de la figura 3.27 si la cámara que apunta hacia el eje z detecta un movimiento en la dirección izquierda y la cámara que apunta hacia el eje y apunta hacia la dirección izquierda, según el marco de referencia que se ha definido para cada una de las cámaras, puede concluirse que el auto se esta moviendo hacia adelante (*forward*), asumiendo claro que el movimiento en linea recta hacia adelante del vehículo este alineado con el eje x como se muestra en la figura 3.26

Resultados del trabajo de *Netramai et al.*

Para probar los resultados de este trabajo hicieron pruebas en un cuarto de 4 x 5 x 3 metros, un conjunto de pruebas con un sistema basado en una sola cámara y otro

X-cam	Z-cam	Y-cam	motion
up	up	-	up
down	down	-	down
left	-	up	left
right	-	down	right
-	left	left	forward
-	right	right	backward
-	down	down	rotate cw around x-axis
-	up	up	rotate ccw around x-axis
up	-	right	rotate cw around y-axis
down	-	left	rotate ccw around y-axis
left	left	-	rotate cw around z-axis
right	right	-	rotate ccw around z-axis
up, right	up, right	-	up and rotate ccw around z-axis

note: cw = clockwise, ccw = counterclockwise

Figura 3.27: Tabla con los posibles movimientos de cada una de las tres cámaras y la estimación del movimiento completo[125].

conjunto de pruebas con un sistema basado en 3 cámaras. Una de las pruebas fue rotar el sistema de cámaras en 15^0, el sistema de tres cámaras hizo una estimación de 16^0 lo cual es una muy buena estimación de la rotación del sistema. Por otro lado para la estimación del movimiento se hicieron traslaciones pequeñas de $10mm$ ante las cuales el sistema de odometría de tres cámaras dio una respuesta de $10{,}73mm$ una estimación muy cercana a la real.

Algunos de los desarrollos anteriormente descritos especialmente el de *Nister et al.* han inspirado algunos desarrollos para sistemas de navegación en donde el terreno es completamente desconocido y no se tiene una referencia global de posicionamiento en ese terreno, es el caso de planetas o lunas distantes.

Para los robots de exploración de Marte (*MER*), por ejemplo, es de suma importancia tener una estimación de la posición a bordo debido a que estos robots deben ejecutar ciertas tareas con mucha precisión. El objetivo planteado por los investigadores de la NASA era tener un sistema de localización cuyas estimaciones de la posición difirieran por menos del 10 % de la posición real en un recorrido de 100 m. Como resultado construyeron un sistema de localización compuesto por una *IMU* (*Inertial Measurement Unit*) y un sistema de *Wheel Odometry*, esta solución cumplía con el objetivo propuesto pero solo para terrenos estables y planos[126].

Dadas las condiciones del sistema de localización anteriormente descrito se usó la odometría visual como suplemento cuando los robots ingresaban en zonas con terrenos inestables y

[126]Yang Cheng, M.W. Maimone y L. Matthies. «*Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging*». En: *Robotics Automation Magazine, IEEE* 13.2 (June), págs. 54-62.

pendientes pronunciadas. Desde la llegada de los robots (*Opportunity* y *Spirit*) a Marte en el 2004, la odometría visual se ha usado de manera extensiva en terrenos inestables y con pendientes pronunciadas. La razón por la cual la odometría visual no se ha usado en toda la navegación de los robots está relacionada con que el procesamiento de las imágenes en el CPU de 20MHz de los robots es lento, por lo tanto, si se quiere mayor velocidad en la navegación no es conveniente usar la odometría visual.

La odometría visual le ha permitido navegar a los robots de manera precisa en recorridos de hasta 8 m y en pendientes mayores a 20°.

Algoritmo de *Cheng et al.*

La estimación del movimiento se basa en identificar un conjunto de características en el fotograma estereoscópico actual y rastrearlas en el fotograma estereoscópico siguiente. A partir del rastreo de estas características se trata de determinar el cambio en la posición y orientación del robot usando MLE (*Maximum Likelihood Estimation*). Es importante mencionar que si no se dispone de suficientes puntos característicos o la estimación del movimiento no converge se usa la estimación de movimiento brindada por el sistema original(*IMU + Wheel Odometry*). A continuación se de detallan las etapas del proceso de estimación del movimiento a partir de los fotogramas estereoscópicos.

- **Detección de características:** En esta etapa las características que puedan ser emparejadas en un fotograma estereoscópico y rastreadas durante el movimiento fácilmente son seleccionadas. Se usa un detector de esquinas (*Harris*, *Forstner*, etc) y los pixeles con los mayores valores de interés son seleccionados como características. Se utilizó un *grid* para reducir el costo computacional de esta etapa.

- **Emparejamiento estereoscópico de características:** Las posiciones 3D de las características se determinan usando el emparejamiento estereoscópico y la triangulación. Debido a que la cámara está bien calibrada, el emparejamiento estereoscópico se hace a lo largo de la línea epipolar teniendo en cuenta algunos pixeles por encima y por debajo de la dicha línea. Para determinar el mejor emparejamiento para cierto punto de interés se usa correlación pseudo-normalizada y un polinomio de grado 4.

  Para hacer la triangulación se interceptan los dos rayos que salen desde los pixeles emparejados en las dos imágenes (fotograma izquierdo y fotograma derecho) y que pasan por los puntos focales. En la realidad, estos rayos no se intersectan y es necesario aproximar dicha intersección. La distancia más corta entre ambos se rayos representa que tan buena es la triangulación, si esta distancia es muy grande la triangulación se considera incorrecta y se desecha.

La calidad de una característica en 3D es una función de la localización relativa, la brecha o diferencia entre los dos rayos estereoscópicos y la agudeza del pico de correlación. Cada característica tiene asociada una matriz de covarianza y el cálculo de esta matriz de covarianza involucra los tres factores mencionados con anterioridad. Los métodos para calcular esta matriz de covarianza se presentan en Matthies and Shafer 1987[127].

- **Rastreo de características**: Después que el vehículo se mueve un poco se adquiere un segundo par de imágenes. Las características en 3D del primer par de imágenes (fotograma anterior) se proyectan en el par de imágenes actual usando la información provista por *wheel odometry*. Posteriormente una búsqueda basada en la correlación restablece las posiciones 2D en el segundo par de imágenes. A partir de estas posiciones 2D es posible hacer emparejamiento estereoscópico y obtener la posición en 3D de esas características. Dado que las posiciones en 3D ya son conocidas por el primer procedimiento es posible realzar un primer filtrado de *outliers* al rechazar los puntos de interés cuyas posiciones en 3D inicial y final varíen de manera considerable.

- **Estimación robusta del movimiento:**

  Si la estimación inicial del movimiento (*wheel odometry*) es adecuada, la diferencia entre las dos estimaciones 3D del paso anterior no debería ser considerable, pero si este no es el caso el error entre ambas estimaciones puede ayudar a determinar el cambio en la posición del robot.

  La estimación del movimiento es realizada en dos pasos:

  1. Una estimación del movimiento menos exacta usando mínimos cuadrados
  2. Una estimación más exacta usando MLE (*Maximum Likelihood Estimation*)

  En la estimación por mínimos cuadrados el error residual se define como:

$$e_j = P_{C_j} - RP_{T_j} - T \tag{3.57}$$

  Donde e_j es el error asociado a la característica j-ésima, P_{C_j} es la posición actual de la característica j-ésima, P_{T_j} es la posición anterior de la característica j-ésima (fotograma estereoscópico anterior) y R y T son respectivamente las matrices de rotación y traslación que desean estimarse.

  La función de costo que desea minimizarse es:

$$M(R,T) = \sum w_j e_j^T e_j \tag{3.58}$$

[127] Matthies y Shafer, óp. cit.

$$w_j = (det(\sum_{T_j}) + det(\sum_{C_j}))^{-1} \tag{3.59}$$

en 3.59 el símbolo $\sum$ se asocia con la matriz de covarianza discutida en la etapa de emparejamiento estereoscópico.

Este problema de mínimos cuadrados tiene una solución que se puede calcular de manera rápida. La desventaja es que esta solución no es tan exacta debido a que solo se toma la calidad (el volumen del error) de las características.

Dado que la solución al problema anterior se puede encontrar de manera rápida se puede fusionar esta estimación con *RANSAC* para hacer una estimación más robusta, para ver los detalles de esta fusión se puede consultar Cheng et al. 2006[128].

En la segunda etapa de la estimación todas las características que hayan sido aceptadas por el *RANSAC* serán usadas para hacer la estimación por MLE, este enfoque tiene en cuenta la diferencia en la posición 3D y los modelos de error asociados.

En la estimación por MLE los tres ejes de rotación θ_R y la traslación T son directamente determinados minimizando la siguiente suma:

$$\sum e_j^T W_j e_j \tag{3.60}$$

Donde e_j tiene el mismo significado que en la ecuación 3.57 y W_j es la matriz inversa de la matriz de covarianza asociada a e_j. La minimización de la suma 3.60 es un problema no lineal y se propone una linealización y la aplicación de un método iterativo para solucionarlo.

Dentro del trabajo de Cheng et al.[129] se afirma que la estimación por MLE es potente debido principalmente a dos aspectos:

- La estimación directa de los tres ejes de rotación θ_R elimina el error inducido por la estimación de la matriz de rotación (R) por el método de los mínimos cuadrados.
- Se incorporan completamente los modelos de error, es decir, se incorpora la forma del error en lugar del volumen, lo que brinda una estimación más exacta del movimiento.

Resultados de *Cheng et al.*

[128] Cheng, Maimone y Matthies, óp. cit.
[129] Ibíd.

A continuación se reportan los resultados de desempeño del sistema de odometría visual descrito. La primera parte de los resultados corresponden a evaluaciones del sistema hechas en el planeta Tierra, la segunda parte corresponde al desempeño real que tuvo el sistema en el planeta Marte.

Dos de los experimentos hechos en la tierra fueron *Marsyard course* y *Johnson Valley course*. El primer experimento comprendía un trayecto de 24 m y se reportó un error menor al 2,5 % en la estimación de la posición y menor a 5° en la estimación de la rotación. El segundo experimento comprendía un trayecto de 29 m y el error en la estimación de la posición fue menor al 1,5 % y menor a 5° en la rotación. En ambos experimentos se uso como *ground truth* un sistema que usaba un teodolito de topografía con un escáner láser, dicho sistema proporcionaba mediciones con un error menor a 2 mm en la posición y menor a 0,2° en la rotación.

En la figura 3.28 se muestra el desempeño del sistema de odometría visual descrito comparado con *wheel odometry + IMU* en un recorrido de 2.4 m en una pendiente pronunciada. El experimento se llevó a cabo en *MER Surface System Testbed Lite rover* y es notable que el error de la odometría visual fue menor al 1 % en todo el recorrido.

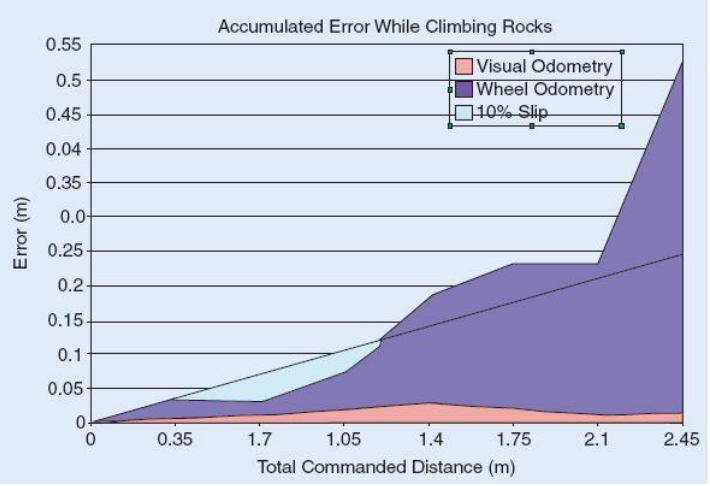


Figura 3.28: Comparación del error acumulado de VO y WO[130].

En Marte la odometría visual fue usada para algunas circunstancias particulares (pendientes pronunciadas, terrenos arenosos, deslizamiento de ruedas, etc.) y la razón está relacionada con el tiempo de cómputo que se necesita para hacer una actualización en la estimación de la posición del robot. Esta actualización tarda alrededor de 2 minutos y por lo tanto retarda considerablemente el avance del robot. Dado que la *IMU* brinda una estimación de la rotación muy aproximada a la rotación real, la odometría visual se

[130] (ibíd.)

usó durante el 2004 y el 2005 en Marte únicamente para actualizar la posición del robot.

La odometría visual ha permitido mejorar la seguridad de los robots de exploración al navegar. Además, ha permitido reducir el tiempo empleado para cumplir algunos de los objetivos de los robots en Marte debido a que se ha mejorado la navegación por terrenos no convencionales. Es importante mencionar que existen situaciones especiales donde la odometría visual no converge por diferentes razones (e.g. carencia de características, movimientos largos, giros largos, etc.) lo cual representa una desventaja.

En la figura 3.29 se muestran los trayectos en que fue utilizada la odometría visual en el robot de exploración *Opportunity* en Marte.

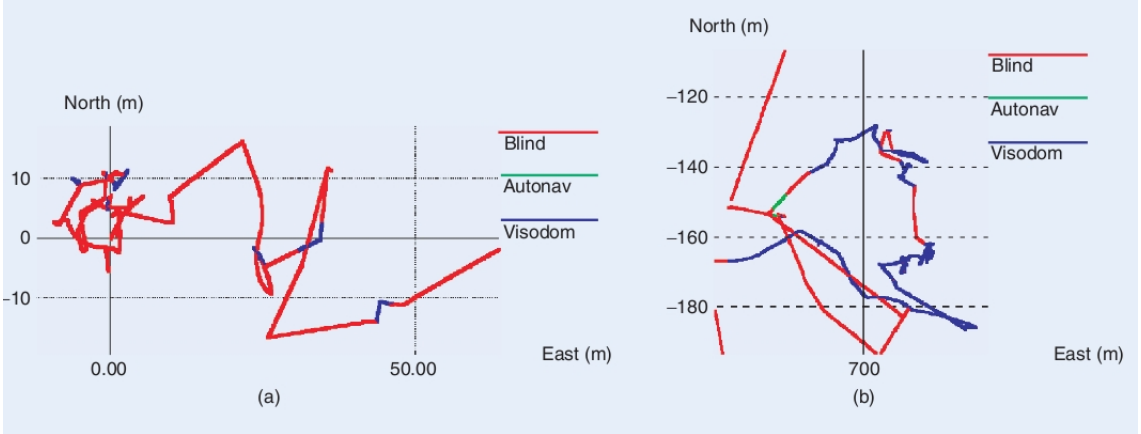


Figura 3.29: Uso de la odometría visual por parte del *Opportunity* en el (a) *Eagle Crater* y en el (b) *Endurance Crater*[131].

La odometría visual se usó con mayor frecuencia en el *Opportunity* que en el *Spirit* debido a que este navegó su primer año en superficies con pendientes pronunciadas. La estimación del movimiento por odometría visual convergió un 95 % del tiempo en el *Opportunity*.

En el *Spirit* la odometría visual solo fue utilizada al llegar a *Columbia Hills* (este lugar se caracteriza por tener pendientes pronunciadas) y convergió a una estimación del movimiento el 97 % del tiempo.

A manera de conclusión se reporta en el trabajo de Cheng et al. que la odometría visual se convirtió en una herramienta efectiva para mantener la seguridad del vehículo y para reducir el tiempo que se tardan los robots en navegar sobre trayectorias complejas.

Desde implementaciones más eficaces y confiables para determinar la posición del vehículo, se han realizado desarrollos por implementar sistemas de odometría visual que den una estimación de la posición del vehículo en poco tiempo para sistemas que funcionen en

[131] (ibíd.)

tiempo real. En la sección 3.4.5 se mostró el algoritmo de *RANSAC* para corregir un modelo de movimiento usando odometría visual, el cuál ha sido ampliamente usado por los diferentes trabajos ampliamente expuestos, sin embargo, en el estudio comparativo desarrollado más adelante(ver sección 4.5) se mostró que el proceso de *RANSAC* puede tardar mucho dependiendo de la mínima cantidad de puntos para estimar un modelo de movimiento correcto, en ese sentido *Nister et al.* con el algoritmo de los cinco puntos y *RANSAC* preventivo, se dio un primer acercamiento a solucionar los problemas de eficiencia de la etapa de *RANSAC* obteniendo resultados satisfactorios. Aún así otros investigadores han demostrado dar un mejor tiempo de respuesta en tiempo real.

El trabajo[132] desarrollado por *Davide Scaramuzza, Friedrich Fraundorfer y Roland Siegwart*, fue el desarrollo de una metodología nueva para hacer la estimación del movimiento de un vehículo y para implementar la eliminación de datos lejanos a un modelo estándar de movimiento (*outliers*). Para dicho objetivo se aprovecharon las ventajas de las propiedades no-holonómicas de un vehículo que se mueve a lo largo de un camino, lo que permite que se pueda plantear una estimación del movimiento de un vehículo parametrizada usando solo un punto característico. Con esta parametrización, que es la mínima posible, se presenta un algoritmo para la eliminación de *outliers* muy eficiente.

A pesar de que se han desarrollado numerosos trabajos en el campo de la odometría visual hasta la fecha, aún juegan un rol secundario en la implementación de vehículos autónomos, debido a las siguientes razones:

- muchos algoritmos aún no funcionan en tiempo real.
- algunos algoritmos necesitan mucha capacidad de procesamiento.
- algunos algoritmos están diseñados para cámaras específicas.
- muchos algoritmos restringen el espacio de percepción de la cámara o solo funcionan si no existen muchos objetos moviéndose en la escena.
- la asociación entre características muchas veces falla debido a diferentes factores.
- Un sistema de odometría visual convencional falla cuando hay poca estructura en la escena, lo que se traduce en una ausencia de puntos característicos a los cuales se les pueda hacer seguimiento.

[132]D. Scaramuzza, F. Fraundorfer y R. Siegwart. «Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC». En: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. 2009, págs. 4293-4299.

Este trabajo da una solución óptima a estos problema por medio de la metodología planteada anteriormente.

Por que se necesita un solo punto para parametrizar la estimación del movimiento

para que un vehículo tenga un movimiento circular, cada una de las cuatro llantas del vehículo deben tener un punto de referencia alrededor del cual sigan una trayectoria circular. Dicho punto se denomina, como punto instantáneo de rotación(*Instantaneous Center of Rotation, ICR*) y se caracteriza por tener velocidad cero en un instante de tiempo determinado, además de que se puede trazar una curva que describa su trayectoria con el tiempo como parámetro. Este punto puede ser calculado hallando la intersección de todos los ejes de giro de cada una de las ruedas. La existencia de este centro de rotación esta dada por el principio de Ackerman, el cual establece que cuando un automóvil da una curva, para que el movimiento sea suave y no desgaste las llantas, el ángulo de rotación de la rueda interior debe ser mas grande que el de la rueda exterior.

Se puede plantear entonces que el movimiento de una cámara instalada en un vehículo puede estimarse con el movimiento circular, por ejemplo un movimiento en linea recta, puede verse como un movimiento circular alrededor de un centro de rotación cuya distancia radial sea infinita.

Para efectos de simplicidad se modelará el movimiento del vehículo solo en dos dimensiones, de manera que para describir el movimiento del vehículo pueden usarse tres parámetros como se muestra en la figura 2.15.

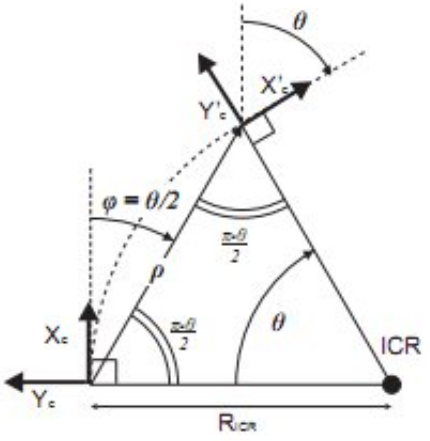


Figura 3.30: modelo del movimiento de un vehículo en dos dimensiones y relación entre los ejes de la cámara en un movimiento circular[133].

[133] (ibíd.)

Los tres parámetros que se necesitan para describir el movimiento del vehículo, estos son θ que representa el ángulo de tracción de las llantas y una pareja ordenada en coordenadas polares (ρ, ϕ) en donde la primera coordenada representa la distancia recorrida, pero como no se conoce la escala ya que el sistema usa una sola cámara, se establece este parámetro igual a 1. la segunda coordenada representa el ángulo de tracción del automóvil. Puede observarse que para determinar el movimiento del vehículo solo se necesita un punto ya que $\phi = \frac{\theta}{2}$, por lo tanto solo tendría que estimarse el ángulo θ con un solo punto. Para el caso de que el vehículo se este trasladando en línea recta $\theta = 0$ y $\phi = 0$.

Independiente del modelo de la cámara, la representación de las coordenadas en la escena se hace por medio de coordenadas esféricas. Por otro lado en un sistema de odometría visual es crucial el cálculo de la matriz esencial, del cual se derivan la rotación y la traslación cuyo cálculo se puede desarrollar partiendo de la restricción epipolar(ecuación 2.11).

$$p'^{T} E p = 0 \tag{3.61}$$

donde p' es un punto en tres dimensiones visto en un fotograma y p es otro punto en 3 dimensiones visto en un fotograma en un instante de tiempo distinto y E es la matriz esencial.

La matriz esencial viene dada por al expresión $E = [T]_X R$.donde:

$$[T]_X = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}, \tag{3.62}$$

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} .T = \rho \begin{bmatrix} cos(\phi) \\ sin(\phi) \\ 0 \end{bmatrix} \tag{3.63}$$

Por lo tanto la matriz esencial E bajo la restricción de que $\phi = \frac{\theta}{2}$ resulta de la siguiente forma:

$$E = \rho \begin{bmatrix} 0 & 0 & sin(\frac{\theta}{2}) \\ 0 & 0 & -cos(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & sin(\frac{\theta}{2}) & 0 \end{bmatrix}, \tag{3.64}$$

Por lo tanto puede verse que solo se necesita de un punto en la imagen para determinar el

ángulo θ y así la matriz esencial que representa el movimiento y la rotación de un punto en la escena. EL ángulo puede ser obtenido fácilmente por medio de la solución de una ecuación al reemplazar (2.14) en (2.13).

Remoción de *outliers(RANSAC)*

En este trabajo se identificaron los *outliers* para la estimación del movimiento, es decir, observaciones que se alejan a la estimación de movimiento obtenida con el algoritmo de estimación, que en este caso se basa en una sola correspondencia para hallar el ángulo θ.

El modelo de remoción de *outliers RANSAC* busca establecer una estimación de movimiento, en este caso, que se ajuste a las observaciones obtenidas. Para establecer esta estimación el *RANSAC* debe hacer iteraciones sobre un conjunto de observaciones hasta obtener una solución correcta, este numero de iteraciones es exponencial con respecto al número de puntos característicos sobre la imagen que se tomen para hacer la estimación de movimiento. El número de iteraciones viene dado por $N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)}$ donde s es el número de datos mínimo para hacer la estimación del movimiento, ϵ es el porcentaje de *outliers* y p es la probabilidad de éxito. Por ejemplo un algoritmo que use 5 puntos[134] o un algoritmo que use 6 puntos de interés para hacer correspondencias tendrá un número de iteraciones mucho mayor que el enfoque propuesto en este trabajo que solo se basa en un punto característico para hacer la estimación del movimiento, lo que permite que el *RANSAC* tenga un modelo de movimiento planar usando solo un parámetro, en vez de dos parámetros como normalmente se acostumbra en otros enfoques. Esto causa inmediatamente que el modelo de estimación basado en un solo punto sea muchísimo mas eficiente que otros enfoques.

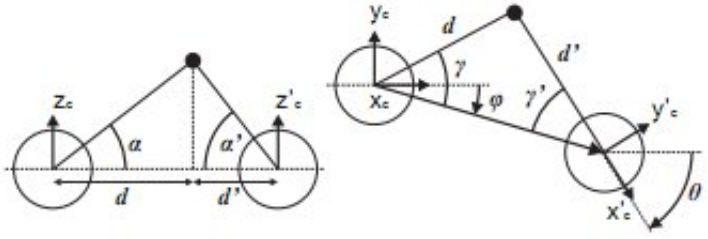


Figura 3.31: Representación de un punto característico desde su proyección en el plano xy(izquierda) y visto desde un lado(derecha)[135].

Para poder tomar el modelo de movimiento que más ajustado sea la realidad se comparara cada una de las posibles correspondencias de un solo punto con otra cantidad n de correspondencias al azar. Los *inliers* son los características que mas se acercan a un modelo

[134]NISTER, D. Et. Al. Óp. cit.
[135] (Scaramuzza, Fraundorfer y Siegwart, óp. cit.)

de movimiento entre dos fotogramas de la imagen generado por una sola correspondencia, en este caso, por un solo punto característico emparejado con su igual en el fotograma siguiente. Para esto se usa una medida de error(error direccional), que me determinara si una estimación de movimiento entre una correspondencia de un punto característico se denomina como *inlier* o como *outlier*. Dicha medida de error se puede deducir de la figura 2.16 en donde se tiene un ejemplo geométrico de como se ve un punto característico en tres dimensiones visto desde un lado y desde arriba. De la figura 2.6 se puede deducir que $dtan(\alpha) = d'tan(\alpha')$ y $dsin(\gamma) = d'sin(\gamma')$ eso quiere decir que $\frac{d'}{d}$ se puede calcular con

$$\frac{sin(\gamma)}{sin(\gamma')} \tag{3.65}$$

y

$$\frac{dtan(\alpha)}{dtan(\alpha')} \tag{3.66}$$

,

por lo tanto la medida de error para decidir si una correspondencia es un *inlier* es si la diferencia entre la expresión (2.15) y la expresión (2.16) es mas baja que un umbral t.

Remoción de *outliers* por medio de un histograma de votos

Una solución más eficiente puede ser planteada para la remoción de *outliers* debido a el hecho de que se tiene solo un parámetro para estimar el movimiento, este método se puede ver como una estimación democrática. Primero se calcula θ para cada correspondencia i de características que se haga, y luego para esa correspondencia i se crea una entrada en una tabla de frecuencias H. Cada nueva característica que se vaya emparejando coloca su voto por una estimación θ(ya existente en la tabla H) que sea igual a su estimación. Por último, la entrada de la tabla H con mas votos será escogida como la estimación de movimiento correcta.

Resultados del trabajo de *Scaramuzza et al.*

En cuanto a los resultados del sistema de odometría basado en un solo punto y modelando el movimiento en un plano polar de dos dimensiones, puede observarse que hay una optimización notable en cuanto a recursos de máquina, ya que la carga de datos para analizar es mucho menor y así también la complejidad del algoritmo. En cuanto a la estimación de la trayectoria, el error acumulativo inherente a un sistema de odometría visual es mucho más grande que en otros enfoques de estimación de movimiento, como el basado en 5 puntos para un movimiento planar, u otros enfoques que se basan en movimiento en 3 dimensiones. Al observar la figura 2.17 y 2.18 puede apreciarse más la diferencia del enfoque de un solo punto característicos con otros enfoques.

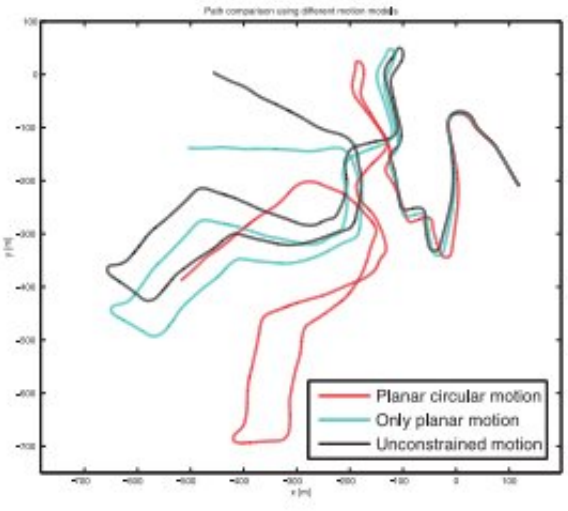


Figura 3.32: Comparación de la estimación del movimiento con un punto característico y movimiento planar(rojo) con otros enfoques, planar (en verde) y en 3 dimensiones(negro)[136]

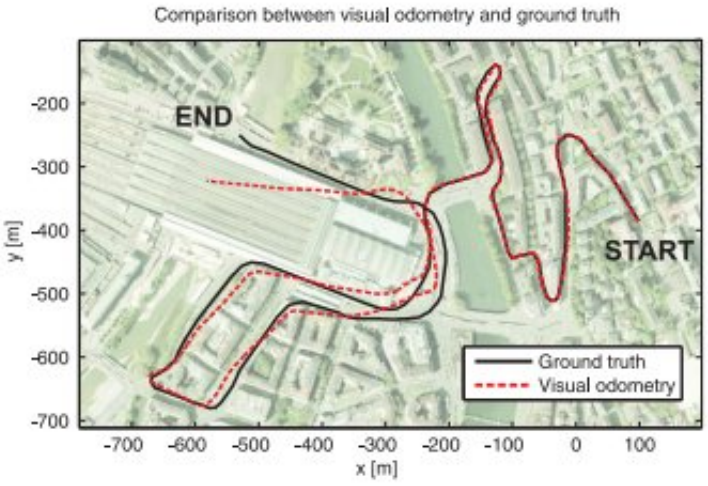


Figura 3.33: Comparación de la estimación del movimiento con un punto característico y movimiento planar con la trayectoria real[137].

Puede verse en la figura 2.18 que la trayectoria estimada por el algoritmo desarrollado en este trabajo presenta una diferencia muy marcada con la trayectoria real, cuando la distancia recorrida aumenta.

[136] (ibíd.)
[137] (ibíd.)

Capítulo 4

Estudio comparativo

4.1. Introducción

En este capítulo se pretende hacer una comparación entre las distintas alternativas existentes para realizar odometría visual. Esta comparación se realizará teniendo en cuenta cada una las descripciones y etapas detalladas y soportadas en la sección 3.4.1:

1. Extracción de características.
2. Emparejamiento de características.
3. Estimación del movimiento.
4. Estimación Robusta.

4.2. Extracción de características

En la figura 4.1[1] se resume el desempeño de algunos detectores de puntos de interés de acuerdo a diversos criterios.

El detector de *Harris* encabeza la lista de los detectores de esquinas. Este detector es de fácil implementación y ha demostrado funcionar satisfactoriamente en sistemas de odometría visual que se ejecutan en tiempo real[3]. En adición a esto tiene la característica de ser repetitivo.

[1]SCARAMUZZA y FRAUNDORFER, óp. cit.
[2] (SCARAMUZZA y FRAUNDORFER, «*Visual Odometry [Tutorial part II]*»)
[3]NISTER, D. Et. Al. Óp. cit.

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CENSURE		x	x	x	x	+++	++	+++	+++

Figura 4.1: Comparación de métodos para extraer características[2].

En la lista de los detectores de *blobs* encabezan la lista *SIFT* y *SURF*(detallado en la sección 3.4.2). *SIFT* y *SURF* son algoritmos invariantes a la escala y a la rotación. Como se evidencia en la figura 4.1 *SIFT* es más robusto que *SURF*, pero este último es más eficiente.

4.3. Emparejamiento de características

En la figura 4.2 se resumen las bondades y debilidades de las medidas de similitud y de las estrategias de emparejamiento, teniendo en cuenta su eficiencia y la precisión para la localización.

Emparejamiento de características					
Medidas de similitud			Estrategia de emparejamiento		
	eficiencia	Precisión para la localización		eficiencia	Precisión para la localización
NCC	'++	'+++	Emparejamiento restringido	'++++	'+++
SAD	'+++	'++	Emparejamiento no restringido	'++	'+++
NNDR	'++++	++++			

Figura 4.2: Comparación de métodos para establecer correspondencias entre dos imágenes I_k e I_{k-1} [4].

Es evidente que *NNDR* presenta mejores resultados en eficiencia y en localización, mientras que *SAD* es el menos preciso para la localizacón como se evidencia en el próximo capítulo

[4]Autores

en la figura 5.4. La medida de similitud *NCC* ha ofrecido resultados satisfactorios como lo demostró *Nister et al.* en su trabajo[5] . Como se especificó en la sección 3.4.3, *NNDR* se utiliza generalmente con descriptores de características. La eficacia del *NNDR* para encontrar correspondencias dependerá en gran parte de la cantidad de información sobre un punto de interés que posea el descriptor utilizado con respecto a un punto característico detectado en alguna región de la imagen.

En cuanto a la estrategia de emparejamiento, se evidencia que un enfoque restringido es mucho más eficiente que el enfoque no restringido, ya que el primero acota el espacio de búsqueda, permitiendo reducir la complejidad algorítmica como ya se puntualizó en la sección 3.4.3. Un enfoque NO restringido es útil cuando no se desea una respuesta en tiempo real o cuando se cuentan con dispositivos de alto procesamiento de datos, como unidades de procesamiento gráfico. El emparejamiento NO restringido además supone un reto aún más grande ya que requiere conocer de antemano cierta información sobre donde puede estar una característica en un fotograma siguiente.

4.4. Estimación del movimiento

Para la estimación del movimiento se condensó en la tabla 4.3 las ventajas y las desventajas de las metodologías planteadas en la sección 3.4.4 que representan las metodologías más generales aplicadas a solucionar el problema de la estimación de movimiento.

La complejidad matemática del método de estimación 2D-2D depende en gran medida de la cantidad de puntos con los que se va a dar una estimación de movimiento. Como se muestra en la sección 3.4.4 la metodología planteada por *Longuet-Higgins* usando 8 puntos, supone la solución de un sistema homogéneo y lineal, mientras que por otro lado el enfoque planteado por *Nister* usando 5 puntos, sugiere la solución de un sistema no lineal que en principio es un problema más complejo que el primero.

4.5. Estimación robusta

En esta sección se compararán los algoritmos descritos en la sección 3.4.5. En la figura 4.4 se muestran sus principales ventajas y desventajas.

[5]NISTER, D. Et. Al. Óp. cit.
[6]Autores
[7]Autores

Métodos de estimación de movimiento	ventajas	desventajas
Método 3D-3D	Se recupera directamente la escala absoluta del movimiento de la cámara.	Si la distancia de la cámara a la escena es mucho más grande que la distancia del baseline(distancia entre las cámaras del par estereoscópico), pueden existir errores en la estimación de la trayectoria.
	Menor complejidad algorítmica, lo que se traduce en mejor tiempo de respuesta.	En la práctica este enfoque ha demostrado ser inferior a 2D-2D y 3D-2D debido a la triangulación.
Método 2D-2D	Su implementación no requiere de visión estereoscópica.	Tiempo de respuesta mayor que en los otros dos enfoques.
		Dificultad a la hora de conocer la escala absoluta del movimiento, solo puede calcularse conociendo información de la escena real.
		Más complejo matemáticamente dependiendo de la cantidad de puntos que se tomen. .
Métodp 3D-2D	mejor que 3D-3D cuando se está hablando de visión estereoscópica.	La triangulación que hace 3D-2D induce bastantes errores en la estimación de movimiento.
	Veloz desempeño en tiempo real.	

Figura 4.3: Comparación de métodos para el cálculo del movimiento dada una secuencia de imágenes[6].

metodologías de RANSAC	ventajas	desventajas
RANSAC	Mayor precisión para localización.	Complejidad algorítmica exponencial sobre el número s de puntos para cosntruir un modelo.
RANSAC preventivo	Menor tiempo de ejecución con respecto al RANSAC tradicional.	Propenso a errores por parte del desarrollador.
	Mayor control sobre los parámetros del algoritmo.	Menos preciso para la localización.

Figura 4.4: Comparación de métodos para *RANSAC*. *RANSAC* tradicional y preventivo[7].

Es válido aclarar que el *RANSAC* preventivo es menos preciso para la localización cuando más de la mitad de las correspondencias entre dos imágenes resultan erróneas[8]. En cuanto al *RANSAC* tradicional en la ecuación 3.50 se evidencia un aumento exponencial del tiempo de ejecución a medida de crece el mínimo número de puntos s para computar un modelo de movimiento. En la figura 4.5 se ilustra el hecho anterior.

[8]Nistér, óp. cit.

Cabe aclarar que además de las iteraciones que se muestran en la figura 4.5 debe tenerse en cuenta que, primero, por cada subconjunto de tamaño s de correspondencias debe calcularse el modelo de movimiento usando alguno de los enfoques previamente descritos(ver sección 3.4.4); segundo, por cada iteración del *RANSAC* tradicional deben validarse que las correspondencias restantes(distintas al subconjunto de tamaño s que se usó para estimar el modelo) sean consistentes con el modelo de movimiento estimado. En conclusión la complejidad algorítmica del *RANSAC* tradicional es $O(NM + NW)$ donde M es la cantidad de correspondencias entre las imágenes I_k e I_{k-1}, N es el número de iteraciones del *RANSAC* y W hace referencia a la complejidad de instanciar un modelo de movimiento dado un conjunto de correspondencias de tamaño s.

Existen otras variantes de *RANSAC* propuestas, entre ellas se encuentran *RANSAC* con incertidumbre[9], *RANSAC* progresivo[10]. Estos últimos son esfuerzos por diseñar enfoques *RANSAC* que tomen menos tiempo para ejecutarse sin sacrificar precisión en la estimación del modelo.

Table 1. Number of RANSAC iterations.

Number of points (s):	8	7	6	5	4	2	1
Number of iterations (N):	1,177	587	292	145	71	16	7

Figura 4.5: Un ejemplo del número de iteraciones de *RANSAC* en función de la cantidad de puntos para estimar un modelo de movimiento[11].

[9] R. Raguram, J.-M. Frahm y M. Pollefeys. «Exploiting uncertainty in random sample consensus». En: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009, págs. 2074-2081.

[10] O. Chum y J. Matas. «Matching with PROSAC - progressive sample consensus». En: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. 2005, 220-226 vol. 1.

[11] (SCARAMUZZA y FRAUNDORFER, óp. cit.)

Capítulo 5

Diseño e implementación

5.1. Introducción

En este capítulo se expondrá, primero, una arquitectura general del sistema por medio de diagramas y segundo, una explicación del algoritmo usado en cada etapa, cada uno con una justificación de su uso y algunos detalles de implementación importantes para asegurar el funcionamiento del sistema de odometría visual.

5.1.1. Arquitectura general del sistema

Si bien la implementación no se encuentra orientada a objetos, se muestra un diagrama de clases en la figura 5.1 representando lo que serían los módulos del sistema de localización.

La implementación de este sistema de odometría visual puede estar dividida en dos grupos de módulos, el primer grupo hace parte de los módulos que ofrece la librería *OpenCV*[2] del lenguaje C++ para ayudar al desarrollador con aplicaciones que tengan que ver con visión por computador. El segundo grupo hace referencia a los módulos construidos por los autores de este trabajo. A continuación se listan los dos grupos:

Primer Grupo(Módulos de *OpenCV*):

- *cv::SurfDetector*

[1] Autores

[2] itseez. *OpenCV: Open source Computer Vision library.* [citado en 2013]. URL: `http://opencv.org/` `autores:http://itseez.com/`.

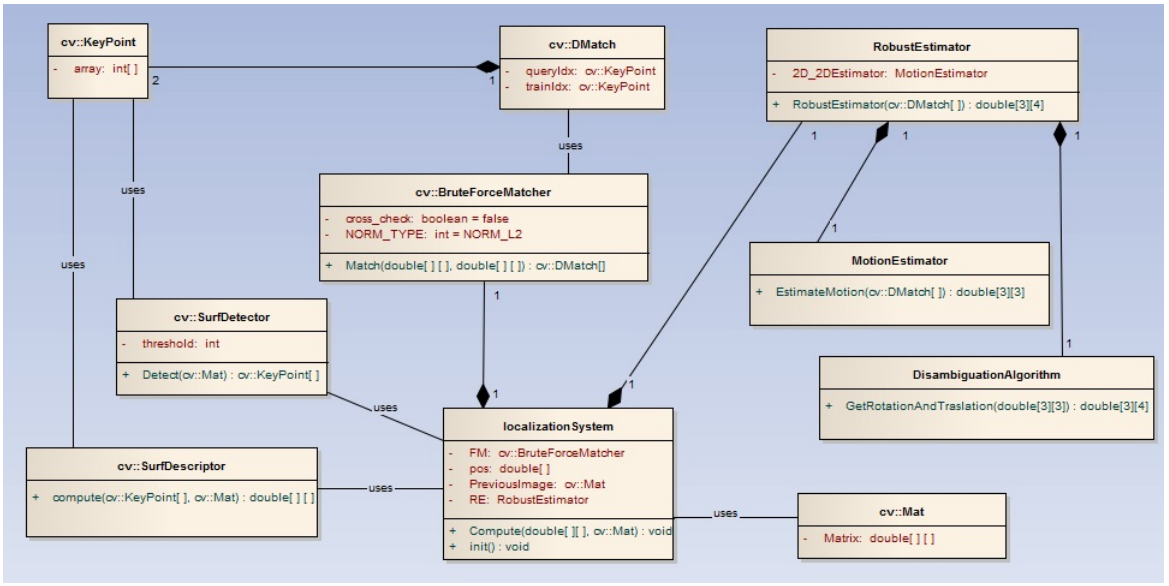


Figura 5.1: Diagrama de módulos del sistema de localización[1].

- *cv::SurfDescriptor*
- *cv::BruteForceMatcher*
- *cv::DMatch*
- *cv::KeyPoint*
- *Mat*

Segundo Grupo(Módulos de los autores de este trabajo):

- *LocalizationSystem*
- *MotionEstimator*
- *RobustEstimator*
- *DisambiguationAlgorithm*

Cada uno de los módulos será detallando en las siguientes secciones siguiendo el esquema de un sistema de odometría visual propuesto en la sección 3.4.1. A continuación se detallan tres de los módulos de *OpenCV*:

- ***Mat***: Mat es una de las clases principales de *OpenCV*, esta clase se compone inicialmente de una matriz de cualquier tipo(entero, punto flotante, etc.), varias de las operaciones que el álgebra lineal describe para aplicar sobre las matrices, tales como, calcular la inversa, la transpuesta, el determinante, etc., son fácilmente aplicables sobre una instancia de esta clase. *Mat* funciona tanto para representar una imagen I como para representar las matrices que se usan en este trabajo para la estimación del movimiento.
- ***cv:KeyPoint***: Este es un tipo de dato o clase que permite representar un punto de interés por medio de la posición (x, y) de dicho punto sobre la imagen.
- ***cv:DMatch***: Este es un tipo de dato o clase que representa una correspondencia entre dos puntos de interés, el primer atributo hace referencia a un punto de interés y el segundo hace referencia a la correspondencia de este punto de interés en una imagen subsecuente.

Para más detalles sobre estos tipos de datos, consultar la página web de *OpenCV*[3].

Para cada etapa del sistema de odometría visual se dará una explicación de los módulos usados por esta y una justificación del uso de cada librería, clase o módulo.

5.2. Extracción de características

Para la extracción de características se revisaron dos algoritmos descritos en la sección 3.4.2. Primero,se revisó el detector de esquinas conocido como detector de *Harris*. Segundo, un descriptor de puntos de regiones de interés o detector de *blobs* conocido como detector *SURF*.

En una primera etapa de implementación del proyecto se trabajo con el detector de *Harris* ya que investigadores como *Nister*[4] y Scaramuzza[5] respaldan el uso de este detector para el desarrollo de un sistema de odometría visual, ver también sección 4.2. Sin embargo las pruebas desarrolladas con este detector generaron resultados satisfactorios para detectar puntos de interés, pero las pruebas del módulo explicado en la sección siguiente 5.3 arrojaron resultados pobres para el emparejamiento de características con este detector.

Luego de evidenciar dichos resultados y luego de una investigación más amplia, se concluye que más allá de un detector de puntos de interés como lo es el detector de *Harris* debía

[3]Ibíd.
[4]NISTER, D. Et. Al. Óp. cit.
[5]SCARAMUZZA y FRAUNDORFER, óp. cit.

buscarse un descriptor, es decir, que además de detectar un punto de interés, describiera ese punto por medio de un vector característico que se compone de información de la región adyacente a dicho punto.

Por la razón anterior se decidió usar el descriptor *SURF* de la librería *OpenCV* para facilitar el trabajo de detectar puntos de interés. Además, este detector muestra buenos resultados con la odometría visual, con la ventaja de que ha sido altamente depurado por los desarrolladores de la librería *OpenCV*.

En la figura 5.2 se pueden apreciar los módulos del sistema involucrados en el desarrollo de esta etapa, cada uno de estos hace parte de las clases que ofrece la librería *OpenCV* para la detección y descripción de puntos de interés.

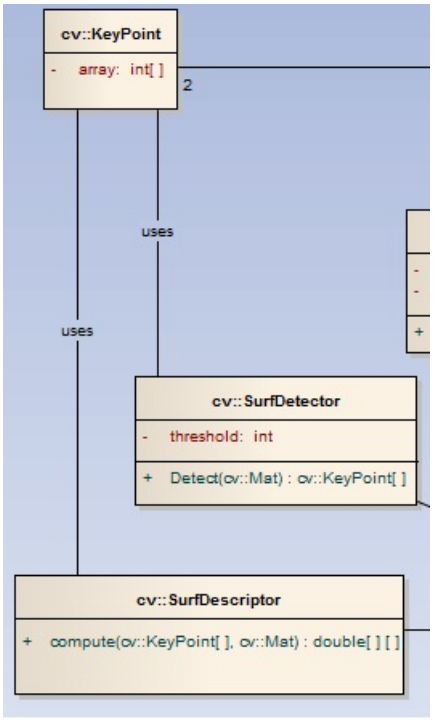


Figura 5.2: Diagrama de módulos de la etapa de extracción característica[6].

- *cv::SurfDetector*: Es una clase que tiene como atributo importante para este trabajo un entero llamado *threshold* o umbral en español. Este valor debe ajustarse para que el detector *SURF* detecte una cantidad determinada de características según la necesidad dado que solo los puntos que generen una respuesta mayor al *threshold* por parte del detector de *SURF* serán considerados como puntos característicos. La función *Detect* se encarga de ejecutar el algoritmo de *SURF* que recibe como

[6] Autores

entrada una imagen de tipo *Mat* y retorna como salida unos puntos de interés de tipo *KeyPoint*.

- *cv::SurfDescriptor*: Es una clase que se compone de un método *Compute* que recibe un arreglo de puntos de interés *KeyPoint[ ]* y una imagen de tipo *Mat*. El método retorna una matriz de punto flotante(puede ser de tipo *Mat*) donde cada fila de la matriz representa un vector que caracteriza a cada punto de interés.

5.3. Emparejamiento de características

En la sección anterior se puntualizó sobre los pobres resultados que se tuvieron con el detector de *Harris* para hacer emparejamiento de características, en la figura 5.4 se puede dar una evidencia de dichos resultados. En la figura 5.3 se puede observar como deberían ser los vectores de flujo óptico si el vehículo se estuviera moviendo en línea recta sobre la carretera y la cámara estuviera apuntando hacia el frente, puede decirse que la cola y cabeza del vector representan la posición de un punto sobre la escena en un instante de tiempo anterior y actual. Una buena estrategia de emparejamiento de características debería generar los vectores de flujo óptico similares a los de la figura 5.3. Sin embargo con el detector de *Harris* y la estrategia de emparejamiento implementada sucedió lo que se ilustra en la figura 5.4(a), es evidente entonces que la estrategia de emparejamiento usada no arrojó los resultados esperados. Por otro lado con la utilización de los módulos ofrecidos por *OpenCV* para establecer correspondencias entre dos imágenes se obtuvo el resultado de la imagen 5.4(b), un resultado más acorde a lo esperado según lo mostrado en la imagen 5.3.

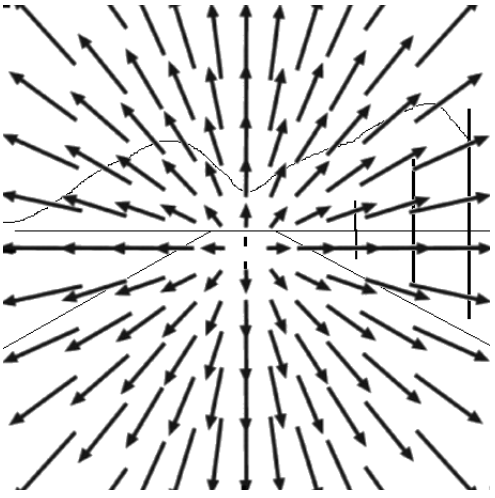


Figura 5.3: Ejemplo de como deberían verse los vectores flujo óptico de un movimiento en línea recta sobre una carretera[7].

Figura 5.4: (a) Imagen de arriba, ejemplo de una estrategia de emparejamiento de características con el detector de *Harris* usando *SAD* como medida de similitud entre un par de características, los segmentos de color negro hacen referencia a los vectores de flujo óptico. (b)Imagen de abajo, ejemplo de una estrategia de emparejamiento de características usando el descriptor *SURF* y los módulos de *OpenCV*[8]

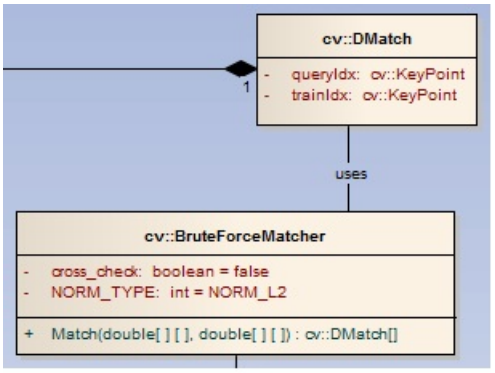


Figura 5.5: Diagrama de módulos de la etapa de emparejamiento de características[9].

Por tal motivo se decidió usar el descriptor *SURF* así como el método de emparejamiento de características que ofrece *OpenCV* para este propósito. En la figura 5.5 se muestra el módulo que se encarga del emparejamiento de características.

[7] Autores

[8] (Andreas Geiger, Philip Lenz y Raquel Urtasun. «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite». En: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012)

[9] Autores

- *cv::BruteForceMatcher* : Este módulo o clase se compone de dos atributos; primero, *cross_check*, que es una variable booleana que indica si debe o no hacerse Chequeo de Consistencia Mutua(*Mutual Consistency Check, MCC*,ver sección 3.4.3); segundo, *NORM_TYPE*, que es una variable de tipo entera que indica el tipo de norma (distancia euclideana, distancia $Manhattan$, etc.) que se usará para comparar que tan similar es una característica con otra. Por último el método *Match* recibe 2 parámetros, primero, el conjunto de descriptores que genera *cv::SurfDetector.Compute* en la primera imagen; segundo, el conjunto de descriptores que genera *cv::SurfDetector.Compute* en la segunda imagen, este método retorna un arreglo del tipo *cv::DMatch* explicado anteriormente.

De esa manera se tienen las correspondencias entre las dos imágenes, luego de esto se procede entonces a realizar una estimación del movimiento. Una aproximación al funcionamiento del método *Match* de la clase *cv::BruteForceMatcher* se muestra en el algoritmo 5.1. En este algoritmo $\phi(x, y)$ hace referencia a la medida de similitud entre dos puntos característicos o de interés.

5.4. Estimación del movimiento

Para la estimación de movimiento se expusieron diferentes metodologías para lograr este cometido, primero, metodología 3D-3D, segundo, metodología 2D-2D y tercero, metodología 3D-2D. Con base a este estudio detallado de cada una de las perspectivas, se tomó la determinación de usar la segunda metodología, ya que según algunos autores (ver sección 4.4) es una metodología que ha dado buenos resultados, un ejemplo de ello es el trabajo desarrollado por *Nister et al.*[10].

El módulo de estimación de movimiento puede apreciarse en la figura 5.6 . A continuación se dará una explicación general sobre este módulo y luego se detallará el algoritmo usado para la estimación de movimiento.

El algoritmo usado para estimar el movimiento es conocido dentro de la comunidad de la visión por computador como el algoritmo normalizado de los ocho puntos[12] (Algoritmo 5.2).

El módulo de estimación de movimiento descrito en esta sección implementa en esencia el algoritmo normalizado de los ocho puntos, excepto por la parte de la normalización y

[10]NISTER, D. Et. Al. Óp. cit.
[11]Autores
[12]Hartley, óp. cit.

Algoritmo 5.1 :Algoritmo para establecer correspondencias entre características de dos imágenes(Método *Match*)

Entrada: Dos imágenes I_k e I_{k-1}, dos conjuntos de descriptores de características F_k y F_{k-1}.

1: **for** $i = 0$ **to** $tam(F_{k-1})$ **do**
2: declarar $menor = \infty$
3: **for** $j = 0$ **to** $tam(F_k)$ **do**
4: **if** $\phi(F_{k-1}^i, F_k^j) < menor$ **then**
5: Establezca como pareja actual para F_{k-1}^i a F_k^j
6: $menor = \phi(F_{k-1}^i, F_k^j)$
7: **end if**
8: **end for**
9: **end for**
10: **for** $i = 0$ **to** $tam(F_k)$ **do**
11: declarar $menor = \infty$
12: **for** $j = 0$ **to** $tam(F_{k-1})$ **do**
13: **if** $\phi(F_{k-1}^j, F_k^i) < menor$ **then**
14: Establezca como pareja actual para F_k^i a F_{k-1}^j
15: $menor = \phi(F_{k-1}^j, F_k^i)$
16: **end if**
17: **end for**
18: **end for**
19: **for** $i = 0$ **to** $tam(F_{k-1})$ **do**
20: **if** la pareja de F_{k-1}^i es un F_k^j y viceversa **then**
21: Establezca una correspondencia entre F_{k-1}^i y F_k^j
22: **end if**
23: **end for**

Salida: Una lista C de correspondencias del tipo *cv::DMatch*.

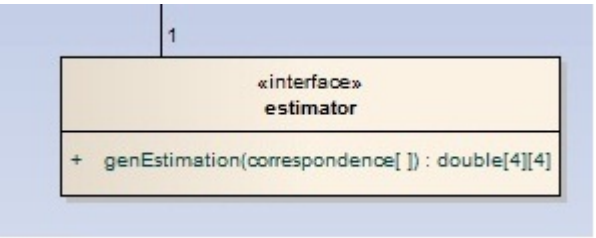


Figura 5.6: Diagrama del módulo de estimación de movimiento[11].

Algoritmo 5.2 :Algoritmo normalizado de los ocho puntos

Entrada: $n \geq 8$ correspondencias $\{x_i \longleftrightarrow x_i'\}$ entre dos imágenes I_k e I_{k-1}.

1: **Normalización:** Transformar las coordenadas de la imagen por medio de una transformación T y T' tal que, $\hat{x_i} = Tx_i$ y $\hat{x_i'} = T'x_i'$, donde T y T' son transformaciones de traslación y escalamiento que buscan que el origen del sistema de coordenadas donde están descritos los puntos x_i y x_i' coincida con el centroide de dichos puntos y que el promedio de la distancia de los puntos x_i y x_i' al origen sea $\sqrt{2}$.

2: Halle la matriz esencial $\hat{E}$ correspondiente a $\hat{x_i} \longleftrightarrow \hat{x_i'}$ usando:

3: **Solución lineal**: Determinar la matriz esencial $\hat{E}'$ usando el método propuesto en la sección 3.4.4 **Estimación de la matriz esencial**.

4:

5: **Reforzar la restricción:** Reemplazar $\hat{E}$ por $\hat{E}'$ de tal forma que $det(\hat{E}') = 0$ usando SVD como se detalló en la sección 3.4.4.

6:

7: **Denormalización:** Declare $E = T'^T \hat{E} T$.

Salida: Una matriz esencial E tal que se cumpla restricción epipolar planteada en 3.39.

denormalización que se implementan en el módulo de la estimación robusta (ver sección 5.5), por lo tanto el algoritmo asume que las n correspondencias dadas como parámetro de entrada han sido previamente normalizadas. Los detalles de esta normalización se detallarán en la sección 5.5.

En el estudio comparativo de la sección 4.4 se mostraron distintas alternativas para desarrollar este módulo y en la sección 3.4.4 se expusieron dos alternativas para solucionar el problema de la estimación de movimiento dadas dos imágenes, usando el algoritmo de los ocho puntos de *longuet y higgins*, y el algoritmo de los cinco puntos de *Nister*. Las razones por las que se escogió el algoritmo de los ocho puntos son principalmente :

- El algoritmo de los ocho puntos es mucho más sencillo en su implementación ya que solo exige resolver un sistema de ecuaciones homogéneo, a diferencia del algoritmo de los cinco puntos que implica hallar solución a un sistema de ecuaciones no lineales.
- A pesar de que el algoritmo de los cinco puntos es más eficiente según el estudio comparativo anterior, la ganancia en precisión para localizar el vehículo sería mínima,

ya que ambos algoritmos tratan de resolver el mismo problema de estimar la matriz esencial. Como primer acercamiento a la construcción de un sistema de odometría visual se justifica el desarrollo de un sistema que permita entender las bases teóricas en esta área de conocimiento y el algoritmo de *Longuet y Higgins* permite dicho propósito.

5.5. Estimación robusta

El algoritmo escogido para eliminar las correspondencias erróneas es *RANSAC* dado su extenso uso en visión por computador para eliminación de *outliers*. Este algoritmo se describe de manera detallada en la sección 3.4.5. Teniendo en cuenta la descripción general de *RANSAC* dada en la sección 3.4.5 quedan algunos parámetros y elementos que deben definirse de manera específica para el sistema de odometría visual descrito en este capítulo.

- Parámetro s: La cardinalidad del conjunto A' que se escoge aleatoriamente en cada iteración del *RANSAC*. Dado que el algoritmo que se está usando para estimar el movimiento es el algoritmo de los 8 puntos normalizado, la elección para este parámetro es $s = 8$, es posible utilizar un s mayor pero esto implica aumentar el número de iteraciones para que *RANSAC* garantice una buena estimación robusta de acuerdo a la ecuación 3.50.
- Función de Error: Esta función define matemáticamente que tan consistente es una correspondencia con respecto a una instancia particular de un modelo de movimiento. La función de error elegida para el sistema de odometría visual implementado es el error de *Sampson* que será descrito en 5.5.1.
- Umbral d: Este parámetro define de manera exacta cuando un dato particular se considera un *inlier* para un modelo de movimiento particular. Si la función de error evaluada en una correspondencia determinada se encuentra por debajo del umbral d entonces esa correspondencia se considerará un *inlier*. Para la implementación se tomó como umbral $d = 0{,}00001$, dado que se obtuvo una proporción de *inliers* mayor que la de *outliers* con esa elección de d.

5.5.1. Error de Sampson

En la práctica las correspondencias obtenidas $\{x_i \longleftrightarrow x_i'\}$ no cumplen la restricción epipolar de manera exacta, ecuación 3.39. Por lo tanto es necesario definir una métrica que

de cuenta acerca de que tan consistente es una correspondencia con la restricción epipolar dada una matriz esencial E determinada. Comúnmente, el error de reproyección es usado como métrica, el error de reproyección se define para este caso como: encontrar $\hat{x}_i$ y $\hat{x'_i}$ tal que la función C en la ecuación 5.1 sea mínima.

$$C(x_i, x'_i) = d(x_i, \hat{x}_i)^2 + d(x'_i, \hat{x'_i})^2 \text{ sujeto a } \hat{x'_i}^T E\hat{x}_i = 0 \tag{5.1}$$

Donde d representa la distancia euclidiana. Existen diversas alternativas para minimizar la ecuación 5.1, pero bajo ciertas circunstancias hay aproximaciones al error de reproyección que resultan ser útiles en la práctica, una de estas aproximaciones será utilizada en el desarrollo de este proyecto y es conocida como el error de *Sampson*.

El error de *Sampson* es una aproximación de primer orden al error de reproyección y se define en la ecuación 5.2.

$$Sampson(x_i, x'_i) = \frac{(x'^T_i E x_i)^2}{(Ex_i)_1^2 + (Ex_i)_2^2 + (E^T x'_i)_1^2 + (E^T x'_i)_2^2} \tag{5.2}$$

Donde $(Ex_i)_k$ corresponde a la k-ésima componente de la multiplicación entre la matriz esencial E y el punto de interés x_i en la primera imagen. $(E^T x'_i)_k$ corresponde a la k-ésima componente de la multiplicación entre la matriz esencial E^T y el punto de interés x'_i en la segunda imagen. De esa manera el error de *Sampson* está representado por un escalar.

Esta aproximación es rápida de calcular y presenta bueno resultados siempre y cuando los errores inducidos por la cámara sean pequeños[13].

5.5.2. Normalización

Como se explicó con anterioridad es necesario realizar una normalización sobre los pares de características correspondientes (x_i, x'_i) de la forma $\hat{x}_i = Tx_i$ y $\hat{x'_i} = T'x'_i$ para evitar inestabilidad numérica en los cálculos necesarios para estimar el movimiento entre fotogramas. Esta normalización se implementa en el módulo que se está describiendo, por lo tanto a continuación se detallará la manera de calcular T y T'.

El procedimiento para calcular T' es análogo al procedimiento para determinar T. Para calcular la matriz de normalización T debe tenerse en cuenta que con esta transformación se buscan básicamente dos cosas[14]:

[13] Hartley y Zisserman, óp. cit.
[14] Hartley, óp. cit.

1. El origen del sistema de coordenadas donde están descritos los puntos x_i debe coincidir con el centroide de los puntos x_i.

2. La distancia promedio al origen de los puntos x_i debe ser $\sqrt{2}$

Teniendo en cuenta los puntos anteriores la Transformación T tiene la siguiente forma:

$$T = \begin{bmatrix} k & 0 & -Centroide_x * k \\ 0 & k & -Centroide_y * k \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

Donde $(Centroide_x, Centroide_y)$ representan las coordenadas del centroide de los puntos x_i y k es el factor de escala calculado como sigue:

$$k = \frac{\sqrt{2} * N}{\sum_{i=0}^{i<N} \|x_i - Centroide\|} \tag{5.4}$$

N en 5.4 representa la cantidad de puntos x_i.

Es importante mencionar que dado que la matriz esencial obtenida por el módulo de estimación se calcula sobre las correspondencias normalizadas es necesario recuperar la matriz esencial en el espacio de las coordenadas no normalizadas. Lo anterior se logra tomando $T'^T ET$ donde E es la matriz Esencial estimada en el espacio de las coordenadas normalizadas.

5.6. Módulo de desambiguación

Dentro de las etapas de construcción de un sistema de odometría visual no se encuentra puntualizada la desambiguación como una de ellas, sin embargo, en la implementación del sistema si se constituye como un módulo diferenciado debido a que deben hacerse algunas operaciones importantes para el algoritmo de odometría visual. En la figura 5.7 se puede apreciar este módulo. A continuación se da una explicación del funcionamiento de este.

El módulo *DisambiguationAlgorithm* solo se compone de un método llamado *GetRotationAndTraslation*, este método pretende extraer la rotación y la traslación a partir de una matriz esencial computada en la etapa de estimación robusta de la sección anterior. En el algoritmo 5.3 se describe el funcionamiento del método *GetRotationAndTraslation*.

[15] Autores

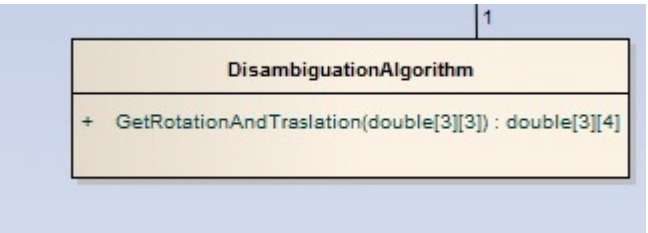


Figura 5.7: Diagrama del módulo que corresponde a la desambiguación[15].

Algoritmo 5.3 :Método para extraer R y $\vec{t}$ de la matriz esencial

Entrada: Una Matriz esencial E.

1: Extraer las cuatro posibles combinaciones $P'_1 = [R_1|t_1], P'_2 = [R_1|t_2], P'_3 = [R_2|t_1] y P'_4 = [R_2|t_2]$ de R y $\vec{t}$ como se específico en la sección 3.4.4(**Extracción de R y $\vec{t}$ de la matriz esencial**).

2: Asumir que la matriz de proyección P de la cámara en la posición anterior C_{k-1} es una matriz de identidad con centro de proyección en el origen, de esa manera $P = [I|0]$

3: Triangular las correspondencias entre las dos imágenes I_k e I_{k-1} usando P con cada una de las combinaciones de P' .

4: Declarar P' a la matriz P'_k que en conjunto con P haya dejado más puntos triangulados al frente de ambas imágenes I_k e I_{k-1}.

Salida: Una matriz de proyección $P' = [R|t]$ para una cámara en la posición C_k.

Para resolver el problema de triangular correspondencias entre dos imágenes I_k e I_{k-1} conociendo sus respectivas matrices de proyección P' y P se debe plantear como un problema de mínimos cuadrados. Una solución a este problema puede encontrarse en el libro de *Hartley*[16].

5.6.1. Hallando la escala real del movimiento

En la sección 3.4.4 se puntualizó sobre el problema de encontrar la escala real del movimiento en un sistema de odometría visual, en dicha explicación se mencionaron tres alternativas para resolver este problema. A continuación se explicará más en detalle una de dichas metodologías, la cuál fue implementada en el desarrollo de este sistema.

El problema de hallar la escala real del movimiento de una cámara sobre un vehículo puede resolverse teniendo conocimiento sobre la escena, por ejemplo, conociendo el tamaño de los objetos en la escena o conociendo las características del vehículo en el cuál se encuentra empotrada la cámara. *Andreas Geiger* propone una metodología[17] para hallar la escala real del movimiento conociendo la altura del vehículo y el ángulo de inclinación de la

[16] Hartley y Zisserman, óp. cit.

[17] Andreas Geiger, Julius Ziegler y Christoph Stiller. «StereoScan: Dense 3d Reconstruction in Real-time». En: *Intelligent Vehicles Symposium (IV)*. 2011.

cámara con respecto a la horizontal, que para el caso dicha horizontal correspondería al suelo.

La metodología propuesta por *Geiger* se fundamenta en el hecho de que si el sistema de detección de puntos de interés detecta puntos en el suelo, la altura y de dichos puntos sería igual a la distancia de la cámara al suelo, esto se cumple si la cámara estuviera mirando hacia el frente(eje z de la cámara alineado con la horizontal que es el suelo), pero en el modelo real, la cámara tiene un ángulo de inclinación con respecto a la horizontal como se muestra en la figura 5.8(a). Esto causa que los puntos de interés que están en el suelo no tengan una coordenada en y común, a pesar de que son colineales.Para corregir este último problema la cámara debe rotarse un ángulo α para que el eje z quede alineado con el suelo. De esta forma todos los puntos detectados en el suelo tendrán una altura y común que indicará la altura de la cámara como se muestra en la figura 5.8(b).

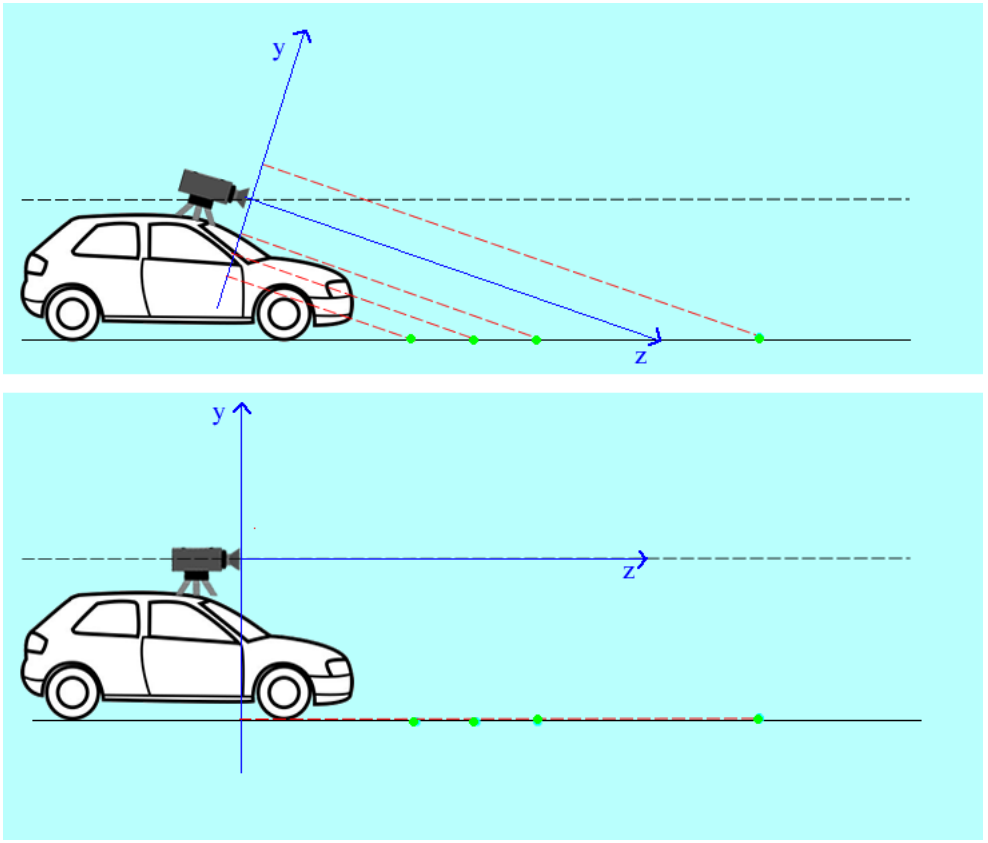


Figura 5.8: (a) Imagen de arriba, corresponde a la configuración real de la cámara con un ángulo α de rotación respecto a la horizontal que es el suelo. Los puntos de color verde corresponden a los puntos de interés, las líneas rojas punteadas hacen referencia a la altura y de cada uno de los puntos de interés y las líneas azules representan el sistema de coordenadas de la cámara. (b) Imagen de abajo, configuración "virtual"de la cámara luego de rotar un ángulo α quedando el eje z alineado con la horizontal que es el suelo[18].

Si se conoce la altura real de la cámara H y si se tiene una altura "virtual"de la cámara h se puede decir que la escala real Γ del movimiento es aproximadamente:

$$\Gamma \approx H/h \tag{5.5}$$

Como ya se sabe que coordenada en y tiene cada punto de interés detectado en el suelo gracias al procedimiento anterior, sería natural concluir que la altura y de cualquiera de estos puntos corresponde a la altura "virtual"h, sin embargo, en la práctica no es recomendable hacer esto debido a errores que hayan podido darse en cualquiera de las etapas anteriores del *pipeline* de la odometría visual, por lo tanto a continuación se mencionan 3 estrategias para resolver este problema, la segunda y tercera estrategia son propuestas por los autores del trabajo expuesto:

1. De todos los puntos de interés que yacen en el suelo, encontrar el punto x_i que agrupe a la mayor cantidad de puntos de interés a su alrededor, método propuesto por *Geiger*.
2. Encontrar el punto de interés x_i sobre el suelo cuya coordenada $|y|$ sea la mayor posible.
3. Encontrar el promedio de todas las alturas $|y|$ de los puntos x que están sobre el suelo.

Aplicada alguna de estas metodologías se tendrá entonces la altura virtual h de la cámara y se podrá dar una aproximación a la escala real como se indicó en la ecuación 5.5. Para la implementación de este trabajo se usa la tercera metodología ya que presentó una estimación más cercana al movimiento real de la cámara que el método propuesto por *Geiger*, debido tal vez a problemas en la parte de emparejamiento de características.

5.7. Sistema de localización

Este módulo no representa una de las etapas para la construcción de un sistema de odometría visual. El módulo del sistema de localización es el módulo principal que se encarga de llamar a los otros descritos anteriormente. A continuación se da una descripción de su funcionamiento. En la figura 5.9 se puede apreciar un vistazo general a este módulo.

[18] Autores
[19] Autores

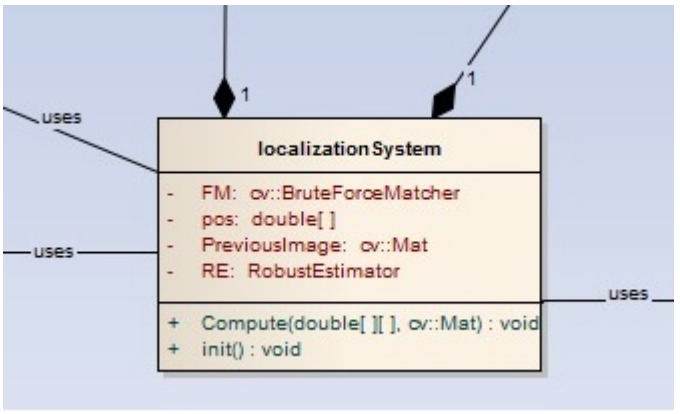


Figura 5.9: Diagrama del módulo que corresponde al sistema de localización[19].

Este módulo además de encargarse de integrar todos los otros módulos del sistema, se encargará de las siguientes tareas:

- Cargar cada uno de los fotogramas I_k de una secuencia de vídeo o de imágenes.
- Multiplicar cada punto característico x_i de una imagen I_k por la inversa de la matriz de calibración de la cámara K^{-1}, esto con el objetivo de transformar los puntos característicos x_i al sistema de coordenadas de la cámara, dicha matriz de calibración se compone de unos parámetros intrínsecos conocidos como el centro de la imagen(c_x, c_y) y la distancia focal f_x, f_y.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

  Lo anterior se debe a que el sistema de coordenadas en el cuál están expresados los puntos característicos sobre una imagen I_k es el espacio de los píxeles, que no es el mismo espacio planteado en el sistema de coordenadas de un modelo de cámara *pinhole*.
- Extracción de R y $\vec{t}$ de la Matriz esencial que retorna el módulo *RobustEstimator*(5.5) usando el método descrito en la sección 3.4.4.
- Cálculo de la posición c_k de la cámara.

Capítulo 6

Resultados

A continuación se presentan los resultados obtenidos por el sistema de odometría visual propuesto como solución al problema descrito en la sección 1.1. Inicialmente se describe el entorno de pruebas en el que se validará el sistema de odometría visual, especificando las variables que se medirán y detallando la manera como se capturará la información. Finalmente se expondrán los resultados obtenidos y adicionalmente se mostrarán algunas estimaciones hechas por un sistema de odometría visual similar (monocular y con estimación $2D - 2D$) conocido como $LIBVISO2$[1].

6.1. Descripción del entorno de pruebas

Para validar el sistema de odometría visual desarrollado se usará un *Dataset* del *Karlsruhe Institute of Technology* conocido como *The KITTI Vision Benchmark Suite*[2]. El objetivo del *KITTI Dataset* es brindar un marco común de evaluación para algoritmos de visión por computador usados en vehículos autónomos y que realizan tareas como emparejamiento estereoscópico, estimación de flujo óptico, detección de objetos, odometría, detección de las líneas de la carretera, rastreo de objetos, etc.

Para la evaluación de algoritmos de odometría visual *KITTI Dataset* proporciona un conjunto de secuencias de imágenes capturadas en la ciudad de *Karlsruhe* en ambientes rurales y urbanos. Las secuencias de imágenes mencionadas están acompañadas de su respectivo *Ground Truth*. El *Ground Truth* representa la referencia con la que se compararán las estimaciones hechas por el algoritmo de odometría visual evaluado.

[1] Andreas Geiger. *LIBVISO2: C++ Library for Visual Odometry 2*. [citado en 2013]. URL: `http://www.cvlibs.net/software/libviso`.

[2] Geiger, Lenz y Urtasun, óp. cit.

La plataforma utilizada en la captura de información para el *Dataset* se muestra en la figura 6.1. Este vehículo se encuentra equipado con dos cámaras a color, dos cámaras en escala de grises (*PointGrey Flea2*) un escáner Láser y un *GPS/IMU* para la localización. Es importante mencionar que el *Ground Truth* del *Dataset* es generado a partir del *GPS/IMU* que cuenta con correcciones *RTK* y que permite obtener una estimación de la posición con un error menor a 10 cm.

Figura 6.1: Plataforma de captura de datos *KITTI Dataset*[3].

Para la presente evaluación se usarán únicamente las imágenes obtenidas directamente en escala de grises por la cámara izquierda. Lo anterior debido a que estas imágenes presentan menos sensibilidad al brillo y al ruido. Es importante mencionar que las imágenes están rectificadas y que la información asociada a los parámetros intrínsecos de la cámara que las capturó está disponible.

Las variables que se medirán en la evaluación serán las siguientes:

- Diferencia de la trayectoria estimada con respecto al *Ground Truth*(D).Esta variable se puede ver como una función del tiempo y representa el error en la estimación de la localización del vehículo por parte del sistema de odometría visual en un instante de tiempo determinado. Para la prueba realizada en este proyecto se tomará el valor de la variable D en algunos puntos sobre la trayectoria (*landmarks*).

- $E_{rotation}$: Esta variable representa el error de rotación (en radianes sobre metros) en función tanto de la longitud de la trayectoria como en función de la velocidad. Para calcular este error se usó la metodología sugerida por Geiger et. al[4] que se resume

[3] (ibíd.)
[4] Ibíd.

para la rotación en la fórmula siguiente:

$$E_{rot}(F) = \frac{1}{|F|} \sum_{(i,j)\in F} < [(\hat{p_j} \ominus \hat{p_i}) \ominus (p_j \ominus p_i)] \tag{6.1}$$

Donde $< [.]$ representa el ángulo de rotación. F es un conjunto de fotogramas. $\hat{p_j}$ y $\hat{p_i}$ son transformaciones de cuerpo rígido que representan la posición real del vehículo *Ground Truth*. p_j y p_i son transformaciones de cuerpo rígido que representan la posición del vehículo estimada por el sistema de odometría visual y $\ominus$ representa el inverso del operador de composición de movimiento. La anterior notación es igualmente válida para la fórmula 6.2.

- $E_{traslation}$: Esta variable representa el error de traslación (como porcentaje de la trayectoria recorrida) en función tanto de la longitud de la trayectoria como en función de la velocidad. Para calcular este error se usó la metodología sugerida por Geiger et. al[5] que se resume para la traslación en la fórmula siguiente:

$$E_{trans}(F) = \frac{1}{|F|} \sum_{(i,j)\in F} \|(\hat{p_j} \ominus \hat{p_i}) \ominus (p_j \ominus p_i)\| \tag{6.2}$$

- Tiempo de ejecución (T). Esta variable representa el tiempo que tarda en procesar un nuevo *frame* el sistema de odometría visual descrito, es decir, T cuantifica el tiempo empleado por el sistema desde la detección de puntos de interés hasta el cálculo de la transformación de cuerpo rígido que tuvo lugar entre el *frame* anterior y el actual.

Las gráficas utilizadas en las siguientes pruebas para ilustrar las trayectorias estimadas y los errores fueron generadas con el kit de desarrollo del *KITTI Dataset*.

6.2. Prueba No. 1

En esta prueba se usará la secuencia de imágenes No. 4 en el *benchmark* de odometría visual del *KITTI Dataset*. Esta secuencia fue capturada en un ambiente de carretera y se compone de 271 fotogramas que representan aproximadamente 27 segundos de recorrido. Para esta prueba es importante mencionar que el automóvil se desplaza a una velocidad entre los 60 y 70 Kilómetros por hora.

En la figura 6.2 se describe la trayectoria estimada junto con el *Ground Truth*.

[5] Ibíd.
[6] Autores

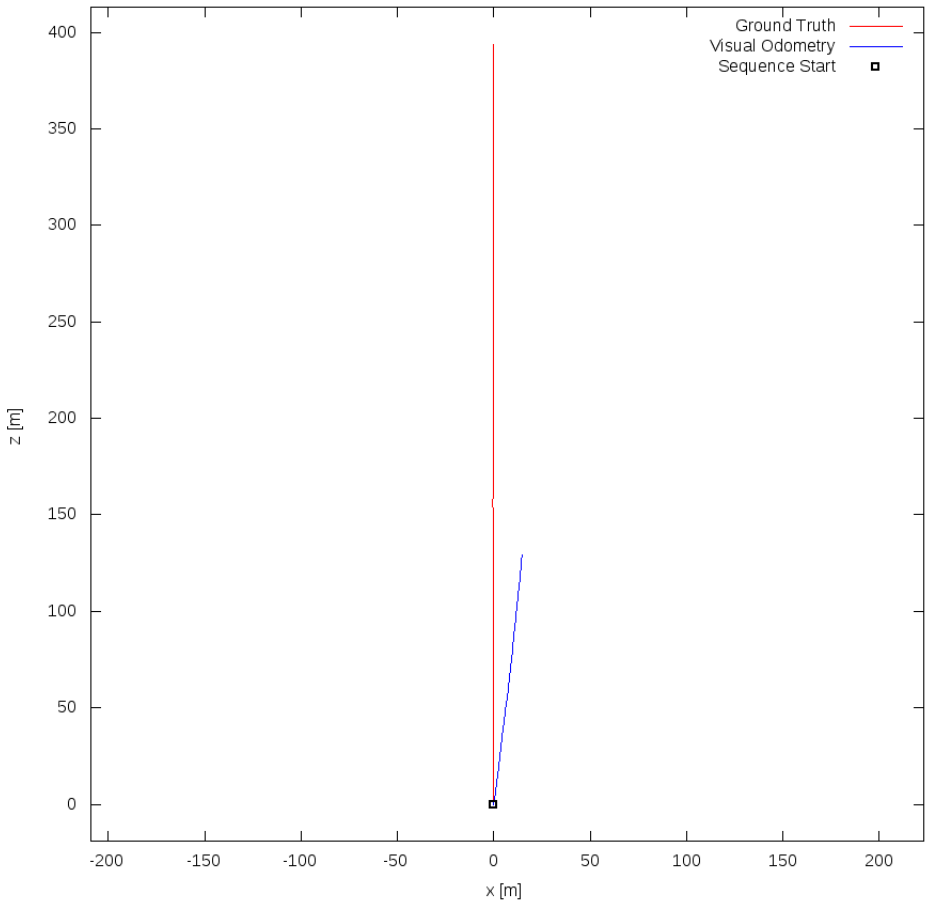


Figura 6.2: Estimación de la Trayectoria, Prueba No . 1^6.

La tabla 6.1 pretende mostrar la diferencia entre la estimación del sistema de odometría visual diseñado y el *Ground Truth* para ciertos fotogramas de la secuencia de imágenes de esta prueba. El error (D) se muestra en metros y es calculado como la distancia euclidiana entre la estimación y el *Ground Truth*.

Fotograma	x (Ground Truth)	z (Ground Truth)	x (estimación)	y (estimación)	D (Error en m.)
0	-0.0	0.0	0.0	0.0	0.0
6	0.009175	7.84547	0.41208	3.16698	4.69580675913
12	-0.002514	15.92862	0.913855	6.94883	9.02642568176
18	-0.00225	24.0029	1.21571	9.30909	14.7442015341
24	0.013601	32.33003	1.60034	12.4023	19.9908019751
30	0.027945	40.63325	1.94202	15.1509	25.554135568
36	0.043871	49.17761	2.29295	17.9667	31.2918401405
42	0.025418	57.67805	2.61292	20.5639	37.2042376192
48	0.007883	66.2914	2.88666	22.9374	43.4494726437
54	-0.030918	74.68229	3.1924	25.596	49.1920079372
60	-0.031054	83.10843	3.40843	27.5605	55.6543131974
66	-0.091427	91.3241	3.68386	29.749	61.6907264663
72	-0.213893	99.53478	3.9147	31.7836	67.8768566711
78	-0.311467	107.5272	3.99006	32.4626	75.1877470582
84	-0.405501	115.5882	4.28302	34.8567	80.8675294628
90	-0.430087	123.5917	4.62874	37.8564	85.8844188238
96	-0.442347	131.7929	4.9239	40.3677	91.5825518421
102	-0.434855	139.838	5.27976	42.9783	97.0281315325
108	-0.453357	147.9966	5.69088	45.8432	102.33801239
114	-0.486234	156.0121	6.15688	49.4058	106.813080488
120	-0.44206	164.1736	6.63664	52.728	111.670182918
126	-0.375204	172.2651	6.94646	55.4035	117.090735408
132	-0.275615	180.5915	7.29422	58.1377	122.687552492
138	-0.269386	188.8851	7.47404	59.6943	129.422654319
144	-0.254337	197.4375	7.74323	62.2007	135.473071686
150	-0.304994	205.9635	8.00902	64.6101	141.597692497
156	-0.312644	214.638	8.30237	67.3514	147.53833741
162	-0.32871	223.5893	8.68609	70.8175	153.037542761
168	-0.325921	232.474	8.78864	71.821	160.911347118
174	-0.331511	241.5849	9.03097	74.121	167.725412069
180	-0.313505	250.6399	9.5067	78.753	172.167194372
186	-0.286234	259.9259	9.95019	83.0358	177.186037414
192	-0.261066	269.1006	10.2687	86.2451	183.158428286
198	-0.247701	278.4458	10.5052	88.5632	190.186820423
204	-0.169993	287.6804	10.6928	90.306	197.673098947
210	-0.106495	296.9895	10.997	93.0251	204.26640465
216	-0.047023	306.6127	11.3859	96.5595	210.364109483
222	-0.041775	316.2361	11.655	99.1409	217.410074303
228	-0.028164	326.0999	12.0156	102.303	224.120736883
234	0.015848	335.7674	12.4071	106.082	230.019403745
240	0.037198	345.506	12.795	109.853	235.998088808
246	-0.000748	355.0231	13.1335	112.999	242.380224959
252	-0.065121	364.5015	13.4028	115.686	249.179730187
258	-0.159178	374.1776	13.7745	119.587	254.971608206
264	-0.215444	383.7599	14.3194	125.312	258.856289675
270	-0.241436	393.5672	14.6209	129.122	264.862516854

Cuadro 6.1: Tabla de errores para la Prueba No. 1

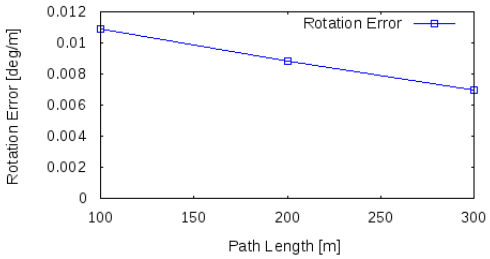


Figura 6.3: Error de rotación en función de la longitud de la trayectoria, Prueba No . 1.

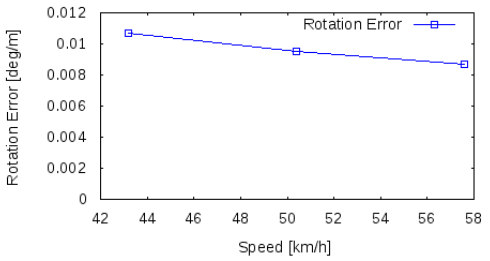


Figura 6.4: Error de rotación en función de la velocidad del vehículo, Prueba No . 1.

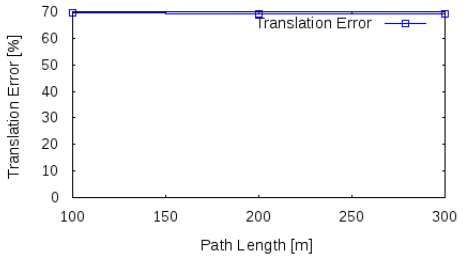


Figura 6.5: Error de traslación en función de la longitud de la trayectoria, Prueba No . 1.

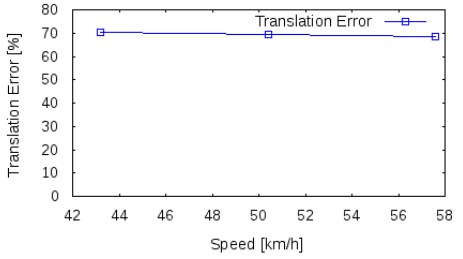


Figura 6.6: Error de traslación en función de la velocidad del vehículo, Prueba No . 1.

6.3. Prueba No. 2

En esta prueba se usará la secuencia de imágenes No. 7 en el *benchmark* de odometría visual del *KITTI Dataset*. Esta secuencia fue capturada en un ambiente residencial y se compone de 1101 fotogramas que representan aproximadamente 1:50 minutos de recorrido.

A continuación se describe la trayectoria estimada junto con el *Ground Truth* de la prueba No. 2.

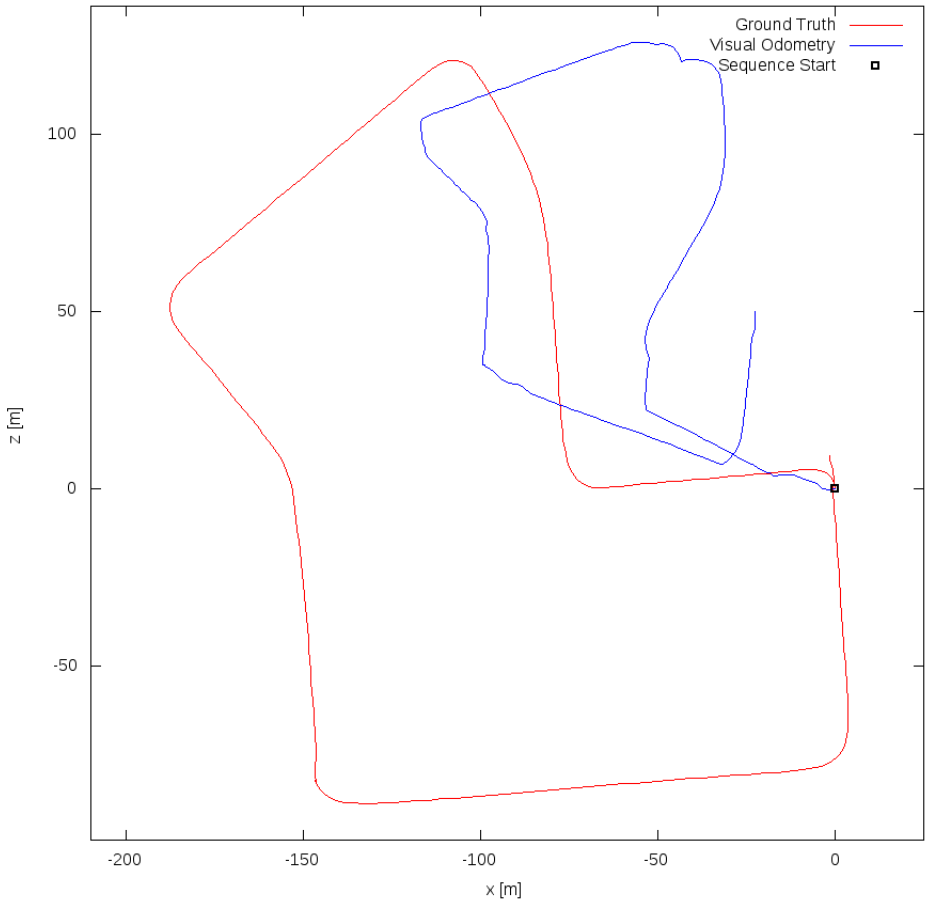


Figura 6.7: Estimación de la Trayectoria, Prueba No . 2.

La tabla 6.2 pretende mostrar la diferencia entre la estimación del sistema de odometría visual diseñado y el *Ground Truth* para ciertos fotogramas de la secuencia de imágenes de esta prueba.

Fotograma	x (Ground Truth)	z (Ground Truth)	x (estimación)	y (estimación)	D (Error en m.)
0	-0.0	0.0	0.0	0.0	0.0
27	-2.416226	4.382651	-5.27882	1.47348	4.08138705786
54	-14.92181	4.699036	-18.8751	4.61856	3.95410902868
81	-36.55351	2.85037	-30.2514	10.5075	9.91706762551
108	-58.30125	0.901153	-44.2824	17.7473	21.9162228513
135	-72.8175	2.626877	-53.2354	29.0959	32.9251851777
162	-77.19076	17.98524	-51.6162	49.1539	40.3180292845
189	-78.69059	38.77605	-45.1076	60.4894	39.9910838258
216	-80.45357	61.6908	-38.0159	72.8841	43.8890168484
243	-83.8341	82.45883	-32.9616	83.5045	50.8832455922
270	-91.82699	101.3333	-31.1198	97.5802	60.8230932895
297	-100.5981	116.2301	-32.2293	114.783	68.3841130077
324	-105.7607	120.4782	-41.7601	120.904	64.0020164214
351	-116.9578	115.7594	-57.1001	125.657	60.6704766344
378	-135.1821	100.4101	-71.3452	120.933	67.0547479752
405	-155.5906	83.26245	-90.5097	114.005	71.9765790053
432	-174.7967	67.24066	-104.1	109.038	82.1281987016
459	-187.1169	54.6867	-115.702	104.676	87.1723468567
486	-183.9667	42.92145	-115.005	93.611	85.5870699703
513	-171.8862	28.42544	-108.814	87.4409	86.376657332
540	-156.6568	8.9962	-101.688	80.5728	90.2484273603
567	-151.3708	-14.6721	-97.7775	65.0133	96.031269793
594	-148.6776	-42.57269	-97.9803	54.8683	109.840624359
621	-146.7032	-68.18509	-98.7976	42.3315	120.452742503
648	-146.6806	-80.70329	-99.0555	34.8443	124.977580809
675	-146.5123	-82.10576	-99.0555	34.8443	126.211982
702	-146.5095	-82.09448	-99.0555	34.8443	126.200476956
729	-146.5765	-82.53671	-99.0555	34.8443	126.635488508
756	-141.5329	-87.50698	-92.2074	29.8898	127.338167509
783	-118.1326	-88.01078	-80.5051	24.7478	118.871048283
810	-86.3386	-85.48913	-66.7412	19.8175	107.114632094
837	-56.57033	-82.91791	-53.9377	15.2289	98.1821116797
864	-28.16153	-80.71248	-43.8052	11.4344	93.4653513598
891	-5.482571	-78.54777	-33.2569	7.26309	90.1937749811
918	3.315913	-69.59111	-28.9142	9.53088	85.4345918557
945	3.052904	-51.88786	-25.9665	18.1048	75.770035386
972	1.637847	-32.57166	-25.0317	26.3129	64.6425258137
999	0.596981	-17.20095	-24.0247	36.1739	58.7801138804
1026	0.087923	-9.181351	-23.8036	38.0995	52.9743687319
1053	-0.465636	-2.543009	-23.8454	38.3573	47.1110246226
1080	-1.228194	7.291404	-22.6154	47.378	45.4350939182

Cuadro 6.2: Tabla de errores para la Prueba No. 2

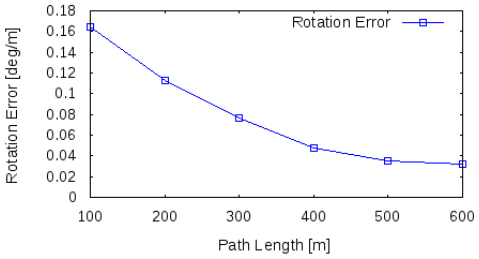


Figura 6.8: Error de rotación en función de la longitud de la trayectoria, Prueba No . 2.

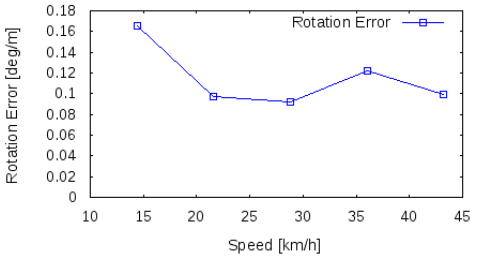


Figura 6.9: Error de rotación en función de la velocidad del vehículo, Prueba No . 2.

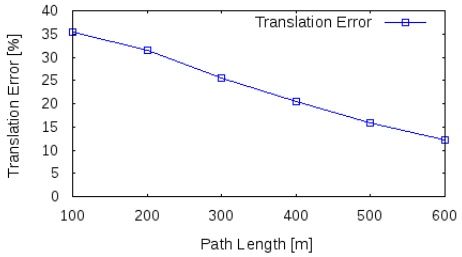


Figura 6.10: Error de traslación en función de la longitud de la trayectoria, Prueba No . 2.

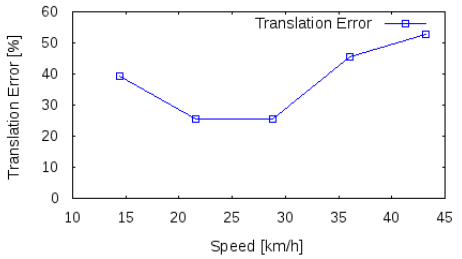


Figura 6.11: Error de traslación en función de la velocidad del vehículo, Prueba No . 2.

6.4. Prueba No. 3

En esta prueba se usará la secuencia de imágenes No. 6 en el *benchmark* de odometría visual del *KITTI Dataset*. Esta secuencia, al igual que en la prueba No. 2, fue capturada

en un ambiente residencial y se compone de 1101 fotogramas que representan aproximadamente 1:50 minutos de recorrido.

A continuación se describe la trayectoria estimada junto con el *Ground Truth* de la prueba No. 3.

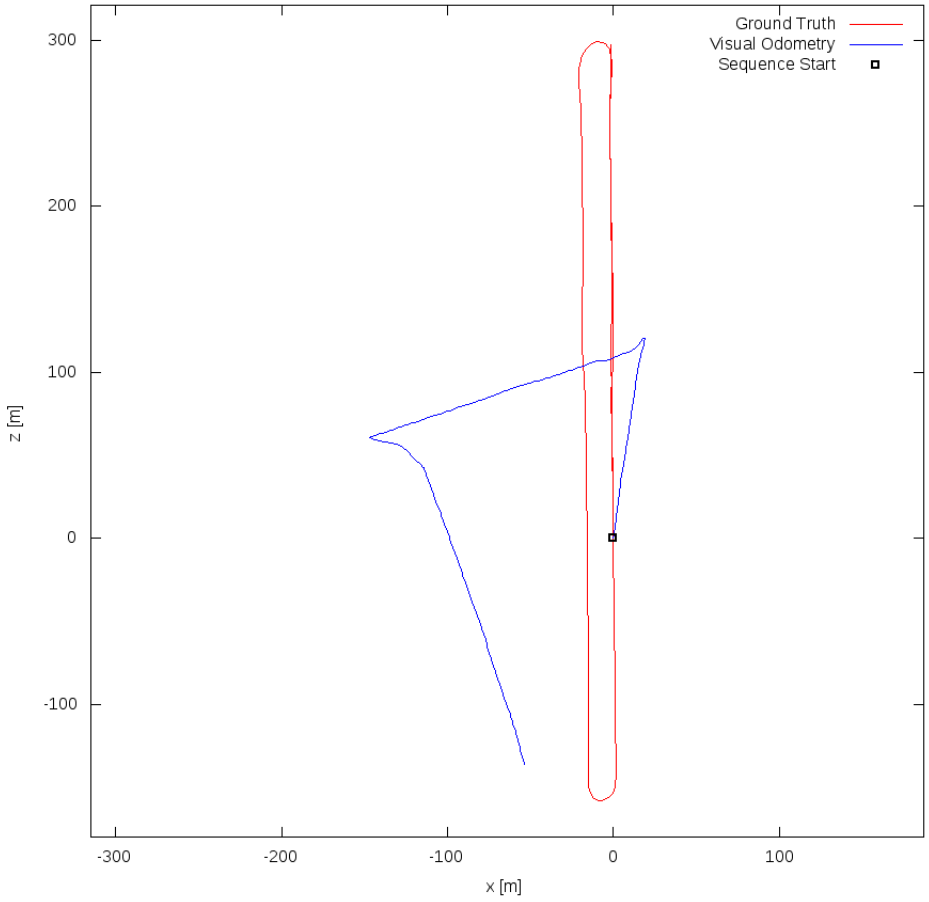


Figura 6.12: Estimación de la Trayectoria, Prueba No . 3.

La tabla 6.3 pretende mostrar la diferencia entre la estimación del sistema de odometría visual diseñado y el *Ground Truth* para ciertos fotogramas de la secuencia de imágenes de esta prueba.

Fotograma	x (Ground Truth)	z (Ground Truth)	x (estimación)	y (estimación)	D (Error en m.)
0	-0.0	0.0	0.0	0.0	0.0
27	-0.389518	30.9112	2.08635	15.2303	15.875155028
54	-0.998598	63.26505	3.75681	28.7682	34.8230751825
81	-0.842406	94.89992	5.85926	43.5431	51.7922319242
108	-0.209523	127.6422	7.01871	49.6831	78.2934775388
135	-0.641745	161.7146	8.89785	62.1521	100.018474688
162	-1.330648	195.6059	10.5698	73.6446	122.540521299
189	-1.757925	224.5915	12.5713	86.8129	138.521728646
216	-2.285245	244.1169	14.0295	97.7851	147.238468464
243	-1.836751	265.6407	16.2625	108.944	157.738513616
270	-1.173751	287.4148	18.4912	118.105	170.447994045
297	-7.313775	298.7204	15.0073	116.794	183.290603704
324	-18.91884	290.5003	-4.68974	107.01	184.041184198
351	-19.79506	260.173	-15.4323	104.61	155.624164717
378	-19.05062	223.1689	-26.613	100.83	122.572411433
405	-18.37716	183.9467	-39.054	96.8055	89.5607081807
432	-18.64019	149.7378	-52.4567	92.7389	66.2754173868
459	-18.86739	129.0544	-54.6093	92.1433	51.3800879102
486	-18.27078	105.4686	-61.1999	90.1341	45.5857020811
513	-16.76028	69.97465	-75.1137	84.8866	60.2286300566
540	-16.21939	30.80497	-88.7125	80.257	87.7539416129
567	-15.81631	-6.487479	-101.443	75.9822	118.883043344
594	-15.42272	-44.72012	-112.759	72.1615	152.104123869
621	-15.04758	-82.31772	-124.867	68.0386	186.191643134
648	-14.8217	-117.7429	-136.882	63.9777	218.908869853
675	-14.95141	-147.3493	-145.711	61.1095	246.075479629
702	-6.85914	-158.0218	-133.826	57.339	250.001715424
729	1.334755	-144.5411	-114.395	42.2783	219.760925573
756	1.286589	-114.0494	-110.351	31.2953	183.270382484
783	0.742127	-74.33667	-104.175	14.7226	137.619610197
810	0.112758	-34.7668	-98.9852	0.542195	105.200429693
837	-0.09919	2.639376	-93.8363	-14.0154	95.2051855455
864	-0.428756	38.79495	-88.6334	-28.5864	110.996871807
891	-1.203577	72.94149	-83.1445	-43.7398	142.579235158
918	-1.390632	102.5403	-79.4177	-54.3953	175.262676825
945	-0.849103	134.3774	-76.413	-63.7957	212.09073552
972	-1.145455	172.4452	-72.4515	-76.8934	259.334319949
999	-1.541969	209.6939	-67.4099	-91.8301	308.63458476
1026	-2.16719	236.9704	-62.9391	-104.279	346.618490628
1053	-2.14347	257.8203	-58.7338	-118.077	380.133194547
1080	-1.520404	282.3283	-55.4692	-130.176	416.017151219

Cuadro 6.3: Tabla de errores para la Prueba No. 3

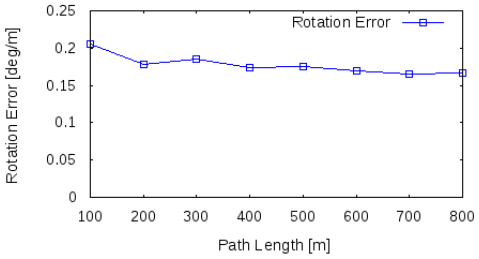


Figura 6.13: Error de rotación en función de la longitud de la trayectoria, Prueba No . 3.

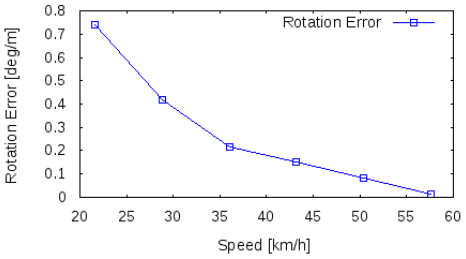


Figura 6.14: Error de rotación en función de la velocidad del vehículo, Prueba No . 3.

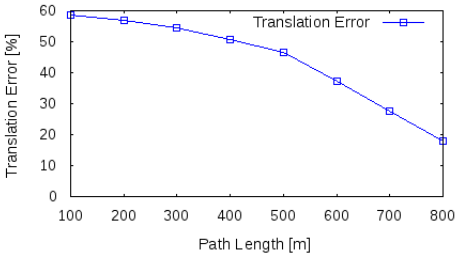


Figura 6.15: Error de traslación en función de la longitud de la trayectoria, Prueba No . 3.

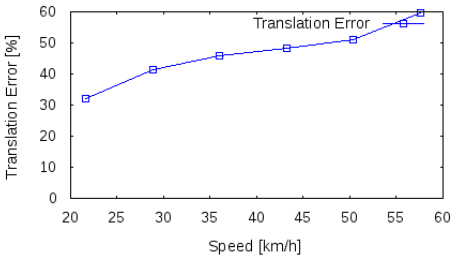


Figura 6.16: Error de traslación en función de la velocidad del vehículo, Prueba No . 3.

6.5. Trayectorias estimadas por el sistema monocular de $LIBVISO2$

A continuación se muestran las estimaciones de trayectoria hechas por el sistema de odometría visual monocular implementado en la librería $LIBVISO2$[7] sobre las mismas secuencias de vídeo utilizadas en la prueba 1, 2 y 3.

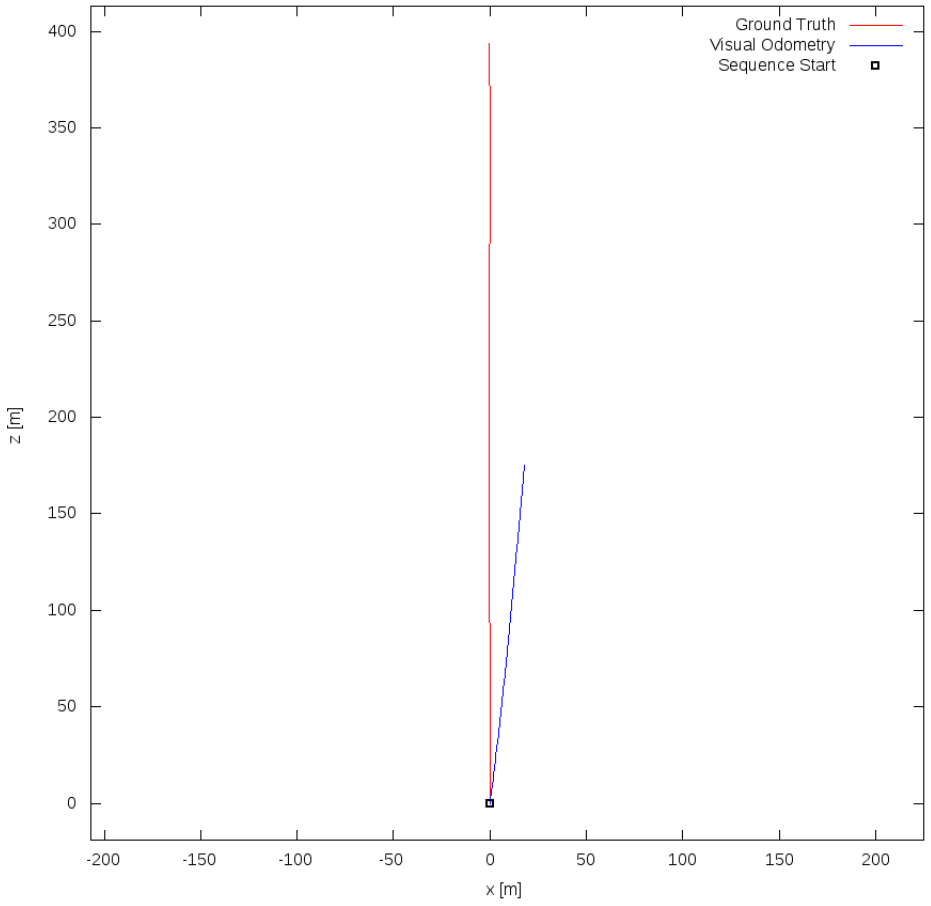


Figura 6.17: Estimación de la trayectoria, Prueba No. 1, LIBVISO2-Monocular.

[7]Geiger, óp. cit.

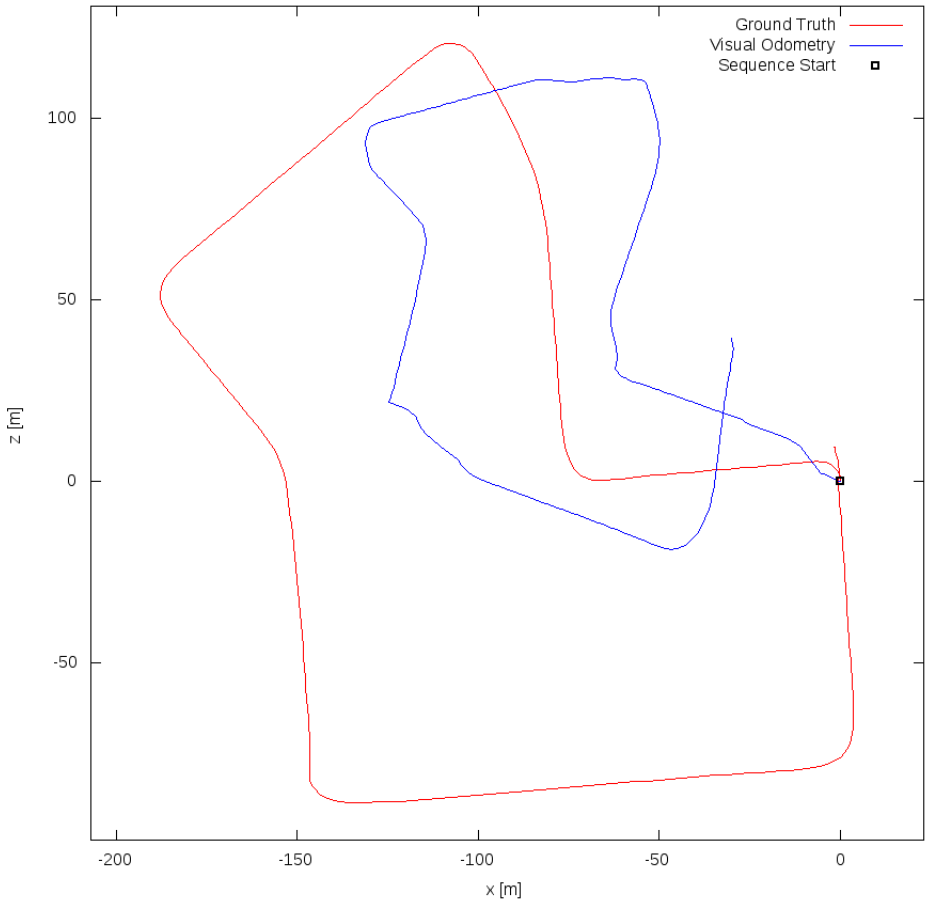


Figura 6.18: Estimación de la trayectoria, Prueba No. 2, LIBVISO2-Monocular.

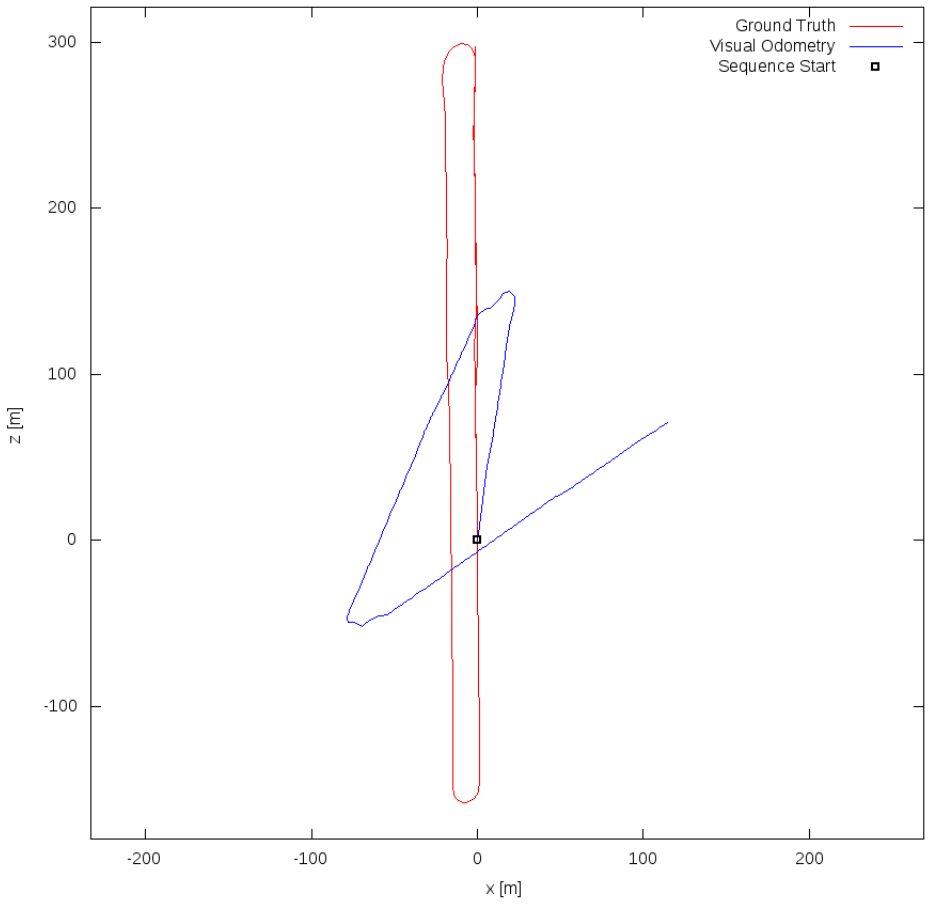


Figura 6.19: Estimación de la trayectoria, Prueba No. 3, LIBVISO2-Monocular.

Número Muestra	Tiempo empleado (s)
1	3.13
2	3.07
3	3.37
4	2.93
5	3.42
6	3.25
7	3.05
8	3.37
9	2.92
10	3.48
11	3.33
12	3.05
13	3.47
14	3.2
15	3.32
16	3.5
17	3.36
18	3.14
19	3.49
20	3.55
21	3.5
22	3.24
23	3.26
24	3.66
25	3.75
26	3.69
27	3.58
28	3.6
29	3.68
30	3.65

Cuadro 6.4: Tiempos de ejecución del algoritmo de Odometría Visual.

6.6. Medición del tiempo de ejecución

En esta sección se mostrarán los tiempos de ejecución empleados por el algoritmo para procesar varios pares de fotogramas. Esto se realiza con el objetivo de determinar la viabilidad de una futura implementación en tiempo real del algoritmo descrito en este proyecto. Las mediciones fueron realizadas con la librería *timer.h* de lenguaje *C*. A continuación se presentan los resultados para el procesamiento de 30 pares de fotogramas.

Teniendo en cuenta la tabla 6.4 se puede concluir que el tiempo promedio de ejecución del algoritmo es aproximadamente 3,367 segundos con una desviación estándar asociada de 0,2338 segundos.

Capítulo 7

Conclusiones

- En este proyecto se implementó un sistema de odometría visual cuyo funcionamiento se puede resumir de la siguiente manera:

  1. Utiliza el detector de $SURF$ como extractor de características.
  2. Usa una estrategia de fuerza bruta para el emparejamiento de características, usando la distancia euclidiana entre descriptores como medida de afinidad.
  3. Utiliza el algoritmo normalizado de los 8 puntos como estimador de movimiento.
  4. Usa un esquema de $RANSAC$ como estrategia para la eliminación de *outliers*.

  El sistema fue probado con un *dataset* que se capturó en un escenario real. Los resultados indican que este sistema no tiene los niveles de exactitud para su uso en un escenario real.

- Al examinar la sección de resultados es evidente que el error traslacional asociado al sistema de odometría visual está lejos de ser despreciable. Este error se asocia principalmente a la falta de un mecanismo efectivo para la estimación de la escala real de la traslación. Por lo tanto, se concluye que el algoritmo de determinación de la escala es fundamental para sistemas de odometría visual basados en correspondencias $2D - 2D$.

- El error acumulativo inherente a los sistemas de odometría visual es un problema significativo a la hora de usar estos sistemas en la realidad (véase la figura 6.2). En este trabajo se resolvió el problema de la localización del vehículo partiendo de la restricción epipolar entre dos imágenes y concatenando una transformación de cuerpo rígido a medida que llegaban más imágenes del recorrido del vehículo, de manera que en ningún momento se implementó una corrección de la trayectoria como la proponen

algunos autores. El control del error acumulativo en sistemas de odometría visual es un aspecto fundamental para el uso real de estos sistemas y por lo tanto deben investigarse y desarrollarse técnicas que permitan evitar que este error acumulativo crezca.

- En el capítulo de resultados (6) se puede observar que si no se tienen en cuenta soluciones efectivas para los dos problemas mencionados anteriormente(escala y error acumulativo) no es recomendable implementar este sistema en un entorno real.

- Se evidencia que el sistema de odometría visual implementado en este trabajo aumenta sus errores en la estimación cuando el vehículo se desplaza a altas velocidades (véanse las figuras 6.6, 6.16, 6.11). Por otro lado cuando el vehículo se encuentra estático por algún intervalo de tiempo, la estimación que hace el sistema de odometría visual es correcta, arrojando que el vehículo registra la misma posición en el espacio en cada imagen siguiente mientras el vehículo esta inmóvil.

- SSD(*Sum Of Square diferences*) no es un descriptor de afinidad adecuado para hacer emparejamiento de características como se evidenció en este trabajo (ver figura 5.4), ya que no mostró buenos resultados para este propósito. Se recomienda usar un descriptor que permita caracterizar de manera efectiva los puntos de interés para lograr hacer un emparejamiento más robusto, por ejemplo, como lo hace el detector y descriptor *SURF*.

- Si no se tiene un método para establecer correspondencias entre dos imágenes que garantice un alto número de *inliers*, no será posible construir un sistema de odometría visual exacto. La calidad de la estimación de un sistema de odometría visual está condicionada por la calidad de los emparejamientos entre imágenes, entre otros factores.

- El sistema de odometría visual construido representa un primer acercamiento a este campo de investigación, se requiere profundizar en el estudio de esta área de conocimiento para poder construir un sistema de odometría visual eficiente y que registre la trayectoria del vehículo con mayor exactitud.

- Un sistema de odometría visual nunca debe usarse como sustituto a otros sensores absolutos como el *GPS*, debe usarse como un complemento para este tipo de sensores, ya que estos también poseen un error inherente que un sistema de odometría visual podría ayudar a corregir por ser un sistema de localización relativo.

- Teniendo en cuenta las mediciones de tiempos de ejecución presentadas en la sección 6.6, se puede concluir que no es suficiente el tiempo de respuesta promedio para lograr una implementación en tiempo real.

.

7.1. Trabajo Futuro

Este trabajo abre una gran cantidad de problemas para ser resueltos en lo que tiene que ver con sistemas de odometría visual y sistemas de localización en general. A continuación se proponen algunos proyectos futuros.

- Uno de los problemas importantes encontrados en el desarrollo de este proyecto ha sido la determinación de la escala real de la traslación, por lo tanto se sugiere investigar acerca de métodos para la determinación de la escala absoluta del movimiento de tal manera que se pueda recuperar satisfactoriamente la trayectoria recorrida por el vehículo.
- Implementación de una estrategia para evitar el crecimiento del error acumulativo en un sistema de odometría visual.
- Una estrategia de emparejamiento de características más eficiente y más eficaz para la odometría visual, diferente a la implementada en este trabajo. Una mejora en el desempeño podría obtenerse al diseñar una estrategia de emparejamiento basada en estructuras de datos eficientes. (*kd-trees* explicados en la sección 3.4.3, Hashing multidimensional, etc.).
- Implementación de una estrategia de fusión de *GPS* y de un sistema de odometría visual, de manera que se pueda tener una estimación más exacta del movimiento. Debe tenerse en cuenta que primero deben solucionarse los problemas actuales del sistema de odometría visual descrito.
- Implementación de un sistema de *SLAM(Simultaneous Localization and Mapping)* que permita tener un modelo del ambiente que rodea al vehículo y que permita identicar cuando el vehículo pasa por un lugar por el que ha pasado con anterioridad (*loop closure*, $VSLAM$).
- El presente trabajo representa un aporte en el subsistema de percepción de un vehículo autónomo. Futuros proyectos podrían enfocarse en el desarrollo de las otras etapas de construcción de un vehículo autónomo, como las que se muestran en la figura 1.1.
- En el enfoque propuesto en este proyecto se estima el movimiento únicamente a partir de 2 fotogramas. Un trabajo futuro podría intentar involucrar un número mayor de fotogramas buscando mejorar la consistencia global de la estimación. En este sentido

podrían tenerse en cuenta técnicas como *local bundle adjustment*, localización basada en grafos, localización de $Markov$, filtros de $Kalman$ etc.

- Los algoritmos de extracción y emparejamiento de características y el algoritmo de $RANSAC$ son paralelizables. Se propone la optimización de los algoritmos implementados con el uso de unidades de procesamiento gráficos (GPU's).

Bibliografía

[1] FONDO DE PREVENCIÓN VIAL. *Publimotos [en línea].* [citado en 22 de julio de 2012]. URL: http://www.publimotos.com/nacionales/accidentes-de-transito-en-la-temporada-de-vacaciones-diciembre-y-enero-en-colombia/?id=3077.

[2] FONDO DE PREVENCIÓN VIAL. *Fonprevial [en línea].* [citado en 22 de julio de 2012]. URL: http://www.fonprevial.org.co/quienes_somos.

[3] DANE. *Estadísticas [en línea].* [citado en 22 de julio de 2012]. URL: http://www.discapacidadcolombia.com/Estadisticas.htm.

[4] MUSTAFA CONKA ALEX FORREST. «Autonomous Cars and Society». En: *Department of Social Science and Policy Studie* (May 1, 2007), pág. 29.

[5] CRISTIAN CAMILO PERILLA RESTREPO. «*Generación de un mapa de entorno tridimensional a partir de la integración entre un escáner láser y una unidad de medida inercial.* Trabajo de grado (Ingeniero Electrónico)». Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2011.

[6] CARNEGIE MELLON UNIVERSITY. «*Autonomous driving in urban environments: boss and the urban challenge.*» En: (19 de junio de 2008), pág. 2.

[7] J IBAÑEZ GUZMÁN. JIUN KEAT ONG. «*Perception Management for the Guidance of Unmanned vehicles.Cybernetics and Intelligent Systems*». En: (Singapore 2004).

[8] GUTMANN JENS Et. Al. «*An Experimental Comparison of Localization Methods*». En: (Germany), pág. 1.

[9] TOOMO INOUE Et. Al. «*Use of human geographic recognition to reduce GPS error in mobile mapmaking learning. Faculty of Science and Technology*». En: *Faculty of Science and Technology, Keio University.* (Japan 2006), pág. 2.

[10] JONATHAN MICHAEL WEBSTER. «*A Localization Solution for an Autonomous Vehicle in an Urban Environment*». Tesis de lic. Virginia Polytechnic Institute y State University, December 3, 2007.

[11] PETER K. ALLEN Et. Al. «*New Methods for Digital Modeling of Historic Sites*». En: *Columbia University.* (November 2012).

[12] NISTER, D. Et. Al. «*Visual Odometry*». En: *Sarnoff Corporation.* (Princeton USA. 2004.).

[13] D. SCARAMUZZA y F. FRAUNDORFER. «*Visual Odometry [Tutorial]*». En: *Robotics Automation Magazine, IEEE* 18.4 (Dec.).

[14] M. MAIMONE, Y. CHENG y L. MATTHIES. «*Two years of visual odometry on the mars exploration rovers: Field reports*». En: *J. Field Robot, vol 24 no. 3* (2007.), págs. 169-186.

[15] RENÉ. GOMEZ y ESTEBAN. CORREA. «*Uso de técnicas de visión por computador para la medición de variables del tráfico en el proyecto Observatorio de movilidad vial de Pereira.* Trabajo de grado (Ingeniero de Sistemas.)» Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2010.

[16] BERND JHÄNE Et. Al. «*Computer vision and applications: A guide for students and practitioners*». En: (2000.).

[17] LI. STAN Z y JAIN. ANIL K. *Handbook of face recognition.* Springer-Verlag, 2011.

[18] E.R. DAVIES. *Computer and Machine Vision: Theory, Algorithms, Practicalities.* Elsevier Science, 2012. ISBN: 9780123869913.

[19] M. SONKA, V. HLAVAC y R. BOYLE. *Image Processing, Analysis, and Machine Vision.* Nelson Education Limited, 2008. ISBN: 9780495082521.

[20] B. SICILIANO y O. KHATIB. *Springer Handbook of Robotics.* Gale virtual reference library. Springer, 2008.

[21] CALIFORNIA INSTITUTE OF TECHNOLOGY. *Mars exploration Rover, [en línea].* [citado en 22 de julio de 2012]. URL: `http://marsrovers.nasa.gov/overview`.

[22] R. E. KALMAN. «*A New Approach to Linear Filtering and Prediction Problems.*» En: *Research Institute for Advanced Study, Baltimore* (USA. 1960.).

[23] GUÍA TÉCNICA COLOMBIANA GTC 51. «*Guía para la expresión de la incertidumbre en las mediciones.*» En: *Colombia* (2000).

[24] LLAMOSA, LUIS ENRIQUE. GOMEZ, JOSÉ. RAMÍREZ, ANDRÉS FELIPE. «*Metodología para la estimación de la incertidumbre en mediciones directas.*» En: *Ciencia y técnica. año XV no. 41. Universidad tecnológica de pereira.* (Mayo del 2009.).

[25] W. SCHMIDT, and R. LAZOS. «*Guía para estimar la incertidumbre en la medición. Centro Nacional de Metrología.*» En: (México, Mayo 2000.).

[26] RUIZ A, MIGUEL Et. Al. «*Error, incertidumbre, precisión y exactitud, términos asociados ala calidad espacial del dato geográfico.*» En: *Departamento de Ingeniería Cartográfica, Geodésica y Fotogrametría. Universidad de Jaén.* (Febrero 2010.).

[27] ODOMETRIA VISUAL. *[en línea].* [citado en 22 de julio de 2012]. URL: `http://simreal.com/content/Odometry`.

[28] PARAMETROS DE ODOMETRÍA. *[en línea].* [citado en 22 de julio de 2012]. URL: `http://optimus.meleeisland.net/images/parametros_odometria.gif`.

[29] DAVID. TABORDA ALVAREZ. «*levantamiento de mapas de entorno por medio de sensores láser.* Trabajo de grado (Ingeniero de Sistemas.)» Universidad Tecnológica de Pereira. Facultad de ingenierías, Pereira 2009.

[30] MAURICIO. GENDE. «*Trilateración.*» En: *facultad de ciencias astronómicas y geofísicas, La plata, Argentina.* (2008).

[31] J.C. NUNNALLY e I.H. BERNSTEIN. *Psychometric theory.* McGraw-Hill series in psychology no. 972. McGraw-Hill, 1994.

[32] T.G. BECKWITH, R.D. MARAGONI y J.H. LIENHARD. *Mechanical Measurements.* Pearson Prentice Hall, 2007.

[33] V. Malyavej y P. Torteeka. «Unmanned ground vehicle localization by dead-reckoning/GPS sensor fusion». En: *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on.* 2010, págs. 508-512.

[34] Soon-Yong Park y col. «Localization of an unmanned ground vehicle using 3D registration of laser range data and DSM». En: *Applications of Computer Vision (WACV), 2009 Workshop on.* 2009, págs. 1-6.

[35] HANS MORAVEC. «*Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*». Tesis doct. Robotics Institute, Carnegie Mellon University y Stanford University, 1980.

[36] CHRIS HARRIS y MIKE STEPHENS. «*A combined corner and edge detector*». En: *In Proc. of Fourth Alvey Vision Conference.* 1988, págs. 147-151.

[37] J.-P. TARDIF, Y. PAVLIDIS y K. DANIILIDIS. «*Monocular visual odometry in urban environments using an omnidirectional camera*». En: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.* Sept. Págs. 2531-2538.

[38] P. CORKE, D. STRELOW y S. SINGH. «*Omnidirectional visual odometry for a planetary rover*». En: *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 4. Sept.-2 Oct. Págs. 4007-4012.

[39] D. SCARAMUZZA, F. FRAUNDORFER y R. SIEGWART. «*Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC*». En: *Robotics and Automation, ICRA '09. IEEE International Conference on*. 2009, págs. 4293-4299.

[40] R. Goecke y col. «*Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform*». En: *Intelligent Vehicles Symposium, 2007 IEEE*. 2007, págs. 450-455.

[41] D. SCARAMUZZA y R. SIEGWART. «*Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles*». En: *Robotics, IEEE Transactions on* 24.5 (), págs. 1015-1026.

[42] L. Matthies y S.A. Shafer. «Error modeling in stereo navigation». En: *Robotics and Automation, IEEE Journal of* 3.3 (June), págs. 239-248.

[43] Larry Henry Matthies. «Dynamic stereo vision». AAI9023429. Tesis doct. Pittsburgh, PA, USA, 1989.

[44] C.F. Olson y col. «Robust stereo ego-motion for long distance navigation». En: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. 2000, 453-458 vol.2.

[45] Olson C.F. y col. «Rover navigation using stereo ego-motion». En: *Robotics and Autonomous Systems* 43.4 (2003), págs. 215-229.

[46] A.I. Comport, E. Malis y P. Rives. «Accurate Quadrifocal Tracking for Robust 3D Visual Odometry». En: *Robotics and Automation, 2007 IEEE International Conference on*. April, págs. 40-45.

[47] Mark Nixon y Alberto S. Aguado. *Feature Extraction & Image Processing, Second Edition*. 2nd. Academic Press, 2008. ISBN: 0123725380, 9780123725387.

[48] D. SCARAMUZZA y F. FRAUNDORFER. «*Visual Odometry [Tutorial part II]*». En: *Robotics Automation Magazine, IEEE* 19.2 (June.).

[49] Herbert Bay, Tinne Tuytelaars y Luc Van Gool. «SURF: Speeded Up Robust Features». En: *Proceedings of the ninth European Conference on Computer Vision*. 2006.

[50] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

[51] Herbert Bay y col. «Speeded-Up Robust Features (SURF)». En: *Comput. Vis. Image Underst.* 110.3 (jun. de 2008), págs. 346-359. ISSN: 1077-3142.

[52] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN: 1848829345, 9781848829343.

[53] A.J. Davison. «Real-time simultaneous localisation and mapping with a single camera». En: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on.* 2003, 1403-1410 vol.2. DOI: 10.1109/ICCV.2003.1238654.

[54] UNIVERSITY OF FLORIDA Alper Üngör. *Computational Geometry: Kd-Trees and Range Trees.* [citado en 24 de junio de 2013]. URL: http://www.cise.ufl.edu/class/cot5520fa09.

[55] J. Shi y C. Tomasi. «Good features to track». En: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on.* 1994, págs. 593-600.

[56] T.S. Huang y A.N. Netravali. «Motion and structure from feature correspondences: a review». En: *Proceedings of the IEEE* 82.2 (1994), págs. 252-268.

[57] K.S. Arun, T. S. Huang y S. D. Blostein. «Least-Squares Fitting of Two 3-D Point Sets». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-9.5 (1987), págs. 698-700.

[58] UNIVERSITY OF ZURICH Davide Scaramuzza. *A tutorial on visual odometry.* [citado en junio de 2013]. URL: http://sites.google.com/site/scarabotix/.

[59] HC Longuet-Higgins. «A computer algorithm for reconstructing a scene from two projections». En: *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds* (1987), págs. 61-62.

[60] R. I. Hartley y A. Zisserman. *Multiple View Geometry in Computer Vision.* Second. 2004.

[61] R.I. Hartley. «In defense of the eight-point algorithm». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.6 (1997), págs. 580-593.

[62] R. Tsai y T.S. Huang. «Uniqueness and estimation of three-dimensional motion parameters of a rigid planar patch from three perspective views». En: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.* Vol. 7. 1982, págs. 834-838.

[63] Dawei Leng y Weidong Sun. «Finding all the solutions of PnP problem». En: *Imaging Systems and Techniques, 2009. IST '09. IEEE International Workshop on.* 2009, págs. 348-352.

[64] L. Kneip, D. Scaramuzza y R. Siegwart. «A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation». En: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* 2011, págs. 2969-2976.

[65] Martin A. Fischler y Robert C. Bolles. «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography». En: *Commun. ACM* 24.6 (), págs. 381-395.

[66] Long Quan y Zhongdan Lan. «Linear N-point camera pose determination». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21.8 (1999), págs. 774-780.

[67] C-P Lu, Gregory D. Hager y Eric Mjolsness. «Fast and globally convergent pose estimation from video images». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.6 (2000), págs. 610-622.

[68] Daniel F Dementhon y Larry S Davis. «Model-based object pose in 25 lines of code». En: *International journal of computer vision* 15.1-2 (1995), págs. 123-141.

[69] Martin A. Fischler y Robert C. Bolles. «Random Sample Consensus: A Paradigm for Model Fitting with Applicationsto Image Analysis and Automated Cartography». En: *Communications of the ACM* 24.6 (1981), págs. 381-395.

[70] David Nistér. «Preemptive RANSAC for live structure and motion estimation». En: *Machine Vision and Applications* 16.5 (2005), págs. 321-329.

[71] M FIALA y A. UFKES. «*Visual Odometry Using 3-Dimensional Video Input*». En: *Computer and Robot Vision (CRV), 2011 Canadian Conference on*. 2011, págs. 86-93.

[72] D.G. LOWE. «*Object recognition from local scale-invariant features*». En: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2, 1150-1157 vol.2.

[73] KURT KONOLIGE, MOTILAL AGRAWAL y JOAN SOLA. «*Large scale visual odometry for rough terrain*». En: *In Proc. International Symposium on Robotics Research*. 2007.

[74] JOERN REHDER y col. «*Global pose estimation with limited GPS and long range visual odometry*». En: *ICRA'12*. 2012, págs. 627-633.

[75] Andreas Geiger, Julius Ziegler y Christoph Stiller. «Stereoscan: Dense 3d reconstruction in real-time». En: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, págs. 963-968.

[76] C. NETRAMAI, H. ROTH y A. SACHENCKO. «*High accuracy visual odometry using multi-camera systems*». En: *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*. Vol. 1. 2011, págs. 263-268.

[77] Yang Cheng, M.W. Maimone y L. Matthies. «*Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging*». En: *Robotics Automation Magazine, IEEE* 13.2 (June), págs. 54-62.

[78] D. Scaramuzza, F. Fraundorfer y R. Siegwart. «Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC». En: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. 2009, págs. 4293-4299.

[79] R. Raguram, J.-M. Frahm y M. Pollefeys. «Exploiting uncertainty in random sample consensus». En: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009, págs. 2074-2081.

[80] O. Chum y J. Matas. «Matching with PROSAC - progressive sample consensus». En: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. 2005, 220-226 vol. 1.

[81] itseez. *OpenCV: Open source Computer Vision library*. [citado en 2013]. URL: `http://opencv.org/autores:http://itseez.com/`.

[82] Andreas Geiger, Philip Lenz y Raquel Urtasun. «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite». En: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[83] Andreas Geiger, Julius Ziegler y Christoph Stiller. «StereoScan: Dense 3d Reconstruction in Real-time». En: *Intelligent Vehicles Symposium (IV)*. 2011.

[84] Andreas Geiger. *LIBVISO2: C++ Library for Visual Odometry 2*. [citado en 2013]. URL: `http://www.cvlibs.net/software/libviso`.