Escuela Técnica Superior de Ingenieros Aeronáuticos
Universidad Politécnica de Madrid

Visión por Computador para UAS

Proyecto Fin de Carrera

Mónico Manuel Ponce González

2012

Escuela Técnica Superior de Ingenieros Aeronáuticos
Universidad Politécnica de Madrid

Visión por Computador para UAS

Proyecto Fin de Carrera

Autor:

Mónico Manuel Ponce González

Directora:

Ana García Álvarez

Tutor:

Pablo Rodríguez de Francisco

2012

A mi padre y a mi madre

por hacer posible que este sueño

se haga, por fin, realidad.

Agradecimientos

En primer lugar, me gustaría agradecer a AERNNOVA ENGINEERING SOLUTIONS IBÉRICA la oportunidad que me ha brindado al ofrecerme este proyecto, y proporcionarme los medios necesarios para la ejecución del mismo. En especial, agradecer la dedicación de mi directora de proyecto, Dña. Ana García Álvarez, del responsable del área de I+D, D. Federico Martín de la Escalera Cutillas y de mis compañeros de oficina, sobre todo de Carmen Barquilla, Ana Reguero, Agustín Oterino, Ignacio Guitián, Gabriel Esteban, Ernesto Llorente, André Rivero y Lara Rufo por soportarme estos meses. Y agradecer también la inestimable ayuda de Aritz Agirre, Ricardo Díaz, Ángel Alcaide y Saverio Corbo; sin la cual, en muchos momentos no se podría haber avanzado el proyecto.

También me gustaría agradecer el apoyo de mi tutor en la Escuela Técnica Superior de Ingenieros Aeronáuticos, D. Pablo Rodríguez de Francisco, que siempre ha estado dispuesto a recibirme para ofrecerme soluciones y conocer en todo momento el estado de los avances de este proyecto. Y agradecer a todos los profesores de la Escuela Técnica Superior de Ingenieros Aeronáuticos que me han enseñado a pensar y razonar, y que tantos conocimientos han hecho que aprenda, especialmente a Dña. María Victoria Lapuerta, por brindarme la oportunidad de realizar el proyecto fin de carrera en empresa, y a Dña. Dolores Sondesa, que tanta ayuda nos prestó tanto a mi compañera Carmen, como a mí, durante nuestro período de becario PAD en la Biblioteca Aeronáutica. De paso, aprovecho para agradecer a D. Luis Manuel Gómez, y a todo el personal de la Biblioteca Aeronáutica, por darme esa oportunidad, donde pasé un año de becario PAD muy bueno y me divertí y aprendí muchísimo con ellos.

También me gustaría mencionar el apoyo del Grupo de Vision For Unmanned Systems de la Escuela Técnica Superior de Ingenieros Industriales, que dirige el profesor D. Pascual Campoy, que ofreció medios y ayuda experta para el avance del proyecto, sobre todo en la parte de programación con la herramienta OpenCV.

En el ámbito más personal, me gustaría agradecer al Colegio Mayor Fundación SEPI, antes conocido como Fundación Empresa Pública, que me acogió en los primeros años de Universidad, en el que hice muchas amistades y pase grandes momentos. Y agradecer a esos amigos que hice allí, como son Joan Cirera, Santiago Alonso, Ignacio Romo, Jaime Calle, Antonio Barea, Ramiro Romero, Ricardo Ortega, Ignacio Ordovás, Alejandro Carreras, Agustín Oterino, Isabel Ruiz, Pablo Fernández de Velasco, Ramón del Rey. Pablo Palmero, Julio César Larriba, Jon Mikel Aboitiz, Carlos Ordás, José Luis de Unzueta, Álvaro Doncel, Fernando Castañón, Marcos Oficialdegui, Émory de Castro, Isabel Rubiales, Julia Rubiales, Pilar Rubiales, María Crespo, Samuel Rodríguez, Carlos Cerdeño, Felipe Juárez, Javier Duro, Jorge Barca, Antonio Barbero y un larguísimo etcétera que no puedo reproducir aquí, ya que estaríamos hablando de escribir un tomo solamente para incluir a toda la gente que se merece estar aquí. Y agradecer a mis actuales compañeras de piso Rocío Fraile y Beatriz Muñoz, que me tienen que aguantar en casa, con mis manías.

Volver a agradecer a André Rivero y a Lara Rufo, esta vez como compañeros de clase, por hacer más fácil y menos pesada la vida en la Escuela desde el primer día que entramos en ella. Y como a

ellos, a otros que también llevan muchos años acompañándome en la Escuela, como David Sampedro, Alejandro Lorenzo, Daniel Rodrigo, Salvador Romero, Carlos Díaz, Carlos Cordero, Garikoitz Calleja, Juan Giraldo, por estar ahí ayudando en clase con apuntes y resolviendo dudas en los duros días previos a los exámenes.

Y agradecer a la gente del Club Deportivo, tanto el de la Escuela Técnica Superior de Ingenieros Aeronáuticos como al de la Escuela Universitaria de Ingenieros Técnicos Aeronáuticos, y a los componentes del Aeronáuticos Club Deportivo Universitario de Madrid, en especial, a Andrea García, Paloma Manzano, Francisco Sánchez, Sergio de Lucas, Manuel Soler, Fernando del Cura, Francisco Barquín, y todos los compañeros de Junta Directiva. Y a mis compañeros del equipo de balonmano, sobre todo a Adrián Morales, Luis Ignacio García, Víctor García, Pablo Marín, Fernando Moreno, Hugo Sainz, Javier Sánchez, Pablo Velasco, David Fueyo, David Hinojosa, Hyo Hwan Cho, Jesús Maturana, Jorge Reverte, Óscar Souto y seguro que me dejo alguno más, que han hecho mucho más llevadero este periodo de mi vida. Y también aprovecho para agradecer al Club Deportivo Balopal, al que le hice la promesa al tener que dejarlo por estos estudios, de que mi primer proyecto tendría un guiño a ellos.

Y también no quiero olvidarme de los amigos de toda la vida que han tenido que ver como no podía estar con ellos todo lo que me hubiera gustado estar para estudiar. Muchos de ellos llevo con ellos desde primero de Párvulos: Luis Ángel Polanco, Miguel Ángel Marcos, Alberto Martínez, Francisco Estébanez, Daniel Molina, Adolfo Ruiz, Fernando Antolín, Giovanni Ibáñez, Ana D. Guerrero, Elena López, Marta Zapatero, María Pérez, Elena Sánchez, Javier Martínez, Javier Martín, Roberto Velázquez, Rubén León, Rubén Becerril, Elvira Tejedor y muchos más que me dejo en el tintero.

Las penúltimas personas a las que quiero mencionar son a los familiares. Quiero agradecer, además de dedicarles este proyecto, a mis padres Manuel y Matilde, por la paciencia que han tenido conmigo, que lo hemos pasado bastante mal pero ya lo hemos terminado. Y al resto de mi familia, mis tíos Melitón, Encarna, José, Carmen, Francisco, Eulalio, Amparo, Diego, Marisol, Antonia, Fernando, Francisca, Honorio, Josefina, José, Mónico, Mercedes, mis primos Soraya, Vanesa, Francisco, Alicia, María del Carmen, Encarna, Manuel, Antonio, Raúl, Francisco, Sergio, José David, Ismael, Diego, Fernando, Elisabeth, Deli, Marisol, Jorge, María de las Nieves, David, y demás, por dar apoyo moral en las reuniones familiares y entender que muchas veces no podía ir a verlos por tener que estar delante de los libros. Y agradecer también los amigos de mis padres, por la ayuda y el apoyo durante todos estos años, sobre todo a Tomás Bretón, Clementina Puertas, José Ángel Bretón y Raúl Bretón por un lado y a Martín García, Begoña Gil y Samuel García por otro.

Lo último que quería hacer es tener un recuerdo por la gente que hubiera estado en estos agradecimientos, que nos han dejado y ya no están con nosotros, y que allá donde estén, estarán orgullosos de mí, de que haya finalizado con éxito mis estudios, y disculparme ante las personas que me han apoyado todos estos años y que debían estar en este capítulo y no están por olvido, no porque no sean importantes y no los he mencionado en estos agradecimientos, porque es tantísima gente, y tantísimo tiempo, que mi memoria me juega malas pasadas.

Resumen

La visión por computador es una parte de la inteligencia artificial que tiene una aplicación industrial muy amplia, desde la detección de piezas defectuosas al control de movimientos de los robots para la fabricación de piezas. En el ámbito aeronáutico, la visión por computador es una herramienta de ayuda a la navegación, pudiendo usarse como complemento al sistema de navegación inercial, como complemento a un sistema de posicionamiento como el GPS, o como sistema de navegación visual autónomo.

Este proyecto establece una primera aproximación a los sistemas de visión artificial y sus aplicaciones en aeronaves no tripuladas. La aplicación que se desarrollará será la de apoyo al sistema de navegación, mediante una herramienta que a través de las imágenes capturadas por una cámara embarcada, dé la orden al autopiloto para posicionar el aparato frente la pista en la maniobra de aterrizaje.

Para poder realizar ese cometido, hay que estudiar las posibilidades y los desarrollos que el mercado ofrece en este campo, así como los esfuerzos investigadores de los diferentes centros de investigación, donde se publican multitud soluciones de visión por computador para la navegación de diferentes vehículos no tripulados, en diferentes entornos.

Ese estudio llevará a cabo el proceso de la aplicación de un sistema de visión artificial desde su inicio. Para ello, lo primero que se realizará será definir una solución viable dentro de las posibilidades que la literatura permita conocer. Además, se necesitará realizar un estudio de las necesidades del sistema, tanto de hardware como de software, y acudir al mercado para adquirir la opción más adecuada que satisfaga esas necesidades.

El siguiente paso es el planteamiento y desarrollo de la aplicación, mediante la definición de un algoritmo y un programa informático que aplique el algoritmo y analizar los resultados de los ensayos y las simulaciones de la solución. Además, se estudiará una propuesta de integración en una aeronave y la interfaz de la estación de tierra que debe controlar el proceso.

Para finalizar, se exponen las conclusiones y los trabajos futuros para continuar la labor de desarrollo de este proyecto.

Abstract

Computer vision is a field of Artificial Intelligence. Among its industrial applications, deficient part detections and robot movement control for manufacturing are two of the most important ones. In aeronautics, computer vision is a navigation tool. It can be used as inertial navigation system complement with an inertial measurement unit (IMU). It also can be used as a complement to a positioning system like GPS, or as autonomous visual navigation system.

This work establishes a first view of computer vision systems and their applications on Unmanned Aerial Vehicles. The application will be developed with the aim of supporting the navigation system, using a tool that gives some instructions to the autopilot, which orders to position the unit in front of the runway landing manoeuvre through the images captured by a camera on-board.

In order to carry out this study, several possibilities were analyzed, as well as some other commercial products that can be also employed to perform this task. A bibliography survey has also been carried out, focussing on almost all the possibilities and commercial developments in this field to perform this task. Some articles about the research efforts of different Research Centers, which publishes many computer vision solutions for navigation of different unmanned vehicles in different environments, had been studied too.

The present study reports all the phases in a computer vision system development. The definition of a working solution is the first issue of that process. In this regard, a literature survey will be required to propose a solution for this specific problem. It will also be required to perform a study of the needs of the system, both hardware and software, and to buy the most appropriate commercial items for these requirement analysis.

The next step will be the planning and development of the application, by defining an algorithm and a software which implements this algorithm. It will also be needed to analyse the results of the tests and simulations performed to the prototype developed. Additionally, a proposal of an aircraft integration will be considered, as well as its monitoring from a ground station. Finally, some conclusions will be summed up and the future work guidelines will be briefly described.

Índice general

Índice de figuras

Capítulo 1

Introducción

La visión por computador es un campo de la inteligencia artificial en el que se busca, a través de lenguajes de programación, que un computador sea capaz de identificar y realizar ciertas órdenes desde el reconocimiento de ciertas características de imágenes, imágenes completas o secuencias de éstas.

Los sistemas de visión artificial se componen de una fuente de luz para que el sistema funcione correctamente, un sensor de imagen, como puede ser una cámara, que captura las imágenes que luego se han de procesar, una tarjeta de adquisición de imágenes, como interfaz entre sensor y el computador y permite a éste último disponer de la información capturada por el sensor de imagen, unos algoritmos de analisis, en los que se procesan, segmentan y se aplican las transformaciones necesarias para obtener los resultados deseados; un procesador que analice las imágenes recibidas ejecutando los algoritmos diseñados y un sistema de respuesta a tiempo real que ejecute los comandos. Es razonablemente habitual que el computador lleve integrados tanto el procesador como la tarjeta de adquisición de imágenes, así como que el sensor de imagen esté embebido en una cámara, ya sea fotográfica o de vídeo.

En el campo de los sistemas aéreos no tripulados, a los que a partir de ahora denominaremos UAS (Unmanned Aerial Systems), la visión por computador forma parte importante para la navegación, el control y en ella se basan numerosas aplicaciones.

Los UAS son aeronaves que navegan sin tripulación humana a bordo y su principal aplicación es el ámbito militar, aunque también tienen una amplia aplicación civil en campos como la extinción de incendios, aunque sea de forma potencial y en centros de investigación, debido a que actualmente, la normativa no permite el uso civil de este tipo de aeronaves. Los UAS se definen como vehículos aéreos reutilizables, capaces de mantener un nivel de vuelo controlado y sostenido y propulsado por un motor de explosión o motor de reacción. Pueden ser guiados remotamente o tener prefijada la misión antes de iniciar el vuelo. Hay una gran variedad de configuraciones, formas y tamaños de estas aeronaves. En sus inicios, eran simples aeronaves guiadas por control remoto pero actualmente se busca el control autónomo, sea desde un plan de vuelo prefijado, controlado desde una estación de tierra o por radiocontrol.

Los UAS comenzaron a utilizarse durante la Primera Guerra Mundial, aunque no fue hasta treinta años después, en la Segunda Guerra Mundial, cuando empezaron a utilizarse aunque no con un fin de reconocimiento o ataque, sino como entrenadores para los operarios de las baterías antiaéreas. Hasta finales de siglo no avanzaron en gran medida para cumplir con todos los objetivos de autonomía bajo radio control, así como potenciaron es uso de materiales compuestos en zonas estructuralmente primarias de las aeronaves para cumplir con requisitos de invisibilidad a los sistemas de posicionamiento como el radar.

En las guerras del Golfo y de Bosnia fueron donde demostraron su gran potencial militar y donde se inició una carrera por poseer la más avanzada de las aeronaves no tripuladas para obtener dominio estratégico militar. Actualmente, el ejército de los Estados Unidos utiliza estos UAS, llamados drones, para las misiones de reconocimiento y ataque de objetivos en la guerra de Afganistán frente a los miembros del grupo terrorista Al-Qaeda, donde se han mostrado muy útiles y con mucha eficacia. Estos drones son dirigidos desde los Estados Unidos, donde el piloto se encuentra en la base con un mando que controla la aeronave. Así, además, se aseguran no poner en riesgo la vida del piloto de la aeronave.

En el manejo de datos, obtención y transmisión de información, se mejoraron las comunicaciones para interactuar de manera más segura y para que esa comunicación sea más difícil de interferir.

Tienen una desventaja muy clave con respecto a otros sistemas, y es la económica. Son sistemas muy caros tanto de fabricar como de mantener, eso hace que las aplicaciones civiles sean muy complicadas de llevar a cabo y por este motivo, los desarrollos se llevan a cabo en centros de I+D con aeronaves de bajo coste, pequeñas, de corto alcance, etc.

A cambio, tiene dos ventajas muy claras, que son el fácil acceso a zonas de alto riesgo o directamente inaccesibles para el sistema tripulado, y el hecho de que no lleve tripulación permite no causas bajas propias en combate.

Otras desventajas pueden tener un motivo más técnico, como la posibilidad de interferir en las comunicaciones entre UAS y tierra, la influencia de fenómenos atmosféricos como la actividad solar o cambios en la ionosfera, o el tiempo de retardo entre el envío de un comando y la ejecución de éste; o tener motivos más éticos, como puede ser que sea el propio UAS, mediante inteligencia artificial, el que determine un objetivo a batir, o la comercialización masiva, que haga llegar este tipo de sistemas a grupos de dudosa moral para cometer delitos con ellos.

Capítulo 2

Objetivos del Proyecto

Este proyecto se enmarca dentro de una de las misiones industriales principales dentro de la visión por computador, que es la ayuda a la navegación de aeronaves.

La operación de un UAS implica no solo la realización de manera autónoma de una misión preconfigurada, sino que también el despegue y aterrizaje de la plataforma debe realizarse sin la asistencia de un piloto humano. Este piloto debe estar presente en la operación como mecanismo de seguridad. La maniobra de aterrizaje exige disponer de una superficie adecuada, así como de un conjunto de sensores y sistemas que permitan el éxito de la operación. La solución más económica y básica supone hacer uso de un GPS, que posiciona tanto plataforma como zona de aterrizaje deseada. Distintos condicionantes no permiten calificar a este tipo de maniobra de precisión. En particular: el error en la posición dado por el GPS y el desarrollo de la maniobra final como planeo, en que no se consigue un control preciso de la senda del avión. Los sistemas basados en visión artificial permiten mejorar la precisión en el aterrizaje, simulando el mecanismo de decisión de un piloto humano. La visión artificial es también aplicable a la navegación, seguimiento y reconocimiento.

Este proyecto supone una primera aproximación a los sistemas basados en visión artificial para UAS. Para ello, se realiza un estudio de la tecnología y de los desarrollos que actualmente se están llevando a cabo.

Otro de los objetivos del proyecto, una vez realizado ese estudio, es definir y desarrollar una aplicación de visión por computador a un caso sencillo, como es la asistencia al aterrizaje de una aeronave, en la que, mediante transformaciones y técnicas de tratamiento de imágenes, se pueda determinar los bordes de la pista, el horizonte y la posición relativa entre la pista y la aeronave para así servir de apoyo al autopiloto a la hora de controlar el aterrizaje

Un objetivo más del proyecto, enfocado a un largo plazo, es la integración en la aeronave y el desarrollo de una herramienta que permita el control de la operación de asistencia desde una estación de tierra. Es por esto, que en los capítulos finales se aborda un estudio de ese software de tierra y una propuesta de integración en una aeronave real.

El objetivo final del proyecto es la puesta en marcha de todos los sistemas de esta aplicación y usarlos en unión con otros sistemas de navegación y posicionamiento como podría ser un GPS para así, poder navegar el UAS de manera autónoma.

Capítulo 3

Estado del arte

Como ya se ha indicado en la introducción, la visión por computador busca que un computador sea capaz de identificar imágenes capturadas y realizar ciertas operaciones, ya sea en las propias imágenes o en los mecanismos de control del sistema.

Dentro de la visión por computador, el control visual es ámbito más utilizado en la industria, y en el que nos centraremos con mayor atención en este capítulo, ya que en él se basan las aplicaciones que se encuentran en los diferentes estudios, investigaciones y desarrollos civiles (con las limitaciones normativas ya especificadas en la introducción) y militares.

En el campo de los sistemas aéreos no tripulados, a los que a partir de ahora denominaremos UAS (Unmanned Aerial Systems), la visión por computador forma parte importante para la navegación, el control y en ella se basan numerosas aplicaciones.

Figura 3.1: MQ-9 Reaper en vuelo

Los UAS son aeronaves que navegan sin tripulación humana a bordo y su principal aplicación es el ámbito militar, aunque también pueden tener una amplia aplicación civil en campos como la extinción de incendios. Debido a su facilidad a la hora de utilizarlos como espías, los UAS con aplicaciones de visión artificial son muy utilizados en el ámbito militar. Además, tienen la ventaja de no poner en riesgo a ningún efectivo del ejército propio en caso de alcance, por no ir tripulados y permiten reconocer, rastrear y conocer puntos estratégicos del enemigo. Destacar los Hermes 450S y Matjaz-1 (IAI Heron) israelíes, y los americanos MQ-9 Reaper de General Atomics, RQ-170 Sentinell de Lockheed Martin y RQ-4 Global Hawk de Northrop Grumman, que ya están involucrados en maniobras con éxito, y el español ATLANTE, que está en desarrollo.

Figura 3.2: Helicóptero COLIBRÍ III, utilizado por el grupo de Vision For Unmanned Systems de la Universidad Politécnica de Madrid

Estos sistemas no tripulados, al ser tan utilizados en el ámbito militar, suelen tener sus datos clasificados y no estar disponibles para el estudio y aplicación civil durante gran parte de su vida útil, lo que no facilita la búsqueda de soluciones para este proyecto. Aún así, hay numerosos grupos de trabajo en centros de robótica, centros de investigación y desarrollo (I+D) y grupos de trabajos de diferentes universidades mundiales que desarrollan potenciales aplicaciones civiles a sistemas no tripulados con visión artificial. En España, fundamentalmente trabajan en este campo el grupo Vision for Unmanned Systems del Centro de Automática y Robótica de la Universidad Politécnica de Madrid y el grupo de desarrollo ICARUS de la Universitat Politècnica de Catalunya. Y aunque hay otros grupos especializados en visión artificial en otras universidades españolas como pueden ser los de la Universidad de Sevilla, la Universidad Politécnica de Valencia, la Universidad de Murcia, la Universidad Pública de Navarra o la Universidad de Zaragoza, no trabajan en aplicaciones para el campo aeronáutico, más bien en aplicaciones informáticas para la industria.

Estos centros de trabajo basan muchas de sus investigaciones en aeronaves de ala rotatoria, por la peculiaridad del vuelo de estas aeronaves; ya que pueden mantenerse en un punto fijo del espacio mientras el sistema de adquisición de imágenes captura, procesa, segmenta y reconoce la imagen para pasar a la siguiente orden del proceso de navegación. Los UAS en los que se basan son helicópteros para aplicaciones exteriores, y quadricópteros y mini-helicópteros en aplicaciones bajo techo, que al ser aeronaves más pequeñas y con una gran versatilidad de actuaciones, son ideales para ensayos en interiores.

Este proyecto, sin embargo, se basa en una aeronave de ala fija. Las aplicaciones de aeronave de ala fija de estos grupos de investigación están enfocadas a vuelos acrobáticos y concursos anuales de robots aéreos de la AUVSI (Association for Unmanned Vehicle Systems International), en los que en cada concurso se fijan las bases de los objetivos a realizar por la aeronave.

Los centros universitarios también investigan en aplicaciones como el aterrizaje y el despegue autónomo del UAS tanto en pista como en portaaviones con la visión por computador como herramienta de cálculo de trayectorias y reconocimiento de horizontes y pistas. En el próximo capítulo se introducirán otros muchos centros de investigación y desarrolladores de prototipos de sistemas de visión

por computador que han publicado artículos en revistas especializadas y que han sido de gran ayuda para la realización de este proyecto.

3.1. Fases en un sistema de visión artificial

Básicamente, el ser humano capta la luz a través de los ojos; y la información captada viaja al cerebro por el nervio óptico. El cerebro procesa la imagen, interpreta la escena y actúa en consecuencia.

Los sistemas de visión artificial buscan reproducir este mismo comportamiento, para lo cual, se definen cuatro fases en el sistema, estas fases son:

1. Captura: Etapa en la que se adquiere la información, las imágenes.
2. Procesamiento: Etapa en la que mediante filtros o transformaciones se eliminan partes no deseadas de las imágenes, facilitando las etapas posteriores.
3. Segmentación: En esta etapa se aíslan los elementos que interesan del resto de la imagen para su comprensión.
4. Reconocimiento: En esta etapa se pretende distinguir los objetos segmentados mediante una serie de características que previamente se han establecido para diferenciarlos.

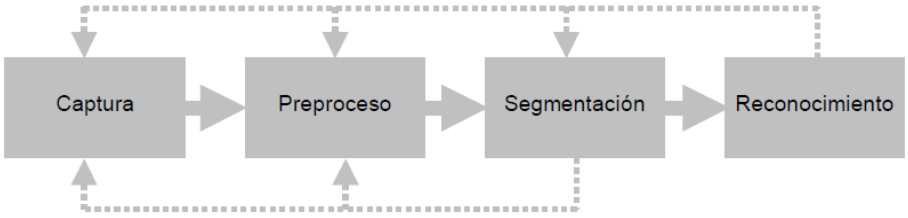


Figura 3.3: Fases de un sistema de visión por computador

Estas fases se siguen de manera secuencial en un sistema habitual de visión artificial, mas puede suceder que no sigan esa secuencia siempre, sino que se realimenten hacia atrás en el ciclo. Así, si la segmentación falla, se puede volver al procesado o a la captura para intentar arreglar el fallo.

3.1.1. Captura

La captura es el proceso en el que se obtiene información sobre una escena mediante un dispositivo óptico. Si la imagen es digital, además de la captura, hay que someterla a una digitalización, en la que transforma la información continua en información con todas sus componentes discretas.

De manera genérica, el proceso de captura se puede realizar mediante dos métodos. El primero se basa en el principio de la cámara oscura, mediante dispositivos pasivos; y el segundo método se basa en el escaneo de imágenes mediante dispositivos activos. También se puede capturar imágenes para el uso en el sistema de visión por computador mediante la construcción de imágenes sintéticas.

El principio de la cámara oscura se puede usar para obtener imágenes del mundo real y proyectarlas en un plano bidimensional. La cámara oscura se basa en la proyección de la imagen a través de una lente fina situada delante de un orificio, que hace que se obtenga una imagen invertida tanto vertical como horizontalmente de la escena que se está produciendo fuera de la cámara. La cámara es una caja cerrada y la luz entra en ella en el orificio que se le produce. La imagen obtenida se proyecta en

la pared opuesta a la que tiene el orificio. Este dispositivo pasivo es un dispositivo muy conocido, utilizado en Bagdad por primera vez en el año 965 por el matemático árabe Alhacen, aunque no fue hasta 1550, cuando Cardan le aplico una lente al orificio para aumentar la luminosidad y Kepler utilizó este principio para crear las bases del telescopio en 1604. Este principio es en el que se basa la cámara estenopeica, o cámara pinhole, precursor de la cámara fotográfica.

El otro método de captura de imágenes es el escaneo. En el, un elemento activo, como puede ser un haz de luz, recorre la escena que se desea capturar. Debido a esto, se necesitan como mínimo un haz de luz, un emisor de luz y un receptor o detector de luz, para que cuando el haz choque con la escena tras ser emitido, se pueda recoger con el detector y representar la escena tras la repetición de este proceso de manera continua. Los escáneres se usan con diferentes fines. Eso nos lleva a que un escáner de tambor, creado para dar representaciones 3D, pueda capturar imágenes de elementos bidimensionales y que escáneres-láser puedan capturar escenas tridimensionales aunque su aplicación fundamental sea la captura bidimensional.

De estos dos métodos, el primero, el que se basa en el modelo de cámara oscura, es el modelo que mejor se adapta a la mayoría de usos, debido a que es una forma muy sencilla y económica de capturar imágenes, a que la velocidad de captura es mucho mayor y a que el funcionamiento es bastante similar al del ojo humano. Es en el proceso de digitalización donde la cámara tiene una desventaja frente al escáner, ya que hasta hace relativamente poco tiempo, era un punto critico, ya que actuaba como cuello de botella, pero con los desarrollos actuales y el auge de la fotografía digital y de las cámaras integradas en dispositivos de telefonía móvil, esta desventaja se ve disminuida casi al máximo.

La digitalización es el otro apartado que se necesita para capturar la imagen y poder pasar a la siguiente fase de procesado de imágenes. Es el paso del mundo analógico continuo al mundo digital discreto. Tiene a su vez dos pasos, que son el muestreo y la cuantización. El muestreo de una señal continua consiste en la discretización respecto a una variable, ya sea tiempo o espacio de manera general. El parámetro fundamental del muestreo es la frecuencia de muestreo, que es el número de veces que se mide un valor analógico por unidad de cambio.

En una imagen, el muestreo convierte la imagen continua en una matriz discreta de N x M píxeles. El píxel es la menor unidad homogénea en color de una imagen digital. El número de muestras por unidad de espacio sobre el objeto original lleva a la resolución espacial de la imagen, concepto definido por la distancia, sobre el objeto original, entre dos píxeles consecutivos. Ahora bien, la unidad más común de resolución espacial es el píxel por pulgada o DPI, medido siempre en el objeto original.

La siguiente operación en la digitalización es la cuantización de la señal, que es la discretización de los posibles niveles de los valores de cada píxel. Los niveles de esa cuantización se hacen en potencias de 2 para facilitar el almacenaje en el computador de esas imágenes, ya que el ordenador usa el byte como unidad de memoria directamente direccionable. Los niveles posibles definen la resolución radiométrica y esos niveles pueden ser 2, 4, 16 o 256 habitualmente.

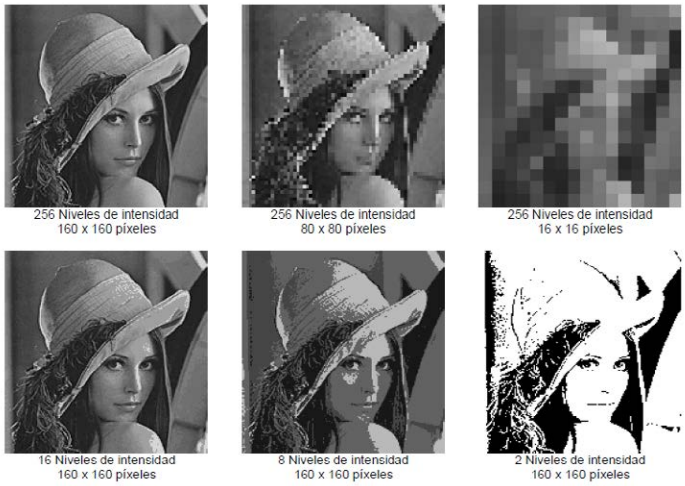


Figura 3.4: Niveles de cuantización

Cuando las imágenes tienen únicamente información sobre el brillo se habla de imágenes en niveles de gris. Se suelen utilizar hasta 256 niveles para representar desde el negro (nivel 0) al blanco (nivel 255).

Si se permiten dos niveles de cuantización solamente, estamos hablando de una imagen binaria o bitonal, y si nos vamos a imágenes en color, se suele usar 256 niveles de intensidad para cada uno de los canales que corresponden a los tres colores primarios, rojo (R), verde (G) y azul (B); obteniéndose de esta forma una imagen de 16 millones de colores (256 X 256 X 256). Estas imágenes se llaman imágenes en color real. En algunos casos se puede necesitar de mayor resolución radiométrica, llegando incluso a 4096 niveles. Además, la imágenes en color no solo se pueden cuantizar en el formato RGB, sino que se puede cuantizar en otros modelos de color, como el HSV (Hue, Saturation, Value (Matiz, Saturación, Valor)), utilizado por Apple e Intel para sus programas y librerías de programación como OpenCV, o el HSL (Hue, Saturation, Lightness (Matiz, Saturación, Luminosidad (o Intensidad))) que es el que usa, por ejemplo, Microsoft en sus programas incluidos en el sistema operativo Windows.

Estos dos procesos (muestreo y cuantización) se pueden realizar de manera uniforme, como se ha explicado, o de manera no uniforme, muestreando una parte de la imagen de forma más nítida, con una resolución espacial mayor, o realizando la cuantización de las imágenes con el uso de paletas, que son un conjunto de colores a los que se les asigna una referencia. Los píxeles de las imágenes que usan paletas contienen como valor de referencia al color de la paleta que quieren presentar. Cuando el número de colores de una imagen es pequeño, el uso de paletas permite un ahorro de memoria y simplificar operaciones al cambiar solo la paleta y no todos los píxeles de la imagen.

El proceso de digitalización requiere evaluar las dos resoluciones, la espacial y la radiométrica, que se precisan para representar de manera adecuada una imagen. O lo que es lo mismo, se necesita conocer la frecuencia con la que se muestrean los píxeles y la gama de colores permitida. A través del teorema de Shannon, al que fue convertida la conjetura de muestreo de Nyquist, se establece que la frecuencia mínima de muestreo para recuperar una señal sin errores debe ser al menos el doble de la máxima frecuencia con la que cambian los elementos que se quieran capturar en la señal. Este teorema, nos lleva a la conclusión de que si tenemos una imagen impresa a 200 DPIs, para su escaneo representando fielmente la imagen impresa necesitaremos al menos un muestreado de 400 DPIs.

También hay que tener en cuenta que dependiendo del uso que se vaya a hacer de la imagen se puede realizar la elección de parámetros de digitalización de manera menos objetiva. A la hora de usar una imagen para reconocimiento, nos puede ser más útil tener una gran variedad de niveles de intensidad pero una resolución espacial pequeña, pero para publicar un periódico en blanco y negro, con 16 niveles de intensidad, una resolución pequeña no será admisible.

3.1.2. Procesamiento

El procesamiento, o procesado de la imagen consiste en la realización de una serie de transformaciones con el objetivo primordial de facilitar la búsqueda de información para obtener las características de la de imagen, o simplemente, mejorar la calidad de la imagen.

El procesado de imágenes va desde simples operaciones aritméticas o geométricas entre píxeles, hasta complejos filtros de contornos como los operadores Canny o Sobel. En este apartado se indicarán las principales operaciones que se realizan para procesar una imagen.

1. Operaciones aritmético-lógicas
2. Operaciones geométricas
3. Operaciones sobre el histograma
4. Filtrado en el dominio de la frecuencia
5. Filtrado en el dominio del espacio
6. Operaciones morfológicas

1. Operaciones aritmético-lógicas

Las operaciones aritmético-lógicas son las operaciones más comunes que se pueden hacer a una imagen, pues se realizan leyendo los valores píxel a píxel y modificándolos según se realiza.

Dentro de este grupo, están las siguientes operaciones

a. Conjunción: Operación lógica AND entre los bits de dos imágenes. Se suele usar para borrar píxeles.

b. Disyunción: Operación lógica OR. Se usa para añadir píxeles a una imagen.

c. Negación: Inversión de los bits que forman una imagen y con ella, se obtiene el negativo de la imagen.

d. Suma: Suma de los valores de los píxeles de dos imágenes.

e. Resta: Resta de los valores de los píxeles de dos imágenes.

f. Multiplicación: Multiplicación de los valores de los píxeles de dos imágenes. Se usa para añadir textura a una imagen.

g. División: División de los valores de los píxeles de una imagen entre los de otra imagen.

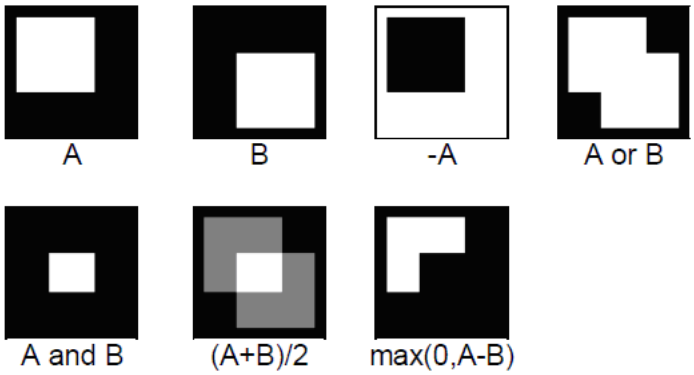


Figura 3.5: Operaciones aritmético-lógicas

En la figura, se observan las diferentes operaciones aritmético-lógicas realizadas en una imagen en blanco y negro para ver sus aplicaciones.

2. Operaciones geométricas

Las operaciones geométricas son operaciones en las que se modifica la posición de la imagen o su forma relativa. Expresadas en coordenadas homogéneas, estas operaciones se pueden expresar como una multiplicación de matrices. Las más frecuentes son

a. Traslación: Movimiento de los píxeles de una imagen según un vector de desplazamiento.

b. Rotación: Giro relativo de los píxeles de una imagen a un eje o a un punto, generalmente el origen de coordenadas.

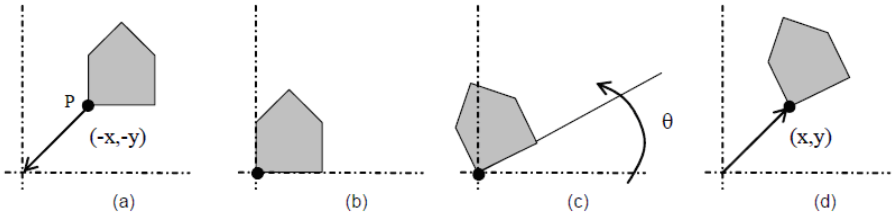


Figura 3.6: Operaciones geométricas en una imagen
La operación (a) es una traslación al origen. La operación (b) define el punto de rotación de la imagen. La operación (c) rota la imagen respecto el origen de coordenadas y la operación (d) vuelve a trasladar la imagen, ya girada, al punto inicial.

c. Escalado: Cambio del tamaño de una imagen al multiplicar las dimensiones de ésta por un factor de escala.

3. Operaciones sobre el histograma

Las operaciones sobre el histograma son operaciones en las que se cambia el contraste y la luminosidad de la imagen. Primero habrá que hablar de qué es el histograma.

El histograma de una imagen es un diagrama de barras en el que se disponen en horizontal los niveles de cuantización y que cada barra del diagrama tiene una altura proporcional al número de píxeles que hay en cada nivel de cuantización. Los histogramas se suelen dar para imágenes en blanco y negro con 256 niveles de intensidad. Si se trata de una imagen a color, se da un histograma para cada una de las componentes RGB de la imagen. Además, el histograma se suele normalizar entre 0 y 1 para evitar dependencias del número de píxeles o de niveles.

Mediante una función de transferencia del histograma, las operaciones en éstos se visualizan de manera muy eficaz. Estas funciones de transferencia, al final, corresponden a aplicaciones, ya que cada punto del dominio tiene un único valor imagen.

Las operaciones más comunes en el histograma son:

a. Ajuste de contraste: El contraste se define como la diferencia de intensidad pronunciada. Pues mediante el uso de una función de transferencia del histograma que aclare los niveles oscuros y oscurezca los claros, podremos aumentar el contraste del conjunto. Y si hacemos la operación contraria, reduciremos el contraste.

b. Ecualización del histograma: El ecualizado consiste en obtener un nuevo histograma de la imagen a partir del original, con una distribución uniforme de los niveles de intensidad.

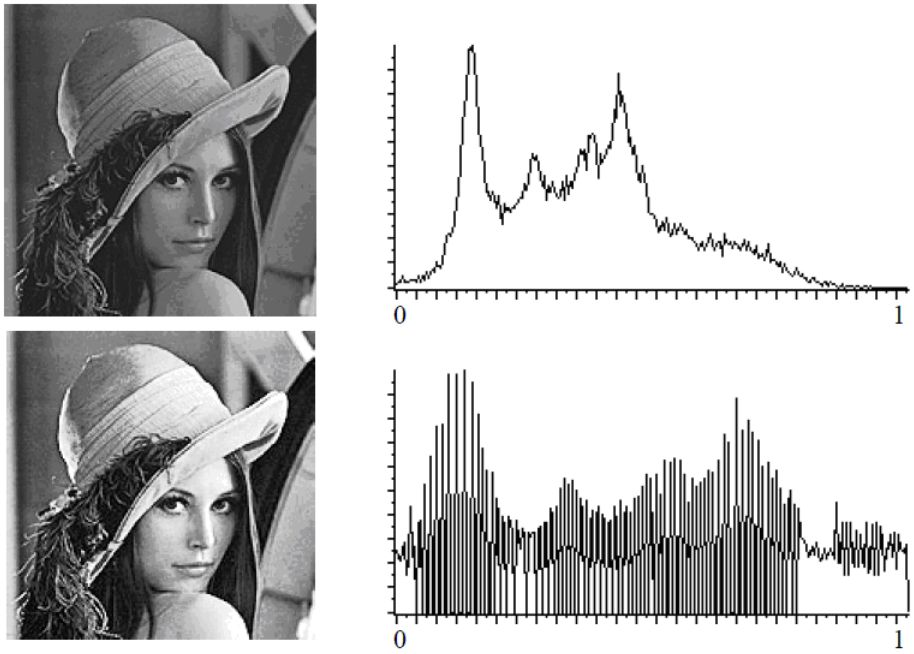


Figura 3.7: Imagen original (a) frente a imagen con el histograma ecualizado (b)

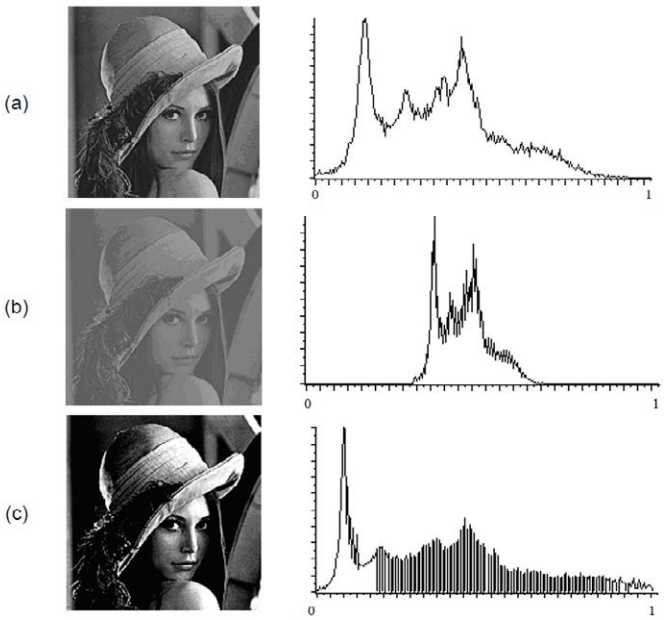


Figura 3.8: Imagen original (a) frente a imágenes con ajustes de contraste (b) y (c)

En las figuras, se observan las diferentes operaciones de ecualizado de histograma y de ajuste de contraste

4. Filtrado en el dominio de la Frecuencia

El filtrado de una imagen en el dominio de la frecuencia procesa una imagen aplicando la transformada de Fourier de la imagen y trabajando sobre el dominio de ésta. Para ello, se modifica el teorema de convolución de señales de forma que lo primero que se realiza es la transformada de Fourier de la imagen, después se aplica el filtro, y por último se realiza la transformada inversa de Fourier para devolverla al dominio espacial.

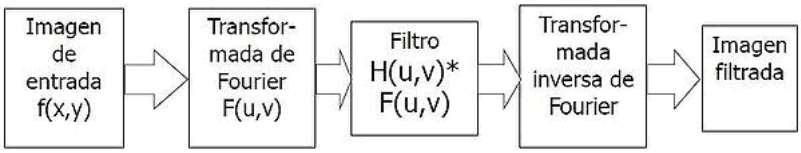


Figura 3.9: Secuencia de transformación en el filtrado de una imagen en el dominio de la frecuencia

Se puede utilizar otras transformadas como la transformada rápida de Fourier, o la bidimensional, o la discreta; pero al aplicar cualquiera de estas, después hay que aplicarle el filtro y luego devolver con la transformada inversa al dominio del espacio para obtener la imagen.

Los filtros más comunes son:

a. Filtro pasa bajos: Atenúa las frecuencias altas y mantiene sin variación las frecuencias bajas. El resultado es un suavizado de las transiciones de la imagen, reduciendo el ruido, pues se atenúan los cambios fuertes de intensidad.

b. Filtro pasa altos: Atenúa las frecuencias bajas y mantiene sin variación las frecuencias altas. El resultado es una mejora en la detección de bordes y un refuerzo en los contrastes de la imagen.

c. Filtro pasa banda: Atenúa las frecuencias muy bajas y muy altas, manteniendo sin variación las frecuencias medias.

Como la multiplicación en el espacio de Fourier es idéntica a la convolución en el dominio espacial, estos mismos filtros podrían implementarse como filtros espaciales.

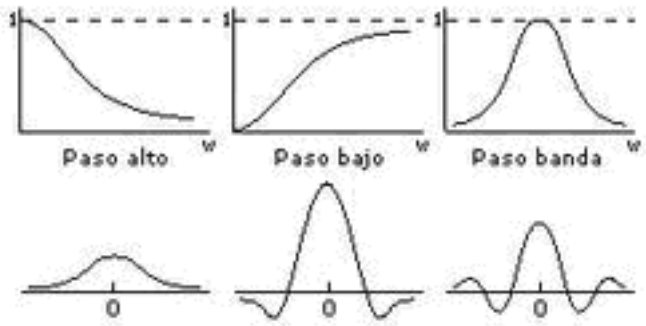


Figura 3.10: Tipos de filtros en el dominio de la frecuencia

Estos filtrados tienen una serie de ventajas, como ser métodos sencillos de implementar, ser rápidos debido al uso del teorema de convolución, fácil asociación de concepto entre tonalidad y frecuencia; y proporcionar mucha flexibilidad en el diseño de soluciones, pero también tiene unos inconvenientes que son la necesidad de conocer varios campos para el desarrollo de una aplicación de procesamiento de imágenes y la imposibilidad de eliminación total del ruido de la imagen.

5. Filtrado en el dominio del Espacio

El filtrado en el dominio del espacio se realiza directamente sobre la imagen. Normalmente se usa un filtro de convolución, aplicando una máscara de convolución, que al final reduce la operación a la multiplicación de los píxeles de la imagen por una matriz de convolución, que será la responsable del filtro. Existen otro tipo de filtros, que son los no lineales, que no se basan en la máscara de convolución. Para realizar el filtrado espacial, se realiza una convolución de la máscara sobre la imagen. Se sigue el Teorema de Convolución en el espacio: $g(x, y) = h(x, y) * f(x, y)$

Cada píxel de la nueva imagen se obtiene del sumatorio de la multiplicación de la máscara por los píxeles contiguos: $g(x, y) = \sum\sum f(i, j)w(i, j)$ y se suele normalizar, dividiendo entre un valor constante, que se obtiene de la suma de los valores de la máscara empleada.

Los filtros espaciales se dividen en los siguientes tipos:

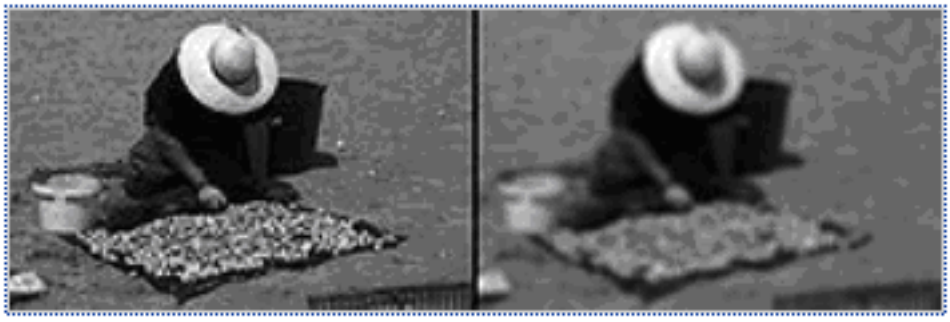
e 

Figura 3.11: Imagen original frente a imagen con un filtro de suavizado

a. Suavizado (Filtro pasa bajos): Utilizados para eliminar ruido o detalles de poco interés, estos filtros dependen mucho de la máscara utilizada. Pueden ser de promedio, promediando los píxeles contiguos con una matriz unidad; de mediana, de media, de pasa bajo en frecuencia, el del bicho raro y el gaussiano. El filtro de la mediana y el del bicho raro son filtros no lineales, el resto son lineales.

b. Atenuación (Filtro pasa altos): Intensifica los bordes, los detalles y los cambios de alta frecuencia, y atenúa las zonas de tonalidad uniforme. Permite mejor identificación de objetos en la imagen, y es común la aparición de ruido.

c. Realce de bordes por desplazamiento y diferencia: Sustrae de la imagen original una copia desplazada de la misma. Su resultado es el realce de bordes existentes según la máscara de convolución utilizada sea horizontal, vertical o diagonal (horizontal/vertical).

d. Laplaciana: Este tipo de filtro realza los bordes en todas las direcciones, ya que trabaja con el operador laplaciano, que permite una mejora de resultados con un aumento de ruido en la imagen.

e. Filtrado de realce de bordes por gradiente direccional: Se emplea para destacar y resaltar con mayor precisión los bordes que se localizan en una dirección determinada. Trabaja con los cambios de intensidad existentes entre píxeles contiguos.

f. Obtención de Contornos: Como los filtros anteriores, estos detectores de contornos se basan en las diferencias de intensidad que se dan píxel a píxel. Requieren un menor coste computacional, ya que trabajan desde el operador gradiente en una máscara que hace que se obtenga el contorno de la imagen y se pueda clasificar las formas existentes en ella. Las más utilizadas son los filtros de Sobel (Horizontal y Vertical), el filtro de Prewitt, el de Roberts y el detector de contornos de Canny.

$$\text{Sobel: Horizontal: } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ Vertical: } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Prewitt: Horizontal: } \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ Vertical: } \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\text{Robert Horizontal: } \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ Vertical: } \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Figura 3.12: Imagen original (izquierda) frente a imagen tratada con el filtro de Canny (centro) y tratada con el filtro de Sobel (derecha)

Canny: Se basa en la primera derivada de una imagen que previamente se le ha realizado un filtro gaussiano.

6. Operaciones Morfológicas

Las operaciones morfológicas tienen su razón de ser en la morfología matemática, basada en la teoría de conjuntos. En la morfología matemática, las imágenes binarias son conjuntos de puntos en dos dimensiones que representan los puntos activos de una imagen. Las imágenes en niveles de gris, son conjuntos en tres dimensiones, donde la tercera componente es el nivel de intensidad de gris.

Puesto que son operaciones basadas en la teoría de conjuntos, sus definiciones se darán también con esa notación. Y además, se darán las definiciones de estas operaciones para imágenes binarias, ya que son más sencillas de entender

Las diferentes operaciones morfológicas que se pueden hacer son:

a. Dilatación: La dilatación de una imagen A con un elemento estructurante B consiste en examinar cada píxel de A, y si esta en valor 0, pasarlo al valor 1 si alguno de sus vecinos esta en la imagen A al ser recorrida por la imagen B (o lo que es lo mismo, tiene valor 1). Normalmente se utilizan como vecinos los 8 píxeles que rodean al estudiado, aunque puede darse una conectividad de 4 píxeles, sin las diagonales.

b. Erosión: Es el proceso inverso a la dilatación Pasa a valor 0 los píxeles con valor 1 si alguno de sus vecinos esta con valor 0.

c. Apertura: Es la operación de erosionar y luego dilatar una imagen.

d. Cierre: Es la operación de dilatar y luego erosionar una imagen y por lo tanto, la operación inversa a la apertura.

e. Gradiente Morfológico: Es la operación resultante de restar la imagen erosionada de la imagen dilatada (Gradiente= Dilatación – Erosión).

f. Top Hat: Esta operación se define como la resta la imagen después de una apertura de la imagen original (Top Hat = Original – Apertura). Esta operación aísla las píxeles de gran intensidad.

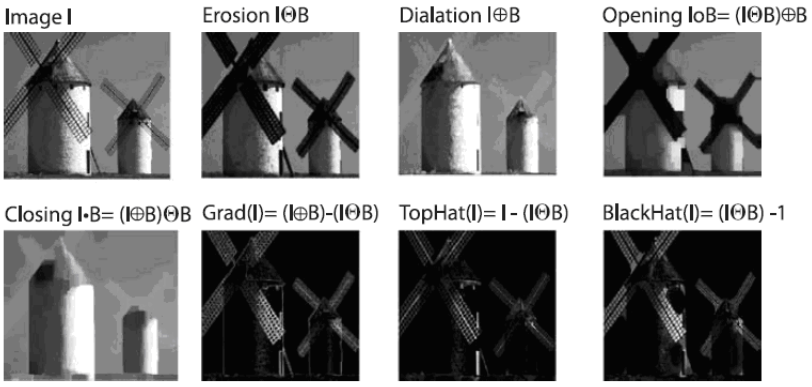


Figura 3.13: Operaciones morfológicas que se le pueden realizar a una imagen

g. Black Hat: Esta operación se define como la resta de la imagen original de la imagen después de un cierre (Black Hat = Cierre – Original) Esta operación aísla las zonas oscuras.

3.1.3. Segmentación

La segmentación es la siguiente fase a considerar dentro de un sistema de visión por computador. La segmentación es un proceso que consiste en dividir una imagen en regiones homogéneas con respecto a una o más características como puede ser el brillo o el color, con el fin de facilitar un posterior análisis o reconocimiento. Esto hace que se puede utilizar tanto para localizar objetos como para encontrar los bordes de los mismos dentro de una imagen.

La segmentación asigna una etiqueta a cada píxel de la imagen de forma que los píxeles que comparten etiqueta tendrán ciertas características similares visuales. Este proceso termina cuando los objetos extraídos de la imagen se corresponden unívocamente con las distintas regiones disjuntas a localizar en la misma. Se habla en este caso de segmentación completa, en oposición a la segmentación parcial, en la que no corresponde de manera unívoca y en la que se deberá hacer algún tipo de tratamiento posterior para conseguir la segmentación completa necesaria para completar el reconocimiento.

Los algoritmos de segmentación están basados en las discontinuidades de los niveles de gris, segmentando la imagen desde los cambios grandes de niveles de gris entre píxeles, como se hace en la detección de líneas, detección de bordes, detección de esquinas, de puntos aislados; o están basados en el efecto contrario, en la similitud de niveles de gris, al agrupar píxeles con características similares mediante umbralización, crecimiento de regiones, etc.

Existen numerosos métodos de segmentación. Se expondrán a continuación los métodos más relevantes y más usados en el mundo de la visión por computador.

a. Método del valor umbral

b. Transformación de Hough

c. Métodos basados en el histograma

d. Clustering o Método de agrupamiento

e. Método de crecimiento de regiones

f. Método del conjunto de nivel

g. Métodos de particionamiento gráfico

h. Watershed o transformación divisoria

i. Segmentación basada en modelos

j. Segmentación multiescala

k. Segmentación semiautomática

l. Redes neuronales de segmentación

m. Segmentación basada en el color

n. Segmentación basada en la textura

o. Segmentación basada en el movimiento

a. Método del valor umbral

El método del valor umbral o umbralización es un método por el que se convierte una imagen de niveles de gris o de color en una imagen binaria, de manera que los objetos de interés tienen una etiqueta distinta al fondo de la imagen. Esta técnica requiere un coste computacional muy bajo, y por lo tanto, es una técnica rápida, pudiendo ser realizada en tiempo real en un computador.

El caso más simple de umbralización es el umbral único o fijo, en el que se establece un umbral fijo sobre el histograma, que marca la separación y es el punto clave del método. El umbral es el límite por el que se divide la imagen, asignando cada píxel a uno de los dos segmentos comparando el umbral y el nivel.

Los métodos del valor umbral son métodos de segmentación completos, pues cada píxel pertenece a uno de los dos segmentos y solo a uno de esos segmentos. La imagen final se calcula con respecto al umbral t de la siguiente manera:

$$T_{global}(g) = 0 \; si \; g < t$$

$$T_{global}(g) = 1 \; si \; g \geq t$$

Existen varias variantes de este método, como es la umbralización de banda, en la que se pone un límite inferior y uno superior; la multiumbralización, que consiste en dar varios umbrales a la imagen dando lugar a regiones diferentes y no a una imagen binaria, la semiumbralización, consistente en mantener la imagen tal cual está a partir del umbral, pero binarizarla si el valor del píxel en inferior al umbral, o la umbralización adaptativa, en la que se permite el cambio del valor umbral según las características del entorno del punto de estudio.

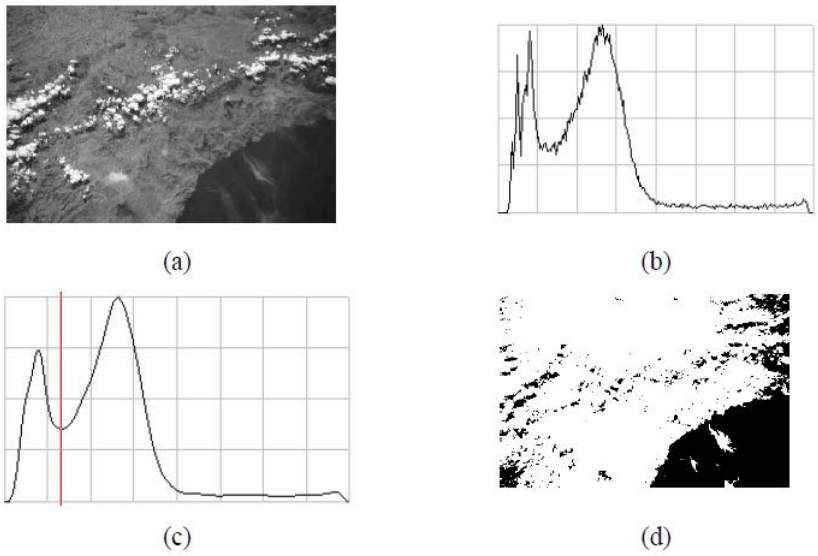


Figura 3.14: Umbralización de Otsu

En figura (b) vemos el histograma de niveles de gris de la figura (a). Para realizar la umbralización, se toma un nivel (c) a partir del cual, se binariza la imagen (d).

La umbralización tiene su explicación matemática en el método de Otsu, llamado así por su creador Nobuyuki Otsu en 1979. Este método utiliza la varianza, calculando el valor umbral de manera que esa varianza sea la menor posible en cada segmento, pero que sea máxima entre segmentos distintos. Se calcula el cociente de ambas varianzas y busca un valor umbral que sea el que maximice ese cociente.

b. Transformación de Hough

La transformación de Hough es un algoritmo creado en 1962, que permite encontrar ciertas formas como rectas, círculos, etc..., dentro de una imagen. La versión más simple es la que encuentra las líneas como características dentro de una imagen. Este algoritmo aprovecha la expresión del haz de rectas que pasan por un punto genérico (x_i, y_i), cuya ecuación es: $(y_i = ax_i + b)$; siendo a y b parámetros variables. Su modo de operación es principalmente estadístico y consiste en que para cada punto que se desea averiguar si es parte de una línea se aplica una operación dentro de cierto rango, con lo que se averiguan las posibles líneas de las que puede ser parte el punto. Esto se continúa para todos los puntos en la imagen, y al final se determinan las líneas de la imagen. Las líneas detectadas serán aquellas que tengan más puntos repetidos en el analisis, ya que la transformación se realiza buscando entre los vecinos del píxel analizado.

c. Métodos basados en el histograma

Los métodos basados en el histograma son métodos muy eficientes, pues solo requieren una única pasada por los píxeles de la imagen. Una vez se calcula el histograma, los picos y los valles de éste se usan para localizar grupos de píxeles en la imagen. Al refinar esta técnica se llega a un algoritmo recursivo de búsqueda en el histograma dividiendo la imagen en regiones cada vez más pequeñas.

Este método tiene la desventaja de que al ir dividiendo en regiones cada vez menores, la identificación de picos y valles cada vez es más compleja.

d. Clustering o Método de agrupamiento

El clustering es una técnica de segmentación iterativa con la cual se puede dividir una imagen en un número determinado de regiones o clusters.

El algoritmo básico es:

1. Escoger K centros de clusters, ya sea de forma aleatoria o basándose en alguno método heurístico.

2. Asignar a cada píxel de la imagen el clúster que minimiza la varianza entre el píxel y el centro del cluster.

3. Recalcular los centros de los clusters haciendo la media de todos los píxeles del cluster.

4. Repetir los pasos 2 y 3 hasta que se consigue la convergencia (por ejemplo, los píxeles no cambian de clusters).

En este caso, la varianza es la diferencia absoluta entre un píxel y el centro del cluster. La diferencia se basa típicamente en color, la intensidad, la textura, y la localización del píxel, o una combinación ponderada de estos factores. El número K se puede seleccionar manualmente, aleatoriamente, o por una heurística.

Este algoritmo garantiza la convergencia, pero puede devolver una solución que no sea óptima. La calidad de la solución depende de la serie inicial de clusters y del valor de K. En estadística y aprendizaje automático, el algoritmo de las k-medias es un algoritmo de agrupamiento para dividir objetos en k grupos, donde k <n. Es similar al algoritmo de maximización de expectativas para las mezclas de gaussianas ya que ambos pretenden encontrar los centros de agrupaciones naturales de los datos.

El modelo requiere que los atributos del objeto correspondan a los elementos de un espacio vectorial. El objetivo es intentar alcanzar la mínima varianza total entre clusters, o, la función de error al cuadrado. El algoritmo de las k-medias fue inventado en 1956.

La forma más común del algoritmo usa una heurística de refinamiento conocido como el algoritmo de Lloyd. El algoritmo de Lloyd comienza dividiendo los puntos de entrada en k conjuntos iniciales, ya sea al azar o usando algunos datos heurístico y a continuación, calcula el punto medio o centro de gravedad de cada conjunto. Se construye una nueva partición, asociando cada punto con el centro de gravedad más cercano. Luego se recalculan los baricentros es para los nuevos clusters, y el algoritmo repite alternando aplicación de estos dos pasos hasta que la converja, que se obtiene cuando los puntos ya no cambiar de cluster (o los centros de gravedad ya no se modifican).

e. Método de crecimiento de regiones

Los métodos de crecimiento de regiones determinan zonas dentro de una imagen basándose en criterios de similaridad y proximidad entre los píxeles de la misma. En estas técnicas la homogeneidad (o falta de homogeneidad) entre regiones adyacentes es el criterio utilizado para unir (o dividir) regiones de la imagen. Dicha homogeneidad se puede definir a partir de criterios como: el nivel de gris medio, el color, la forma, etc. El resultado de la segmentación es una partición de la imagen en regiones homogéneas.

Las técnicas de segmentación basadas en contornos tratan de encontrar fronteras entre regiones. En general, las técnicas basadas en regiones trabajan mejor en imágenes con ruido, en donde los contornos son difíciles de localizar. La segmentación resultante de una detección de contornos y la basada en crecimiento de regiones, aplicadas a una misma imagen no producen normalmente el mismo resultado. Es posible combinar los resultados producidos por ambos tipos de segmentaciones.

Las primeras técnicas de crecimiento de regiones fueron las del crecimiento a partir de semillas; en las que se dan una serie de semillas como entrada junto con la imagen. Las semillas marcan cada uno de los objetos que tienen que ser segmentados. Las regiones crecen iterativamente mediante la comparación de todos los píxeles vecinos no asignados a ninguna región.

f. Método del conjunto de nivel

La propagación de curvas es una técnica popular en el análisis de imágenes para la extracción de objetos, seguimiento de objetos, la reconstrucción en 3D, etc.

La idea central de este enfoque consiste en desarrollar una curva hacia el menor potencial de una función de coste que en su definición refleja la tarea a la que está dirigida e impone ciertas limitaciones de suavidad. Las técnicas de Lagrange se basan en la parametrización de acuerdo con alguna estrategia de muestreo y luego desarrollar cada elemento de acuerdo a la imagen y sus condiciones internas. Si bien esta técnica puede ser muy eficiente, también sufre varias limitaciones, como decidir sobre la estrategia de muestreo, la estimación de las propiedades geométricas internas de la curva, el cambio de su topología, abordar los problemas de dimensiones superiores, etc. En cada caso, una ecuación en derivadas parciales llamada la ecuación del conjunto de nivel es resuelto por diferencia finita.

El método del conjunto de nivel fue propuesto inicialmente para realizar un seguimiento de interfaces móviles por Osher y Sethian en 1988 y se ha diseminado a través de varios dominios de imágenes a finales de los noventa. Se puede utilizar para hacer frente de manera eficiente al problema de la propagación de la curva o superficie de una manera implícita. La idea central consiste en representar la evolución del contorno usando una función con signo, donde su nivel cero corresponde al contorno actual. Entonces, de acuerdo a la ecuación de movimiento de las curvas de nivel, se puede obtener fácilmente un flujo similar de la superficie implícita que cuando se aplica al nivel de cero reflejará la propagación del contorno. El método del conjunto de nivel tiene numerosas ventajas como son que es un método implícito, intrínseco, sin parámetros, que ofrece una manera directa para estimar las propiedades geométricas de la estructura que evoluciona, y que puede cambiar la topología.

g. Métodos de particionamiento gráfico

Los métodos de particionamiento gráfico se pueden usar con eficacia en la segmentación de imágenes. En estos métodos, la imagen se modela como un grafo ponderado no dirigido. Por lo general, un píxel o un grupo de píxeles se asocian con los nodos y los pesos de las aristas definen la similitud entre los píxeles vecinos. El gráfico (imagen) se divide de acuerdo a un criterio de diseño para modelar "bien" los clusters. Cada una de las particiones de nodos (píxeles) da como salida de estos algoritmos los objetos segmentados que hubiese en la imagen. Algunos algoritmos populares de esta categoría son cortes normalizados, camino aleatorio, el mínimo corte, particionamiento isoperimétrico y árboles de expansión mínima.

h. Watershed o transformación divisoria

La transformación divisoria o watershed calcula las líneas divisorias o líneas de aguas. Una imagen en escala de grises puede ser vista como un relieve topográfico, donde se interpreta el nivel de gris de un píxel como su altura en el relieve. También, se puede considerar la magnitud del gradiente de una imagen como una superficie topográfica. Los píxeles que tienen las más altas intensidades de gradiente corresponden a las líneas divisorias, que representan los límites de las regiones.

El agua puesta sobre cualquier píxel encerrado por una línea divisoria común fluye "colina abajo" a un mínimo de intensidad local común. Los píxeles que drenan a un mínimo común forman una cuenca, que representa un segmento de la imagen (un objeto).

i. Segmentación basada en modelos

La segmentación basada en modelos tiene como hipótesis fundamental que las estructuras de interés de la imagen tienen una forma geometría repetitiva. Por lo tanto, se puede buscar un modelo probabilístico para explicar la variación de la forma de la estructura y luego cuando se segmenta una imagen se imponen limitaciones para tomar la imagen como el modelo elegido a priori.

Esta tarea implica:

1. La selección de los ejemplos de entrenamiento (ejemplos que se usan para probar los modelos).

2. La representación probabilística de la variación de los ejemplos seleccionados.

3. La inferencia estadística entre el modelo y la imagen. El estado del arte para la segmentación basada en el conocimiento implica la forma activa y los modelos de apariencia, contornos activos y una plantilla deformable y métodos basados en niveles.

j. Segmentación multiescala

Las segmentaciones de la imagen se calculan en múltiples escalas y a veces se propagada de gran escala a pequeña escala. Los criterios de segmentación pueden ser arbitrariamente complejos y se pueden tener en cuenta tanto criterios globales como locales. Un requisito común es que cada región debe estar conectada en algún sentido.

Los métodos de segmentación multiescala más importantes son el de segmentación jerárquica de señales unidimensionales y el de segmentación de imágenes

k. Segmentación semiautomática

En este tipo de segmentación, el usuario define las regiones de interés con clics del ratón y los algoritmos se aplicarán de forma que se elige el camino que mejor se ajusta al borde de la imagen. Hay bastantes métodos que utilizan este tipo de segmentación interactiva como las tijeras inteligentes o el SIOX

l. Redes neuronales de segmentación

Las redes neuronales de segmentación se basan en el procesamiento de pequeñas áreas de una imagen utilizando una red neuronal artificial o un conjunto de redes neuronales. Después de este proceso de decisión se construye un mecanismo que marca las áreas de una imagen de acuerdo a la categoría reconocida por la red neuronal.

Un tipo de red diseñada especialmente para esto es el mapa de Kohonen. Las redes neuronales de parejas de pulsos (PCNNs) son modelos neuronales propuesto por modelos corteza visual de un gato y desarrollado para un alto rendimiento de procesamiento de imágenes biomiméticas.

El modelo de Eckhorn proporciona una herramienta sencilla y eficaz para estudiar la corteza visual de mamíferos pequeños, y pronto fue reconocido por ser una aplicación con un gran potencial en el procesamiento de imágenes. En 1994, el modelo de Eckhorn fue adaptado para ser un algoritmo de procesamiento de imágenes por Johnson, quien calificó este algoritmo como Pulse-Coupled Neural Network.

Aproximadamente desde el año 2000, PCNNs (por sus siglas en inglés) han sido utilizados para una variedad de aplicaciones de procesamiento de imagen entre las que se incluyen entre otras la segmentación de imágenes, la generación de características, la generación de funciones, la extracción de rostros, la detección de movimiento, la detección de regiones en crecimiento y la reducción de ruido.

Un PCNN es una red neuronal de dos dimensiones. Cada neurona en la red corresponde a un píxel en una imagen de entrada, recibiendo la información del color de su correspondiente píxel como un estímulo externo. Cada neurona se conecta con sus neuronas vecinas, recibiendo estímulos locales de ellas. Los estímulos externos y locales se combinan en un sistema de activación interna, que acumula los estímulos hasta que se excede un umbral dinámico, dando como resultado una salida de pulsos. A través de cálculos iterativos, las neuronas PCNN producen series temporales de impulsos de salidas. La serie temporal de impulsos de salidas contiene información de la imagen de entrada y puede ser utilizado para varias aplicaciones de procesamiento de imágenes, tales como la segmentación de la imagen y la generación de características. Comparado con los medios convencionales de procesamiento de imágenes, PCNNs tienen varias ventajas importantes, incluida la robustez frente al ruido, la independencia de las variaciones en los patrones geométricos de entrada, capacidad para pasar por pequeñas variaciones en los patrones de intensidad de entrada, etc.

m. Segmentación basada en el color

Como ya se ha visto anteriormente, el color se puede suponer como la unión de tres planos, tres canales, cada uno con sus niveles de intensidad en cada punto respecto las componentes de una base de color, normalmente la RGB. Una de las técnicas de segmentación basada en el color consiste en aplicar las técnicas anteriormente estudiadas a cada uno de los tres planos e integrar los resultados en la imagen, para que salga la imagen segmentada en color.

El clustering y la segmentación umbralizada en el modelo de color HSV son las técnicas que más se utilizan para segmentar basándose en el color.

n. Segmentación basada en la textura

En la segmentación basada en la textura, lo primero que tenemos que hacer es definir modelos de texturas para tratar la imagen como si fuese una función y no una colección de píxeles y así poder transformar una ventana de la imagen en un vector de características.

Los modelos de texturas se dividen en tres categorías, la primera basada en estructuras piramidales, la segunda en campos aleatorios y la tercera está basada en métodos estadísticos.

Dependiendo del número de texturas a reconocer y de si se tiene el conocimiento a priori de éstas, la segmentación basada en la textura estará clasificada en supervisada o no supervisada.

El objetivo de esta segmentación es el de reconocer regiones con un vector de características similar y asignarles la misma etiqueta. Es una segmentación compleja, pues necesita de la medida de la distancia entre esos vectores de características para discernir si son similares o no.

o. Segmentación basada en el movimiento

La segmentación basada en el movimiento se utiliza para discernir objetos animados de una secuencia de imágenes respecto de fondos estáticos. Esta técnica, conocida como sustracción de fondo, estudio la imagen resultante de la resta de dos imágenes consecutivas de la secuencia y deja los píxeles que se desplazan con valores distintos de cero al restar las imágenes, mientras que los que no se desplazan se mantienen con valores nulos al realizar la resta. Este tipo de segmentación depende muy fuertemente de la iluminación del entorno, pues la sustracción de fondo, al restar dos imágenes, se debe tomar un umbral en el que binarizar la imagen y que dependerá bastante de esa iluminación.

3.1.4. Reconocimiento

El reconocimiento es la última fase del sistema de visión artificial. En esta fase, el sistema interpreta finalmente la imagen, la clasifica y toma las decisiones pertinentes que previamente se le ha programado.

El reconocimiento tiene una primera parte de adquisición de datos, una extracción de características compleja y por último la toma de decisiones. Este proceso debe llevar al sistema de reconocimiento a asignar a cada objeto su categoría o clave que es el conjunto de entidades que comparten alguna característica que las diferencia del resto.

El reconocimiento lleva a la clasificación, entendida como la asignación de las diferentes partes del vector de características a grupos o clases, y basándose en las características extraídas, llegar al aprendizaje automático, desarrollando técnicas que permiten a las computadoras "aprender", reconocer formas, tamaños, objetos u otras características, obtenidas desde patrones de estudio.

Los algoritmos de aprendizaje se caracterizan por partir de un conjunto de funciones discriminantes con unos valores en sus parámetros que hacen que con se pueda discriminar optimizando las funciones respecto al objeto a reconocer de forma acertada. El proceso termina cuando el conjunto de entrenamiento se clasifica correctamente.

Los clasificadores utilizan alguno de estos procedimientos:

a. Geométricos, con los patrones con características que se pueden obtener mediante gráficos. En éste enfoque se emplea el cálculo de distancias, geometría de formas, vectores numéricos, puntos de atracción.

b. Estadísticos, que se basan en teorías probabilísticas y de estadística, como análisis de varianzas, covarianzas, dispersión, distribuciones, etc.

Supone que se tiene un conjunto de medidas numéricas con distribuciones de probabilidad conocidas y a partir de ellas se hace el reconocimiento.

a. Sintáctico-estructural: se basa en encontrar las relaciones estructurales que guardan los objetos de estudio, utilizando la teoría de lenguajes formales, teoría de autómatas, etc. El objetivo es construir una gramática que describa la estructura del universo de objetos.

b. Neuro-reticular: se utilizan redes neuronales que se 'entrenan' para dar una cierta respuesta ante determinados valores.

c. Lógico-combinatorio: se basa en la idea de que el modelado del problema debe ser lo más cercano posible a la realidad del mismo, sin hacer suposiciones que no estén fundamentadas. Se utiliza para conjuntos difusos y utiliza lógica simbólica, circuitos combinacionales y secuenciales, etc.

Según tengamos constancia o no de un conjunto previo que permita al sistema aprender, la clasificación puede ser supervisada, parcialmente supervisada o no supervisada.

a. Clasificación supervisada: también es conocida como clasificación con aprendizaje. Se basa en la disponibilidad de áreas de entrenamiento. Se trata de áreas de las que se conoce a priori la clase a la que pertenecen y que servirán para generar una signatura espectral característica de cada una de las clases. Se denominan clases informacionales en contraposición a las clases espectrales que genera la clasificación no supervisada. Algunos métodos de la clasificación supervisada:

i. Funciones discriminantes: si son dos clases, se busca obtener una función g tal que para un nuevo objeto O, si $g(O) \geq 0$ se asigna a la clase 1 y en otro caso a la 2. Si son múltiples clases se busca un conjunto de funciones g_i y el nuevo objeto se ubica en la clase donde la función tome el mayor valor.

ii. Vecino más cercano: un nuevo objeto se ubica en la clase donde esté el objeto de la muestra original que más se le parece.

iii. Redes neuronales artificiales: Se supone que imitan a las redes neuronales reales en el desarrollo de tareas de aprendizaje.

b. Clasificación parcialmente supervisada: también conocida como de aprendizaje parcial. En estos problemas existe una muestra de objetos sólo en algunas de las clases definidas.

c. Clasificación no supervisada: también conocida como clasificación sin aprendizaje. Se utilizan algoritmos de clasificación automática multivariante en los que los individuos más próximos se van agrupando formando clases.

Por último, destacar la clasificación de estos clasificadores en a priori o a posteriori, según sean clasificadores de un solo paso que utilizan la muestra de aprendizaje para el cálculo de las funciones discriminantes y un cálculo exacto (a priori) o clasificadores con aprendizaje, de procedimientos iterativos de entrenamiento, en el cual, el clasificador aprende a reconocer de forma progresiva los patrones de la muestra de aprendizaje (a posteriori).

Los clasificadores, como base del sistema de reconocimiento, tienen una gran aplicación además de la visión artificial, donde se usan para reconocimiento de caracteres, ya sea caracteres escritos a mano o impresos, reconocimiento facial, reconocimiento de objetos y reconocimiento de huellas dactilares, como son:

1. Previsión meteorológica: poder clasificar todos los datos meteorológicos según diversos patrones, y con el conocimiento a priori que tenemos de las diferentes situaciones que pueden aparecer nos permite crear mapas de predicción automática.

2. Reconocimiento de voz: el análisis de la señal de voz se utiliza actualmente en muchas aplicaciones, un ejemplo claro son los teleoperadores informáticos.

3. Aplicaciones en medicina: análisis de biorritmos, detección de irregularidades en imágenes de rayos X, detección de células infectadas, marcas en la piel.

4. Interpretación de fotografías aéreas y de satélite: gran utilidad para propuestas militares o civiles, como la agricultura, geología, geografía, planificación urbana.

5. Predicción de magnitudes máximas de terremotos.

6. Reconocimiento de música: identificar el tipo de música o la canción concreta que suena.

3.2. Control Visual

La visión por computador es uno de los sensores más estudiados de la robótica en los últimos tiempos, y su aplicación a UAS está muy relacionada con el control de la aeronave. El control visual es la aplicación más relevante de la visión artificial en UAS, y la que más se ha desarrollado últimamente debido a la mejora de velocidad de los procesadores y de las tarjetas digitalizadoras, al igual que la bajada de coste en las cámaras. Esto ha hecho que el procesamiento de imágenes a tiempo real, el control servo visual y el control de servo-manipuladores sea una realidad.

El control visual en vehículos autónomos necesita el uso de la información visual del procesamiento de imágenes para el control de la velocidad, posición relativa o absoluto y control de la orientación de UAS. Esa información, si se emplea para el posicionamiento de la aeronave, estaremos en el control servo visual del UAS.

El control visual de un UAS se puede realizar única y exclusivamente con los datos que el autopiloto recibe del control visual, o se puede realizar de manera clásica, con unidades inerciales de medida (IMU) o con sistemas GPS y usando el control visual como complemento de ayuda y de comprobación.

El control visual tiene diferentes enfoques para su clasificación que se exponen a continuación.

3.2.1. Clasificación por control

3.2.1.1. Directo

Las consignas de error provenientes del análisis visual de las imágenes se introducen directamente como consignas de control a los actuadores del sistema y el bucle de control interno no se usa.

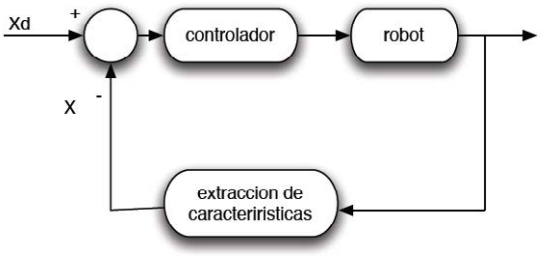


Figura 3.15: Control Directo

Es el comúnmente llamado control servo visual.

3.2.1.2. Indirecto

El sistema de control tiene la estructura de dos etapas.

El sistema posee un bucle interno rápido o controlador interno, y otro externo visual y más lento que genera las referencias en todo momento a ese controlador interno.

– Estático

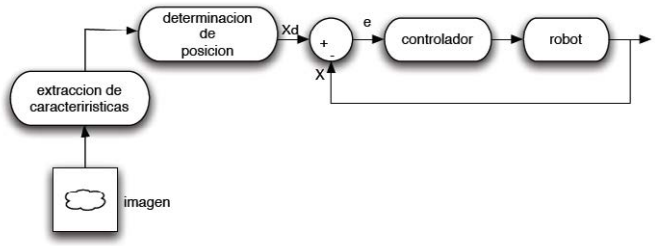


Figura 3.16: Control indirecto estático

Los bucles operan de forma secuencial.

– Dinámico

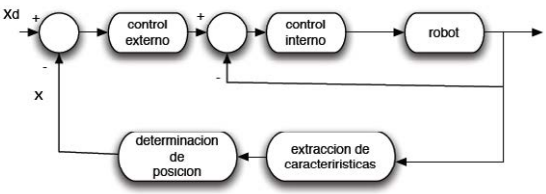


Figura 3.17: Control indirecto dinámico

La información está presente en todo momento, incluso cuando el sistema está en movimiento.

3.2.2. Clasificación por tipo de controlador

3.2.2.1. Image-Based Vision Servoing (IBVS)

Control visual basado en la imagen

En este método, se estima la posición de la aeronave y se proyecta sobre el plano de la imagen. La señal de error que se genera en la imagen es con la que se trabaja como señal de control.

Es un método que se usa sobre todo para el control 2D, pues el error se calcula en el plano de la imagen, que es bidimensional

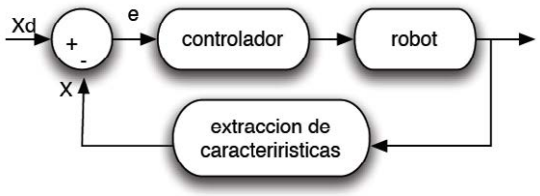


Figura 3.18: Control Image-based Vision Servoing

Es más lento de procesar pero es más preciso y robusto.

3.2.2.2. Position-Based Vision Servoing (PBVS)

Control visual basado en la posición.

Se conoce la posición de la aeronave y se corrige el error con la estimación de la posición que se calcula con la imagen. Requiere un modelado de la cámara, un calibrado de ésta y un modelo geométrico del objetivo. El control que minimiza el error se realiza en el espacio de trabajo de la aeronave. Usado para el control 2D y para el 3D. Normalmente, se utiliza en 3D, debido a que el error se calcula en el espacio de la aeronave, que es un modelo geométrico en tres dimensiones cargado previamente en la memoria del sistema.

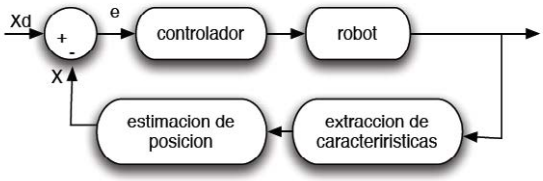


Figura 3.19: Control Position-based Vision Servoing

Más rápido de procesar y se puede realizar el control directo de la trayectoria pero requiere más hardware y necesita conocer bien la posición inicial.

3.2.2.3. Híbridos

Usan las dos teorías para mejorarlas y quedarse con lo bueno de cada una y así poder tener control en tiempo real.

Se usa sobre todo para reconocimiento de áreas, bordes y esquinas, para que el tiempo de procesamiento de la imagen sea el menor posible.

Los hay híbridos propiamente dichos, que son los llamados 2'5D, y los métodos particionados.

3.2.3. Clasificación por localización de la cámara

3.2.3.1. Eye-in-Hand

La cámara se mueve con el efector, pues está acoplado a él. El sistema de referencia de la cámara y del efector es el mismo. El control visual de UAS es el sistema que utiliza, pues normalmente lleva embarcada la cámara en la aeronave.

3.2.3.2. Eye-to-Hand

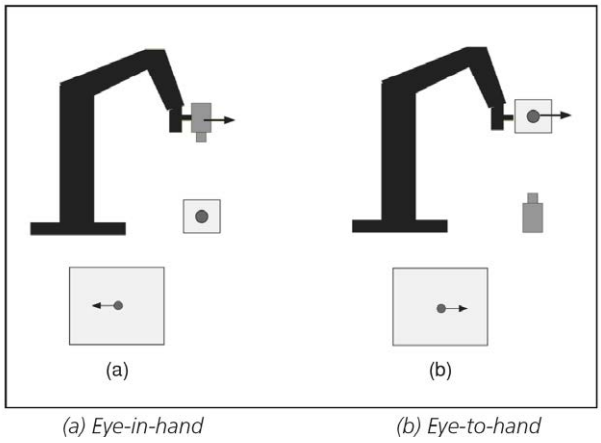


Figura 3.20: Control Eye-in-hand y Control Eye-to-Hand

El efector se mueve y la cámara está fija a un punto exterior a él. Se debe conocer la posición de la cámara respecto del efector en todo momento.

3.3. Aplicaciones en UAS

Las aplicaciones de la visión por computador y el control visual son muy amplias y variadas. Fundamentalmente, como ya se ha explicado, son aplicaciones militares, pues la aviación no tripulada está bastante desarrollada; ahora bien, en el ámbito civil, aún siendo aplicaciones potenciales, no pudiendo tener objetivos comerciales por imperativo legal, tenemos otras aplicaciones muy importantes:

1. Aplicaciones en ambientes de alta toxicidad o radiación, para control de muestras con alto peligro para la vida humana.
2. Cooperación en misiones de narcotráfico y terrorismo.
3. Cartografía: Realización de mapas y realización de modelos de elevaciones del terreno de alta resolución.
4. Agricultura y gestión de cultivos.
5. Servicios forestales: seguimiento de las áreas boscosas, control de incendios y búsqueda de puntos calientes tras el proceso de control.

6. Geología.
7. Hidrología.
8. Medio ambiente: estado de la atmósfera.
9. Control de obras y evaluación de su impacto.
10. Seguimiento de la planificación urbanística.
11. Gestión del Patrimonio.

Además, los centros universitarios de investigación y desarrollo que tuvieron su mención en la introducción de este capitulo han estado estudiando y desarrollando investigaciones para poder aplicar la visión artificial y dar a conocer al mundo académico, y al resto de la humanidad, sus avances en sus estudios a través de artículos en revistas especializadas, conferencias, tesis y demás proyectos de investigación. Entre las más relevantes, se encuentran las que se muestran a continuación:

1. Seguimiento de objetivos, ya sean móviles o fijos.
2. Reconocimiento de objetos y características del terreno para aterrizajes.
3. Reconocimiento del terreno libre de obstáculos para aterrizajes seguros.
4. Reconocimiento y búsqueda de líneas eléctricas.
5. Localización de Puntos Calientes en el terreno tras un incendio.
6. Persecución y Evasión de enemigos.
7. Diseño de rutas preprogramadas, asegurándose el seguimiento de la ruta al obligar a volar por una serie de puntos de paso.
8. Despegue y Aterrizaje autónomo, ya sea de manera clásica o mediante maniobra de encaramado (a similitud de algunas aves).
9. Vuelo a punto fijo de aeronaves, ya sea de ala fija o de ala rotatoria.

3.4. Hardware

Los sistemas de visión artificial son sistemas que requieren de elementos computacionales y de captura para realizar su función. Como ya se ha estudiado, estos sistemas necesitan de hardware en la que se adquiera las imágenes y se procese la información, así como se necesita de un software, una serie de programas que se implementen en el hardware y realicen las misiones programadas.

Dentro del hardware de los sistemas de visión por computador, es necesario un sensor de captura, que normalmente suele ir dentro de una cámara, ya sea fotográfica o de vídeo, un procesador donde se implementan el software que realiza el procesado, segmentación, reconocimiento de imágenes y da órdenes al resto de sistemas para realizar sus misiones. Adicionalmente, en aplicaciones para UAS, donde el sensor de captura y el procesador de imágenes normalmente van embarcados en la aeronave, se necesitará algún sistema para conectar el procesador de imágenes al autopiloto de la aeronave, y además, otro sistema, normalmente mediante transmisión inalámbrica de datos entre la aeronave y una estación de seguimiento en tierra que vigile las actuaciones de la aeronave y permita controlar el vuelo del UAS desde tierra en cualquier momento mediante algún programa informático. Puede darse el caso en el que el procesado de las imágenes se haga en la estación de tierra, obteniendo la ventaja del ahorro de peso, pues solo debe ir embarcada la cámara y la mejora

del procesado de la imagen, pues en tierra se puede tener procesadores mucho más potentes que embarcados, donde tienen que tener un compromiso entre el peso y la potencia de cálculo, pero tiene el inconveniente de que la transmisión de vídeo necesita de canales de transmisión de datos potentes y de transmisión de vídeo de baja calidad, añadiendo además el problema del retardo en las comunicaciones.

3.4.1. Sensores de captura

Los sensores de captura son los encargados de capturar la luz a la que están expuestos mediante componentes que se excitan con la luz, y componer la imagen fotográfica que buscamos. Funciona convirtiendo la luz que capta en señales eléctricas que almacena, mide y convierte en una representación electrónica del patrón de iluminación que llegó al sensor.

Actualmente, la mayoría de los sensores comerciales de imagen digital se basan en el efecto fotoeléctrico, emitiendo electrones al ser excitados con los fotones que reciben.

Los sensores de captura más utilizados son los siguientes:

- CCD

El sensor CCD (Charge Couple-Device) es el sensor más utilizado en cámaras y videocámaras digitales. En estos aparatos, el CCD es el sensor con diminutas células fotoeléctricas que registran la imagen. Desde allí la imagen es procesada por la cámara y registrada en la tarjeta de memoria. La capacidad de resolución o detalle de la imagen depende del número de células fotoeléctricas del CCD. Este número se expresa en píxeles. A mayor número de píxeles, mayor resolución. Actualmente las cámaras fotográficas digitales incorporan CCD con capacidades de hasta ciento sesenta millones de píxeles (160 megapíxeles) en cámaras Carl Zeiss.

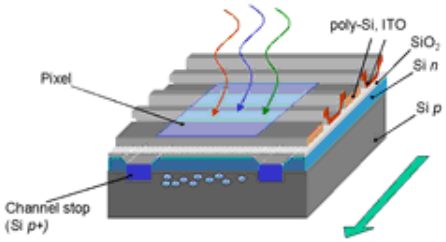


Figura 3.21: Detector CCD

Los píxeles del CCD registran gradaciones de los tres colores básicos: rojo, verde y azul (RGB), por lo cual tres píxeles, uno para cada color, forman un conjunto de células fotoeléctricas capaz de captar cualquier color en la imagen. Para conseguir esta separación de colores la mayoría de cámaras CCD utilizan una máscara de Bayer que proporciona una trama para cada conjunto de cuatro píxeles de forma que un píxel registra luz roja, otro luz azul y dos píxeles se reservan para la luz verde (el ojo humano es más sensible a la luz verde que a los colores rojo o azul). El resultado final incluye información sobre la luminosidad en cada píxel pero con una resolución en color menor que la resolución de iluminación. Se puede conseguir una mejor separación de colores utilizando dispositivos con tres CCD acoplados y un dispositivo de separación de luz como un prisma dicroico que separa la luz incidente en sus componentes rojo, verde y azul. Estos sistemas son mucho más caros que los basados en máscaras de color sobre un único CCD. Algunas cámaras profesionales de alta gama utilizan un filtro de color rotante para registrar imágenes de alta resolución de color y luminosidad pero son productos caros y tan solo pueden fotografiar objetos estáticos.

Los detectores CCD, al igual que las células fotovoltaicas, se basan en el efecto fotoeléctrico, la conversión espontánea de luz recibida en corriente eléctrica que ocurre en algunos materiales. La sensibilidad del detector CCD depende de la eficiencia cuántica del chip, la cantidad de fotones que deben incidir sobre cada detector para producir una corriente eléctrica. El número de electrones producido es proporcional a la cantidad de luz recibida (a diferencia de la fotografía convencional sobre negativo fotoquímico). Al final de la exposición los electrones producidos son transferidos de cada detector individual (fotosite) por una variación cíclica de un potencial eléctrico aplicada sobre bandas de semiconductores horizontales y aisladas entre sí por una capa de SiO2. De este modo, el CCD se lee línea a línea, aunque existen numerosos diseños diferentes de detectores.

En todos los CCD el ruido electrónico aumenta fuertemente con la temperatura y suele doblarse cada 6 u 8 °C. En aplicaciones astronómicas de la fotografía CCD es necesario refrigerar los detectores para poder utilizarlos durante largos tiempos de exposición.

Históricamente la fotografía CCD tuvo un gran empuje en el campo de la astronomía donde sustituyó a la fotografía convencional a partir de los años 80. La sensibilidad de un CCD típico puede alcanzar hasta un 70 % comparada con la sensibilidad típica de películas fotográficas en torno al 2 %. Por esta razón, y por la facilidad con la que la imagen puede corregirse de defectos por medios informáticos, la fotografía digital sustituyó rápidamente a la fotografía convencional en casi todos los campos de la astronomía. Una desventaja importante de las cámaras CCD frente a la película convencional es la reducida área de los CCD, lo que impide tomar fotografías de gran campo comparable a algunas tomadas con película clásica. Los observatorios astronómicos profesionales suelen utilizar cámaras de 16 bits, que trabajan en blanco y negro. Las imágenes en color se obtienen tras el procesamiento informático de imágenes del mismo campo tomadas con diferentes filtros en varias longitudes de onda.

Las imágenes obtenidas por una cámara CCD son sometidas a un proceso de corrección que consiste en restar de la imagen obtenida la señal producida espontáneamente por el chip por excitación térmica (campo oscuro) y dividir por una imagen de un campo homogéneo (campo plano o flat field) que permite corregir las diferencias de sensibilidad en diferentes regiones del CCD y corregir parcialmente defectos ópticos en la cámara o las lentes del instrumento utilizado.

- CMOS

También se le conoce como Active píxel Sensor (APS). El APS también se basa en el efecto fotoeléctrico, ya que está formado por tantos fotositos como píxeles tenga la imagen, que producen una corriente eléctrica en función de la intensidad recibida. En el CMOS, a diferencia del CCD se incorpora un amplificador de la señal eléctrica en cada fotosito y es común incluir el conversor digital en el propio chip. En un CCD se tiene que enviar la señal eléctrica producida por cada fotosito al exterior y desde allí se amplifica. La ventaja es que la electrónica puede leer directamente la señal de cada píxel con lo que se soluciona el problema conocido como blooming, por el que la recepción de una gran intensidad lumínica en un punto influye en los píxeles adyacentes (un brillo fuerte produce líneas blancas en la imagen).

La desventaja es que entre los receptores de luz (fotositos) se encuentra mucha electrónica que no es sensible a la luz, lo que implica que no pueda captar tanta luz en una misma superficie del chip. La solución al problema vino no sólo por una mayor densidad de integración, por lo que la electrónica no sensible se reducía en tamaño, sino por la aplicación de microlentes que a modo de lupa concentran la luz de cada celda en su fotosito Debido a que no se podía alcanzar la densidad de integración necesaria para competir con el CCD, esta tecnología careció de importancia durante los años 70, 80 y mitad de los 90.

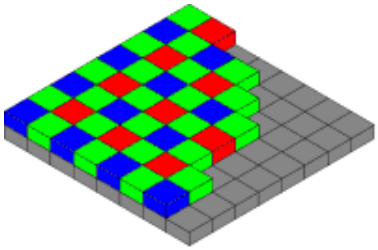


Figura 3.22: Detector CMOS con filtro de Bayer

Al igual que ocurre con el CCD, los fotositos captan únicamente intensidad lumínica, para lo que se suele emplear un filtro conocido como máscara de Bayer para la distinción de los colores. Mediante esta máscara unos fotositos tienen un filtro para recoger solo la luz roja, otros para la verde y otros para el azul.

Según los fabricantes de CCDs, los sensores CMOS tienen un elevado ruido de patrón fijo (ruido que no varía con el tiempo y que se ve como un fondo fijo en la imagen) pero sus defensores indican que tienen un bajo consumo de energía lo que redunda en una mayor autonomía de la cámara. Al parecer, el 'ruido' mencionado se debe a que los sensores CMOS convencionales tienen un amplificador por separado en cada píxel y estos amplificadores normalmente no serán uniformes por todo el chip y la desigualdad residual será la que genere el ruido. Por el contrario, todos los píxeles de un CCD se activan a través de una etapa común del amplificador, de modo que se evita este problema. Por otro lado, los fabricantes de CMOS argumentan que los sensores CCD necesitan una electrónica externa compleja que eleva el coste. En la práctica, es posible encontrar implementaciones de alta calidad de ambas tecnologías. Finalmente, se achaca a los sensores CMOS una escasa sensibilidad a la luz ultravioleta e infrarroja.

Las ventajas y desventajas dependen en parte de cada dispositivo puesto que es posible encontrar sensores CCD con características similares a los CMOS y viceversa. Sin embargo, es posible listar las características típicas como siguen:

Ventajas

- Consumo eléctrico muy inferior.
- Económico (necesita pocos componentes externos).
- Lectura simultánea de mayor número de píxeles.
- El conversor digital puede estar integrado en el mismo chip.
- Escaso Blooming ("Smear") o inexistente.
- Mayor flexibilidad en la lectura (Previsualización más rápida, vídeo,...).
- Los píxeles pueden ser expuestos y leídos simultáneamente.
- Otras topologías posibles (el sensor Super CCD de Fujifilm emplea una construcción en forma de panel (octogonal) para los píxeles).
- Distintos tipos de píxeles (según tamaño y sensibilidad) combinables.
- Muy alta frecuencia de imagen en comparación a un CCD del mismo tamaño.

Desventajas

- Menor superficie receptora de la luz por píxel.
- Menor uniformidad de los píxeles (mayor ruido de patrón fijo-FPN).

Debido a su bajo coste, el APS comenzó a emplearse masivamente en webcams y en las cámaras de los teléfonos móviles. Sin embargo, hoy día también se utiliza en cámaras réflex digitales de objetivo único pues no sólo superan en luminosidad a los sensores CCD, sino que también producen menos ruido. En el campo de las videocámaras se usan sensores CCD, con la excepción de las cámaras de alta definición Sony HDR-HC1, HDR-HC3 y la Sony FX7 (que utiliza 3 sensores CMOS). También se emplea el APS en cámaras industriales.

- Super CCD

El sensor Super CCD es un Sensor CCD propietario desarrollado por Fujifilm en 1999.

El Super CCD utiliza una geometría de píxeles octogonal en lugar de rectangular y presenta una mayor superficie fotosensible al aprovechar mejor el área que los CCD dedican al cableado entre elementos fotosensibles permitiendo que haya más área disponible para la recolección de la luz entrante.

En enero de 2003 Fujifilm anunció la cuarta generación de sensores SuperCCD en dos variaciones: SuperCCD HR y SuperCCD SR. HR viene de "High Resolution" (alta resolución) y SR de "Super dynamic Range" (rango súper-dinámico). El sensor SR tiene dos fotodiodos por elemento fotosensible lo cual permite generar una mayor gama de luminancia desde el negro hasta el blanco (rango dinámico).

La cuarta generación de Super CCD HR tiene sensores situados a 45^{o} de la línea horizontal (algo posible gracias a la geometría octogonal) y por lo tanto las columnas de píxeles están escalonadas en comparación con su posición estándar lo cual reduce la separación efectiva del píxel. Para obtener imágenes con la orientación normal horizontal y vertical interpola un píxel entre cada par de sensores, produciendo por tanto la grabación de 12Mpx desde 6Mpx efectivos. Como contrapartida, Fujifilm promete que la quinta generación de sensores HR será también de 45 grados pero no usará interpolación.

En teoría, el resultado de la optimización del área fotosensible es una mejor sensibilidad y un menor ruido de lo que se generaría usando el mismo área global de píxeles cuadrados.

Esto permite en definitiva una mayor resolución vertical y horizontal (a expensas de la resolución diagonal) frente a los sensores CCD tradicionales con el mismo número de píxeles.

- FoveonX3

El Foveon X3 es un sensor CMOS de imagen, de la marca Foveon, formado por tres capas apiladas verticalmente: cada elemento de la matriz del sensor está formado por tres capas cada una de las cuales es sensible a uno de los colores primarios (RGB).

Cada una de las diferentes longitudes de onda de los colores primarios se absorben en distintas capas, pues las ondas más largas (rojas) tienen una mayor profundidad de penetración en el silicio que las más cortas (azules). Mediante el uso de un filtro que bloquea la luz infrarroja se consigue en las capas del sensor una sensibilidad al color similar a la de los conos del ojo humano. En la película fotográfica de color se emplea el mismo principio, que también emplea distintas capas sensibles al color, unas sobre otras.

Habitualmente se utilizan sensores CMOS o CCD con una máscara de Bayer, de modo que un píxel de color se forma con 4 elementos sensores adyacentes que reciben distintos tonos y que luego se interpolan. Por el contrario en el sensor Foveon X3, cada elemento sensor recibe la información completa del color. Por ello la resolución es mayor que en los sensores convencionales a igual número de píxeles.

El sensor Foveon X3 es distinto en la forma de apilar los sensores rojo, verde y azul, pues lo hace uno sobre otro en lugar de colocarlas lado a lado como es el caso del filtro de Bayer. Esto quiere decir que, en lugar de limitarse a una componente de color, cada elemento del sensor puede resolver todo un color suprimiendo la interpolación de los datos de color.

Una propiedad interesante del Foveon X3 es que un mayor porcentaje de los fotones que entran en la cámara serán detectados por el fotosensor; en principio lo serán casi todos frente al tercio que con los otros sensores Bayer.

Resulta también interesante el hecho teórico de que mientras los sensores CCD y CMOS de filtro Bayer tienen una mayor resolución de luminancia que de color (captan mejor los matices de iluminación que de color), en el chip Foveon X3 las dos resoluciones son teóricamente iguales. Es por esto que existe una cierta controversia con respecto a la cuantificación de la mejora que supone este sistema realmente.

Otras ventajas del sensor Foveon X3 son la reducción de artefactos, la obtención de un color más real y un detallado de texturas más ajustado. Según algunos como desventaja tiene una alta producción de ruido en fotografías de exposiciones largas, pero otros análisis parecen concluir que tanto su rango dinámico, su riqueza cromática y la producción de ruido son mejores en este tipo de sensor

A principios del año 2010 las únicas cámaras fotográficas que equipan el sensor Foveon X3 son las Sigma SD14 réflex y las compactas Sigma DP1 y DP2. En septiembre de 2008 Sigma anunció el desarrollo del modelo SD15, que sucederá al SD14 pero que también equipara sensor Foveon X3.

3.4.2. Distintas cámaras

Las cámaras fotográficas se dividen en primera instancia en analógicas y digitales. La cámara analógica se basa en la exposición de la luz en una cámara oscura proyectándose en negativo a una película de material fotosensible.

Las cámaras fotográficas constan de una cámara oscura cerrada, con una abertura en uno de los extremos para que pueda entrar la luz, y una superficie plana de formación de la imagen o de visualización para capturar la luz en el otro extremo. La mayoría de las cámaras fotográficas tienen un objetivo formado de lentes, ubicado delante de la abertura de la cámara fotográfica para controlar la luz entrante y para enfocar la imagen, o parte de la imagen. El diámetro de esta abertura, conocido como apertura, suele modificarse con un diafragma, aunque algunos objetivos tienen apertura fija.

Mientras que la apertura y el brillo de la escena controlan la cantidad de luz que entra por unidad de tiempo, en la cámara durante el proceso fotográfico, el obturador controla el lapso en que la luz incide en la superficie de grabación. Por ejemplo, en situaciones con poca luz, la velocidad de obturación será menor y por lo tanto estará mayor tiempo abierto para permitir que la película reciba la cantidad de luz necesaria para asegurar una exposición correcta. Las cámaras digitales, por el contrario, capturan la imagen con alguno de los sensores anteriormente descritos, y comprimen la imagen para su almacenamiento en memorias flash o en discos duros. Las cámaras digitales se clasifican según su sensor de captura, su espectro de frecuencia que capturan o según su protocolo de conexión con el exterior. Según el espectro de frecuencia, se distinguen cámaras infrarrojas, cámaras de espectro visible, cámara de rayos X, cámaras ultravioletas, etc.

Según el protocolo de conexión con el exterior se tienen cámaras USB, cámaras Firewire o cámaras de conexión de 27 pines (Camera Link). Hay también tarjetas para conectar la cámara a la placa base del ordenador, mediante conexión PCI Express.

Por último, se tienen que hablar de las cámaras estenopeicas o pinhole, donde la apertura de la luz es tan pequeña como la cabeza de un alfiler, haciendo fotografías con profundidad de campo infinita, lo que lleva al encuadre de cualquier punto de la imagen, y de las cámaras réflex, en las que el fotógrafo ve directamente la imagen que va a tomar a través de un visor óptico sin errores de paralaje.

Las cámaras de vídeo, que también tiene esa diferencia entre analógico y digital, tienen un funcionamiento similar a las cámaras fotográficas en el mundo digital, pues lo que realiza la cámara de vídeo es capturar imágenes a 1/24 de segundo para que el vídeo se vea continuo y no se puedan distinguir los fotogramas que lo componen.

Las cámaras de vídeo analógicas funcionan de la siguiente manera. La cámara de vídeo transforma la secuencia de escenas ópticas en señales eléctricas. Se compone de un objetivo, un tubo de cámara, y los dispositivos electrónicos de control. La luz se enfoca dentro del tubo de cámara sobre una superficie fotosensible para crear la señal de vídeo. Esta señal, consistente en una onda que describa la intensidad de cada punto de la línea de la pantalla a través de la amplitud, separa la información de las señales de control. La conversión de la señal lumínica en eléctrica se realiza sobre la diana, de diámetro entre 12 mm y 30 mm, que es otra superficie fotosensible que da la resolución de la cámara.

Finalmente, si lo que se quiere obtener de la cámara analógica es un vídeo digital, se necesita una tarjeta digitalizadora, que depende del número de muestras que es capaz de tomar de la señal de vídeo y de la resolución radiométrica que es capaz de alcanzar.

3.4.3. Procesadores

El procesador es el elemento donde se lleva la información para que se le apliquen los tratamientos que se han ido definiendo en las distintas fases del sistema de visión artificial.

Además, es el encargado de ejecutar todos los programas, empezando por el sistema operativo, los accesos de memoria, los programas escritos en lenguajes de programación de bajo nivel, etc.

Para aplicaciones embarcadas en UAS, se requiere un compromiso entre el tamaño del conjunto placa base-procesador, el peso y la potencia de cálculo, ya que normalmente, a mayor peso y tamaño, mayor potencia de cálculo. Además, se requiere suficiente memoria de acceso aleatorio (RAM) para que los programas que se deben ejecutar, se procesen a velocidad adecuada y no de manera lenta.

Hay numerosos modelos y tipos de procesadores, debido al auge de la informática en los últimos veinte años. Eso hace que sea inabordable dar una lista de todos los modelos de procesadores. Lo que se mostrará a continuación son los modelos más comunes, con su arquitectura, y el modelo a usar en este proyecto.

- Procesadores **PC**
- Intel
  Posibilidad de doble y cuádruple núcleo de proceso.
  - Modelos:
    1. Atom
    2. Core
    3. Xeon
- AMD
  Tecnología de 64 bits.
  - Modelos:
    1. Athlon
    2. Phenom
    3. Sempron
    4. FM1
- Procesadores ARM
  Procesadores para sistemas móviles y embarcados, como son teléfonos móviles, tablets, PDA.
  Posibilidad de usarlos como ordenadores de sobremesa o embarcados de bajo coste.
  - Modelos:
    Gumstix:
    * Verdex Pro
    * Overo:
      1. FE
      2. IronSTORM
      3. FireSTORM
      4. Air
      5. AirSTORM
      6. Earth
      7. EarthSTORM

    Rapsberri Pi
    BeagleBoard:
    * Rev. xm -C
    * Rev. C5

    BeagleBone A5
    PandaBoard:
    * Modelo A4
    * Modelo B1

    ARM Cortex:
    * Serie M
    * Serie A

3.5. Software

El software es el componente lógico que hace posible la realización de tareas específicas, o lo que es lo mismo, la aplicación informática que da la información y la orden de la realización de la tarea.

En los sistemas de visión artificial, el software se programa en lenguaje de programación para realizar las tareas de cada fase anteriormente estudiada, y poder así discernir en las imágenes las características adecuadas.

3.5.1. Programas utilizados en visión por Computador

Los programas a utilizar en las fases de un sistema de visión por computador son muchos y variados. A continuación se dará una lista de algunos de los principales y se centrará la información en la librería más utilizada para la programación en lenguajes de programación como C, C++ o Python, que es la librería OpenCV.

* Software Privativo:
  - DALSA:
    - Inspect
    - Sherlock
  - Common Vision Blox
  - MVTec Software GmbH:
    - ActiVisionTools
    - HALCON

* Software Abierto:
  - OpenCV
  - Code::Blocks
  - Qt
  - SimpleCV
  - EmguCv
  - ITK
  - VTK
  - ROS
  - TINA

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel (IPP).

OpenCV se implementa como librería para códigos de programación en C, C++, Python, Java de manera muy sencilla y con mucho material de información tanto en Internet como en manuales, lo que ha hecho que sea la más utilizada para sistemas de visión por computador. En este proyecto, se implementará la aplicación de visión por computador en C, a través de Code::Blocks, que es un es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C y C++. Está basado en la plataforma de interfaces gráficas WxWidgets, lo cual quiere decir que puede usarse libremente en diversos sistemas operativos, y está licenciado bajo la Licencia pública general de GNU. OpenCV no permite la creación de interfaces gráficas a través de botones, lo que se solucionará implementando las librerías de Qt, que es otro entorno de desarrollo de programas, diseñado por Nokia, que si permite este tipo de interfaces gráficas y que se complementa con OpenCV.

3.5.2. Algoritmos

Los algoritmos son la base de los programas que luego se implementaran en el software del sistema de visión artificial. Estos algoritmos dan la base de lo que tiene que realizar el programa en cuestión

Los algoritmos más conocidos, además de los ya explicados en la segmentación y en el procesado de imágenes, que se usan en sistemas de visión por computador son los que siguen a continuación.

– Lucas - Kanade

Se usa para seguimiento de objetivos. En visión por computador, el método de Lucas - Kanade es un método diferencial para la estimación del flujo óptico de gran uso desarrollado por Bruce D. Lucas y Takeo Kanade. Asume que el flujo es constante en la vecindad de los píxeles en consideración y resuelve las ecuaciones básicas del flujo óptico para todos los píxeles en esa vecindad, bajo el criterio de mínimos cuadrados. Combinando información de todos los píxeles cercanos, este método resuelve la ambigüedad inherente a la ecuación del flujo óptico. Es un método poco sensible al ruido pero por otro parte, no ofrece información del flujo en el interior de regiones uniformes de la imagen.

– Filtros de Kalman

Se usa para predecir la posición de un móvil y poder hacer aplicaciones See & Avoid. El filtro de Kalman es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para poder identificar el estado oculto (no medible) de un sistema dinámico lineal, al igual que el observador de Luenberger, pero sirve además cuando el sistema está sometido a ruido blanco aditivo. La diferencia entre ambos es que en el observador de Luenberger, la ganancia K de realimentación del error debe ser elegida "a mano", mientras que el filtro de Kalman es capaz de escogerla de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema.

– Segmentación, normalmente Binaria, para Reconocimiento del terreno

– Reconocimiento de áreas por relaciones de vecindad entre píxeles

Aplicación de IR para seguimiento de ríos.

– Reconocimiento de bordes y registro de imágenes por relaciones angulares entre características lineales

Aplicación para reconocimiento de pistas y colocación respecto a ellas.

– Transformación de Hough

La transformación de Hough, como se especificado anteriormente, es un algoritmo desarrollado en 1962, que permite encontrar ciertas formas geométricas como rectas o círculos dentro de una imagen. La versión más simple de este algoritmo busca líneas como características dentro de una imagen. Este algoritmo se basa en la expresión del haz de rectas que pasan por un punto genérico (x_i, y_i), cuya ecuación es: $(y_i = ax_i + b)$; siendo a y b parámetros variables. Se profundizará algo más en ella en los capítulos posteriores de este proyecto.

– Algoritmo de búsqueda de esquinas

La detección de esquinas es un método utilizado en los sistemas de visión por ordenador para extraer ciertos tipos de características e inferir el contenido de una imagen. La detección de la esquina se utiliza con frecuencia en la detección de movimiento, registro de imagen, seguimiento por cámara, mosaicos de imágenes, composición panorámica, modelado 3D y reconocimiento de objetos. Este método de detección de esquinas se engloba el tema de la detección de puntos de interés.

Una esquina se puede definir como la intersección de dos bordes. Una esquina también se puede definir como un punto para el que hay dos direcciones de borde dominantes y diferentes en una zona local del punto. Un punto de interés es un punto en una imagen que tiene una posición bien definida y puede ser detectado de forma clara y robusta. Esto significa que un punto de interés puede ser una esquina, pero puede ser también, por ejemplo, un punto aislado de máxima intensidad local o de mínima intensidad local, un final de línea, o un punto en una curva donde la curvatura es localmente máxima.

En la práctica, la mayoría de los llamados métodos de detección de esquinas detectan los puntos de interés en general, en lugar de esquinas en particular. Como consecuencia de ello, si sólo detecta las esquinas es necesario hacer un análisis local de los puntos de interés detectados para determinar cuáles de ellas son esquinas reales. Ejemplos de detección de bordes que se pueden utilizar con post-procesamiento para detectar esquinas son el operador de Kirsch y el conjunto de enmascaramiento de Frei-Chen.

"Esquina", "punto de interés" y "característica" se utilizan indistintamente en la literatura, haciendo más confuso la comprensión de los métodos. En concreto, hay varios detectores de blobs que se puede denominar como "operadores de puntos de interés", pero que a veces erróneamente se han denominado "detectores de la esquina". Además, existe un método de detección de crestas o surcos, para capturar la presencia de objetos alargados.

Los detectores de esquina no suelen ser muy robustos y con frecuencia requieren la supervisión de expertos o de redundancias con el fin de evitar el efecto de los errores individuales que dominan la tarea de reconocimiento.

Una determinación de la calidad de un detector de esquina es su capacidad para detectar la misma esquina en múltiples imágenes similares, bajo diferentes condiciones de iluminación, traslación de la imagen, rotación de la imagen y otras transformaciones.

Se tiene un enfoque simple para la detección de esquinas de las imágenes a través de correlación, pero esto se vuelve muy costoso computacionalmente hablando. Un enfoque alternativo utiliza con frecuencia se basa en un método propuesto por Harris y Stephens que a su vez es una mejora de un método por Mora.

– SIFT (Scale-Invariant Feature Transform)

Para cualquier objeto en una imagen, los puntos de interés en el objeto pueden ser extraídos para proporcionar una "descripción de la característica" del objeto. Esta descripción, extraída de una imagen de entrenamiento, a continuación, se puede utilizar para identificar el objeto e intentar localizarlo en una imagen de prueba que contiene muchos otros objetos. Para realizar el reconocimiento de manera fiable, es importante que las características extraídas de la imagen de entrenamiento sean detectables incluso bajo cambios en la escala de la imagen, cambio en el nivel de ruido o cambios de iluminación. Estos puntos generalmente se encuentran en regiones de alto contraste de la imagen, como pueden ser los bordes del objeto.

Otra característica importante de estas características es que las posiciones relativas entre ellos en la escena original no deben cambiar de una imagen a otra. Por ejemplo, si sólo las cuatro esquinas de una puerta fueron utilizadas como características, funcionará independientemente de la posición de la puerta, pero si los puntos en el marco se utilizan también, el reconocimiento puede fallar dependiendo si la puerta está abierta o cerrada. Del mismo modo, las características de los objetos situados en puntos articulados o flexibles no suelen servir si tienen algún cambio en su geometría interna entre dos imágenes de la serie que se está procesando.

Sin embargo, en la práctica SIFT detecta y utiliza un número mucho mayor de características de las imágenes, lo que reduce la contribución de los errores causados por estas variaciones locales en el promedio de todos los errores de emparejamiento.

– SURF (Speeded Up Robust Feature)

SURF es un detector robusto de características locales, presentado por primera vez por Herbert Bay en 2006, que puede ser utilizado en tareas de visión por computador, como el reconocimiento de objetos o la reconstrucción 3D.

En gran parte se inspira en el descriptor SIFT. La versión estándar de SURF es bastante más rápido que SIFT y sus autores aseguran que es más robusto frente a las transformaciones en imágenes diferentes que SIFT. SURF se basa en las sumas de las respuestas del modelo 2D wavelet de Haar y hace un uso eficiente de imágenes integrales.

Se utiliza una aproximación a números enteros para el determinante de detector Hessaino (Hessian blob detector), que puede calcularse de forma extremadamente rápida con una imagen integral pues son tres operaciones de enteros. Para las características, utiliza la suma de la respuesta del wavelet de Haar alrededor del punto de interés. De nuevo, estos se pueden calcular con la ayuda de la imagen integral.

– Aplicación de estos algoritmos a RGB para cámaras a color, a través de los histogramas de color de la cámara

– Flujo Óptico

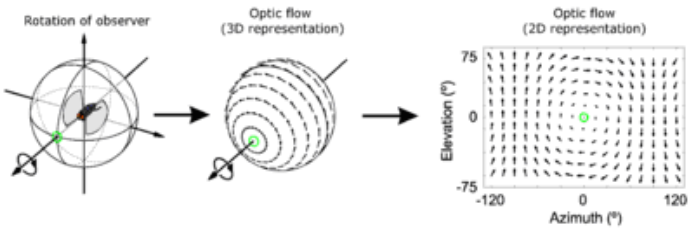


Figura 3.23: Flujo Óptico

El flujo óptico es el patrón del movimiento aparente de los objetos, superficies y bordes en una escena causado por el movimiento relativo entre un observador (un ojo o una cámara) y la escena. El concepto de flujo óptico se estudió por primera vez en la década de 1940 y, finalmente, fue publicado por el psicólogo estadounidense James J. Gibson como parte de su teoría de la affordance (una acción que un individuo puede potencialmente realizar en su ambiente). Las aplicaciones del flujo óptico tales como la detección de movimiento, la segmentación de objetos, el tiempo hasta la colisión y el enfoque de cálculo de expansiones, la codificación del movimiento compensado y la medición de la disparidad estereoscópica utilizan este movimiento de las superficies y bordes de los objetos.

Capítulo 4

Definición de un caso de aplicación viable

La visión por computador es un campo de estudio dentro de la robótica muy amplio, donde caben aplicaciones de muy distinto tipo, como puede ser el reconocimiento y clasificación de piezas en una fábrica, el posicionamiento de robots respecto algún punto significativo o la microscopia digital.

En UAS, se ha visto anteriormente numerosas aplicaciones tanto civiles como militares, en las que la visión por computador puede ser un sistema vital de navegación, o simplemente una ayuda más.

Por eso, se ha de definir una aplicación viable a la que hacer el seguimiento, el desarrollo teórico y los ensayos. Para ello, se iniciará la exposición dando unas reseñas sobre varias soluciones ya existentes, y después se definirá la aplicación en dos pasos.

Figura 4.1: Helicópteros Yamaha R-50, Yamaha RMAX y Bergen Industries Twin utilizados para sus investigaciones por el grupo de David Shim en la UC at Berkeley (California, USA)

Hay una gran cantidad de artículos y de libros sobre aplicaciones de la visión por computador a los UAS, como anteriormente se introdujo. La gran mayoría de los investigadores se decantaron en su

día, por la aeronave de ala rotatoria debido a sus peculiaridades de vuelo. Así, el grupo de Vision for Unmanned Systems de la Universidad Politécnica de Madrid, con Pascual Campoy a la cabeza, usa el COLIBRI, en sus dos versiones, de gas y eléctrico, en el que hacen sus investigaciones y sus aplicaciones como el detector de objetos, en su caso un globo rojo, y posterior seguimiento, o el control del helicóptero basado en la información visual que recibe; el grupo de la UC at Berkeley, propusieron un esquema de aterrizaje autónomo con un helicóptero Yamaha R-50, basado en la localización de patrones, dentro de su proyecto BEAR; al igual que el grupo de la universidad de Tübingen, que proponen un aterrizaje y despegue autónomo sobre una plataforma en movimiento. Otro grupo de investigación que trabajan con helicópteros son los investigadores de la Southern California University, en el proyecto AVATAR, y los desarrolladores del proyecto ICARUS de la Universitat Politècnica de Catalunya usan un helicóptero para la monitorización de incendios forestales, además del estudio del control de éste a través de un autopiloto virtual.

Figura 4.2: Prototipo del quad-rotor X4-Flyer diseñado por Robert Mahony

Robert Mahony (Australian National University) y Tarek Hamel (Universite Nice Sophia Antipolis) proponen un control visual basado en la imagen para robots aéreos, implementándolo en un quadricóptero X4. Los investigadores de la universidad de Chemnitz también tienen su quadricóptero en el que ensayan otro modelo de patrón para aterrizar, los investigadores de la universidad de Ljubljana también controlan su X-3D-BL mediante técnicas de control visual basado en imágenes, y aterrizajes basados en reconocimiento de patrones y los del grupo de Vision for Unmanned Systems de la Universidad Politécnica de Madrid, también tienen investigaciones con quadricópteros en los que la aeronave realiza un seguimiento de un objeto y se mantiene en vuelo a una distancia determinada del obstáculo. En el laboratorio de sistemas inteligentes de la École Polythecnique Federal de Lausana desarrollaron un sistema de control basado en el flujo óptico, que permite aterrizajes y despegues autónomos a quadricópteros. Otro grupo de investigación que realiza importantes investigaciones en visión artificial en el campo de los quadrotores es el grupo PIXHAWK de la Eidgenössische Technische Hochschule Zürich (ETH Zürich), que utiliza el mismo hardware que se va a utilizar en el desarrollo de este proyecto.

Figura 4.3: SMARTBAT. UAS de ala fija de la UC at Berkeley

Figura 4.4: Helicóptero Electra, utilizados para sus investigaciones por el grupo de David Shim en la UC at Berkeley (California, USA)

Hay otros muchos grupos de investigación que basan sus desarrollos en aeronaves de ala fija, siendo bastante más complejo de implementar y ensayar, debido a que la imagen debe procesarse en tiempo real. En una aeronave de ala rotatoria, al poder mantenerse en punto fijo de manera sencilla, se puede permitir retrasos en el procesado de la imagen, pues no compromete al vuelo, pero en aeronaves de ala fija eso no se permite. Así, la universidad de North Carolina, ha resultado victoriosa en varias ocasiones de la competición del AUVSI con una aeronave de ala fija controlada por visión a través de una cámara y software de posicionamiento GIS. Otro grupo que trabaja con aeronaves de ala fija es el que François Chaumette dirige en el IRISA-INRIA francés, en el que trabajan mediante simulaciones informáticas con el videojuego Microsoft Flight Simulator, en el aterrizaje automático de un avión tanto en pista como en portaaviones. Otros grupos son los de Randall Beard en la

Brigham Young University de Provo, Utah, que trabaja con mini UAVs con control tanto por voz como por imágenes, del vuelo; o el de Andrew Miller, en la Central Florida University. En la UC at Berkeley, también se han desarrollado aplicaciones de UAS de ala fija, especializándose en la detección de bordes de ríos y lagos y el seguimientos de éstos debido al uso de cámaras infrarrojas. La planificación de rutas es otra aplicación que se puede llevar a cabo de manera eficaz con visión artificial, y eso es lo que han desarrollado en la Carnegie Mellon University.

Figura 4.5: Aerial Robotics Club de la Universidad de Carolina del Norte, ganadores del Concurso de AUVSI en 2010

Robert Mahony y Tarek Hamel, en colaboración con Florent Le Bras (Delegation Générale pour l'Armement), han desarrollado un sistema de visión por computador en el que se controla aeronaves de ala fija, y no solo rotatorias, mediante un control servo visual basado en la imagen con buenos resultados.

Hay otras aplicaciones que aunque no son aplicaciones en UAS, pueden servir como inspiración, como idea, para implementar en un UAS. Así, el proyecto GOLD de los italianos Massimo Bertozzi y Alberto Broggi de la Universidad de Parma o el proyecto YARF del profesor Kluge de la Universidad de Michigan son una gran ayuda al reconocimiento de pistas de aterrizaje, aun siendo ideados para el seguimiento y reconocimiento de carreteras.

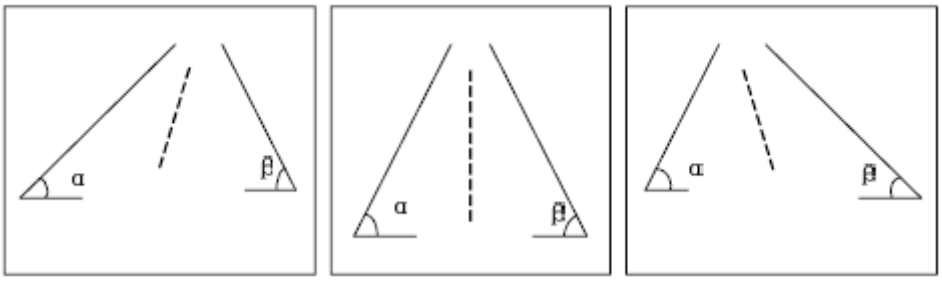


Figura 4.6: Casos de visión en el que el UAS ve la pista y decide hacia donde ha de moverse. Es la base del proyecto de Zhogke Shi y Jiajia Shang

Los investigadores de la Universidad de Xi'an Zhongke Shi y Jiajia Shang han propuesto un reconocimiento de pistas de aterrizaje para aeronaves de ala fija en el que basándose en características visuales, se puede controlar la aeronave. La idea principal está en que, mediante segmentación en banda, algoritmos de contornos de Sobel, dilatación de la imagen, transformadas de Hough y ajuste

por mínimos cuadrados se obtenga de la imagen, una representación de la línea del horizonte y de los bordes de la pista, para, al comparar el ángulo que forman ambos bordes con la horizontal, conocer la posición relativa del avión con la pista y poder corregirla si fuera necesario. Para ello, trabajaron también con Flight Simulator, incorporando su algoritmo a un programa informático con Visual C.

En la imagen de la izquierda se ve que el ángulo α es menor que el β, con lo cual se tiene que la aeronave circula dejando a la derecha la pista, por lo que para controlar el vuelo, se necesita que la aeronave vire hace la izquierda, y al contrario, si se ve que el ángulo β es menor que el α, con lo cual se tiene que la aeronave circula dejando a la izquierda la pista, y se debe corregir volando hacia la derecha. Si ambos ángulos son iguales, la aeronave volará alineada con el centro de la pista y no se requerirá ninguna corrección. Esta investigación será la base del desarrollo de este proyecto, pues es una aplicación conceptualmente sencilla, pero de compleja implementación.

4.1. Actuación sobre el rudder del UAS para modificar la dirección de su trayectoria en función de la imagen obtenida por una cámara embarcada, monitorizada desde tierra

La aplicación que se desarrollará en este proyecto será la actuación sobre el rudder de un UAS para modificar la dirección de la trayectoria en función de la imagen obtenida por una cámara embarcada, monitorizada desde tierra. Esta aplicación, se engloba bien en la anteriormente descrita, pues lo que se mostrará es la corrección de la trayectoria de la aeronave durante el descenso, para alinearse con la pista.

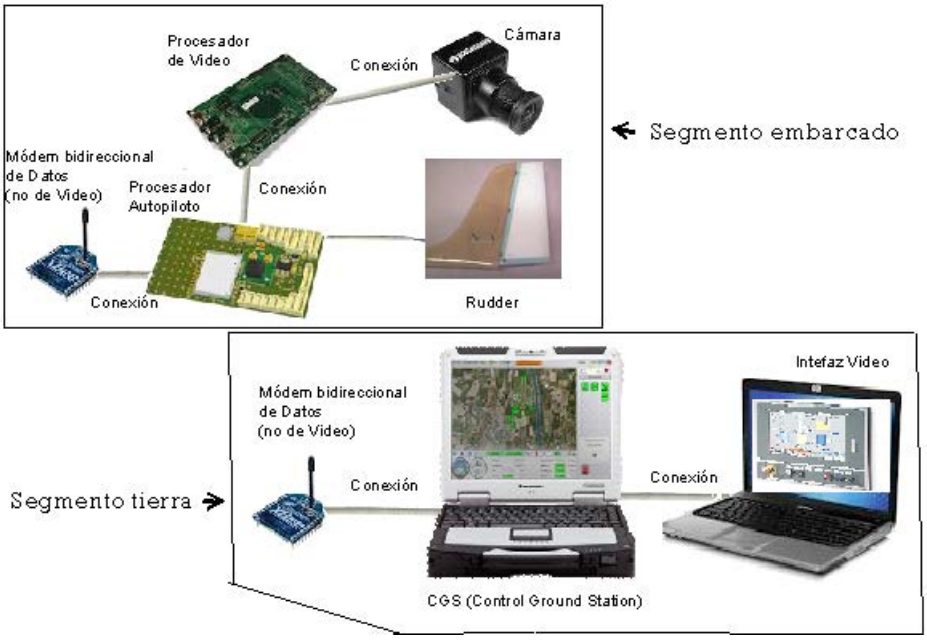


Figura 4.7: Croquis de montaje del sistema de actuación sobre el rudder

La aplicación que se ha desarrollar consiste en embarcar tanto la cámara, como el procesador de vídeo, y actuar sobre el autopiloto, que será el encargado de transmitir la orden al actuador. Con eso, se necesita de un compromiso en el procesador entre potencia de cálculo, peso y dimensiones. Además, se necesita una conexión eficaz entre los procesadores del autopiloto y del procesador de vídeo, y se necesita mucha eficacia en el programa de procesado, segmentación y reconocimiento de imágenes, pues debe de ser lo más rápido posible, para que la aeronave pueda responder en tiempo

real, o al menos, con el suficiente retraso como para no ver peligrar su seguridad. Para no comprometer esa seguridad, la aeronave puede ser controlada desde tierra vía estación de radiocontrol.

Como se ve en la imagen, el desarrollo de la aplicación consta de dos partes bien diferenciadas, que son el segmento embarcado y el segmento de tierra.

El segmento embarcado, del que ya se ha hecho referencia anteriormente, consta de la cámara, para la adquisición de imágenes, del procesador de imágenes, del autopiloto, del actuador y de las conexiones entre todos ellos. Además, consta de un módem bidireccional de datos para enviar la información a tierra, y recibir del segmento de tierra las instrucciones de encendido y apagado. Como un posible trabajo futuro, se puede incluir otra conexión inalámbrica entre tierra y avión para la transmisión del vídeo capturado por la cámara, pues el envío de los paquetes de vídeo es un envío complejo desde un solo módem, pues satura los canales de información y encima el vídeo debe de estar muy comprimido y ser de muy baja calidad, lo que dificulta su recepción. Esa transmisión de vídeo estaría comprendida entre el procesador de vídeo y la estación de tierra del procesador de vídeo, sin interferir en la transmisión de información del autopiloto.

El segmento de tierra se compone de otro módem bidireccional de datos, una estación en tierra para el procesado de imágenes y la estación en tierra del autopiloto. La estación en tierra para el procesado de vídeo será la que se ha de programar a conciencia, dando una interfaz para el usuario en la que dé el estado del sistema, si el sistema está encendido o apagado, y si está encendido, dé la posición de la aeronave respecto la pista y su corrección.

Además, el autopiloto debe ser la responsable del activación y la desactivación del sistema de corrección de trayectoria basado en visión a través de una orden y debe generar una señal de salida en la interfaz de tierra.

4.2. Actuación sobre un servo en función de la imagen obtenida por una cámara embarcada, monitorizada desde tierra

La idea básica es la misma que la de la aplicación anterior. Únicamente cambia el elemento sobre el que se actúa. Esto es debido a que los conocimientos técnicos y prácticos de los que se dispone no nos permite afrontar el objetivo último del proyecto que es la aplicación anterior.

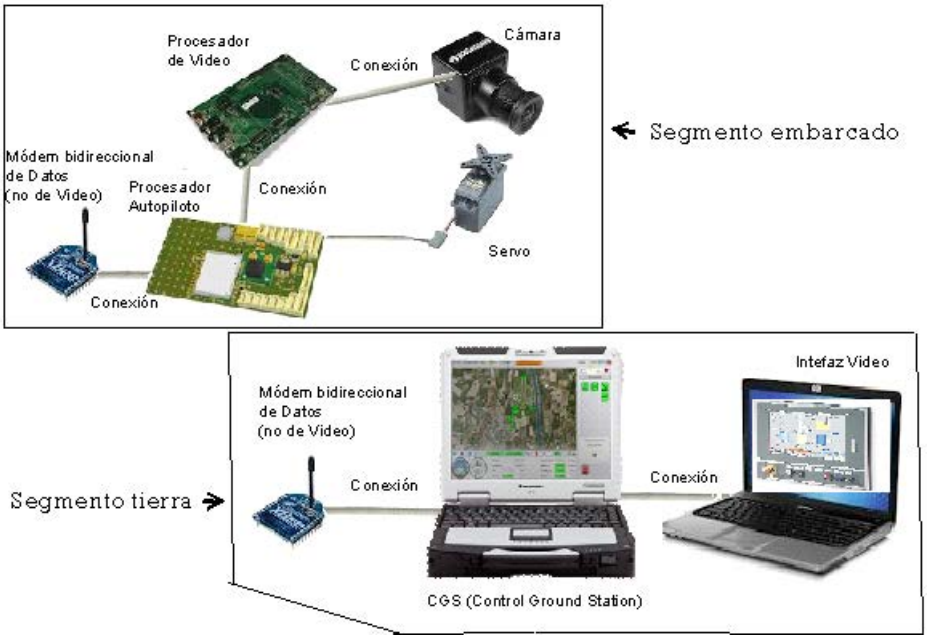


Figura 4.8: Croquis de montaje del sistema de actuación sobre el servo

Como se ha avanzado, el hecho de realizar esta aplicación en lugar de la anterior, es una cuestión puramente formal. La esencia del proyecto es la misma, pero esta aplicación permite tde blobsener una seguridad con la aeronave que no permite la otra. El hecho de mover un servo y no el actuador que mueve el rudder del avión permite que el avión no se vea en riesgo en las misiones de ensayo del sistema, pues la aeronave no se verá afectada por la puesta en marcha y la prueba del sistema de visión por computador y además, se tiene la potestad de poder pasar a control manual en cualquier momento desde tierra a través de una estación de radiocontrol que gobierna las actuaciones del autopiloto.

El desarrollo del proyecto implica una serie de conocimientos en programación que han de ser adquiridos a la vez que el proyecto se realiza, con lo que el servo se hace mucho más seguro a la hora de programar que el alerón. Todo esto permite a los futuros desarrolladores trabajar desde la aplicación segura hacia el control automático de la aeronave en su descenso, y en un futuro no muy lejano, el control total de la aeronave con la imagen capturada por las cámaras.

La aplicación que se ha desarrollar consiste en embarcar tanto la cámara, como el procesador de vídeo, y actuar sobre el autopiloto, que será el encargado de transmitir la orden al servo, además, monitorizar es estado del sistema desde una interfaz en tierra, como se ve en la imagen anterior.

El segmento embarcado, del que ya se ha hecho referencia anteriormente, consta de la cámara, para la adquisición de imágenes, del procesador de imágenes, del autopiloto, del servo y de las conexiones entre todos ellos. Además, consta de un módem bidireccional de datos para enviar la información a tierra. El segmento de tierra se compone de otro módem bidireccional de datos, una estación en tierra para el procesado de imágenes y la estación en tierra del autopiloto, de manera idéntica a la aplicación sobre el actuador.

Capítulo 5

Análisis de necesidades

El sistema de visión por computador, como ya se ha explicado, tiene por objetivo que un computador sea capaz de reconocer ciertas características dentro de una imagen y dar órdenes a otro sistema basándose en ese reconocimiento. Para ello, se requiere como mínimo de un ordenador con su procesador y de un elemento que capture de imágenes como es la cámara de vídeo.

Además de esos elementos físicos, el sistema requiere disponer en la memoria del computador, los programas para arrancar el sistema y compilar el código de programación para realizar las tareas programadas y los algoritmos necesarios para ejecutar dicho programa y el reconocimiento de imágenes.

Anteriormente ya se incluyeron otros algoritmos muy usados en visión artificial, tales como el de Lukas-Kanade o el detector de esquinas, para servir de inicio y base a posibles trabajos futuros. En este apartado, se presentará el algoritmo en el que se basa el programa de detección de características en la imagen.

5.1. Hardware

El hardware de un sistema informático lo forman todos los componentes físicos, ya sean eléctricos, electrónicos, electromecánicos o mecánicos.

Dentro del hardware, para un sistema de visión artificial, la unidad central de procesamiento o procesador y la cámara son los elementos más importantes, pero hay otros muchos como son las memorias, tanto las de acceso aleatorio (RAM) como las de almacenamiento masivo, la unidad de procesamiento gráfico (GPU) y los periféricos de entrada, de salida y de entrada y salida (E/S).

Este apartado se centrará en los dos componentes básicos debido a un motivo estrictamente formal, ya que comercialmente, en la mayoría de los casos, tanto las memorias, como la GPU y los periféricos están integrados en una placa base junto con el procesador, por lo que forma parte de sus características.

5.1.1. Procesador

Los sistemas de visión por computador embarcados en UAS son sistemas que han de tener una capacidad potente de cálculo para trabajar en tiempo real, pero a su vez deben de ser sistemas livianos, para evitar disminuir la carga de pago del UAS. Por ello, se ha de llegar a una solución de compromiso. Gracias al auge de los sistemas para telefonía móvil, hay una gran variedad comercial que aúnan las dos características esenciales y nos permite poder hacer una selección amplia.

El procesador es el componente fundamental del procesador y es el encargado de ejecutar las instrucciones y de procesar los datos. La memoria RAM o de acceso aleatorio es la que utiliza el ordenador para el almacenamiento transitorio y de trabajo, donde almacena temporalmente la información, los datos y los programas que el procesador lee, procesa y ejecuta. La memoria de almacenamiento masivo es el dispositivo donde se almacena la información de trabajo como el sistema operativo del ordenador. Es de gran capacidad y normalmente debe alimentarse a la corriente desde la fuente de alimentación. La tarjeta gráfica o unidad de procesamiento gráfico tiene por objetivo liberar a la CPU de la tarea de cálculo de operaciones gráficas, y así aprovechar mejor la potencia de cálculo de la CPU y la memoria RAM del sistema.

El procesador que se requiere para este cometido, como ya se ha especificado, debe tener un compromiso entre el peso y la potencia de cálculo. Por ello, se han estudiado varias opciones, cuyas características se expondrán a continuación y se ha seleccionado una de esas opciones alegando los motivos que se expondrán al término del capítulo.

Las opciones que se barajaron son:

1. Gumstix Overo
   - Características Gumstix Overo IronSTORM:
     - Arquitectura: ARM Cortex-A8
     - Rango de Temperaturas:
       Componentes -40 ^{o}C $<$ T $<$ 85 ^{o}C
       Ranura Tarjeta MicroSD: -25 ^{o}C $<$ T $<$ 85 ^{o}C
       Módulo Bluetooth/Wifi: -20 ^{o}C $<$ T $<$ 75 ^{o}C
     - Procesador en C.O.M.: Texas Instruments DM3730 Digital Video Processor
     - Velocidad del Procesador: 800 MHz
     - Procesador de Señales Digitales: (DSP) High Performance Image, Video, Audio (IVA2.2™) Accelerator Subsystem
     - OpenGL: POWER SGX™ Graphics Accelerator (Acelerador Gráfico)
     - RAM: 512 MB
     - NAND: 512 MB
     - Performance: Up to 2,000 Dhrystone MIPS Bluetooth Included 802.11 b/g
     - Incluye un conector de Cámara 1 x 27-pines, donde se conectará la cámara del sistema de visión por computador
     - Incluye Ranura de tarjetas microSD
     - Power Management: Texas Instruments TPS65950
     - Conectores 2 x 70-pin AVX .
     - C.O.M. Montaje: Cuatro (4) x #0 agujeros de montaje para asegurarla a la placa de expansión Overo-series, o similar
     - Alimentación: vía placa de expansión (Overo series o similar, se utiliza la placa de expansión Gumstix Overo Tobi) conectado a conector dual de 70-pines
     - Peso GS3703FE @ 5.6g GUM3703FE @ 42.6g (incl. caja de a bordo y antenas)
     - Longitud: 58 mm
     - Anchura: 17 mm

2. BeagleBone
   - BeagleBone A5
   - Características:
     - Procesador superescalar ARM Cortex™-A8 cuya velocidad $>$ 700 MHz
     - Memoria RAM DDR2 de 256 MB
     - 1 puerto USB 2.0 host
     - 10/100 Ethernet Integrada

• Ranura de tarjetas microSD y tarjeta microSD con validación y imagen de demostración con la distribución Angstrom de UNIX

• Puerto USB 2.0 flexible para dispositivos con capacidad para alimentar

• Puerto USB-a-Puerto serie/JTAG embebido compartido con el puerto USB de dispositivos

• 3.3V - 2 Cabezales de expansión de 46 pines para periféricos con señales LCD multiplexadas y control de batería

• Tamaño: 8,6 × 5,4 cm.

3. BeagleBoard
   - BeagleBoard –xM (rev. C)
   - Características:
     - Super-scalar ARM Cortex-A8 1GHz.
     - Memoria RAM LPDDR de 512 MB
     - Puerto USB 2.0 OTG de alta velocidad que opcionalmente puede alimentar a la placa.
     - Hub embebido de 4 Puertos USB 2.0 de alta velocidad con 10/100 Ethernet.
     - Salida de video DVI-D (Para monitores y televisiones digitales).
     - Salida de video S-video (Para televisiones analógicas).
     - Audio Estéreo tanto de entrada como de salida.
     - Ranura para tarjetas microSD de alta capacidad and tarjeta microSD de 4 GB.
     - Interfaz JTAG
     - Puerto para Cámaras.

4. PandaBoard
   - PandaBoard B1
   - Características:
     - Núcleo Lógico con Procesador OMAP4460 para aplicaciones.
     - Doble núcleo ARM® Cortex™-A9 MPCore™ con multiprocesamiento simétrico (SMP) a una velocidad mayor de 1.2 GHz cada uno.
     - Codificación y Descodificación Full HD (1080p) multi-standard de video
     - Imagination Technologies' POWERVR™ SGX540 núcleo gráfico con soporte a la mayoría de API's incluyendo OpenGL® ES v2.0, OpenGL ES v1.1, OpenVG v1.1 y EGL v1.3 y que ofrece un rendimiento mejor comparado con el anterior núcleo SGX530
     - Audio de baja potencia
     - Memoria
       - 1 GB DDR2 RAM de baja potencia
       - Ranura de tarjetas SD/MMC con soporte para tarjetas de alta velocidad y de alta capacidad.
     - Display
       - Conector HDMI v1.3 (Tipo A) para conectar dispositivos HD
       - Conector DVI-D (puede conectar un segundo dispositivo, mediante display simultáneo, requeriendo adaptador de HDMI a DVI-D)
       - Cabezales de expansión para LCD
     - Soporte DSI
     - Audio
       - 3.5" Jack de Audio tanto entrada como salida
       - Salida de audio HDMI
       - Soporte de entrada de audio estéreo
     - Conectividad
       - 10/100 Ethernet Embebido
       - Conectividad Wireless 802.11 b/g/n (basada en WiLink™ 6.0)

- Bluetooth® v2.1 + EDR (basada en WiLink™ 6.0). Bluetooth de baja energía.
- Puerto USB 2.0 OTG de alta velocidad
- 2 Puertos USB 2.0 host de alta velocidad
- Cabezales de expansión para cualquier periférico (I2C, GPMC, USB, MMC, DSS, ETM)
- Cabezales de expansión para cámaras de video
- Señales de expansión LCD usando un único conjunto de bancos de resistencias
- Debug (Depuración de Programación interna)
  - JTAG
  - UART/RS-232
  - 2 LEDs de status (configurable)
  - 1 Botón GPIO
  - Botón de arranque en la placa
- Dimensiones
  - Alto: 114,3 mm
  - Ancho: 101,6 mm
  - Peso: 82 gramos

– PandaBoard A4
  – Características:
    - Dimensiones: 11,5 x 10,2 mm
    - OMAP 4430 dual core ARM Cortex-A9 procesador @ 1 GHz
    - POWERVR SGX540 núcleo gráfico que soporta OpenGL® ES v2.0, OpenGL ES v1.1, OpenVG v1.1 and EGL v1.3
    - Ranuras de tarjetas SD/MMC de tamaño completo
    - 10/100 Ethernet
    - 802.11 b/g/n Wi-Fi
    - Bluetooth 2.1+EDR
    - HDMI v1.3 – Full HD y DVI-D
    - 3.5" audio de entrada/salida
    - 1 puerto USB 2.0 OTG
    - 2 puertos host USB 2.0
    - 14-pin JTAG
    - UART vía conector DB-9
    - 26 GPIO pins en cabezales de expansión para compatibilidad con I2C, GPMC, USB, MMC, DSS, and ETM
    - LEDs configurables para la depuración
    - Alimentación vía USB o vía de conector de 5V DC

5. Rapsberry Pi
  – Características Raspberry Pi Linux
    - SoC Broadcom BCM2835 (CPU, GPU, DSP, y SDRAM)
    - CPU: Procesador de 700 MHz con núcleo ARM1176JZF-S (Familia ARM11)
    - GPU: Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC decodificador de perfil alto.
    - Memoria (SDRAM): 256 Megabytes (MiB)
    - Salidas de Video: Composite RCA, HDMI
    - Salidas de Audio: 3.5 mm jack, HDMI
    - Almacenamiento Onboard: Ranuras para tarjetas SD, MMC, SDIO
    - 10/100 Ethernet RJ45 conexión de red embarcada
    - Almacenamiento vía ranuras para tarjetas SD/ MMC/ SDIO

Se ha seleccionado la opción del procesador Gumstix Overo IronStorm FE, con la placa Gumstix Overo Tobi como complemento para conectar los periféricos como el módem bidireccional debido a la facilidad de interacción con el autopiloto Paparazzi del UAS, ya que se conoce un manual online de conexión entre ambos. Además, tenía la ventaja frente a alguno de sus competidores de ser un sistema mucho más ligero y compacto, aún siendo algo menos económico y con una potencia de cálculo un poco menor que la del PandaBoard B1, su mejor competidor.

5.1.2. Cámara

La cámara es el objeto que captura la luz y genera las imágenes que posteriormente serán tratadas en el procesador. La cámara elegida para ir embarcada ha de ser una cámara ligera, debe tener una buena resolución y una buena calidad de imagen y además tener una conexión adecuada con el procesador, ya sea vía USB, vía Firewire o vía conector específico.

Las opciones que se barajaron para la cámara fueron las siguientes:

– Gumstix Caspa
- Características
  - Tecnología de imagen Micron® DigitalClarity® CMOS
  - Formato: Wide-VGA, activo 752H x 480V (360,960 píxeles)
  - Obturador Global de fotodiodos; Integración y lectura simultánea
  - Monocromática o Color: Utiliza longitudes de onda dentro del infrarrojo cercano con mayor rendimiento para usos en el espector infrarrojo cercano no visible (Lente sin filtro IR)
  - Eficiencia del Obturador: $>$99 %
  - Interfaz simple doblemente cableado en serie
  - Tamaño de pantalla: Programable por el usuario a cualquier formato más pequeño (QVGA, CIF, QCIF). La Resolución de imágenes puede mantenerse independiente al tamaño de la pantalla.
  - Binning: 2 x 2 y 4 x 4 de la resolución completa
  - Controles Automáticos: Control de Exposición y de ganancia automáticos (AEC y AGC)
  - Formato de datos de salida:
    - Modo sencillo del sensor:
      - 10-bit en paralelo/autónomo
      - 8-bit ó 10-bit en serie LVDS
    - Stereo sensor mode:
      - 8-bit en serie LVDS Intercalados

– Point Grey
- FireFly MV
  - Características
  - Interfaz: USB 2.0 (libdc1394/OpenCV o Point Grey API en Linux)
  - Sensor: CMOS (Aptina MT9V022)
  - Resolución: 752 x 480
  - Frecuencia: Hasta 90 fotogramas por segundo (FPS) (El driver de Point Grey para OpenCV soporta 60 Hz)
  - Obturador Global
  - Alimentación: 5.0 V, 300-350 mA (1500 - 1750 mW)
  - Peso: 18g (con la lente)
  - Detalles de la interfaz del software
    - OpenCV para interactuar con esta cámara.
    - OpenCV en Gumstix Overo / BeagleBoard para FireFly MV:

–Driver Linux : libdc1394 sobre libusb-1.0

- Chamaleon
  - Sensor: Sony 1.3MP 1/3" ICX445 EXview HAD CCD
  - Resolución and Frecuencia: 1296x964 a 18 FPS
  - Interfaz Digital: USB 2.0 5-pin Mini-B interface
  - Dimensiones: 25.5 mm x 41 mm x 44 mm
  - Entrada/Salida: Conector de 7 pines para trigger y strob
  - Software: El programa de Point Grey FlyCapture® SDK está diseñado para el soporte de adquisición de imágenes y el control de cámara para todos los productos USB y FireWire de Point Grey.
  - FlyCapture es compatible con los sistemas operativos Microsoft® Windows® 2000/XP/Vista, e incluye el driver del dispositivo PGRCAMTM, el software de las librerías de API completo, programas de demostración y ejemplos de programas con el código fuente escrito en C/C++.
  - Tiene su propio software (Point Grey's FlyCapture® SDK) y funciona con los otros sistemas operativos como Mac y Linux. Además, funciona con el resto del software disponible.
- Flea
  - Flea 2
    - Sensor: Sony 1/3", 1/2", 1/1.8", 2/3" CCDs, Color/BW
    - Resolución: 0.3, 0.8, 1.3, 1.4, 2.0, 5.0 Megapíxeles
    - Frecuencia: Desde 7.5 FPS (FL2G-50S5) hasta 80FPS (FL2-03S2)
    - Dimensiones: 29 x 29 x 30 mm
    - Interfaz: 9-pin IEEE-1394b 800Mb/s interfaz
  - Flea 3
    - Sensor: Sony 1/2", 1/4", 1/1.8", CCDs, Color/BW
    - Resolución: 0.3, 1.4, 2.0 Megapíxeles
    - Frecuencia: Desde 15 FPS (FL3-20S4) hasta 120 FPS (FL3-03S1)
    - Dimensiones: 29 x 29 x 30 mm
    - Interfaz: Cabezal de 9 pin con protocolo IEEE-1394b hasta 800Mb/s

– Sentech
- Cámaras USB 2.0
- Características
  - Resolución VGA ~ UXGA a color o blanco y negro
  - Integración extendida de baja iluminación (con reducción de ruido compleja)
  - Exploración parcial variable y control de velocidad del reloj
  - Dibujos / anotaciones & volteado horizontal y vertical
  - Conectores USB estándar o con seguro
  - Controladores DirectX, Twain, Linux, Halcon y NI LabVIEW
- Modelos de AH y ASH
  - Carcasa de precisión con diseño de disipador de calor e indicador LED
  - Características de visión de máquinas
    - Disparador (hardware o software)
    - Señal de terminación integrada, control de lectura de salida
    - 4 E/S configurables
    - Serie USB 2.0
    - Alto rendimiento, Versátil
  - Éstas son cámaras USB compactas, de exploración progresiva con CCD a color o blanco y negro que incluyen disparador de software, captura de imágenes, zoom digital y un menú de configuración y control de usuario completos. Esta serie está disponible con un conector USB superior o posterior. Las cámaras USB de Sentech incluyen un SDK, DirectX, controladores Twain y Linux así como el software de visualización Sentech.

– IDS
  •Modelos
  – USB 2 uEye LE
    – USB 2.0
    – Sensores rápidos CMOS
    – C/CS-Mount, distancia focal ajustable
    – S-Mount (12/14 mm) con lente de filtro
    – Disparador, strobe y 2 GPIO (sólo en los modelos board-level)
  – USB 2 uEye SE
    – USB 2.0
    – Sensores CMOS y CCD Sony
    – E/S digitales ópticamente desacopladas
    – Conectores C-Mount atornillables
    – Versiones OEM disponibles
  – USB 2 uEye RE
    – USB 2.0
    – Sensores CMOS y CCD Sony
    – E/SO digitales ópticamente desacopladas
    – IP 65 / IP 67 Carcasa, tubo de protección opcional
    – Conectores atornillables
    – C-Mount
  – USB 2 uEye ME
    – USB 2.0 Sensores CMOS y CCD Sony
    – E/S digitales ópticamente desacopladas
    – Conectores atornillables
    – C-Mount
    – Versiones OEM disponibles

– Otras Cámaras
  • Webcams USB
  – Las Webcams USB 2.0 ofrecen una resolución relativamente buena, un bajo coste y un peso relativamente pequeño a costa de la compatibilidad de la cámara con el procesador, la dificultad de encontrar los drivers para usarse en sistemas operativos libres y de la fragilidad del conector USB.

La elección de la cámara resultó más sencilla que la del procesador. Se seleccionó la cámara Gumstix Overo Caspa FS, debido a que es la que mejores prestaciones presentaba, además de que permite la grabación en espectro infrarrojo, que asegura una aplicación muy amplia para futuros trabajos.

5.2. Software

El software de un sistema informático lo forman todos los componentes intangibles, lógicos, que hacen posible la realización de tareas específicas. Aquí se especificarán los lenguajes de programación y los programas utilizados para la realización de los programas de reconocimiento de imágenes y la interfaz gráfica de la estación de tierra y el algoritmo que se siguió en el procesado de imágenes

5.2.1. Programas y Lenguajes de programación

5.2.1.1. Lenguaje C

El lenguaje de programación C fue creado por Dennis M. Ritchie en los Laboratorios Bell en 1972, como una evolución del lenguaje B, basado en el Basic Combined Programming Languaje (BPCL) de Martin Richards.

El lenguaje C es un lenguaje orientado a la implementación en Sistemas Operativos, especialmente los sistemas UNIX, y se le tiene mucho aprecio debido a la alta eficiencia del código que produce y por ello, es el lenguaje más popular para crear software de sistemas y aplicaciones.

El lenguaje C es un lenguaje fuertemente tipificado de medio nivel pero con muchas características de bajo nivel y dispone de estructuras de los lenguajes de alto nivel, controladas a muy bajo nivel mediante construcciones del lenguaje. Los compiladores ofrecen extensiones al lenguaje que posibilitan la mezcla de código en ensamblador con código C o posibilitan el acceso directo a la memoria o a dispositivos periféricos.

C presenta una serie de ventajas y desventajas frente a sus competidores que relataremos a continuación. C es un lenguaje muy eficiente ya que permite utilizar las características de bajo nivel para realizar aplicaciones óptimas, además es el lenguaje con la mayor portabilidad que existe en la informática, existiendo compiladores para todos los sistemas operativos conocidos, y proporciona facilidades para realizar los programas en forma de módulos, y para utilizar código o bibliotecas existentes como puede ser la de OpenCV para el procesado de imágenes o la de Qt para la interfaz gráfica de los programas. La principal desventaja de C es la velocidad de desarrollo, para el principiante es muy complejo comenzar con la programación, pues C asigna y libera espacio de memoria de forma explícita, lo que complica la labor de aprendizaje. Y además, tiene un mantenimiento costoso, ya que aunque se presente en órdenes cortas, son muy enrevesadas, y si no se siguen ciertas normas, el código se hace muy complejo de leer.

La compilación de un programa en lenguaje C se realiza en tres fases automatizadas que son el preprocesado, la compilación y el enlazado. El preprocesado modifica el código fuente C por unas instrucciones para facilitar la tarea al compilador. La compilación o el compilado, genera el código objeto a partir del código ya procesado. Esto genera un archivo que puede ser ejecutado mediante compiladores de otros lenguajes de programación como Fortran. El enlazado une los objetos con las bibliotecas externas y los distintos módulos para generar el programa ejecutable final.

C es el lenguaje común para programar sistemas embebidos El código ligero que un compilador C genera, combinado con la capacidad de acceso a capas del software cercanas al hardware son la causa de su popularidad en estas aplicaciones.

Una característica donde C demuestra comodidad de uso particularmente valiosa en sistemas embebidos es la manipulación de bits. Los sistemas contienen registros mapeados en memoria (en inglés, MMR) a través de los cuales los periféricos se configuran. Estos registros mezclan varias configuraciones en la misma dirección de memoria, aunque en bits distintos. Con C es posible modificar fácilmente uno de estos bits sin alterar el resto. Este tipo de manipulación es muy tediosa o sencillamente imposible en otros lenguajes de alto nivel, dado que se utilizan operaciones comunes del lenguaje ensamblador como las operaciones a nivel bit OR, AND, SHL y CPL pero que están disponibles en el lenguaje C. Otras características de C consideradas desventajas en la programación para PC -como la ausencia de control de memoria automático- se convierten en ventajas cuando los sistemas embebidos necesitan código pequeño y optimizado. Ese es el caso de los sistemas basados en microcontroladores de poca potencia como el Intel 8051 o muchos sistemas ARM, como el Gumstix Overo IronSTORM que se usará en este proyecto. Ese es uno de los motivos fundamentales en la elección de C como lenguaje de programación para nuestra aplicación en vuelo.

Debido a su gran portabilidad, C es el inicio a muchas ramas de evolución a otros lenguajes, entre los que destaca sobre todo C++, creado por Bjarne Stroustrup, intentando proporcionar orientación a objetos y es la variante más difundida, pues combina la flexibilidad de C y la programación

orientada a objetos como abstracción y ocultación. Otros lenguajes inspirados en C, son Java, C# y Objective-C.

Para finalizar con la descripción del lenguaje C, se hablara de las bibliotecas. Una biblioteca de C es una colección de funciones utilizadas en el lenguaje de programación C. Las bibliotecas más comunes son la biblioteca estándar de C y la biblioteca del estándar ANSI C, la cual provee las especificaciones de los estándares que son ampliamente compartidas entre bibliotecas. La biblioteca ANSI C estándar, incluye funciones para la entrada y salida de archivos, alojamiento de memoria y operaciones con datos comunes: funciones matemáticas, funciones de manejo de cadenas de texto y funciones de hora y fecha. Otras bibliotecas C son aquellas utilizadas para desarrollar sistemas Unix, las cuales proveen interfaces hacia el núcleo. Estas funciones son detalladas en varios estándares tales como POSIX y el Single UNIX Specification.

Ya que muchos programas han sido escritos en el lenguaje C existe una gran variedad de bibliotecas disponibles. Muchas bibliotecas son escritas en C debido a que C genera código objeto rápido; los programadores luego generan interfaces a la biblioteca para que las rutinas puedan ser utilizadas desde lenguajes de mayor nivel, tales como Java, Perl y Python.

Finalmente como apéndice al apartado, se muestra una cronología de los principales lenguajes de programación en el apéndice B.

5.2.1.2. Code::Blocks

Code::Blocks es un entorno de desarrollo integrado (Integrated Development Environment (IDE)) libre y multiplataforma para el desarrollo de programas en lenguaje C y C++. Code::Blocks está basado en la plataforma de interfaces gráfica WxWidgets y eso hace que pueda usarse libremente en varios sistemas operativos. Esta IDE está bajo licencia pública general GNU.

Code::Blocks surgió de la necesidad de una IDE creada en los lenguajes C y C++ para esos lenguajes, ya que existía Dev-C++, pero Dev-C++ está creada en Delphi, lo que dificultaba la interacción. Code:Blocks está construido como un núcleo expansible mediante complementos, y su funcionalidad viene provista de los complemente predeterminados. Es una plataforma muy dinámica y potente pues puede incluirse nuevas funcionalidades de manera sencilla y además permite usarla para construir otras herramientas de desarrollo solamente añadiendo más complementos.

Code::Blocks se enlaza a una gran variedad de compiladores, ya que sólo es la interfaz del entorno de desarrollo. A los compiladores en los que se enlaza, este programa los configura para que se pueda trabajar directamente desde el entorno gráfico. Los compiladores compatibles más comunes son el de Microsoft Visual Studio Toolkit, el GCC, en sus versiones de Windows (MINGW o Cygwin) o GNU/Linux y Borland C++ Compiler.

Las características del entorno de Code::Blocks son las consideradas clásicas, y sirven de apoyo a la programación, entre ellas destacan el uso de espacios de trabajo para combinar proyectos, el uso de espacios de trabajos adaptables a la tarea que se realiza o según se cambie la configuración, la navegación de proyectos, el editor tabulados, el intercambio rápido entre archivos fuente y archivos de cabecera y la lista de tareas. Además, en la edición, Code:Blocks permite el coloreo de la sintaxis, para mejorar la comprensión por parte del programador, la tabulación inteligente del código, el autocódigo y el autocompletado de código, mediante listas desplegables, vistas de argumentos de las funciones y la selección múltiple de las funciones sobrecargadas, como funciones más destacadas, que ayudan y mucho a la hora de programar.

En la compilación, permite realizar la construcción de archivos sin la necesidad de makefiles, lo que ayuda a que se compile de forma rápida y soporta compilación en paralelo para varios núcleos. Esta cualidad fue la principal que le hizo ser el elegido para el desarrollo de la aplicación. La otra cualidad que fomenntó su elección fue que en sistemas operativos Windows, Code::Blocks permite un aprendizaje de los lenguajes de programación de manera muy sencilla y muy cómoda; lo que provocó que se pudiera desarrollar la aplicación de visión por computador de manera satisfactoria.

Code::Blocks trae integradas plantillas para generar varias clases de programas, ya sea la clásica aplicación de consola, bibliotecas estáticas o dinámicas, o proyectos completos enlazados con populares bibliotecas como OpenGL y SDL; sin embargo, Code::Blocks integra sólo las plantillas, las bibliotecas deben instalarse por separado.

5.2.1.3. OpenCV

Como ya se indicó anteriormente, OpenCV es una biblioteca libre de visión artificial. Fue desarrollada por Intel, en una primera versión que apareció en el mes de enero de 1999 y que se ha utilizado en infinidad de aplicaciones como pueden ser los sistemas de seguridad con detección de movimiento o aplicaciones de control de procesos donde se requiere reconocimiento de objetos. La gran aceptación de estas librerías se deben a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es una librería multiplataforma, existiendo versiones para los principales sistemas operativos. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo sencillo, eficaz, robusto y eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo y las propiedades que ofrecen esos lenguajes de programación para sistemas embarcados. OpenCV puede además utilizar un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

OpenCV se implementa como librería para códigos de programación en C, C++, Python, Java de manera muy sencilla y como se indicó anteriormente se dispone de multitud de material de información tanto en Internet como en manuales, lo que ha hecho que sea la más utilizada para sistemas de visión por computador. OpenCV permite un diseño muy sencillo de órdenes de procesado de imágenes, que serían operaciones bastante complejas si no estuviera instalada esta librería, como los detectores de borde de Sobel o de Canny, o la segmentación de imágenes, ya que habría que utilizar todo el potencial del cálculo matricial píxel a píxel de manera manual, lo cual es una gran fuente de posibles fallos de programación.

5.2.1.4. Qt

Qt es una biblioteca multiplataforma usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como pueden ser herramientas para la línea de comandos y consolas para servidores.

Qt está desarrollada como un software libre y de código abierto dentro del Qt Project, donde participan desarrolladores como Nokia y Digia y la comunidad de desarrolladores libres. Anteriormente al Qt Project, Qt era desarrollado por la división de software de Nokia, cuando ésta adquirió Trolltech, que era el primer desarrollador de Qt, en junio de 2008. Qt se distribuye bajo los términos de la

Licencia Pública General Reducida de GNU, y también se distribuyen licencias comerciales de Qt a través de Digia desde marzo de 2011.

Qt utiliza el entorno de escritorio de sistema GNU/Linux KDE y el lenguaje C++ de forma nativa, pudiendo agregar otros lenguajes a través de bindings, que son adaptaciones de bibliotecas escritas en un lenguaje de programación distinto al que se va a usar. Qt funciona en las principales plataformas y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos de acceso a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para el manejo de archivos además de las estructuras tradicionales de datos.

Qt cuenta actualmente con un sistema de triple licencia: GPL v2/v3 para el desarrollo de software de código abierto y software libre, la licencia de pago QPL para el desarrollo de aplicaciones comerciales, y a partir de la versión 4.5 una licencia gratuita pensada para aplicaciones comerciales, LGPL.

Qt es utilizada principalmente para aplicaciones embebidas en electrodomésticos, o embarcadas en aeronáutica y automoción, así como para aplicaciones para móviles. Los principales usuarios de Qt son Autodesk Maya, Dassault DraftSight, Google, KDE, Adobe, ESA, Siemens, Volvo, Walt Disney Animation Studios, Skype, Samsung, Panasonic y Mathematica.

Las principales aplicaciones creadas con Qt son Adobe PhotoShop Album, Google Earth, KDE, Last.fm Player, LyX, Launchy, Mathematica, Quantum GIS, Skype, MythTV, VLC Media Player VirtualBox, y QT Creator.

5.2.2. Algoritmos

El algoritmo que sigue el procesado de imágenes se basa en el barrido por píxeles de la imagen. La imagen, una vez capturada, se carga en la memoria del ordenador en blanco y negro, como si fuera una matriz de puntos. La inmensa mayoría de lenguajes informáticos de programación trabajan con las imágenes de ese modo. Así, una vez cargado, y con el fin de que el programa sea mucho más eficiente, se le pide que capture algunos fotogramas más, pero que no los guarde, para que solo analice uno de cada cinco fotogramas.

Una vez guardado y segmentado a cero, se realiza un barrido en vertical, con un paso determinado, de píxeles, buscando el nivel de intensidad de gris entre cada píxel. Y se compara ese nivel de intensidad con el del anterior. Ahí se toma la primera decisión. Si la diferencia de nivel es mayor a un umbral determinado, se captura dicho punto. Si no lo es, se desecha. El barrido vertical asegura que captura todos los puntos del horizonte, ya que en el horizonte siempre hay un cambio brusco de intensidad de gris.

Una vez capturados los puntos, se le pide que busque el horizonte, que deberá encontrarse alrededor del tercio central de la imagen como se estudia en fotografía, pues si se encuentra a un tercio del borde superior, se le da énfasis al plano corto mientras que si se coloca entorno al tercio inferior de la imagen, se enfatiza el fondo. Una vez buscado los puntos en ese tercio central, se le pide al algoritmo que, por teoría de mínimos cuadrados, con todos los puntos que ha guardado, genere una recta, que será la línea del horizonte.

Este algoritmo permite conocer el horizonte. Mediante una operación matemática sencilla, permite conocer el balanceo de la imagen y por tanto, del propio avión, si la cámara se coloca solidaria a éste. Ese balance, será la entrada para el programa de detector de bordes de pista, pues se necesitará para cuadrar la imagen. El algoritmo del detector de bordes de pista se realiza de modo similar al

anterior, cambiando el barrido, que pasa a ser en horizontal y las condiciones lógicas para guardar los puntos, que pasan a ser condiciones de entornos.

En este detector de bordes, se realiza otra vez la comparativa de nivel de intensidad de gris entre píxeles y se guardan los puntos detectados. Una vez realizada esa operación, se dan las condiciones lógicas de para la discretización de ambos bordes y se guardan los puntos que el detector decida que son útiles en dos grupos de puntos, unos son los del borde izquierdo y los otros los del borde derecho. Para ello, se le pide al algoritmo que los puntos del borde izquierdo aumentan su coordenada vertical y disminuyen su horizontal, mientras que los del izquierdo aumentan sus dos coordenadas. Una vez conocidos los dos bordes de pista, se realiza la comparación de ángulos entre ellos y pasa el dato al autopiloto, que será el responsable de que la aeronave se desplace hacia donde debe hacerlo.

El árbol de decisiones ilustra el proceso.

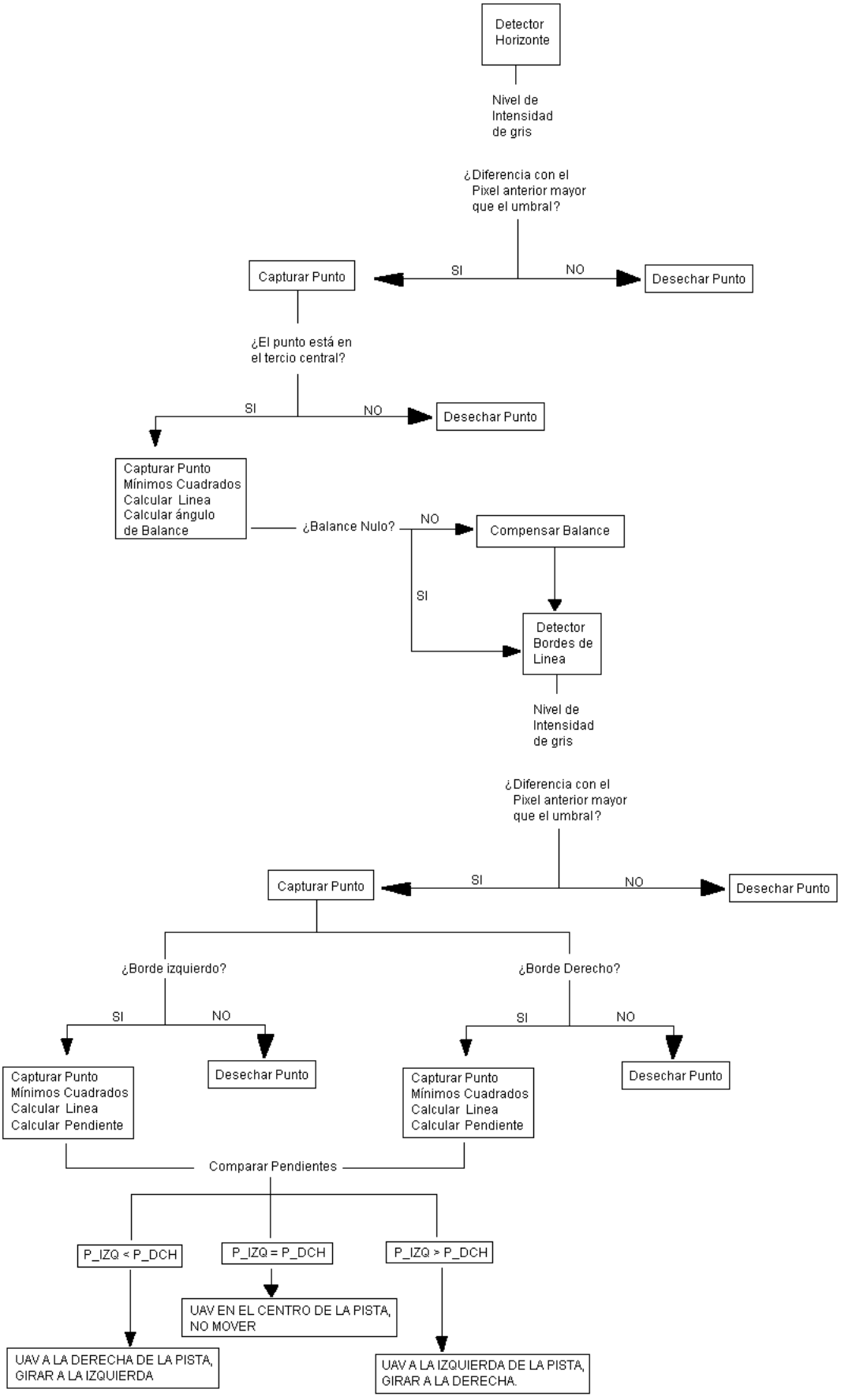


Figura 5.1: Árbol de Decisiones del Algoritmo de búsqueda

Capítulo 6

Planteamiento de una solución

La visión por computador, como se ha ido explicando en los distintos capítulos de este proyecto, es un campo con un potencial de aplicación muy grande, y que permite realizar muchísimas tareas, como la segmentación, el procesado de imágenes y que luego se aplicará a un sistema industrial para llevar a cabo su cometido, como puede ser la selección de elementos de fabricación defectuosos en factorías o la navegación visual en una aeronave.

Como también se ha especificado anteriormente, un programa en lenguaje C o C++, apoyado en la librería de visión por computador de Intel OpenCV es la herramienta con mayor potencial y mayor referencias que existe actualmente en el mundo de la investigación de UAS, por lo cual, la solución a plantear irá inexorablemente en ese dirección.

En este proyecto, la solución que se plantea es el desarrollo de una aplicación que detecte bordes de pista y horizonte por el que se ayude a la navegación de una aeronave y proponer una posterior integración en un avión, de un procesador para el trabajo con las imágenes, que será el Gumstix OVERO IronSTORM, que se emplaza en una placa base Gumstix OVERO Tobi, de una cámara para la captura de las imágenes, y la integración con el autopiloto Paparazzi, que es el que lleva las órdenes de movimiento al avión.

Para poder plantear la solución, lo que se ha buscado es realizar un programa en C para que el procesador pueda realizar la tarea de búsqueda de bordes que el algoritmo que en el capítulo anterior expuso. Y una vez realizada la tarea, transferir los datos al autopiloto, para que este, a través de los datos que ha recogido del procesador de imágenes, de la orden de moverse al avión.

Además, se ha realizado el programa de tal forma que, además de dar los datos al autopiloto, el procesador de imágenes, guarda las pendientes de las dos rectas que detecta como bordes de pista en un archivo, lo que permite que ese archivo sea enviado a tierra y sea la entrada al programa de la interfaz gráfica de la estación de seguimiento, pues leyendo el contenido de ese archivo, se puede identificar la maniobra que está realizando el autopiloto.

El programa que se planteó como solución realiza primero una detección de horizonte, con el fin de determinar el balance de la aeronave si la cámara está solidariamente colocada en ésta, para después, plantear la detección de bordes. Para ello, se llevará a cabo, primero, un paso de la imagen de color a blanco y negro, con el fin de simplificar el trabajo posterior del procesador, luego una segmentación de las imágenes, y posteriormente, al aplicar el algoritmo de detección con un nivel de umbral definido, seleccionar los puntos que son del borde (o del horizonte, dependiendo del caso)

Capítulo 7

Desarrollo de la solución

La solución se desarrolla en el entorno de Code::Blocks, para la edición del código de lenguaje C, que permite tener con bastante claridad, control sobre nuestro programa.

Como ya se indico anteriormente, el programa que se planteó como solución realiza primero una detección de horizonte, con el fin de determinar el balance de la aeronave si la cámara está solidariamente colocada en ésta, para después, plantear la detección de bordes. Para ello, se llevará a cabo, primero, un paso de la imagen de color a blanco y negro, con el fin de simplificar el trabajo posterior del procesador, luego una segmentación de las imágenes, y posteriormente, al aplicar el algoritmo de detección con un nivel de umbral definido, seleccionar los puntos que son del borde (o del horizonte, dependiendo del caso)

7.1. Desarrollo Teórico

La solución que se propone requiere del estudio de técnicas de segmentación y procesado de imágenes, así como de cálculo matricial y de cálculo geométrico para finalizar con el cálculo de la posición respecto a la pista y su colocación a través de las órdenes del autopiloto.

Dentro de este desarrollo teórico, se van a explicar las teorías de segmentación de Otsu, con un esquema de las opciones de segmentación que OpenCV tiene programadas por defecto; las teorías de la transformación de Hough, de los detectores de contorno de Sobel y Canny y el método de mínimos cuadrados, que es el que OpenCV utiliza para calcular y dibujar la líneas una vez se le da una nube de puntos.

Se hablarán de las transformadas de Hough y de los detectores de Sobel y Canny, aunque no se utilizaron dentro del programa debido a que son las teorías desde las que se deben partir para mejorar este programa, por lo tanto, desde donde partir para los trabajos futuros; ya que únicamente no se utilizaron debido a la falta de experiencia con el lenguaje de programación a la hora de trabajar con imágenes, ya que, por ejemplo, el detector de Sobel que OpenCV trae instalado cambia la profundidad de campo de la imagen de 8 bits a 32 bits, mientras que otras muchas funciones necesarias para la detección de la línea obliga a trabajar en 8 bits; o el detector de Canny de OpenCV sólo funciona con imágenes en blanco y negro de 8 bits, aunque después se puede presentar en pantalla la imagen con los bordes en color. Pero el hecho de no haberse utilizado no quita para que lo presentado en estas teorías ayude a la misión final de nuestro programa, que es la detección de la pista y la colocación frente a ella como base a la ayuda a la navegación.

Una vez finalizadas las teorías, se especificará la teoría del método que sigue el programa para llevar a cabo su tarea

7.1.1. Método de Otsu de segmentación por valor umbral

La primera de las teorías a explicar es el método de Otsu para la segmentación de una imagen. Este método lo inventó Nobukuyi Otsu en 1979, y se basa en utilizar técnicas estadísticas, para resolver el problema. En concreto, se utiliza la variancia, que es una medida de la dispersión de valores – en este caso se trata de la dispersión de los niveles de gris.

El método de Otsu calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para ello se calcula el cociente entre ambas variancias y se busca un valor umbral para el que este cociente sea máximo.

Como punto de partida para la explicación matemática del método de Otsu tomamos dos segmentos de puntos ($K_0(t)$ y $K_1(t)$), que serán definidos a partir del valor umbral t. t es la variable que buscamos, y los dos segmentos son el resultado deseado en la segmentación.

Sea p(g) la probabilidad de ocurrencia del valor de gris $0 < g < G$ (G es el valor de gris máximo). Entonces la probabilidad de ocurrencia de los píxeles en los dos segmentos es:

Para K_0:

$$P_0(t) = \sum_{g=0}^{t} p(g)$$

Y para K_1

$$P_1(t) = \sum_{g=t+1}^{G} p(g) = 1 - P_0(t)$$

Si tomamos dos segmentos (o sea un solo valor umbral) la suma de estas dos probabilidades dará evidentemente 1.

Si g es la media aritmética de los valores de gris en toda la imagen, y g_0 y g_1 los valores medios dentro de cada segmento, entonces se pueden calcular las variancias dentro de cada segmento como:

$$\sigma_0^2(t) = \sum_{g=0}^{t} (g - \overline{g_0})^2 p(g) \qquad \sigma_1^2(t) = \sum_{g=t+1}^{G} (g - \overline{g_1})^2 p(g)$$

La meta es mantener la variancia dentro de cada segmento lo más pequeña posible y conseguir que la variancia entre los dos segmentos sea lo más grande posible.

Así obtenemos:

$$Q(t) = \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)}$$

La variancia entre los segmentos es:

$$\sigma_{zw}^2(t) = P_0(t) \cdot (\overline{g_0} - \overline{g})^2 + P_1(t) \cdot (\overline{g_1} - \overline{g})^2$$

La variancia dentro de los segmentos se obtiene de la suma de ambas:

$$\sigma_{in}^2(t) = P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t)$$

El valor umbral t se elige de manera que el cociente Q(t) sea máximo. Q(t) es por lo tanto la medida buscada. De esta forma elegimos un valor umbral que optimiza los dos segmentos en términos de variancia.

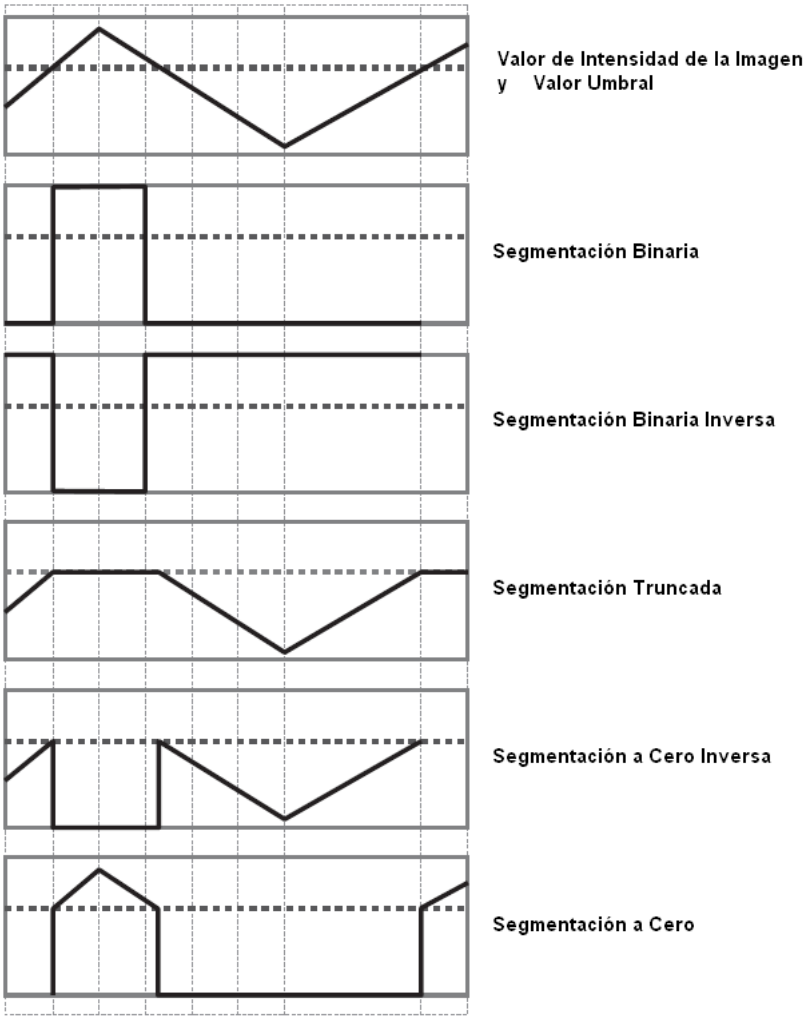


Figura 7.1: Tipos de Segmentación de OpenCV

OpenCV utiliza este método para realizar su segmentación por el valor umbral, aunque también utiliza otro método más sencillo, en el que se da el valor umbral y el valor máximo de intensidad, y según sea la opción, segmenta la imagen de una manera u otra.

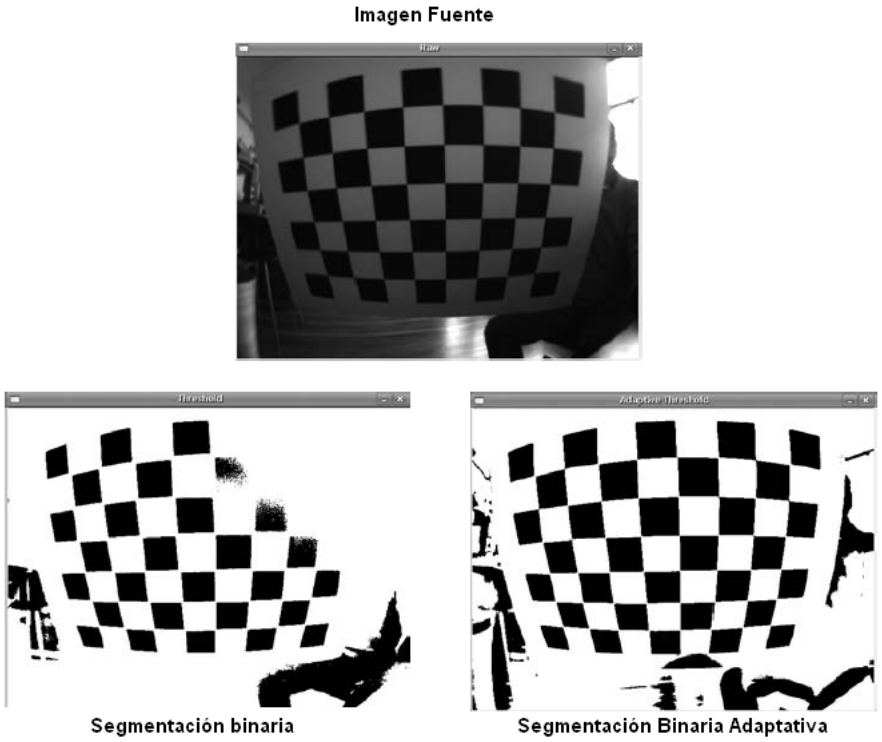


Figura 7.2: Segmentación Adaptativa de OpenCV

Además, OpenCV tiene la opción de la segmentación por valor umbral de forma adaptativa, en la cual, la segmentación no es la misma en toda la imagen, sino que se adapta a los niveles de intensidad de cada zona de la imagen, mejorando las segmentaciones en zonas de sombra.

7.1.2. Transformada de Hough

La siguiente teoría a explicar es la transformada de Hough. Esta transformada, como se indicó anteriormente, es un algoritmo creado en 1962, que permite encontrar ciertas formas como rectas o círculos dentro de una imagen. La versión más simple es la que encuentra las líneas como características dentro de una imagen. Este algoritmo aprovecha la expresión del haz de rectas que pasan por un punto genérico (x_i, y_i), cuya ecuación es: $y_i = ax_i + b$; siendo a y b parámetros variables. Si además se obliga a pasar por otro punto (x_j, y_j), los parámetros a y b se conocen al resolver el sistema, llamándolos a' y b'.

Su modo de operación es principalmente estadístico y consiste en que para cada punto que se desea averiguar si es parte de una línea se aplica una operación dentro de cierto rango, con lo que se averiguan las posibles líneas de las que puede ser parte el punto. Esto se continúa para todos los puntos en la imagen, al final se determina qué líneas fueron las que más puntos posibles tuvieron y esas son las líneas en la imagen.

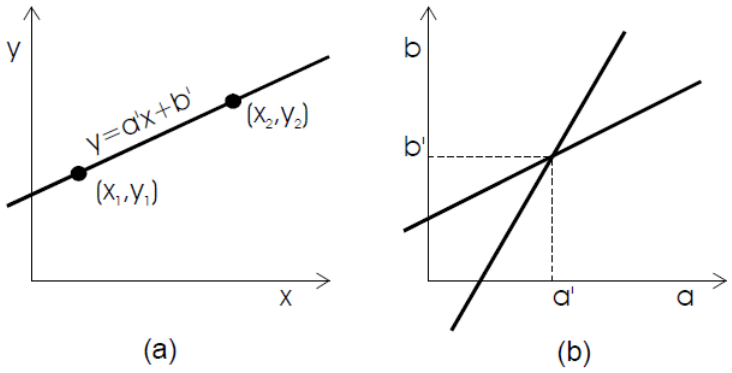


Figura 7.3: Transformada de Hough

Si se representa el espacio de los parámetros (a,b), el número de veces que una recta pasa por el punto (a',b') determina el número de píxeles que comparten la misma recta. Por ello, el simple conteo de las veces que se repiten los mismos valores para a y b sirve de indicador para encontrar las rectas que existen en una imagen. Más concretamente, si se representa el espacio de parámetros (a,b) de las rectas que pasan por los puntos(x_1, y_1) y (x_2, y_2) se obtiene la figura que se expone a continuación. La intersección de las rectas de esta figura determina el punto (a',b') que se corresponde con los parámetros de la recta $y = a'x + b'$, que contiene los puntos(x_1, y_1) y (x_2, y_2). De manera práctica, suele discretizarse el plano de los parámetros (a,b), para realizar el conteo. Además, la representación $y = ax + b$ plantea el problema de que ni los valores de a ni los de b están acotados, complicándose aún más en el caso de las rectas verticales en los que el parámetro a tiende a infinito. Esto nos lleva al uso de coordenadas polares que después se trata.

Su modo de operación es principalmente estadístico y consiste en que para cada punto que se desea averiguar si es parte de una línea se aplica una operación dentro de cierto rango, con lo que se averiguan las posibles líneas de las que puede ser parte el punto. Esto se continúa para todos los puntos en la imagen, al final se determina qué líneas fueron las que más puntos posibles tuvieron y esas son las líneas en la imagen.

La transformada de Hough emplea una representación paramétrica de formas geométricas. Una recta, por ejemplo se representa por un módulo ϕ (phi) (perpendicular a la recta y que pasa por el origen (0,0) y un ángulo ρ (rho) (formado por el módulo y el eje positivo de las x's). Se representa así:

$$x\,cos(\rho) + y\,sen(\rho) = \phi$$

La ventaja de este método es que evita singularidades, como por ejemplo rectas de pendiente infinita. Si se representa ρ y ϕ en un plano cartesiano, una recta queda determinada mediante un punto con coordenadas (ϕ(recta), ρ(recta)), mientras que un punto, se representa como una función senoidal. Si por ejemplo tenemos dos puntos, tendremos dos sinusoidales desfasadas α grados dependiendo de las coordenadas de los puntos. Dichas sinusoides se irán cruzando cada 180^{o}. La interpretación geométrica de este hecho, es que la función seno de cada punto, representa las infinitas rectas que pasan por cada punto, cuando dos puntos comparten la misma recta, sus representaciones senoidales se cruzan, se obtiene un punto. Cada vez que se da media vuelta (ρ=180^{o}) se vuelve a repetir la misma recta, por lo que volvemos a obtener otro punto, que de hecho es la misma recta.

7.1.3. Detector de bordes de Sobel

El detector de bordes de Sobel es un procedimiento utilizado en el análisis de imágenes. Es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad

de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de éste vector.

El operador Sobel calcula el gradiente de la intensidad de una imagen en cada punto. Así, para cada punto, este operador da la magnitud del mayor cambio posible, la dirección de éste y el sentido desde oscuro a claro. El resultado muestra cómo cambia una imagen en cada punto analizado y, en consecuencia, la probabilidad de que éste represente un borde en la imagen y, también, la orientación a la que tiende ese borde. En la práctica, el cálculo de la magnitud -probabilidad de un borde- es más fiable y sencillo de interpretar que el cálculo de la dirección y sentido. Matemáticamente, el gradiente de una función de dos variables (en este caso, la función de intensidad de la imagen) para cada punto es un vector bidimensional cuyos componentes están dados por las primeras derivadas de las direcciones verticales y horizontales. Para cada punto de la imagen, el vector gradiente apunta en dirección del incremento máximo posible de la intensidad, y la magnitud del vector gradiente corresponde a la cantidad de cambio de la intensidad en esa dirección. Lo dicho en los párrafos anteriores implica que el resultado de aplicar el operador Sobel sobre una región con intensidad de imagen constante es un vector cero, y el resultado de aplicarlo en un punto sobre un borde es un vector que cruza el borde (perpendicular) cuyo sentido es de los puntos más oscuros a los más claros.

Debido a que la función de intensidad de una imagen digital sólo se conoce mediante puntos discretos, las derivadas de estas funciones no pueden ser definidas a menos que asumamos que existe una función continua que ha sido muestreada en los puntos de la imagen. Con algunas suposiciones adicionales, la derivada de la función continua de intensidad puede ser calculada como una función de la función de intensidad muestreada, por ejemplo de la imagen digital. De lo anterior resulta que las derivadas en cualquier punto particular son funciones de los valores de intensidad, virtualmente, en todos los puntos de la imagen. Sin embargo, aproximaciones a estas funciones diferenciales pueden ser definidas con el nivel de precisión requerido teniendo en cuenta únicamente una pequeña región de puntos alrededor del estudiado. El operador Sobel representa una primera aproximación imprecisa del gradiente de la imagen, pero es de calidad suficiente para ser de uso práctico en muchas aplicaciones. Más precisamente, éste operador utiliza sólo valores de intensidad en una región de 3x3 alrededor de cada punto analizado para calcular el gradiente correspondiente, además de que utiliza sólo números enteros para los coeficientes que indican la aproximación del gradiente. Como una consecuencia de su definición, el operador Sobel puede ser implementado mediante simples definiciones tanto en hardware como en software: sólo son utilizados ocho puntos de la imagen alrededor del punto a analizar para calcular el punto correspondiente de la imagen resultante, además sólo se requiere aritmética con números enteros para calcular una aproximación del vector gradiente.

Matemáticamente, el operador utiliza dos kernels de 3×3 elementos para aplicar convolución a la imagen original para calcular aproximaciones a las derivadas, un kernel para los cambios horizontales y otro para las verticales. Si definimos como la imagen original, el resultado, que son las dos imágenes y que representan para cada punto las aproximaciones horizontal y vertical de las derivadas de intensidades, es calculado como:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \qquad G_Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

En cada punto de la imagen, los resultados de las aproximaciones de los gradientes horizontal y vertical pueden ser combinados para obtener la magnitud del gradiente, mediante:

$$G = \sqrt{G_x^2 + G_Y^2}$$

Con esta información, podemos calcular también la dirección del gradiente:

$$\Theta = arctg\left(\frac{G_Y}{G_x}\right)$$

donde, por ejemplo, Θ es 0 para bordes verticales con puntos más oscuros al lado izquierdo.

7.1.4. Detector de bordes de Canny

El algoritmo de Canny es un operador desarrollado por John F. Canny en 1986 que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes. Lo más importante es que Canny también desarrolló una teoría computacional acerca de la detección de bordes que explica por qué la técnica funciona.

El propósito de Canny era descubrir el algoritmo óptimo de detección de bordes. Para que un detector de bordes pueda ser considerado óptimo debe cumplir los siguientes puntos:

•Buena detección: El algoritmo debe marcar el mayor número real en los bordes de la imagen como sea posible.

•Buena localización: los bordes de marca deben estar lo más cerca posible del borde de la imagen real.

•Respuesta mínima: El borde de una imagen sólo debe ser marcado una vez, y siempre que sea posible, el ruido de la imagen no debe crear falsos bordes.

Para satisfacer estos requisitos Canny utiliza el cálculo de variaciones que esuna técnica que encuentra la función que optimiza un funcional indicado. La función óptima en el algoritmo de Canny es descrito por la suma de cuatro términos exponenciales, pero se puede aproximar por la primera derivada de una gaussiana.

El algoritmo de detección de bordes de Canny utiliza un filtro basado en la primera derivada de una gaussiana. Ya que es susceptible al ruido presente en datos de imagen sin procesar, la imagen original es transformada con un filtro gaussiano. El resultado es una imagen un poco borrosa respecto a la versión original. Esta nueva imagen no se ve afectada por un píxel único de ruido en un grado significativo.

Un ejemplo de un filtro gaussiano 5x5 sería el siguiente:

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A$$

El borde de una imagen puede apuntar en diferentes direcciones, por lo que el algoritmo de Canny utiliza cuatro filtros para detectar horizontal, vertical y diagonal en los bordes de la imagen borrosa. El operador de detección de bordes (Roberts, Prewitt, Sobel, por ejemplo) devuelve un valor para la primera derivada en la dirección horizontal (G_Y) y la dirección vertical (G_X). A partir de éste, se pueden determinar el gradiente de borde y la dirección:

$$G = \sqrt{G_x^2 + G_Y^2}$$

$$\Theta = arctg\left(\frac{G_Y}{G_x}\right)$$

7.1.5. Método de los mínimos cuadrados

El método de mínimos cuadrados es una técnica de análisis numérico encuadrada dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados de una variable independiente y una variable dependiente y una familia de funciones, se intenta encontrar la función, dentro de dicha familia, que mejor se aproxime a los datos, o sea, un mejor ajuste, de acuerdo con el criterio de mínimo error cuadrático. OpenCV utiliza como mínimo error cuadrático para calcular las líneas, ρ=r, que al final, no es más que una regresión lineal.

En su forma más simple, intenta minimizar la suma de cuadrados de las diferencias ordenadas también llamadas residuos, entre los puntos generados por la función y los correspondientes en los datos. Desde un punto de vista estadístico, un requisito implícito para que funcione el método de mínimos cuadrados es que los errores de cada medida estén distribuidos de forma aleatoria.

El teorema de Gauss-Márkov prueba que los estimadores mínimos cuadráticos carecen de sesgo y que el muestreo de datos no tiene que ajustarse, por ejemplo, a una distribución normal. También es importante que los datos recogidos estén bien escogidos, para que permitan visibilidad en las variables que han de ser resueltas

La técnica de mínimos cuadrados se usa comúnmente en el ajuste de curvas, aunque también se puede utilizar para otros fines, como minimizar la energía o maximizar la entropía de un sistema.

Las rectas de regresión lineal son las rectas que mejor se ajustan a la nube de puntos generada por una distribución binomial, en nuestro caso, la matriz de puntos que detecta el algoritmo.

Matemáticamente, son posibles dos rectas de máximo ajuste:

- La recta de regresión de Y sobre X; que es la que OpenCV tiene por defecto:

$$y = \overline{y} + \frac{\sigma_{XY}}{\sigma_X^2}(x - \overline{x})$$

- La recta de regresión de X sobre Y:

$$x = \overline{x} + \frac{\sigma_{XY}}{\sigma_Y^2}(y - \overline{y})$$

La correlación r de las rectas determinará la calidad del ajuste. Si r es cercano o igual a 1, el ajuste será bueno y las predicciones realizadas a partir del modelo obtenido serán muy fiables (el modelo obtenido resulta verdaderamente representativo); si r es cercano o igual a 0, se tratará de un ajuste malo en el que las predicciones que se realicen a partir del modelo obtenido no serán fiables (el modelo obtenido no resulta representativo de la realidad).

7.1.6. Desarrollo teórico del programa

Este desarrollo teórico del programa abarcará tanto las teorías que se utilizan en el algoritmo de búsqueda de bordes que anteriormente se ha especificado, como las diferentes teorías que se han ido utilizando en las pruebas hasta llegar al programa final.

Así, la primera teoría que se buscó fue aplicar combinándos un detector de bordes de Canny y la transformada de Hough, en su modo polar. Eso llevaba a que la salida del programa era adecuada, pues detectaba todos los bordes presentes en la imagen, pero tenía un problema que fue el que hizo que se descartara. Ese problema era que no se podía hacer una discretización cómoda de los bordes, pues la nomenclatura de las lineas que dibujaba el programa en su subrutina de transformada de Hough se hace de manera local, y no permitía entrar en el código para hacerle decidir con qué linea quedarse.

El siguiente paso fue activar la segmentación binaria que OpenCV ofrece antes de aplicar los diferentes filtros. No mejoraba sustancialmente la detección de bordes, pero mejoraba el algoritmo de búsqueda de bordes de Canny, lo que hizo pensar en mantener este manera de proceder. Y también se probó con las operaciones morfológicas de gradiente, top-hat, black-hat y el detector de Sobel; pero ninguna de estas operaciones llevó a un resultado satisfactorio para iniciar un método para crear un algoritmo de visión por computador a partir del cual trabajar y avanzar en esa dirección, ya que esas operaciones morfológicas implicaban cambios estructurales en las configuraciones de las imágenes que ralentizaban el proceso de cálculo, por lo que también se desechó su uso.

Al no poder usar la transformada de Hough, se pensó en un detector de color com el que aparece en el manual de Robert Laganiére "OpenCV2 Computer Vision Application Programming Cookbook". Este detector de color, lo que hace es comparar el valor de los niveles de intensidad de color entre un punto y sus vecinos, y si no hay una variación muy grande, decide que el punto es del mismo color. Una vez el programa dictamina de que color es cada punto, segmentaba de forma binaria con un color objetivo. Los puntos que son similares al color objetivo se segmentan de blanco y los que no, de negro. Esa manera de proceder, llevada a nuestro objetivo, haría que todo lo que tuviese el color del asfalto de la pista, lo segmente de una forma, y lo que no, de otra.

Como idea era muy buena, pero la implementación del propio ejemplo del manual, y por ende, después aplicarlo a nuestro caso, no era tan sencilla como se esperaba, y daba numerosos errores; lo que llevó a descartar también este modelo, aunque aportó la idea de la comparativa de niveles de intensidad entre píxeles.

El programa, siguiendo el algoritmo que anteriormente se ha avanzado y que es de creación propia, lo primero que hace es realizar una segmentación a cero, a un nivel bastante alto de intensidad, para que tome la diferencia entre el terreno adyacente a la pista y la propia pista; y además, marque también la diferencia entre cielo y tierra, incluso cuando el tiempo sea nublado. Se propuso ese

método de segmentación a cero frente a la segmentación binaria debido precisamente a que en una segmentación binaria, en los casos de tiempo nublado, el propio cielo sería un elemento que podría falsear los resultados. OpenCV permite realizar una coloración de zonas conforme el color de los puntos vecinos, y esa es otra de las operaciones que se probaron para evitar el problema del cielo, pero no es una operación ni eficaz ni robusta, pues requiere que se le dé como entrada un punto donde iniciar la búsqueda de vecinos, y para automatizar eso no se tenían los conocimientos suficientes para hacerlo de manera que no afectara al tiempo de cálculo.

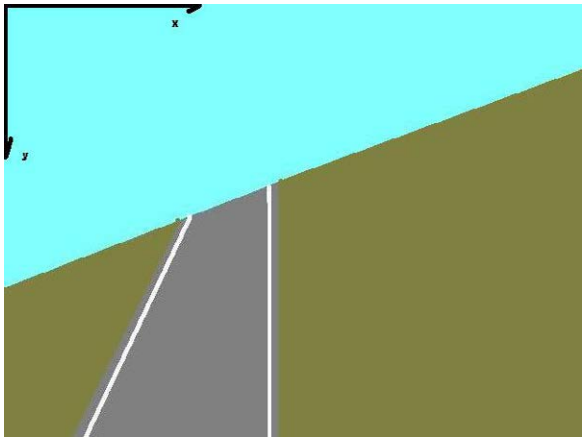


Figura 7.4: Ejes de cálculo

Como se introdujo en el apartado donde se explica el algoritmo, el programa primero realiza un barrido en vertical, siendo el origen de coordenadas la esquina superior izquierda, el eje x, el eje horizontal y el eje y, el vertical descendente, para la búsqueda del horizonte, y luego uno en horizontal para la búsqueda de bordes de pista. Esto surgió casi por casualidad, al intentar aplicar el detector de color que anteriormente se mencionó. Se probaron ambas configuraciones sobre una fotografía de una carretera, y precisamente ese fue el germen del algoritmo.

La comparación entre niveles de intensidad de gris entre píxeles se realiza mediante el valor absoluto de la resta entre el valor de un píxel y el anterior. Eso nos lleva a que se puede cuantizar esa comparación, entre 0 (dos puntos iguales) y 255 (un punto blanco y el otro negro), por lo que podemos establecer un nivel para separar los puntos en puntos válidos (que guarda el detector en su memoria) y puntos no válidos (que no guarda el detector).

Para el detector de horizonte, lo que se aplica tras la comparación de niveles de intensidad de gris entre píxeles es una teoría de fotografía que nos indica que el horizonte en una fotografía debe estar alrededor de un tercio del borde superior, si queremos enfatizar en primer plano, o alrededor de un tercio del borde inferior, si queremos dar preponderancia al fondo de la imagen. Por ello, se le pide que para discretizar, encuentre los puntos del horizonte en esa zona.

La siguiente parte que se le pide, es que una vez guarda los puntos en una matriz, mediante mínimos cuadrados, pinte la línea del horizonte; y con dos puntos $(x_1, y_!)$ y (x_2, y_2) de esa línea, por geometría básica, calcular el ángulo θ de balance del avión, pues:

$$\theta = arctg\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

Una vez tenemos el ángulo de balance, se le realiza una rotación de la imagen respecto el punto de fuga, el punto donde se van a cruzar ambos bordes de pista. Generalmente esa rotación se realiza desde el centro de la imagen. Esa rotación, OpenCV la hace como una transformación afín, manteniendo los ángulos relativos entre los dos bordes de pista, y además, permite, como antes se ha indicado, colorear los puntos que quedan vacios al realizar la rotación con el color medio de los 8 vecinos.

En este momento, se realiza el detector de bordes de pista. Para guardar los puntos de cada borde, primero se realizó un algoritmo muy basto, y genérico, por el cual, para el borde izquierdo se le pedía al programa que solamente buscase en la mitad izquierda, y para el borde derecho, que buscase en la mitad derecha. Eso no es nada robusto, ni eficaz, ni eficiente ni lógico, y además, en casos de que ambos bordes estén en el mismo lado, daría errores. Por ello, se buscó un algoritmo lógico de geometría básica por el cual, el borde izquierdo tiene pendiente positiva y el derecho pendiente negativa. Por lo tanto, al barrer en horizontal, con la misma coordenada y, si un punto tiene la coordenada x mayor que anterior y la coordenada y aumenta, es del borde derecho, y si la coordenada x es menor que el anterior, es del borde izquierdo. Además, se realiza varias relaciones lógicas para que solamente coja puntos de los dos bordes, y no de los bordes de las imágenes, ni que mezcle los bordes.

Al guardarlos en matrices separadas, se puede calcular las dos rectas de manera fácil, con el método de mínimos cuadrados, sin que un borde influya en el otro. Y una vez calculados los dos bordes, calculamos los dos ángulos de las pendientes de las rectas, con el mismo método que se utilizó para calcular el ángulo de balance.

El ángulo de la pendiente negativa se cambia de signo y se comparan como se habló en el apartado del algoritmo, como Zhognke Shi introdujó en su artículo "Vision-based Runway Recognition for UAV Autonomous Landing". La comparación se hace de manera que se simule una navegación por visión humana, en la que los ángulos se suponen iguales, y por lo tanto, no se da orden al autopiloto de corregir la trayectoria, si al compararlos no hay una diferencia mayor de 5^{o} entre ellos.

Una vez todo resuelto y probado, se buscó como última opción, para mejorar el tiempo de cálculo, que no en todos los fotogramas calculase los bordes, sino que lo hiciese cada cierto tiempo. Se probó primero cada 300 milisegundos. El resultado de este paso fue satisfactorio en parte, pues mejoraba el tiempo de cálculo, pero la salida no era del todo lo correcta que debería, pues en el tiempo en el que capturaba la imagen, la salida pasabaa de color a imagen segmentada, lo cual era bastante molesto. Por eso, este método se descartó y loque se hizo fue hacer que la librería de OpenCV cargase varios fotogramas seguidos, para que no calculase nada en ellos, y darle una espera de entrada de teclado sin que trabajase.

7.2. Implementación del Software

La implementación del software se inició con numerosos problemas de adaptación a la nueva arquitectura ARM que nuestro Gumstix Overo IronSTORM tiene implementada, ya que cambia bastante de la arquitectura de PC a la que se está acostumbrado. Eso fue un reto bastante ambicioso y que hizo que se tuviesen que afrontar situaciones complejas a nivel informático, que se resolvieron con mucho esfuerzo.

La alimentación de la placa base fue el primer problema que se tuvo que afrontar. Se desconocí el hecho de que Gumstix oferta de forma separada la alimentación de sus productos, y eso hizo que al llegar la compra de EE.UU., no se tuviera forma de alimentar de potencia a la placa. El conector que usa la placa Gumstix es un conector estándar de 2 mm, muy común en EE.UU., pero no lo es

Figura 7.5: Gumstix Overo alimentada por pilas recargables

tanto en Europa, donde está en desuso. Por ello, este conector es bastante complicado encontrarlo en las tiendas de componentes electrónicos españolas. Nokia lo utilizaba en sus terminales de telefonía móvil, y gracias a ello, se encontró un conector para continuar avanzando. La alimentación embarcada se realizará mediante pilas recargables a 5V DC, mientras que en tierra, la placa se alimenta con un adaptador de 5V DC 1A conectado a la red comercial de 220 V AC.

Una vez alimentado el procesador Gumstix Overo IronSTORM, se probó el sistema operativo que Gumstix nos ofrecía. Una distribución UNIX Angstrom con entorno gráfico Enlightment. Esta distribución que aporta el fabricante del computador viene sin ningún compilador para crear nuestro programa, ni con las librerías de OpenCV, básicas para la realización de este proyecto; y además, esta distribución que Gumstix vende está sin actualizar. Eso hace que el primer paso a dar es realizar una actualización para después instalar los programas que necesitamos. La actualización se realiza vía internet, conectando la placa a la red. La conexión utilizada fue un punto wireless a través de un teléfono móvil con tarifa de Internet.

Para evitar riesgos de fallos del sistema operativo a la hora de actualizar el sistema operativo, y así, trabajar con mayor seguridad, se realizan varias tarjetas microSD arrancables, para que la placa cargase el sistema operativo desde la tarjeta SD y no desde la memoria interna, siguiendo el manual "How to create a SD Bootable" de la web de Gumstix Developers. Las tarjetas microSD que se disponen tienen una capacidad de 2 GB la más pequeña, y de 8 GB las otras. Las de 8 GB dieron muchos problemas a la hora de particionarlas por ser superiores de 4 GB; problema inherente a las tarjetas SD y los lectores de tarjetas comerciales, pero con un poco de pericia se consigue realizar las particiones. En una de ellas, se probó cargar el sistema operativo Ubuntu 10.04, y en la otra el Angstrom que Gumstix oferta de fábrica en su web. La tarjeta de Ubuntu se creó, con la aplicación rootstock y el programa qemu, que emula diferentes arquitecturas, siguiendo los pasos que se indican en el manual "Installing Ubuntu 10.04 on Gumstix Overo" de la wiki de Gumstix. En la prueba, el sistema cargaba y permitía realizar operaciones, pero a los pocos minutos, se bloqueaba, como si fallase algún sistema de señales eléctricas, lo cual hizo que se desechara seguir por este camino. La otra tarjeta de 8 GB se cargó con el sistema operativo de fábrica que ofrece Gumstix. Ese sistema operativo sólo puede arrancar desde memoria interna, y no desde una memoría externa como es una tarjeta microSD. Ese provocó que esta distribución del sistema operativo cargada en la tarjeta de 8

GB también fuera desechada.

Con la tarjeta de 2 GB, lo que se buscó fue una distribución UNIX de Angstrom, que Gumstix oferta en la web, pero en vez de ser la que viene de fábrica, se eligió la última versión que los desarrolladores han creado. Y esta versión, si que viene con el compilador instalado por defecto.

El sistema operativo se monta en la tarjeta de 2 GB del siguiente modo. En la tarjeta, se hacen al menos dos particiones, una arrancable en formato FAT de unos 75 MB llamada boot y otra con sistema de archivos linux con el resto de la capacidad llamada rootfs. En esa segunda partición es donde se creará el sistema de archivos para el ordenador. En la partición arrancable, se instalan tres archivos, en este orden, un MLO, un u-boot.bin y un uImage, que son los archivos de arranque. Y en la otra partición, se descomprime el sistema de archivos que hemos descargado de Gumstix. Se propone realizar estos pasos en la tarjeta de 8 GB para tener más espacio disponible en la memoria del computador, aunque Gumstix Overo permite conectar memorias externas vía USB.

Al probarlo, arrancándolo desde terminal con una conexión serie desde un portátil, como nos indica el fabricante en su web "Setting up a serial connection", no dio ningún error más allá del tiempo de arranque, que es algo mayor de lo esperado. Ese tiempo de arranque tan alto solamente sucede al cargar el sistema operativo por primera vez, porque ha de ser configurado. Tras la configuración, el arranque se hace en un tiempo razonable. Y en está distribución, actualmente no hay que actualizar ningún paquete. Además, a través de los repositorios de Angstrom, se puede instalar las librerías de OpenCV; con lo cual el programa, que se había realizado y comprobado en Windows XP, se puede trasladar al computador que irá embarcado, cambiando algunos códigos que cambian de Windows a UNIX. Actualmente, se han compilado programas de código C con llamadas a librerías de OpenCV en esta distribución, tanto propios como ejemplos propuestos por el distribuidor de OpenCV. Se ha implementado parte del algoritmo en el computador Gumstix, comprobando su funcionamiento. Como trabajo futuro se propone terminar la implementación completa del algoritmo.

El montaje de la placa se realiza con una salida a monitor digital HDMI - DVI-D, un teclado y un ratón, conectados en un hub y de ahí al mini USB -A, que es el que debe alimentar a los periféricos, el USB host se conectará a una memoria USB para almacenamiento de datos y el mini USB - B, o USB Console, se utilizara para la conexión serie. La placa del procesador, se coloca mediante presión a la placa base Gumstix Overo Tobi, que es la que tiene las salidas a los periféricos.

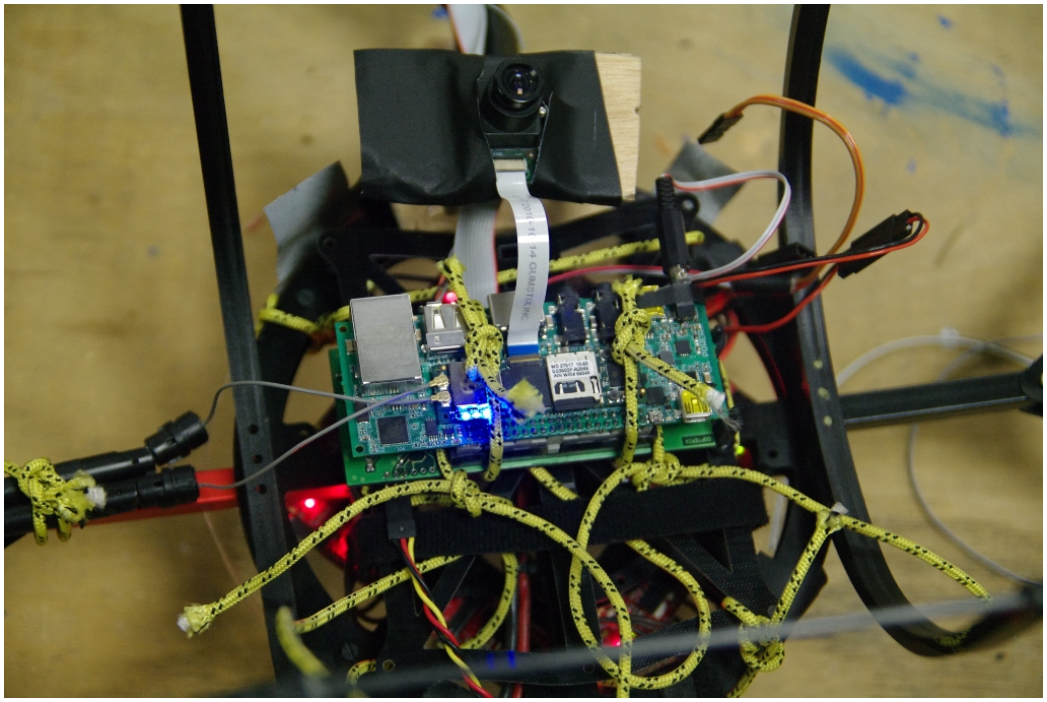


Figura 7.6: Gumstix Overo Tobi y Gumstix Caspa montadas en el proyecto Copterix

El montaje final es similar al que el grupo francés de investigación de la Telecom ParisTech, Copterix, hace para sus desarrollos, como vemos en la imagen y como se verá en los siguientes capítulos.

7.3. Evaluación

La evaluación del sistema se llevó a cabo con una serie de experimentos con imágenes y vídeo, y se realizó en modo de aprendizaje, probando cada parte del sistema por separado para ir incluyendo poco a poco más funciones y tareas a realizar por parte del procesador.

El resultado de estas simulaciones dio lugar a la corrección de varios de los fallos que tanto el compilador, la versión 4.2 del General Compiler C (GCC), que fallaba por problemas técnicos inherentes a él con ciertos tipos de archivos de vídeo, como del propio programa, como podía ser la falsa detección de puntos (puntos que detecta como de uno de los bordes, pero no lo son, por ser del otro, o del horizonte, o de otras marcas en la via) o la adquisición de puntos que debían ser descartados por fallo de programación (la esquina superior izquierda, origen de coordenadas, en algunos casos, por fallos de asignación de memoria se seleccionaba sin ser un punto de los susceptibles a pertenecer a los bordes) Ese problema se soluciona en la placa Gumstix, pues Angstrom viene que la versión 4.3.3 de GCC, que evita ese bug.

Esto hizo que el programa fuese robusto frente a distintas imágenes, en la que el programa realizaba todas las funciones, distintos vídeos, distintos umbrales tanto para la segmentación, como para el nivel de gris y distintos pasos de búsqueda entre píxeles.

Además, se llevaron a cabo una serie de experimentos para corroborar los distintos pasos que ejecuta el programa, devolviendo como salida una serie de vídeos, en los que se ve como va eligiendo los puntos para los detectores o la segmentación a cero de la imagen en blanco y negro.

Capítulo 8

Análisis de resultados y conclusiones

Figura 8.1: Imagen Original de la Carretera, que se utiliza para las pruebas

Lo primero que se consiguió fue el detector de horizontes en imágenes, en una imagen en formato *.jpg, con una resolución de 640 X 480 píxeles, que es la resolución estándar de grabación de vídeo. La imagen que se utilizó era una imagen real de una carretera de dos carriles, ya que nos permite simular con un grado de exactitud bastante alto a una pista de aterrizaje, sobre todo para el tipo de UAS de ala fija para el que se planea integrar el sistema.

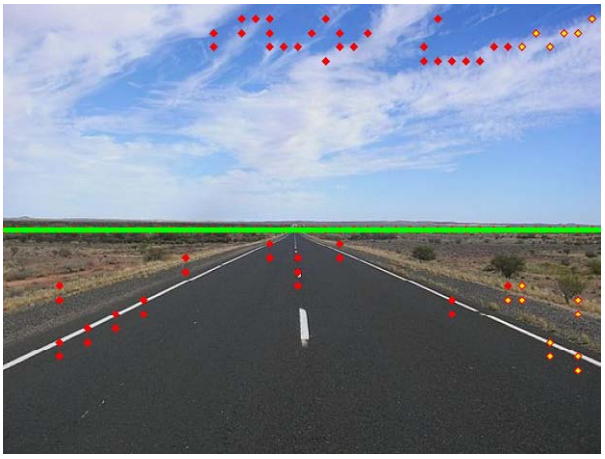


Figura 8.2: Salida de ensayo de Nivel de Horizonte
El ensayo se realiza con un paso de barrido entre píxeles de 15 píxeles, una segmentación binaria de 135 y un umbral de cambio de intensidad de gris de 70. Se ve el problema del cielo nublado que se ha comentado anteriormente.

Figura 8.3: Salida de ensayo de Nivel de Horizonte
El ensayo se realiza con un paso de barrido entre píxeles de 25 píxeles, una segmentación a cero de 175 y un umbral de cambio de intensidad de gris de 100.

Y con esa imagen se realizo la detección del horizonte con distintos pasos entre píxeles en el barrido, distintos umbrales de nivel de intensidad de gris para capturar puntos y distinto niveles de umbral en la segmentación. Al final, para terminar de comprobar la robustez del programa, se ejecutó el programa con otras imágenes, como un paisaje marítimo, o el paisaje visto desde el ala de un avión, llegando a la misma conclusión. Como se verá en las distintas imágenes de los resultados, el horizonte se detectará dibujándolo con una línea de color verde, el borde izquierdo con una línea de color negro y el borde derecho con una recta de color azul. Los distintos puntos detectados tendrán una variación en su forma de salida, siendo rojos en el detector de horizontes, verdes en la primera versión del detector de bordes, y rojos (Horizonte), naranjas (Bordes), verdes (Borde Izquierdo) y morados (Borde Derecho) en el detector final.

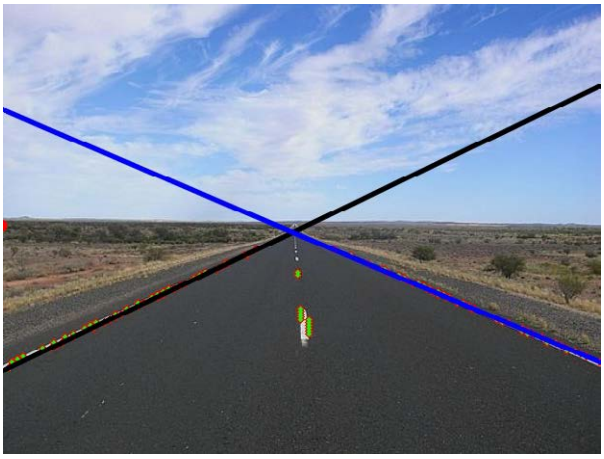


Figura 8.4: Salida de ensayo de Detección de Borde de Pista
El ensayo se realiza con un paso de barrido entre píxeles de 3 píxeles, una segmentación a cero de 175 y un umbral de cambio de intensidad de gris de 120.

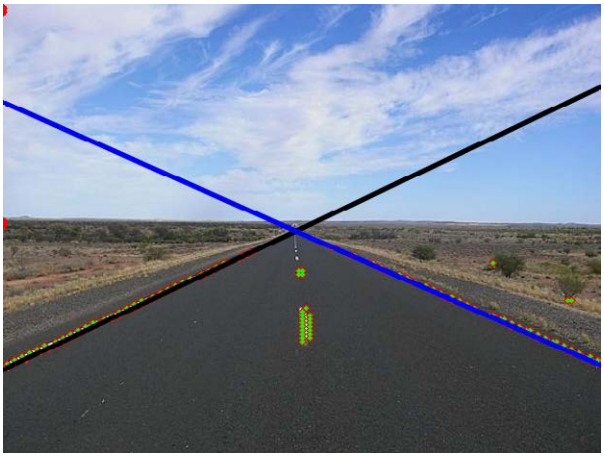


Figura 8.5: Salida de ensayo de Detección de Borde de Pista
El ensayo se realiza con un paso de barrido entre píxeles de 3 píxeles, una segmentación a cero de 175 y un umbral de cambio de intensidad de gris de 80.

El siguiente paso fue descubrir que al barrer la imagen en el sentido contrario, la imagen detectaba los bordes de la pista, y no el horizonte (en caso de estar en horizontal, sin ángulo de balance), y se realizó el mismo experimento, con las mismas condiciones anteriores. La imagen de la carretera con la que se inician los ensayos da una buena idea de la robustez del programa, y de los posibles errores que se pueden cometer, ya que no solo se detecta el borde, sino que la línea central que separa ambos carriles también es detectada, y en ciertos casos, si no se han escogido de manera correcta los paramétros del programa, puede falsear los resultados.

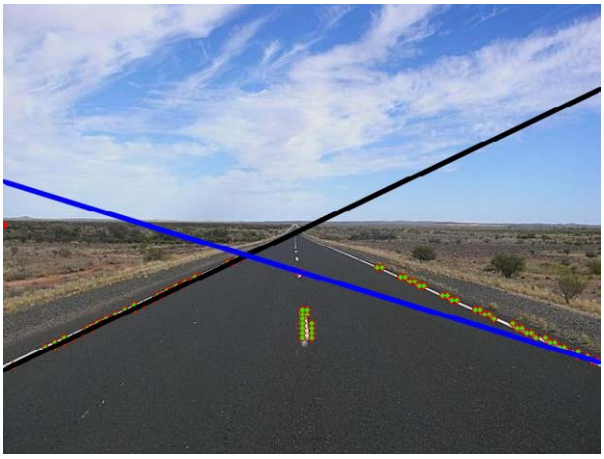


Figura 8.6: Salida de ensayo de Detección de Borde de Pista
El ensayo se realiza con un paso de barrido entre píxeles de 4 píxeles, una segmentación a cero de 175 y un umbral de cambio de intensidad de gris de 120. Se comprueba el bug en el borde derecho.

Figura 8.7: Original y Salida del ensayo de Deteccion de Borde de Pista

La imagen más significativa en la que se ven los resultados correctos de este programa la obtenemos del tratamiento de una imagen de otra carretera de montaña, donde el horizonte y el borde derecho apenas tienen distinción, y sin embargo, el detector de bordes funciona perfectamente.

Figura 8.8: Imagen original de los ensayos

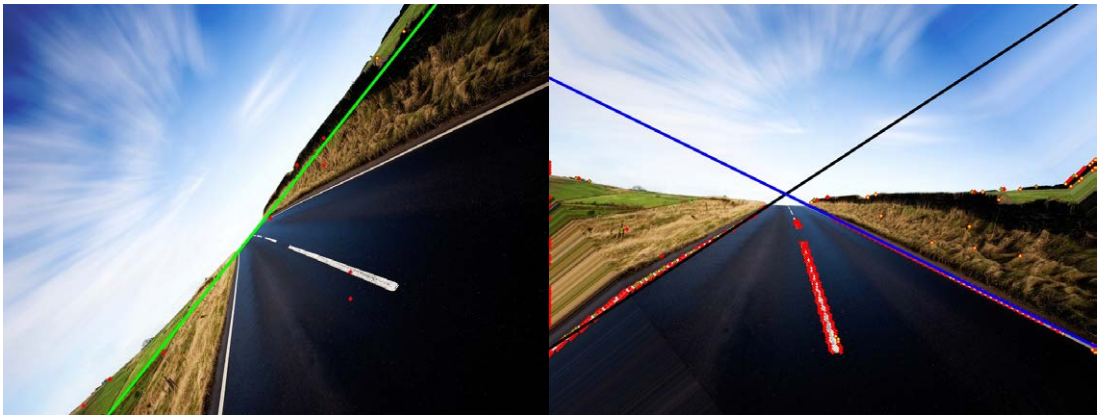


Figura 8.9: Salida del ensayo de Detección de Nivel de Horizonte y Salida del ensayo de Detección de Borde de Pista
El ensayo de detección de horizonte está realizado con un nivel de umbralización al cambio de intensidad de gris de 110 y un paso en el barrido de 50 píxeles.

Una última imagen de otra carretera, similar a la anterior pero con el horizonte girado unos 50º, sirvió para comprobar que ambos detectores funcionaban bastante bien, primero el de horizonte; y permitió el ensayo del mecanismo de cálculo de ángulo de balance y su corrección para la detección de la pista

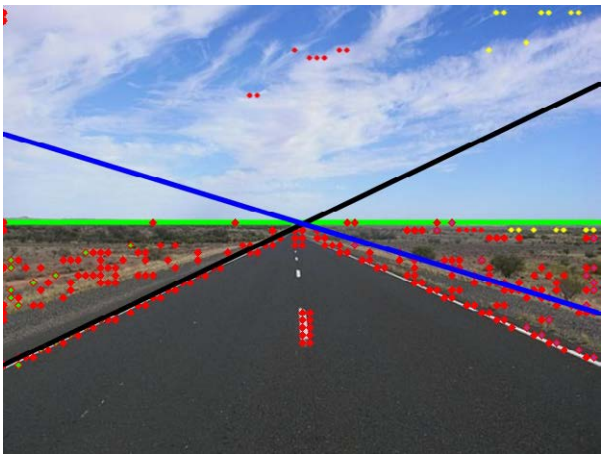


Figura 8.10: Salida del ensayo de Detección de Nivel de Horizonte y Detección de Borde de Pista para imágenes
El ensayo se realiza con un paso de barrido entre píxeles de 8 píxeles, una segmentación a cero de 110 y un umbral de cambio de intensidad de gris de 50 para horizonte y de 55 para bordes.

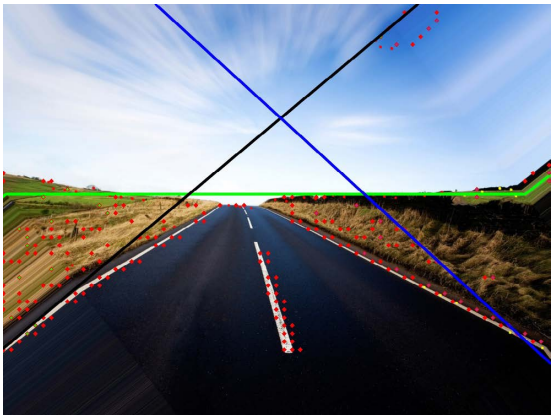


Figura 8.11: Salida del ensayo de Detección de Nivel de Horizonte y Detección de Borde de Pista para imágenes
El ensayo se realiza con un paso de barrido entre píxeles de 15 píxeles, una segmentación a cero de 110 y un umbral de cambio de intensidad de gris de 60 para horizonte y de 95 para bordes.

La siguiente fase fue la implementación de los dos detectores en un mismo programa, lo cual, aunque parece sencillo, dio problemas en la interpretación de algunos de los comandos y de las ordenes por duplicidad de nomenclatura. Además, se implementó el código por el cual, si el ángulo de balance no era nulo, el programa rotaba la imagen y la dejaba con la línea del horizonte en horizontal, y rellenaba los píxeles que se quedan vacíos tras el giro con el valor medio de los píxeles vecinos.

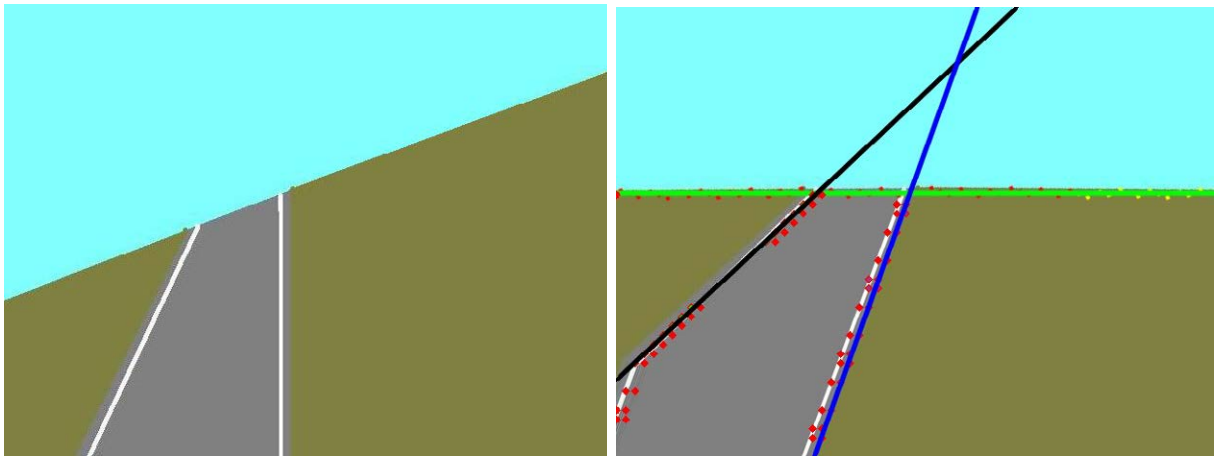


Figura 8.12: Original y Salida del ensayo de Detección de Nivel de Horizonte y Detección de Borde de Pista para imágenes
El ensayo se realiza con un paso de barrido entre píxeles de 10 píxeles, una segmentación a cero de 190 y un umbral de cambio de intensidad de gris de 50 para horizonte y de 95 para bordes.

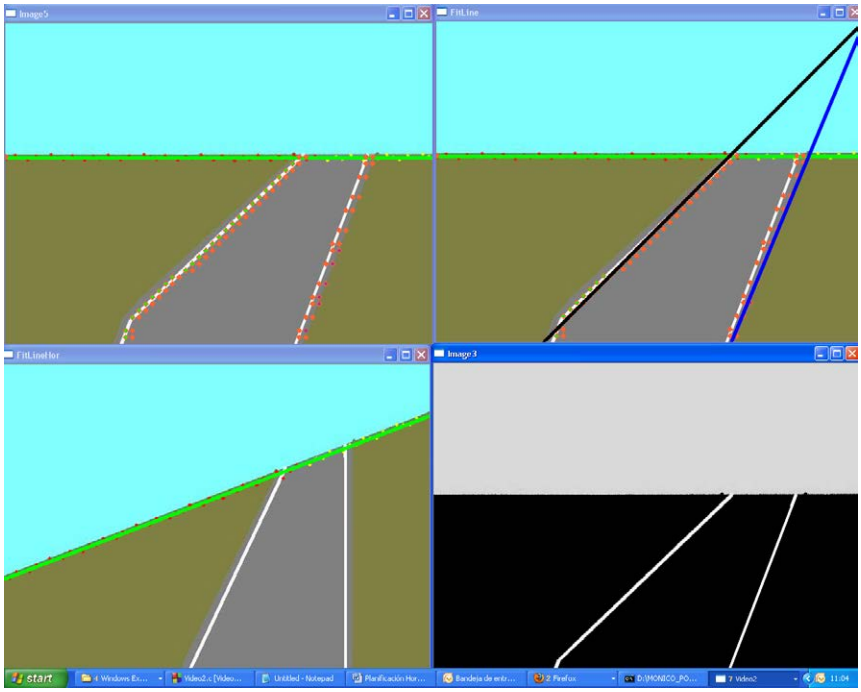


Figura 8.13: Ejecución del programa en simulación de vídeo

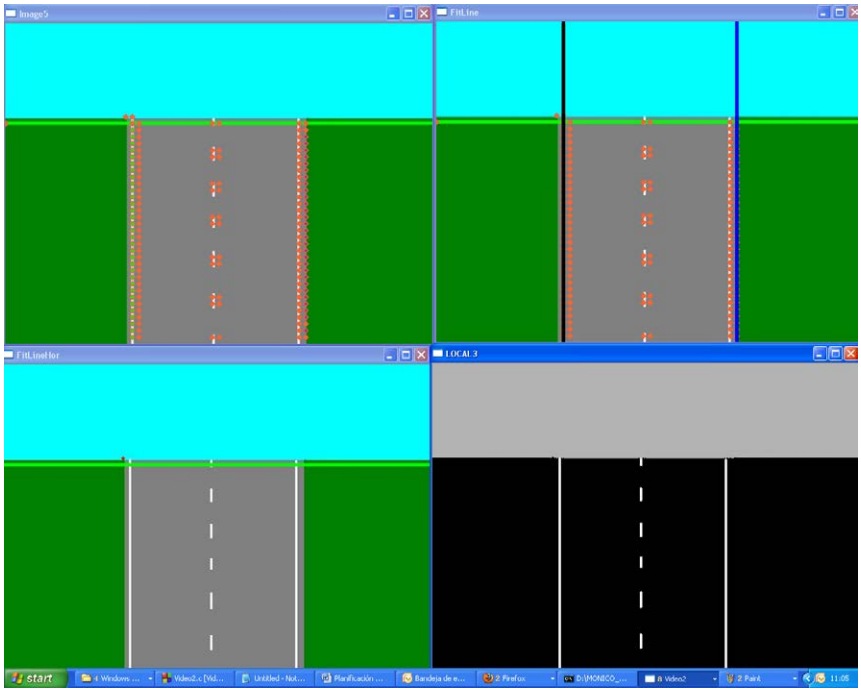


Figura 8.14: Ejecución del programa en simulación de vídeo

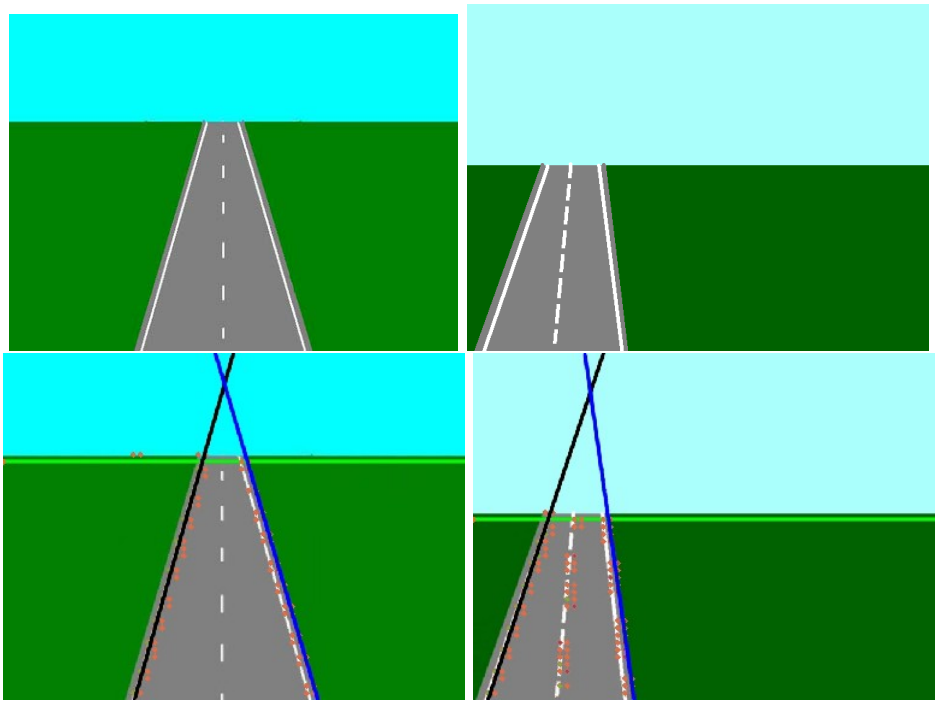


Figura 8.15: Original y Salida del programa en simulación de vídeo

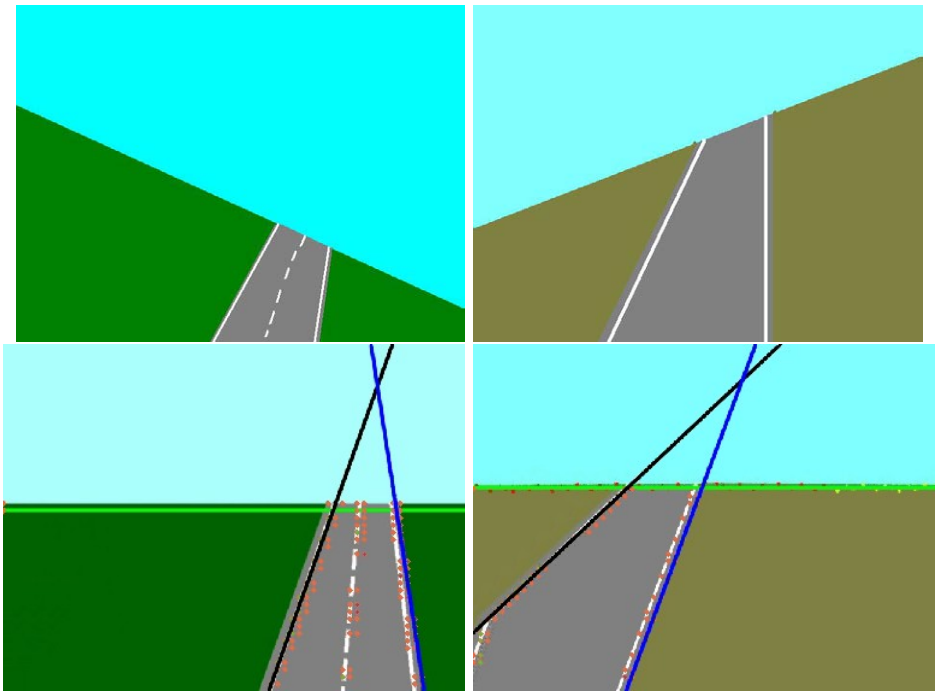


Figura 8.16: Original y Salida del programa en simulación de vídeo

Una vez comprobado que el programa realizaba ambas funciones y presentaba en pantalla el ángulo de los bordes de pista, y por lo tanto, la dirección que debería tomar nuestro UAS para colocarse respecto a ella, el siguiente paso fue la implementación de vídeo en diferentes formatos, como son el *.wmv y el *.avi. Fue un paso costoso por el motivo que hemos avanzado anteriormente. El compilador daba bastantes fallos al hacerle presentar en pantalla muchos resultados en vídeo y además, mandaba un aviso de error interno al arrancar cualquier aplicación de vídeo. Ese error se corrige en la siguiente versión del compilador GCC (4.3), pero no en la que Code::Blocks implementa

por defecto (4.2) y por eso, se buscaron formas alternativas para realizar los ensayos del programa, mediante vídeos creados a partir de imágenes distintas pero de la misma resolución con la aplicación Windows Movie Maker, para comprobar la eficacia en el cambio de fotogramas y con la captura vacía de fotogramas para agilizar la ejecución del programa.

Fotograma Capturado	Ángulo Borde Izquierdo	Ángulo Borde Derecho
1	74	73
2	74	73
3	73	73
4	74	73
5	73	74
6	74	74
7	74	73
8	74	73
9	73	81
10	71	82
11	71	69
12	71	80
13	70	81
14	70	77
15	70	84
16	70	84
17	71	63
18	56	90
19	57	90
20	57	88

Cuadro 8.1: Datos de Salida del fichero generado para servir de entrada al programa de la estación de tierra

El último ensayo del programa se realizó con la idea de capturar fotogramas por tiempo, variando en tiempo de captura, en vez de por fotogramas, pero ese ensayo no dio buenos resultados, ya que capturaba el fotograma, pero seguía trabajando con el fotograma anterior del vídeo, y además, después de la captura, en la salida de vídeo donde se ve la imagen del horizonte y de los bordes de la pista, se mostraba un parpadeo negro muy molesto, debido a que en ese instante, si que trabajaba con el fotograma capturado, que está en blanco y negro. El que sí dio buenos resultados fue el ensayo de salida por archivo de los datos de los ángulos de los bordes de la pista, aunque hay que refinarlo, ya que hay que conseguir que el archivo que se guarde sea el que se mande a tierra, y para ello, hay veces que debe abrir el archivo y actualizarlo y otras debe borrar el contenido y generarlo de nuevo. En otras ocasiones, interesara que el archivo ni se borre ni se actualice, sino que se guarde con otro nombre, para mantener una copia de seguridad y realizar un seguimiento pero este requerimiento no se ha conseguido realizar todavía de manera eficiente, sin que el programa devuelva una serie de errores, como la no actualización del archivo, o sustituir el borrado del archivo por la actualización o viceversa.

Como se ve en los resultados, el programa es bastante robusto ante cambios de paso entre píxeles en el barrido, ante cambios de umbral en el nivel de gris, y en la segmentación, y ante cambios de imagen; y que también da bastante buenos resultados en cuando la imagen de la pista (o la del horizonte) tiene un contraste alto con la imagen del fondo.

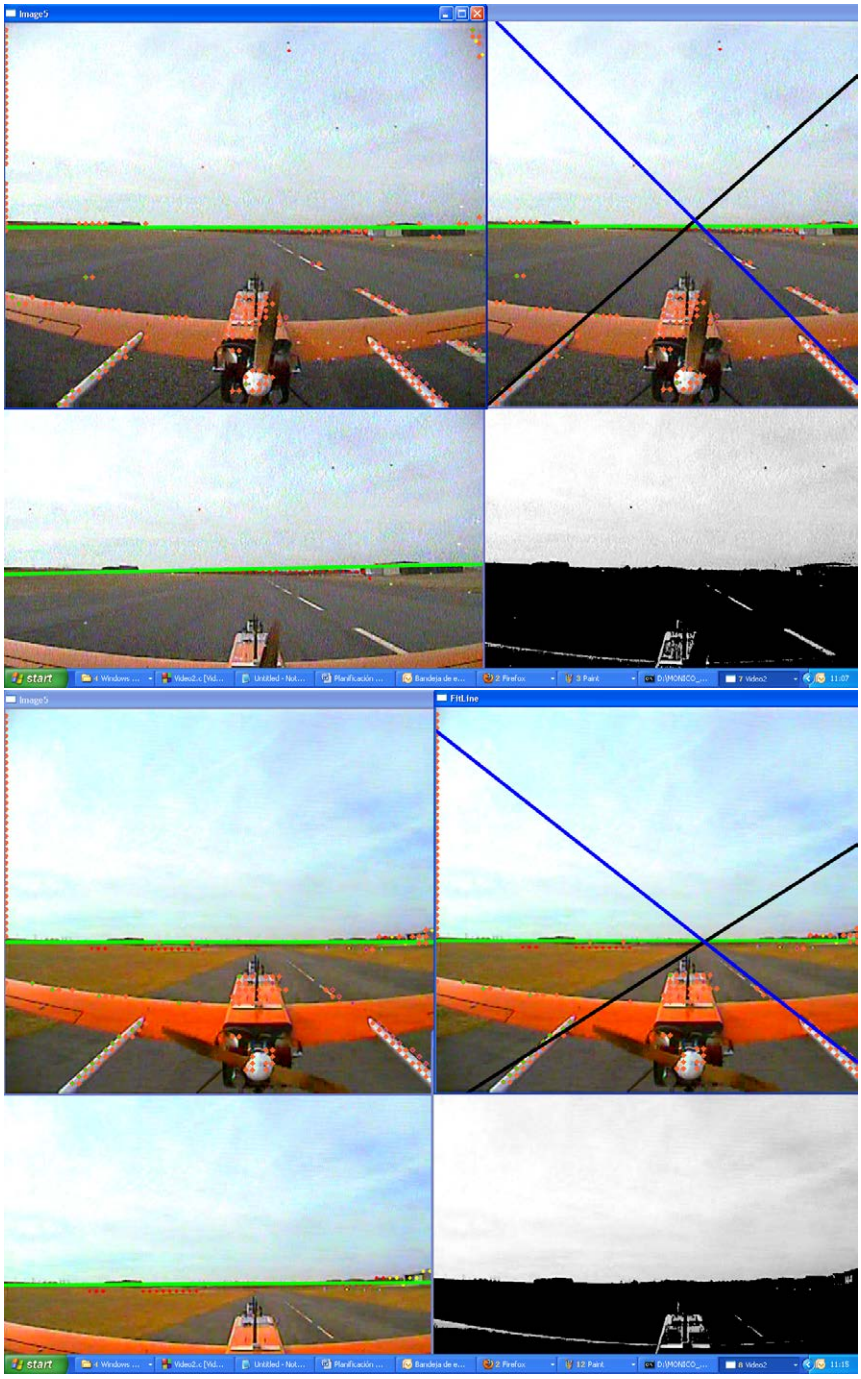


Figura 8.17: Ejecución del programa en vídeo de vuelo real

Se han capturado dos momentos del vuelo, en ellos, se comprueba la salida por pantalla del programa, donde se ve, de arriba a abajo y de izquierda a derecha, la detección de puntos del detector de bordes, la imagen final, con los bordes y el horizonte detectados, la imagen de salida del detecto de horizonte y la imagen segmentada.

Se ve además, que el programa, puede llegar a detectar otro tipo de bordes si la integración de la cámara no es la correcta, como se ve en los resultados del vídeo de la aeronave con la cámara en cola, que lo que detecta son las barras metálicas que unen el fuselaje a la cola que tienen un contrate mayor, y no la pista.

El tiempo de respuesta es adecuado, pues los cálculos del procesado de imagen tardan alrededor de 200 milisegundos por fotograma, 5 fotogramas por segundo, una velocidad bastante aceptable,

como indicaron desde el grupo del profesor Campoy, que especificaron que hasta 10 fotogramas por segundo era una velocidad muy adecuada para ser un programa que se inicia desde cero.

En este capítulo se han visto algunos de los resultados más significativos de los ensayos. Para profundizar más en los ensayos y sus resultados, consultar el Apéndice A de este proyecto, dónde se incorporan el resto de los resultados de los ensayos

Capítulo 9

Propuesta de integración en un UAS

Uno de los objetivos de este proyecto es la ayuda a la navegación durante la fase de aterrizaje de una aeronave no tripulada. Para llevar a cabo ese objetivo, se va a proponer un modelo de integración del sistema en la aeronave no tripulada Mugin.

La placa Gumstix Overo Tobi, donde se coloca el procesador Gumstix Overo IronSTORM es una placa que permite la conexión de una cámara a través de un puerto USB, o conectar la cámara Gumstix Caspa a través de un conector de 27 pines. Ese conector, es un conector que está situado justo por encima del procesador, y según nos marca el fabricante, se conecta con un cable flexible de 80 mm a la cámara. Hay que tener en cuenta es la longitud de cable, pues puede ser crítica a la hora de integrar el sistema a la aeronave.

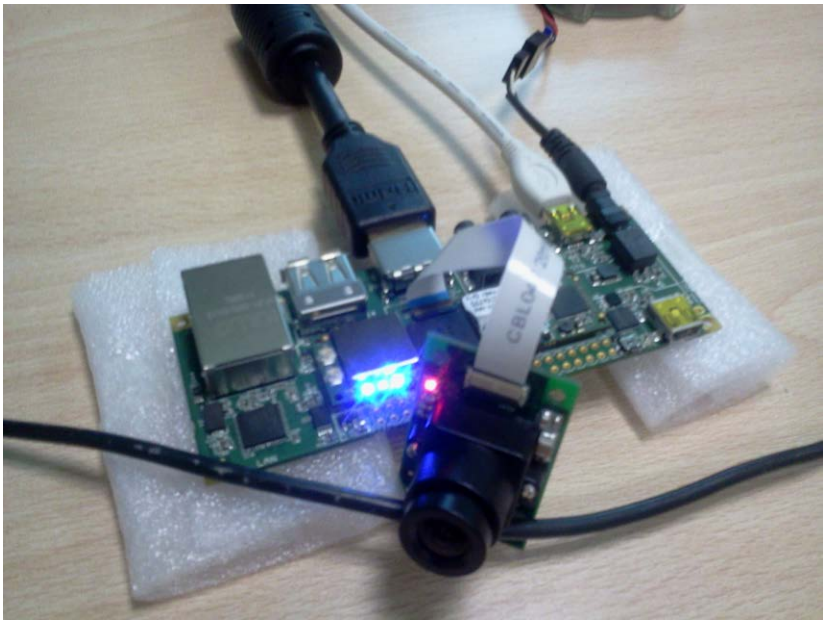


Figura 9.1: Sistema de Visión por computador Gumstix Overo (IronSTORM-Tobi-Caspa)

Esta placa, con la cámara montada, se probó en tierra, conectada a un monitor DVI, a un teclado y a un ratón USB a través de un hub, comprobando su correcto funcionamiento. Se adjunta una imagen de sistema montado en tierra para su comprobación.

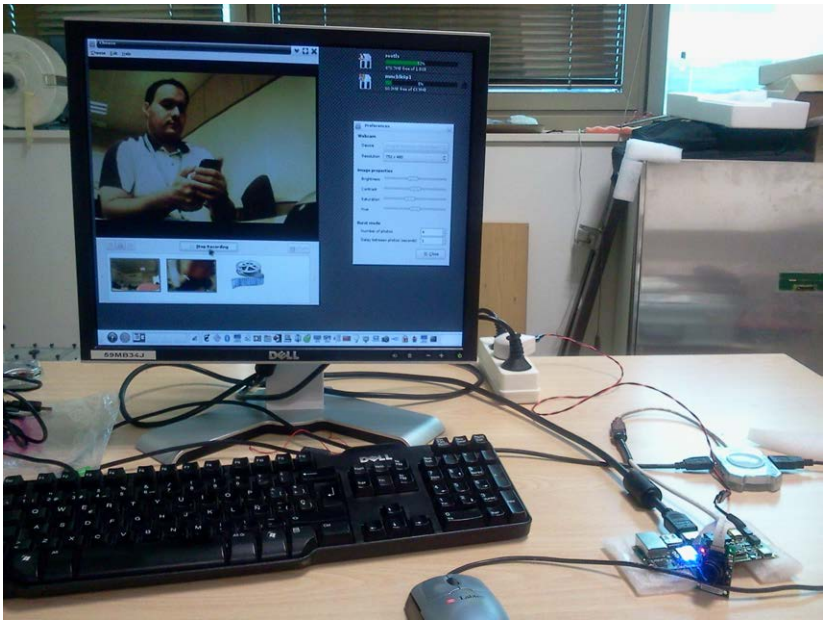


Figura 9.2: Comprobación de funcionamiento del Sistema de Visión por computador Gumstix Overo (IronSTORM-Tobi-Caspa)

Una de las características a la hora de seleccionar este computador era su compatibilidad con el autopiloto Paparazzi que es el que está instalado en la aeronave Mugin donde se pretende hacer funcionar el sistema de visión por computador. Paparazzi es un sistema que incluye tanto el autopiloto embarcado como la conexión en tierra para control de la misión mediante modem bidireccionales. Es un sistema abierto desarrollado por la École National de'Aviation Civile francesa y que permite una integración con sistemas OMAP, como es el Gumstix Overo.

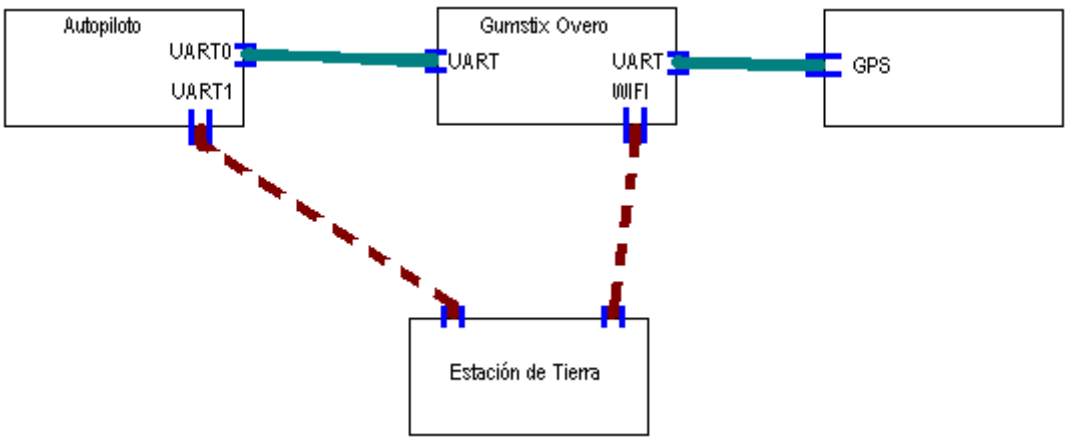


Figura 9.3: Conexión Overo - Paparazzi

La conexión entre Overo y Paparazzi se realiza conforme el esquema que se incorpora. Si se requiere de mayor información sobre la integración entre Overo y Paparazzi, se recomienda visitar la wiki de Paparazzi ("http://paparazzi.enac.fr/wiki/Overo"), donde se indica tanto el procedimiento a seguir en la conexión del hardware y del software, como las características de la conexión.

La aeronave Mugin posee las siguientes características:

Envergadura: 3 m.

Longitud: 2'25 m.
Peso: 9 kg.
Velocidad mínima: 40 km/h = 11.11 m/s
Velocidad máxima: 150 km/h = 41.67 m/s
Autonomía: 3 horas
Motor bicilíndrico de gasolina de 55 cc.
Rango de RPM: 1600-7800
Peso Seco: 1.78 Kg (sin sistema de arranque DC-CDI de 4.8V)

Figura 9.4: UAS Mugin

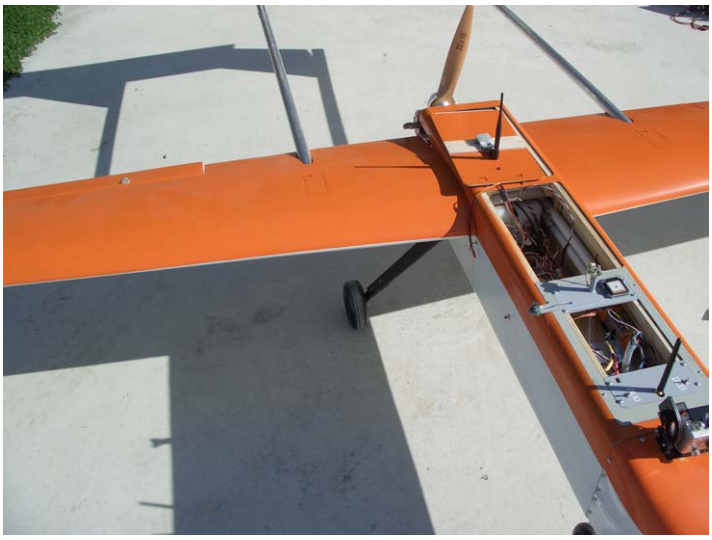


Figura 9.5: Interior Uas Mugin

Una vez conocidas las características de la aeronave, se procede a explicar las posibles integraciones en el avión. Una posible colocación del sistema podría ser la parte trasera de la aeronave. Eso sería crítico, pues la cámara debería ir colocada a suficiente altura como para salvar toda la aeronave en su campo de visión para poder llevar a cabo su cometido sin perturbaciones. Eso a día de hoy, es inviable por diversos motivos. El fundamental es por una limitación física en la conexión entre la cámara y el computador, debido a que la longitud del cable que los conecta es de 80 mm., por lo que el computador debería ir dentro de la superficie de cola, o al aire, en la estructura que se

diseñaría para elevar la cámara, y como se ve, eso no es viable. Por lo tanto, esta solución se debería descartar.

Figura 9.6: UAS Mugin con sistema para la colocación de una cámara en el morro del avión

Eso hace que la solución óptima sea un sistema similar al de la imagen, con la cámara en la parte superior del morro de la aeronave, y el procesador embarcado en el primer compartimento del fuselaje del avión. Para evitar que el procesador sufra movimientos bruscos y se desconecte en vuelo, se propone anclarlo al fuselaje del avión a través de unas patillas que se ajustan en los agujeros de las cuatro esquinas que tiene la placa de expansión. Eso también tendrá un efecto beneficioso para la disipación del calor que emite el procesador en su funcionamiento. Además, habrá que tener en cuenta la longitud del cableado de conexión entre la cámara y el procesador, por lo que se intentara acercar al máximo posible la placa al morro, para acercar a la cámara a su emplazamiento final.

También se deberá tener en cuenta la conexión del ordenador de procesado de imágenes con el autopiloto, que también se hará de manera cableada, conectándose entre si por dos puertos series. Sobre todo se tendrá que tener en cuenta que el cableado no afecte a ningún otro sistema de la aeronave.

Capítulo 10

Estudio de la interacción software

En este capítulo se introducirá la interacción entre el segmento embarcado y el segmento en tierra del sistema de visión artificial una vez realizada la propuesta de integración en la aeronave. Para ello, se introduce una propuesta simple de la salida en pantalla de la estación en tierra. También se hablará de los mecanismos para la selección del modo "visión", que es el modo en el que se pone en marcha el sistema, y el mecanismo de recuperación del control manual, para finalmente terminar por especificar las compatibilidades del sistema, tanto las compatibilidades entre ambos ordenadores embarcados, como las compatibilidades entre segmento de tierra y los componentes embarcados.

La entrada de datos se realiza mediante un archivo de texto en el que se guardan las pendientes de los bordes detectados. Ese archivo de texto se envía a tierra mediante un módem bidireccional embarcado, y se recibe en tierra con otro módem. Y se realiza de este modo, debido a que los canales de transmisión de datos obligan a que los datos a enviar y recibir no sean demasiado grandes. Eso también nos limita para el envio de la señal de vídeo, que para mandarla a tierra es necesaria que sea de muy baja resolución y de muy baja calidad, ya que el envio de vídeo a través de este tipo de modem satura los canales de información y ralentiza las conexiones.

La interfaz de la estación de tierra que se propone es una salida por pantalla en la que lo primero que indique es si el sistema de visión está encendido o apagado. Si está encendido, indicará también si el archivo de datos está llegando o si por el contrario, no se está conectando con el módem, y por lo tanto, el archivo no llega a su destino.

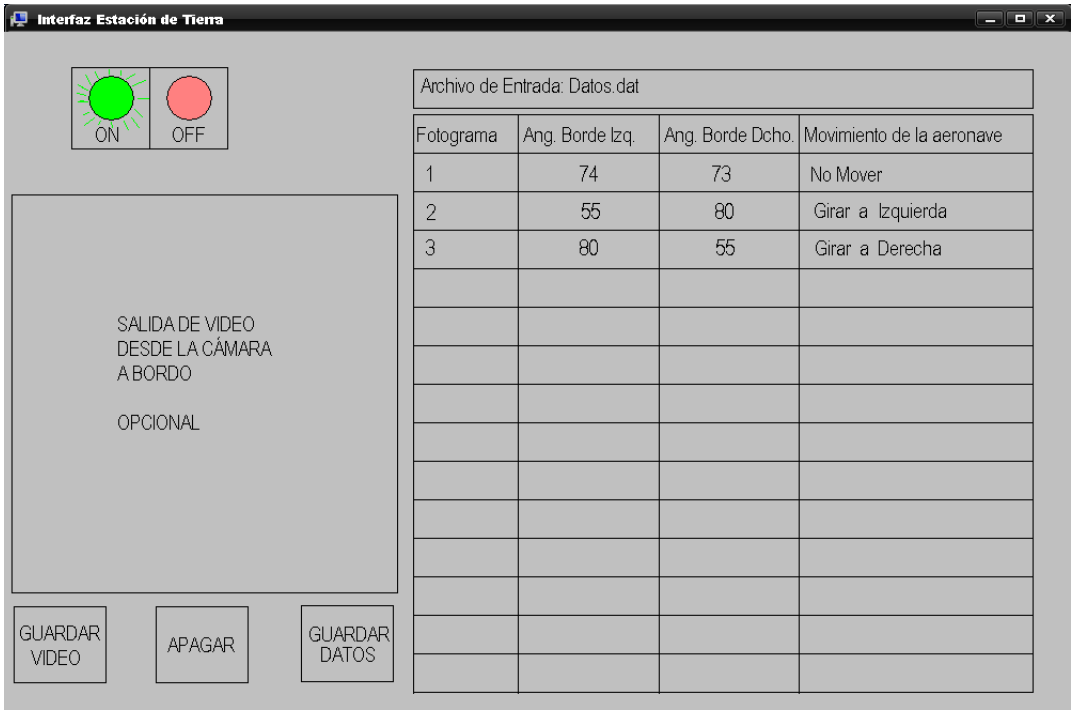


Figura 10.1: Posible Interfaz del programa de la Estación de Tierra

Si el archivo llega a su destino, el archivo se lee y se muestra por pantalla, indicando el fotograma examinado, los ángulos de las pendientes de los dos bordes y la actuación para corregir la trayectoria que debe hacer la aeronave. Como solo se envían ambos ángulos, tanto el fotograma como el movimiento se conocerán a través de un programa que haga un contador de líneas de datos, y un comparador de ángulos, en los que nos indique si hay que moverse a izquierda, a derecha o mantenerse en la misma trayectoria.

Opcionalmente, se puede enviar la salida de vídeo a tierra, para controlar lo que la cámara está viendo, con las peculiaridades que anteriormente se han especificado, ya que la señal de vídeo suele ser una señal que lleva mucha información asociada a ella. Se pueden enviar en vez del vídeo, las imágenes ya tratadas, con los bordes detectados.

Por último, se colocaría una opción por la cual, se podría hacer un seguimiento para realizar distintos ensayos guardando los datos del programa para tratarlo a modo de registro, y se podría colocar la opción de guardar el vídeo si se considera su emisión, para comprobar y corroborar el ensayo.

10.1. Mecanismo de selección del modo "visión"

El modo “visión” es el modo en el que se pone en marcha el sistema de visión por computador. Este modo será el modo importante del sistema de visión artificial.

Como el autopiloto tiene un modo por el cual, activa el autopiloto o se recupera el control manual mediante la activación de un interruptor de seguridad desde una emisora de radiocontrol, se puede usar ese mismo mecanismo para activar el modo visión. Usar ese mecanismo no implica usar la misma orden de activación del autopiloto para activar el sistema de visión, pero en un futuro, sería la opción correcta.

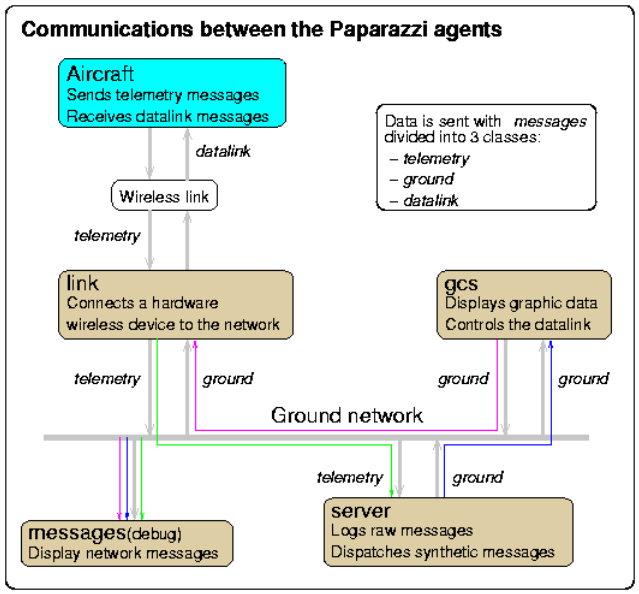


Figura 10.2: Arquitectura de comunicación de Paparazzi

En el momento en el que se activa el modo de visión, se le da una orden desde Paparazzi a Overo, para que se active el programa de visión por computador. Para ello, se le da una orden de activación a través de una receta Bitbake, que hace el que sistema se ponga en marcha y ayude al control la trayectoria. Esa es la manera automática que los sistemas UNIX permiten para realizar órdenes y hacer que un programa se ejecute o se cierre sin tener que montar una interfaz gráfica para controlar el sistema operativo.

Bitbake es un compilador multiplataforma para sistemas UNIX que ante la entrada de un archivo escrito en algún lenguaje de programación (Python, C, C++, etc...), genera un archivo ejecutable al modo de los sistemas operativos Windows. Por eso, una receta Bitbake es la opción más razonable para automatizar el arranque del programa en el sistema que se está diseñando. Otro modo para automatizar ese arranque sería generar un script que se ejecutase al arrancar el sistema operativo y que dé la orden de ejecutar el archivo objeto precompilado del programa. Ese método es más rudimentario, porque cada vez que se ejecuta el programa, hay que generar el archivo ejecutable y eso es un consumo innecesario de tiempo y memoria si se tiene otra herramienta como es Bitbake.

Paparazzi, además, puede controlar el movimiento de la cámara, a través de una opción que tiene en la estación de tierra del autopiloto, que puede ser distinta a la del procesador de imágenes.

Paparazzi utiliza un bus de software llamado Ivy Software Bus para la gestión y la transmisión de datos entre la estación de tierra y el autopiloto embarcado. Ese bus de software hace que el control de tierra monitorice, mediante datos telemétricos, la navegación autónoma del UAS. Esos datos telemétricos son distribuidos en una red local por un canal de datos inalámbricos a través de los módems bidireccionales. Este sistema se ejecutan de forma simultánea en el software Paparazzi Center, que es el que se carga en la estación de tierra.

Ivy es un bus de software basado en mensajes de texto que es utilizado por Paparazzi como fuente principal de comunicación. Ese bus de software puede ser utilizado por cualquier otro componente incorporando sus librerías y sus archivos específicos. Eso hace que se pueda comunicar también con Paparazzi. El protocolo de comunicación es independiente del proyecto Paparazzi y eso permite que pueda ser utilizado por otros sistemas.

Ivy normalmente funciona con una función de envío con una cadena de texto mostrada en pantalla por terminal y otra de enlace. La función de enlace se compone de dos parámetros, una llamada y una expresión regular. Cuando algún componente tiene un mensaje que enviar que concuerda con una expresión regular de uno o varios enlaces, se dispara la llamada asociada con la cadena de texto como parámetro.

10.2. Compatibilidades

Las compatibilidades del sistema vienen dadas en dos tramos diferenciados. Los tramos son las compatibilidades entre autopiloto y el procesador de imágenes, y el otro tramo es la compatibilidad entre el segmento de tierra y el segmento embarcado con los dos módems bidireccionales y la emisora de radiocontrol.

El autopiloto Paparazzi se conecta vía puerto serie USB al procesador de imágenes. Eso da una limitación de datos transmitidos, de velocidad de transmisión y otra limitación física por la longitud del cable.

Otra compatibilidad que hay que tener en cuenta es la propia compatibilidad entre elementos embarcados en la aeronave, entre las distintas cargas de pago que soportará el avión, ya que puede ser un problema y grave si alguno de los elementos del sistema de visión interfiere en la misión de otro sistema embarcado, como puede ser el autopiloto por radiación electromagnética, o el motor. O si por el contrario, esos otros elementos interfieran en la operación del sistema de visión. Por eso, también hay que estudiar los elementos que van embarcados en esta aeronave y la interacción con el sistema de visión artificial, para ver si son físicamente compatibles.

La compatibilidad entre el segmento de tierra y el embarcado viene determinada por las posibles interacciones electromagnéticas entre los módems, la estación de radiocontrol y los propios procesadores. Hay que tener mucho cuidado con las distintas frecuencias a las que emiten los módems y la estación de radiocontrol, ya que puede interferir las dos señales y perderse información.

Además, hay otra fase de compatibilidad que hay que tener en cuenta, que es una limitación física. Es el viaje que debe hacer la información entre módems. Hay que tener cuidado con los posibles obstáculos que se presenten durante el vuelo que puedan apantallar la señal y evite la conexión; y hay que tener en cuenta también la longitud del viaje, ya que para emitir la señal a posiciones muy alejadas del avión, hay que colocar una antena muy potente para que la señal llegue con suficiente calidad para ser recibida. Esa es una de las compatibilidades que es más crítica.

Como se ve, las compatibilidades del sistema son muy importantes para que el sistema no se caiga en vuelo y no se requiera del control manual de emergencia. En el futuro, una de las tareas a realizar es la selección del sistema de emisión y recepción de datos conforme a estas compatibilidades y al resto de características de los diferentes componentes que el mercado ofrece para satisfacer esta necesidad.

Capítulo 11

Conclusiones y Trabajos Futuros

Es este último capítulo se especifican las conclusiones del proyecto y las líneas de trabajo futuras en las que avanzar en los posteriores cursos. Esas líneas de trabajo se fundamentaran en el trabajo realizado durante estos últimos meses y que concluyeron con la redacción de este proyecto.

En el capítulo de resultados, se ve cómo el programa realiza la detección de bordes de pista de manera bastante eficiente, robusta y eficaz, pues al ensayar variando varios parámetros del sistema, el sistema sigue funcionando de manera correcta. Eso nos lleva a comprobar la fiabilidad y la robustez del algoritmo de búsqueda que se ha desarrollado durante varios meses con el trabajo en las oficinas de AERNNOVA ENGINEERING SOLUTIONS IBÉRICA.

También se ha enseñado en los diferentes capítulos cómo se han ido resolviendo los diferentes problemas que han ido apareciendo durante el período de beca en el desarrollo del proyecto y cómo se han ido cumpliendo uno tras otro los objetivos primordiales del proyecto. Estos objetivos eran, como se ha indicado, el desarrollo de una aplicación viable de un sistema de visión por computador, el cual sirva como complemento para la ayuda al aterrizaje de una aeronave no tripulada; el estudio de las posibles interacciones con una estación de tierra y la propuesta de una integración del sistema dentro de la aeronave Mugin en la que la cámara se sitúa en la parte superior del morro de la aeronave y el computador se integra dentro del fuselaje, anclándose a él mediante patillas.

Además se ha resuelto un objetivo secundario que no se había fijado en el inicio del proyecto, que es la implementación del sistema operativo en la placa Gumstix Overo de arquitectura ARM. Ese objetivo forma parte de un technichal report de otro proyecto fin de carrera de la Universidad de Granada.

Los trabajos futuros a realizar se organizan en cuatro grupos que a continuación se exponen.

1. Completar la implementación satisfactoria del programa de detección de bordes de pista en el procesador Gumstix Overo. Actualmente, la placa Gumstix Overo con la distribución Angstrom permite compilar programas, desde terminal, que lleven la librería de OpenCV y tiene detectada la cámara con su driver Linux. Por lo tanto, únicamente hay que conseguir implementar el código que se desarrolló en este proyecto en un sistema operativo Windows XP en el computador Gumtix Overo, con los cambios necesarios para compilar en Linux. Esos cambios son sustanciales, ya que GCC en UNIX no compila el programa si genera avisos de incompatibilidades en el código, aunque el código no tenga errores. Además UNIX trabaja de manera distinta con los datos dispuestos en arrays y con las rutas de los sistemas de archivos.

2. Llevar a cabo la propuesta de integración en el UAS Mugin, corrigiendo los posibles errores que se han podido cometer en la definición de esa integración, así como comprobar que las patillas en las que se anclará la placa se pueden fijar de forma idónea a la aeronave

3. Realizar las pruebas del sistema en vuelo real en la aeronave. Una vez integrado en la aeronave, uno de los trabajos futuros es el ensayo del sistema en vuelo. Se puede realizar los ensayos anteriores a los embarcados en vuelo, en una plataforma radiocontrolada como puede ser un coche teledirigido, para comprobar si el sistema responde de manera correcta y con la fiabilidad necesaria como para embarcar el sistema sin riesgos para la aeronave.

4. Realizar una mejora continua del sistema. Se han avanzado diversas teorías que pueden ayudar a realizar de una manera más sencilla la implementación del algoritmo. Además, se puede mejorar tanto dicho algoritmo como el programa a implementar en la placa ARM.

Como se ha indicado y como conclusión final, se establecen cumplidos de manera satisfactoria los objetivos principales del proyecto.

Apéndices

Apéndice A

Anexo de Resultados

En este anexo, se incorporan los resultados de los diferentes ensayos y simulaciones que se han ido realizando para la comprobación del algoritmo y su implementación en el programa que realizara el tratamiento de imagénes

Se ha realizado este anexo para no cargar al lector en el capítulo de resultados de numerosas imágenes muy similares entre sí. Además, servirá para corroborar la robustez del algoritmo y la eficacia del programa.

Este anexo se dividirá en tres secciones, que son los tres programas donde se comprobaba cada parte del algoritmo. Además, la última sección se subdivide en dos, una para los ensayos sobre imágenes y otro para los ensayos del programa al ser ejecutado sobre videos.

A.1. Detector de Horizonte

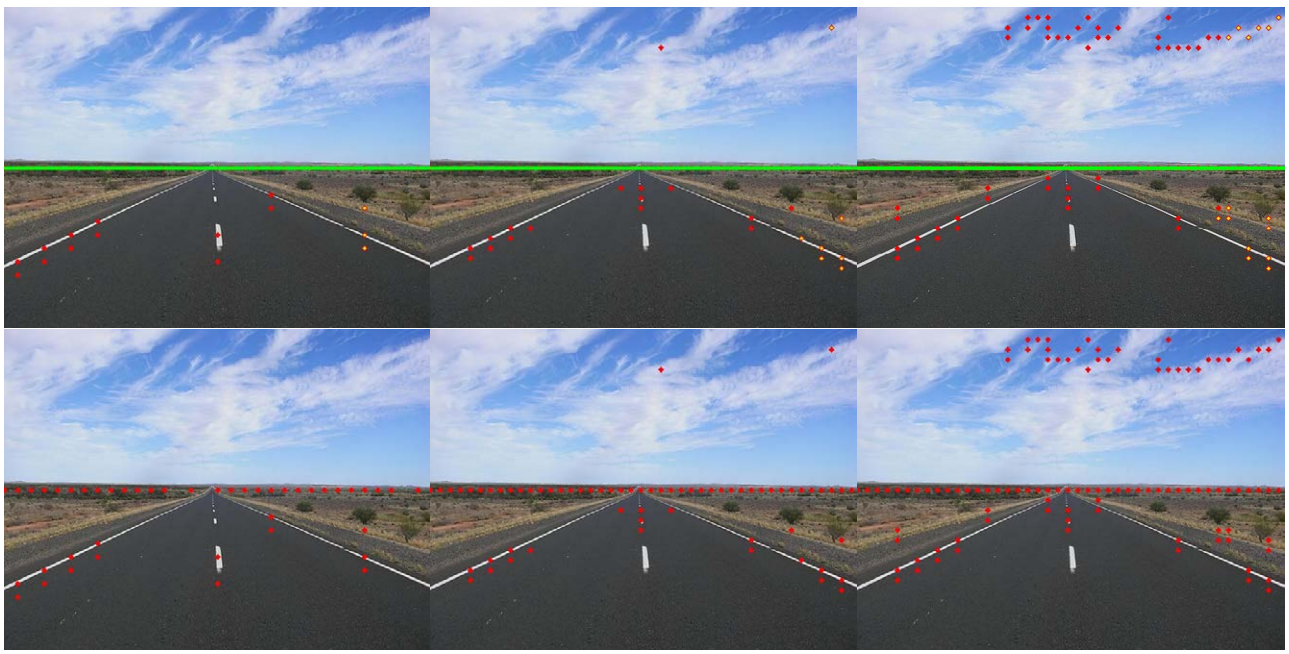


Figura A.1: Ensayo 1A-2A-3A Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 1A: Nivel de Segmentación: 175; Nivel del comparador: 80; Paso:20
Ensayo 2A: Nivel de Segmentación: 175; Nivel del comparador: 70; Paso:15
Ensayo 3A: Nivel de Segmentación: 135; Nivel del comparador: 70; Paso:15

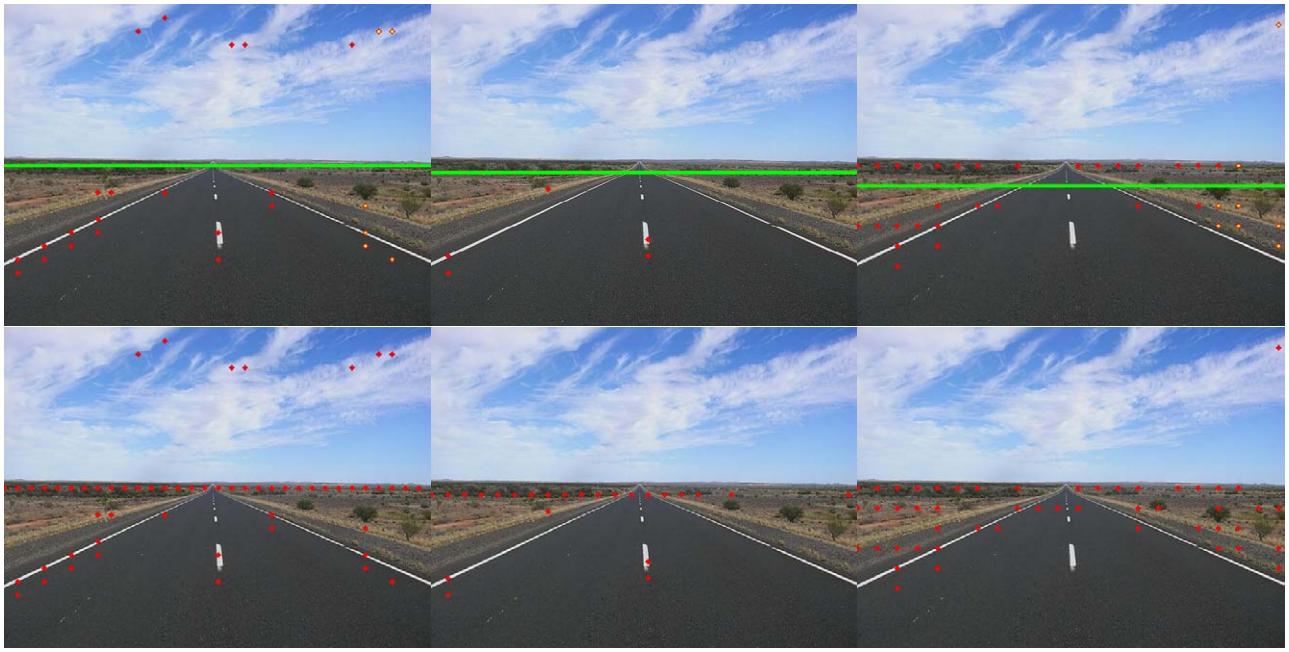


Figura A.2: Ensayo 4-5-6 Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 4A: Nivel de Segmentación: 175; Nivel del comparador: 60; Paso:20
Ensayo 5A: Nivel de Segmentación: 175; Nivel del comparador: 100; Paso:25
Ensayo 6A: Nivel de Segmentación: 100; Nivel del comparador: 90; Paso:30

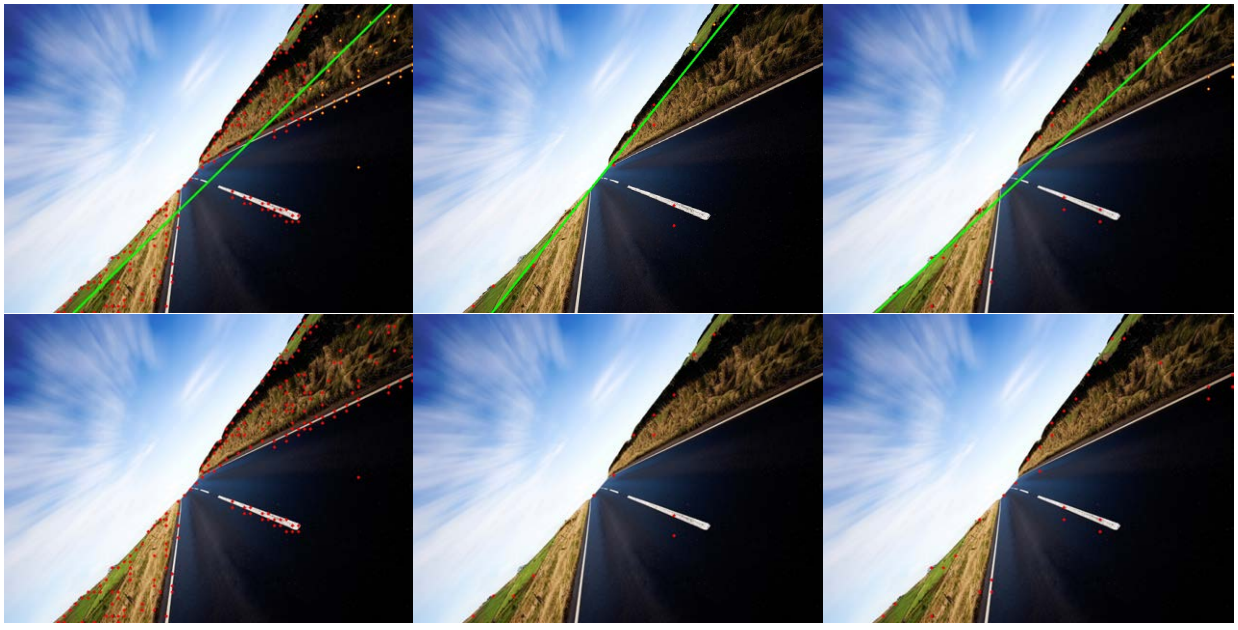


Figura A.3: Ensayo 7A-8A-9A Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 7A: Nivel de Segmentación: 175; Nivel del comparador: 50; Paso:15
Ensayo 8A: Nivel de Segmentación: 175; Nivel del comparador: 110; Paso:50
Ensayo 9A: Nivel de Segmentación: 100; Nivel del comparador: 90; Paso:30

El detector de horizonte se probó en varias imágenes. Los ensayos tuvieron una salida que se expone en este anexo.

A.2. Detector de Bordes de Pista

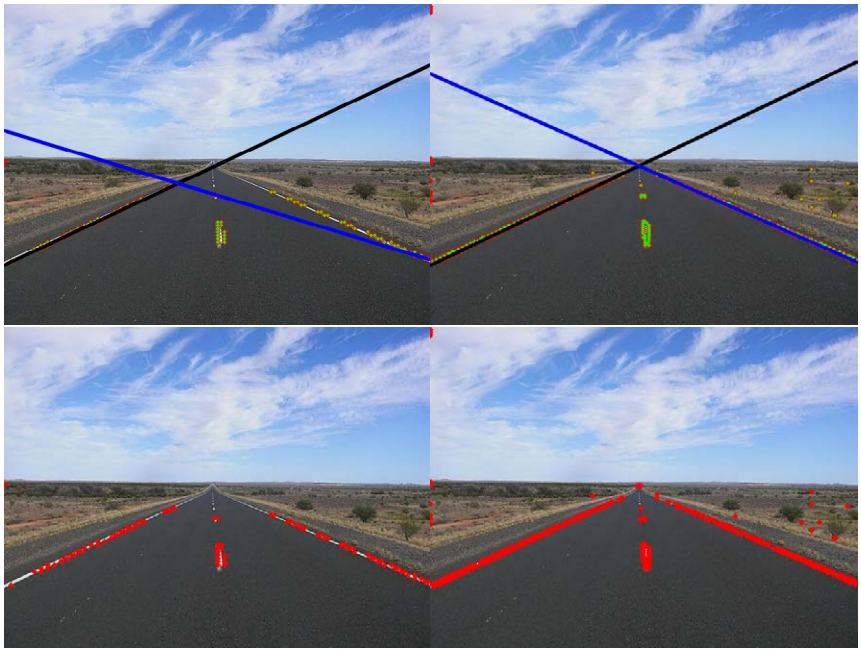


Figura A.4: Ensayo 1B-2B Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 1B: Nivel de Segmentación: 175; Nivel del comparador: 120; Paso:4
Ensayo 2B: Nivel de Segmentación: 175; Nivel del comparador: 60; Paso:3

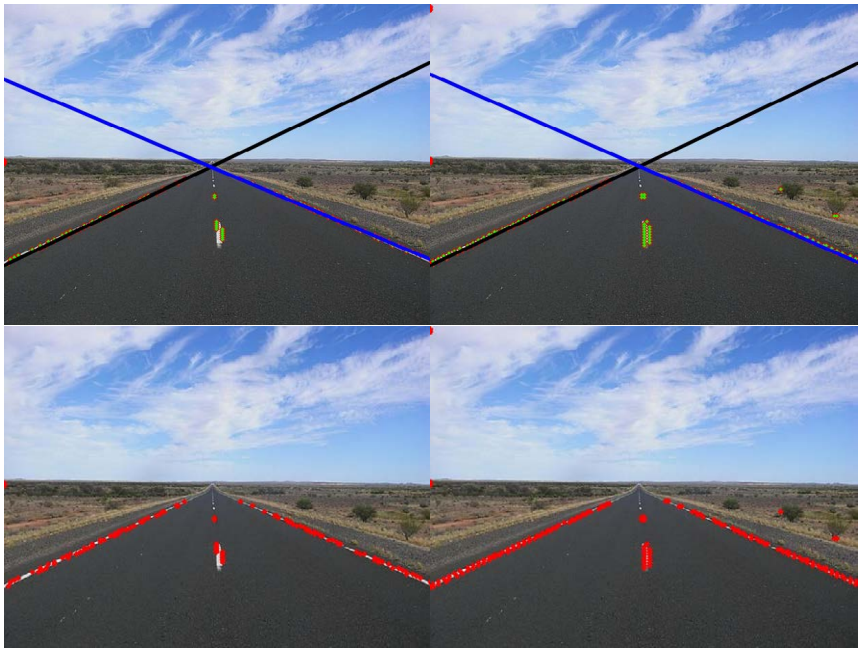


Figura A.5: Ensayo 3B-4B Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 3B: Nivel de Segmentación: 175; Nivel del comparador: 120; Paso:3
Ensayo 4B: Nivel de Segmentación: 175; Nivel del comparador: 80; Paso:4

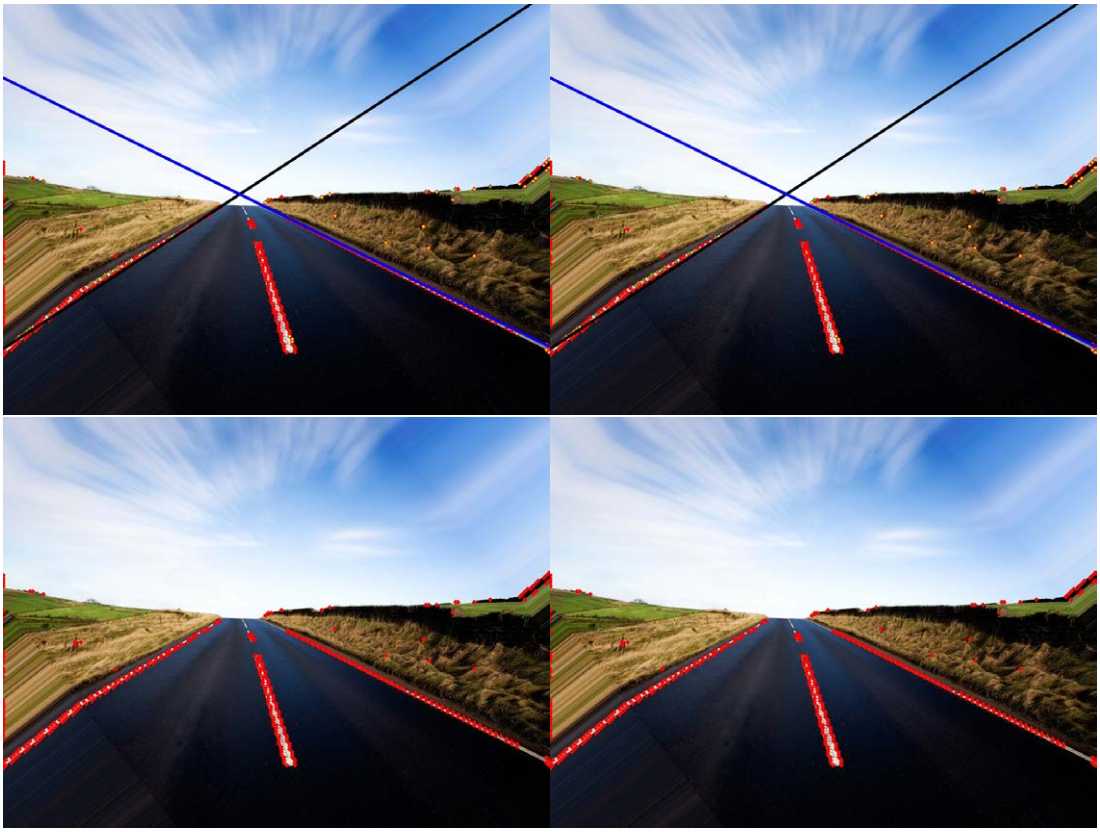


Figura A.6: Ensayo 5B-6B Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 5B: Nivel de Segmentación: 175; Nivel del comparador: 90; Paso:5
Ensayo 6B: Nivel de Segmentación: 175; Nivel del comparador: 110; Paso:5

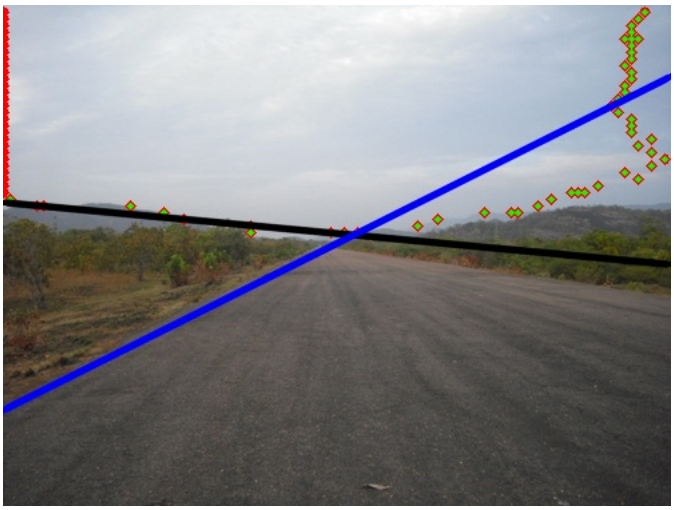


Figura A.7: Ensayo 7B Salida del Detector
Ensayo 7B: Nivel de Segmentación: 175; Nivel del comparador: 90; Paso:5

El detector de bordes de pista se probó en varias imágenes. Los ensayos tuvieron una salida que se expone en este anexo.

A.3. Detector de Horizonte y de Bordes de Pista

A.3.1. Detector funcionando sobre Imágenes

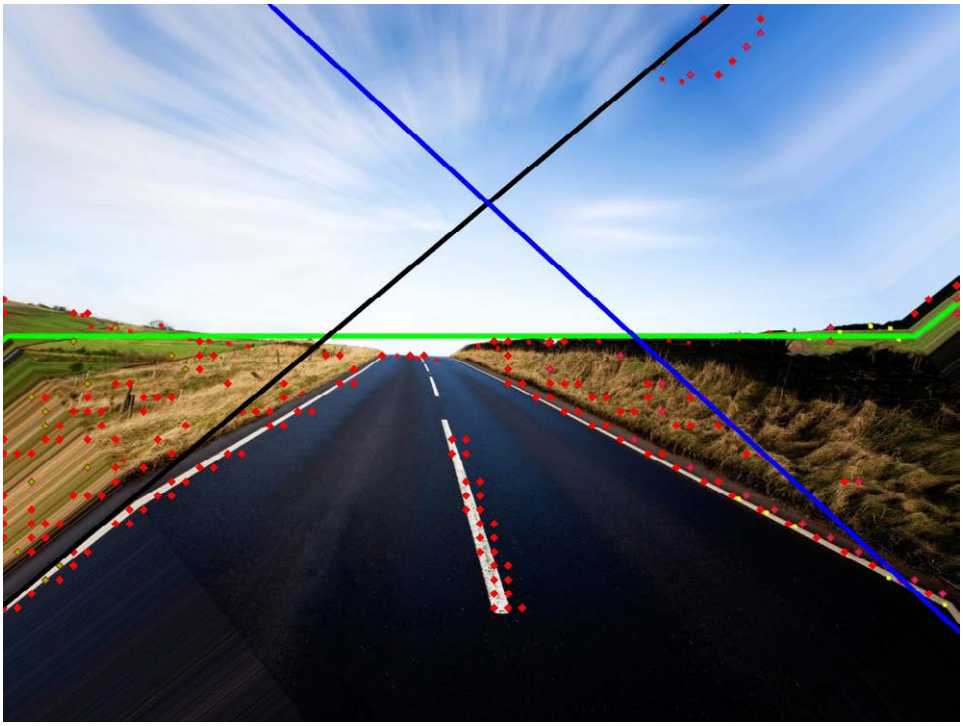


Figura A.8: Ensayo 1C. Salida del Detector
Ensayo 1C: Nivel de Segmentación: 110; Nivel del comparador de horizonte: 60; Nivel del comparador de bordes:95 ; Paso:15

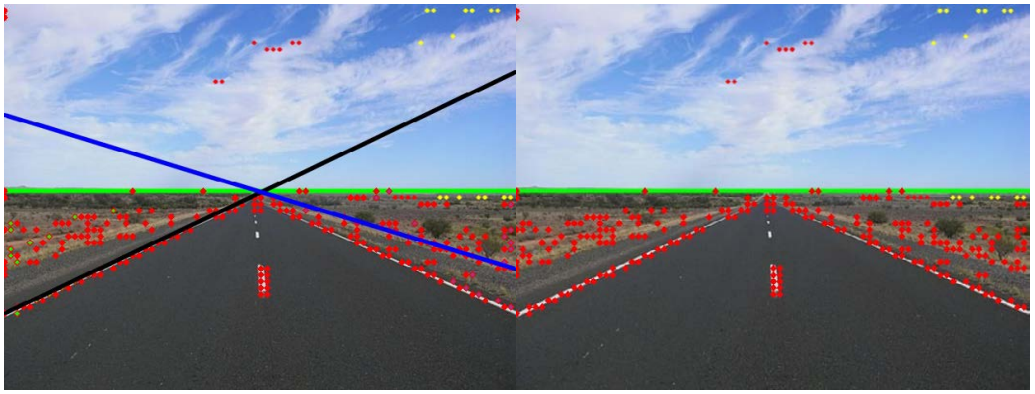


Figura A.9: Ensayo 2C. Derecha: Salida del Detector. Izquierda: Puntos Detectados
Ensayo 2C: Nivel de Segmentación: 110; Nivel del comparador de horizonte: 50; Nivel del comparador de bordes: 55; Paso: 8.

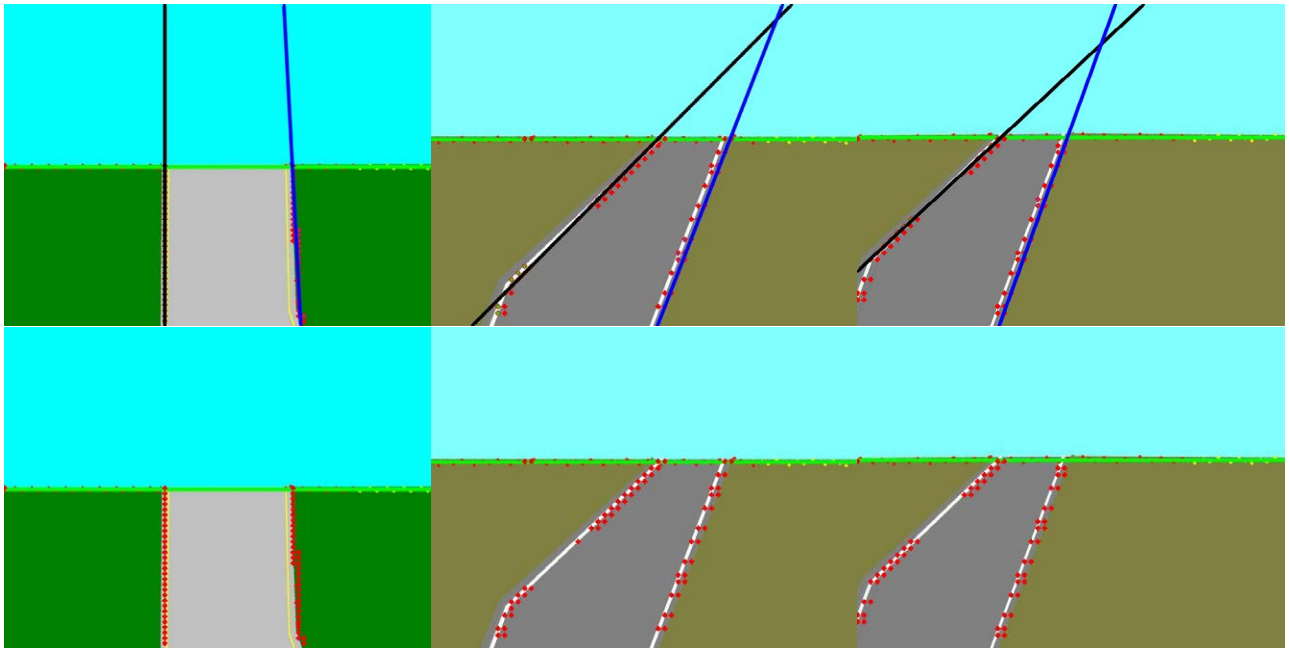


Figura A.10: Ensayo 3C-4C-5C Arriba: Salida del Detector. Abajo: Puntos Detectados
Ensayo 3C: Nivel de Segmentación: 150; Nivel del comparador de horizonte: 45; Nivel del comparador de bordes: 80; Paso: 10.
Ensayo 4C: Nivel de Segmentación: 115; Nivel del comparador de horizonte: 50; Nivel del comparador de bordes: 95; Paso: 10.
Ensayo 2C: Nivel de Segmentación: 190; Nivel del comparador de horizonte: 50; Nivel del comparador de bordes: 95; Paso: 10.

El detector global se probó en varias imágenes. Los ensayos tuvieron una salida de resultados que se expone en este anexo.

A.3.2. Detector funcionando sobre Vídeo

El detector global probado en vídeos se ensayó con distintos archivos de vídeo. Los ensayos tuvieron los resultados que se exponen en este apartado.

Se exponen diferentes fotogramas de las salida de vídeo de distintos ensayos. En los ensayos en los que no se pudo realizar un vídeo de la salida del programa se expone una salida de la pantalla, donde se ve además de la salida del detector, la imagen segmentada, los puntos detectados y la salida parcial del detector de horizonte. Los ensayos de vídeo se hicieron todos con las condiciones que se especifican a a continuación, con diferentes entradas de archivos de vídeo.

Esas condiciones son: Paso: 10

Nivel de Segmentación: 165

Nivel del detector de horizonte: 50

Nivel del detector de bordes: 35

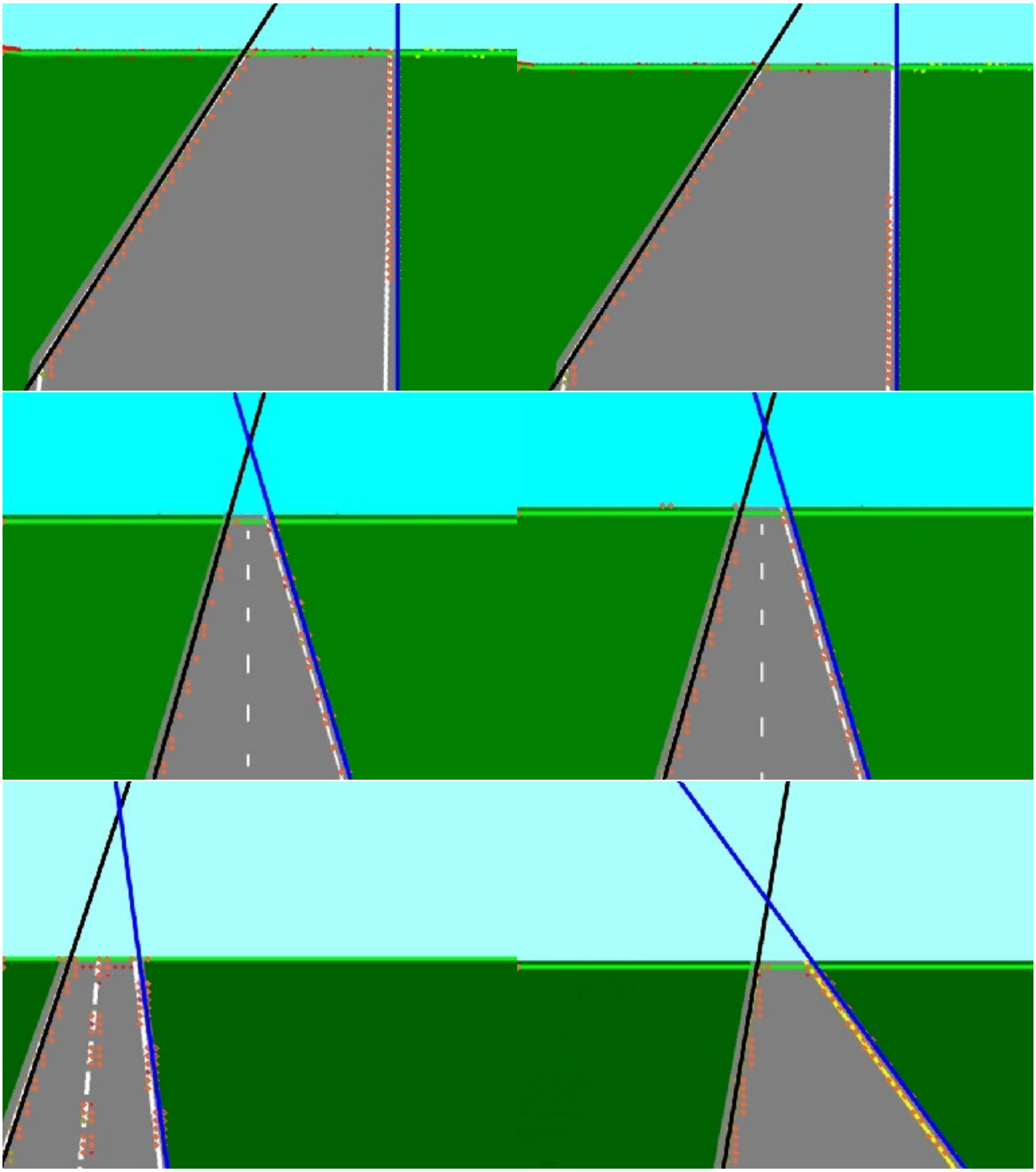


Figura A.11: Ensayo 1D

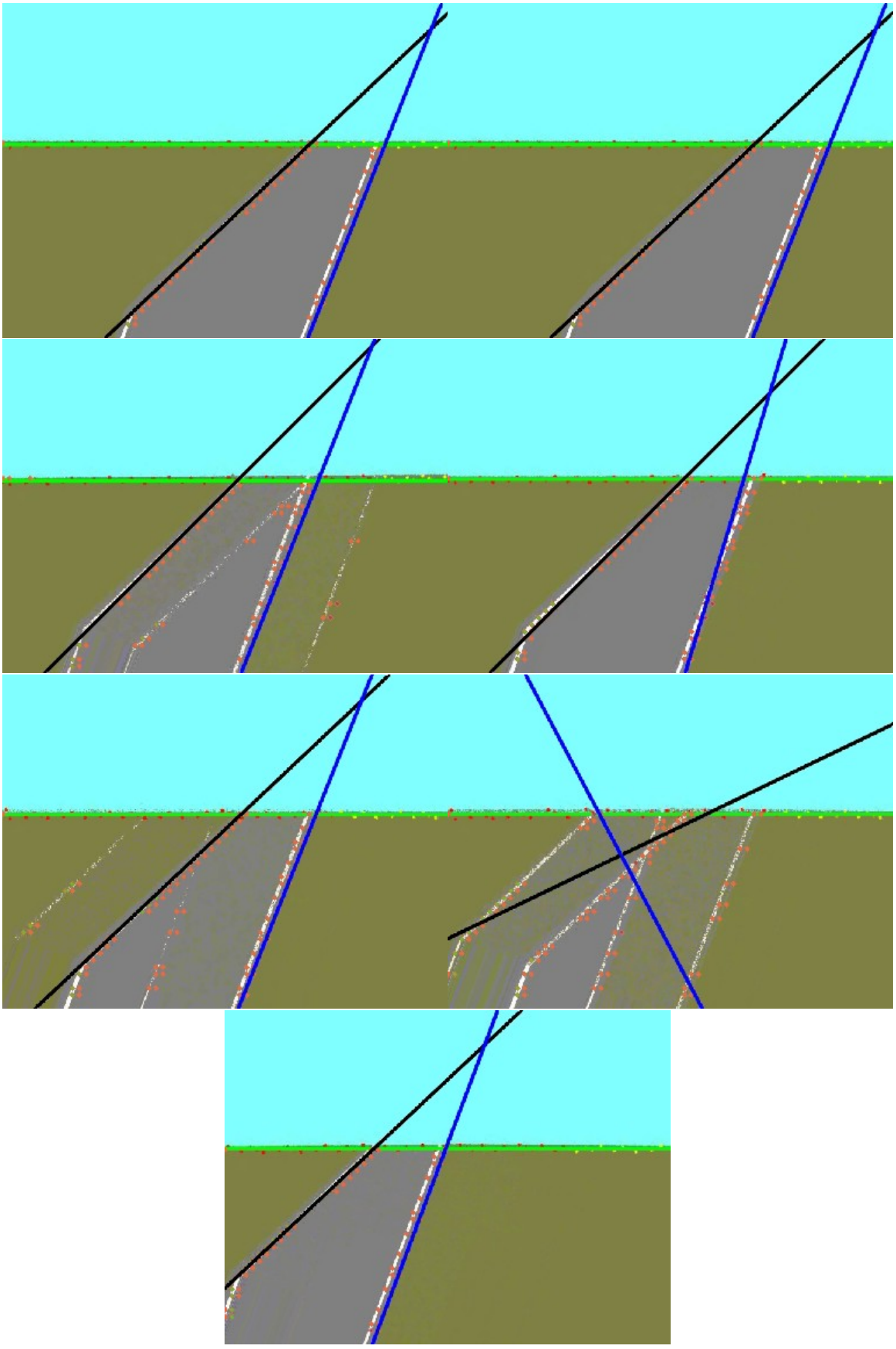


Figura A.12: Ensayo 2D

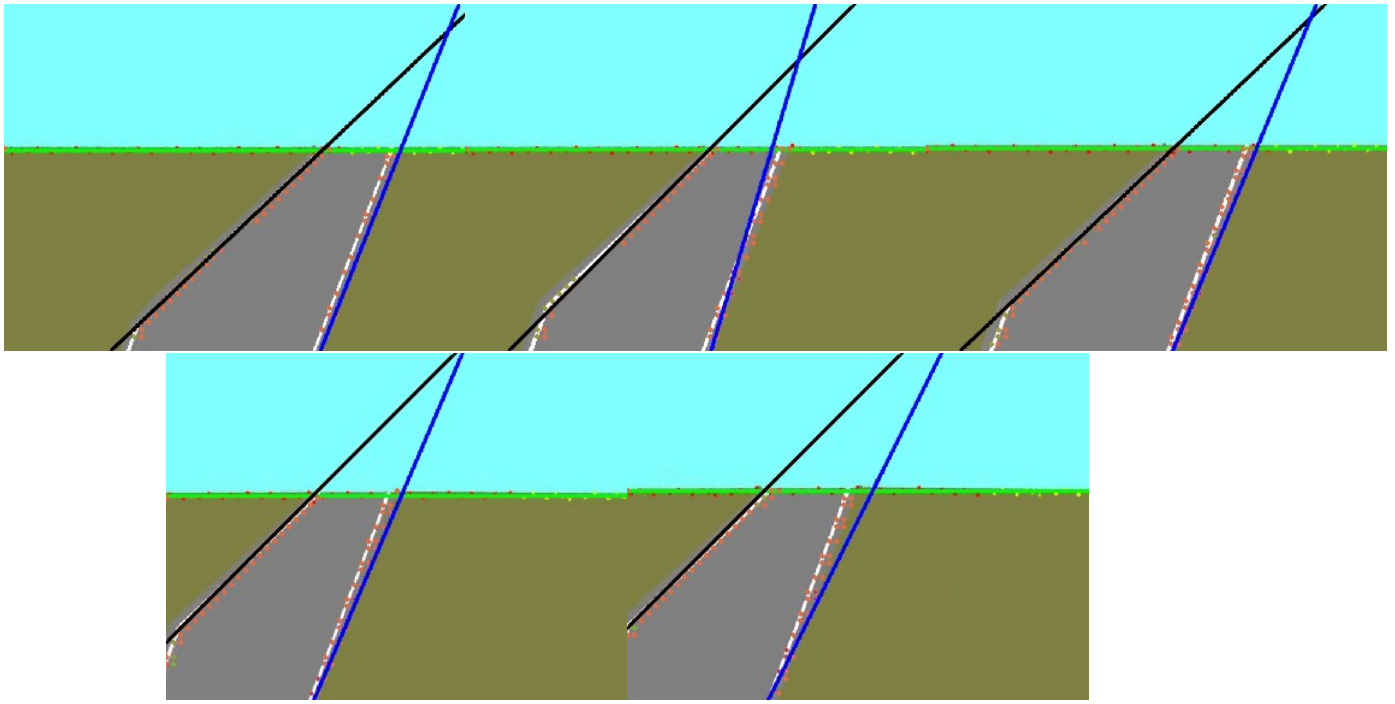


Figura A.13: Ensayo 3D

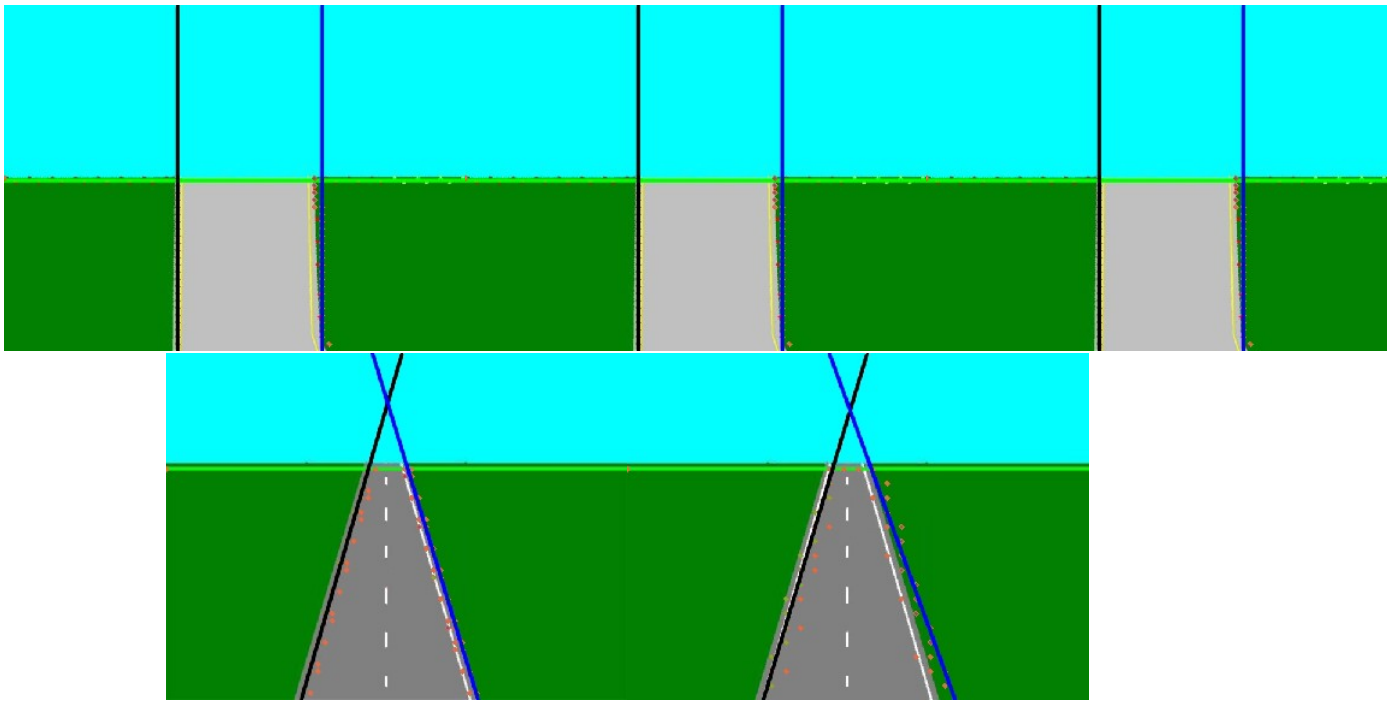


Figura A.14: Ensayo 4D

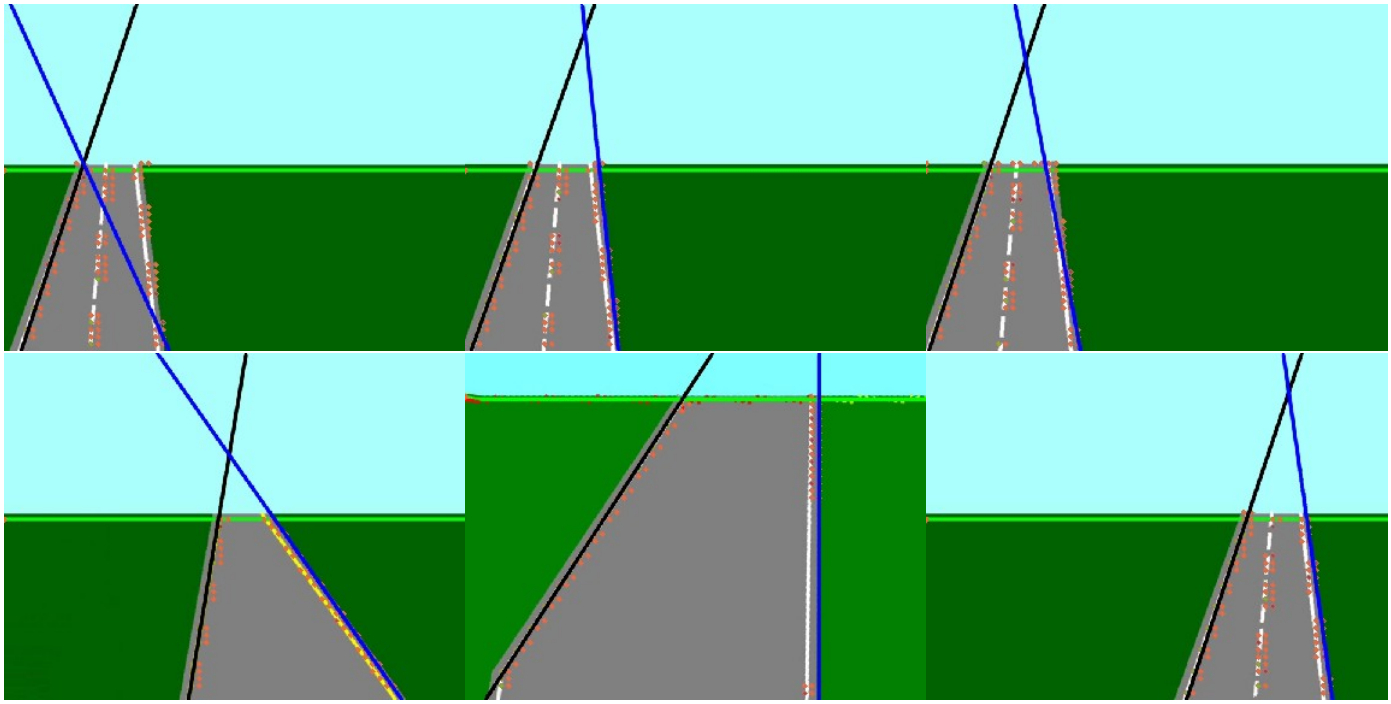


Figura A.15: Ensayo 5D

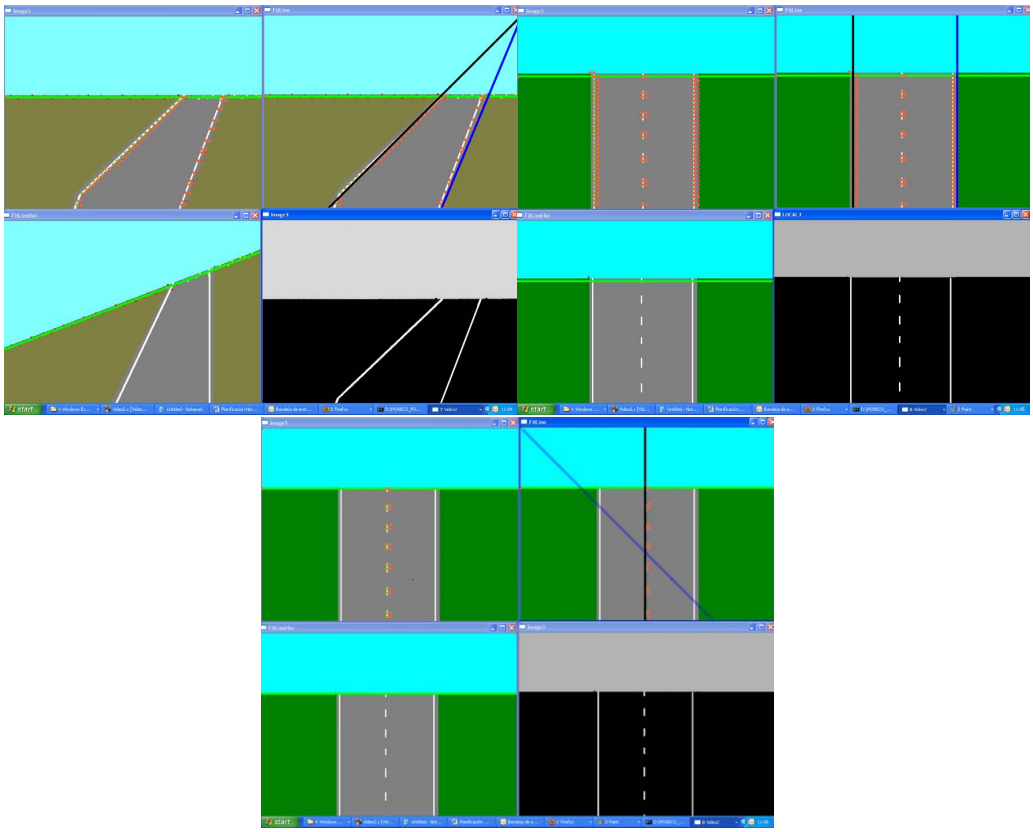


Figura A.16: Ensayo 6D

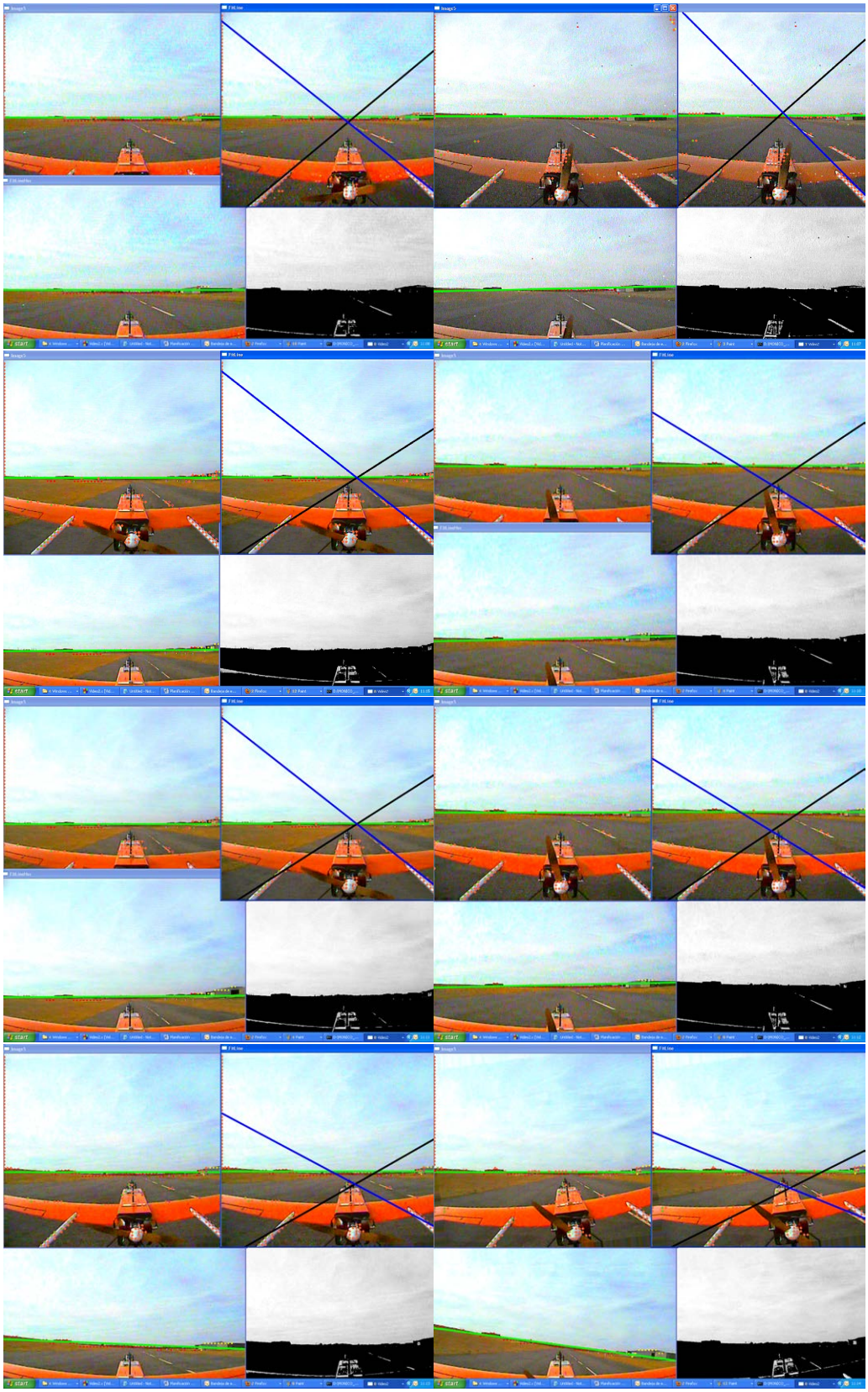


Figura A.17: Ensayo 7D

Este ensayo se realizó con un vídeo real de la aeronave Mugin con la cámara en la parte trasera del avión. Se comprueba que la colocación de la cámara no es la adecuada para el cometido de este proyecto.

Apéndice B

Cronología de los Lenguajes de Programación

En este anexo se indicará una cronología por los diferentes lenguajes de programación y sus creadores.

Se dividirán aproximadamente en décadas para una búsqueda más rápida y sencilla.

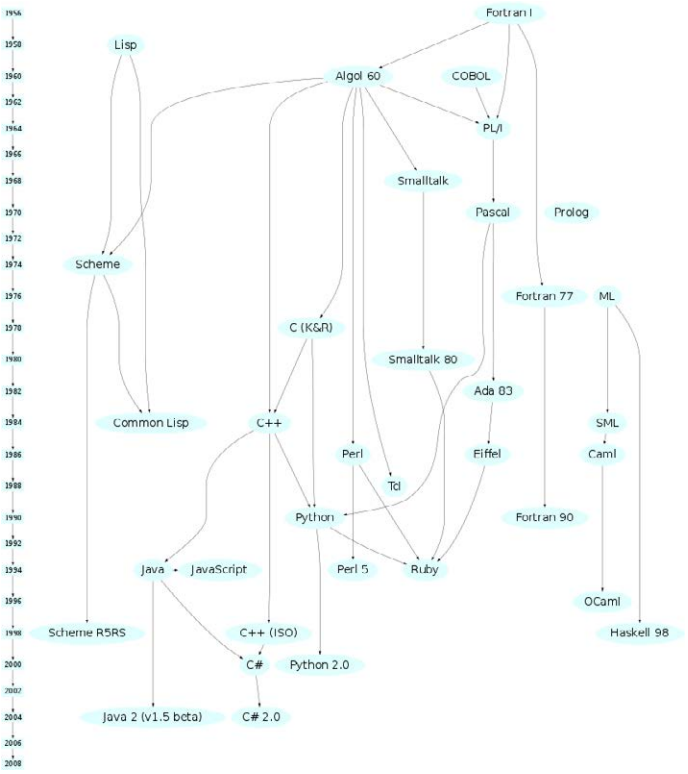


Figura B.1: Cronologia de los principales lenguajes de programación (Hasta 2008)

Además, con esta cronología se puede tener una idea de que lenguaje es la base de los siguientes.

Año	Lenguaje de Programación	Desarrollador/Creador
1840	PRIMER PROGRAMA	Ada Lovelace
1945	PLANKALKÜL	Konrad Zuse
1952	A-0	Grace Hopper
1954	MARK I AUTOCODE	Tony Brooker
1954-1955	FORTRAN "0"	John W. Backus (IBM)
1954	ARITH-MATIC	Grace Hooper
1954	IPL V	Allen Newell, Cliff Shaw y Herbert Simon
1955	FLOW-MATIC	Grace Hooper
1956-1958	LISP	John McCarthy
1957	COMTRAN	Bob Bemer
1957	FORTRAN "I"	John W. Backus (IBM)
1957	COMIT	
1958	FORTRAN II	John W. Backus (IBM)
1958	ALGOL 58	Esfuerzo Internacioal
1958	IPL V	Allen Newell, Cliff Shaw y Herbert Simon
1959	COBOL	Comité CODASYL
1959	LISP	John McCarthy
1959	TRAC	Calvin Mooers
1960	ALGOL 60	
1960	COBOL 61	Comité CODASYL
1961	COMIT	
1962	FORTRAN IV	
1962	APL	Kenneth E. Iverson
1962	MAD	Bruce Arden
1962	Simula	
1962-1963	SNOBOL	Ralph Griswold,
1963	SNOBOL3	Ralph Griswold,
1963	ALGOL 68	Adriaan van Wijngaarden
1963	JOSS I	Cliff Shaw (RAND)
1964	COWSEL	Rod Burstall, Robin Popplestone
1964	PL/1	IBM
1964	BASIC	John Kemeny y Thomas Kurtz
1964	TRAC	Calvin Mooers
1964	IITRAN	
1965	TELCOMP	Bolt, Beranek y Newman
1966	JOSS II	Chuck Baker (RAND Corporation)
1966	FORTRAN 66	
1966	ISWIM	Peter J. Landin
1966	CORAL66	
1967	BCPL	Martin Richards
1967	MUMPS	Massachusetts General Hospital
1967	APL	Kenneth E. Iverson
1967	SIMULA 67	Ole-Johan Dahl, Bjørn Myhrhaug, Kristen Nygaard (Norsk Regnesentral)
1967	SNOBOL4	Ralph Griswold, y otros
1967	XPL	W. M. Mckeeman, (Universidad de California Santa Cruz) Y J. J. Horning (Universidad de Stanford)

Cuadro B.1: Cronología Lenguajes de Programación (-1967)

Año	Lenguaje de Programación	Desarrollador/Creador
1968	POP-1	Rod Burstall, Robin Popplestone
1968	DIBOL-8	DEC
1968	Forth	Chuck Moore
1968	LOGO	Seymour Papert
1968	REFAL	Valentin Turchin
1969	ALGOL 68	Adriaan van Wijngaarden
1969	PL/1	IBM
1970	Forth	Chuck Moore
1970	POP-2	
1971	Pascal	Niklaus Wirth, Kathleen Jensen
1971	Sue	Holt (Universidad de Toronto)
1972	Smalltalk-72	Xerox PARC
1972	C	Dennis Ritchie
1972	INTERCAL	
1972	Prolog	Alain Colmerauer
1973	COMAL	Børge Christensen, Benedict Løfstedt
1973	LIS	Ichbiah (CII Honeywell Bull)
1974	GRASS	Tom DeFanti
1974	BASIC FOUR	BASIC FOUR CORPORATION
1975	Scheme	Gerald Jay Sussman, Guy L. Steele, Jr.
1975?	Modula	Niklaus Wirth
1975	Altair BASIC	Bill Gates, Paul Allen
1975	CS-4	Benjamin M. Brosgol (Intermetrics)
1976	Smalltalk-76	Xerox PARC
1976	Ratfor	Brian Kernighan
1977	FP	John Backus
1977	Bourne Shell (sh)	Stephen Bourne
1977	IDL	David Stern (Research Systems Inc)
1977	Standard MUMPS	
1977	ICON	Ralph Griswold
1977	Green	Ichbiah (CII Honeywell Bull)
		Departamento de Defensa de los Estados Unidos
1977	Red	Benjamin M. Brosgol (Intermetrics)
		Departamento de Defensa de los Estados Unidos
1977	Blue	Goodenough (SofTech)
		Departamento de Defensa de los Estados Unidos
1977	Yellow	Spitzen (SRI International)
		Departamento de Defensa de los Estados Unidos
1978	FORTRAN 77	
1978?	Modula-2	Niklaus Wirth
1978?	MATLAB	Moler (Universidad de Nuevo Mexico)
1978?	SMALL	Dr. Nevil Brownlee (Universidad de Auckland)
1978	VISICALC	Dan Bricklin, Bob Frankston en VisiCorp
1979	REXX	Mike Cowlishaw
1979	Awk	Aho, Weinberger, Brian Kernighan
1979	ICON	Ralph Griswold
1979	Vulcan dBase-II	Ratliff

Cuadro B.2: Cronología Lenguajes de Programación (1968-1979)

Año	Lenguaje de Programación	Desarrollador/Creador
1980	C con Clases	Bjarne Stroustrup
1980	Smalltalk-80	Xerox PARC
1982	Objective-C	Brad Cox
1983	Ada 83	Departamento de Defensa de los Estados Unidos
1983	C++	Bjarne Stroustrup
1983	True BASIC	John George Kemeny, Thomas Kurtz (Dartmouth College)
1984?	Korn Shell (ksh)	Dave Korn
1984	Standard ML	
1984	CLIPPER	Nantucket
1984	Common Lisp	Guy Steele
1985	1984 MUMPS	
1985	PARADOX	Borland
1985	PostScript	John Warnock
1985	QuickBASIC	Microsoft
1985	Miranda	David Turner (Universidad de Kent)
1986	LabVIEW	National Instruments
1985	Eiffel	Bertrand Meyer
1986	Informix-4GL	Informix
1986	PROMAL	
1987	Self	Sun Microsystems Inc.
1987	HyperTalk	Apple Computer
1987	SQL-87	
1987	Perl	Larry Wall
1988	Octave	
1988	dBase-IV	
1988	Tcl	John Ousterhout
1988	Object REXX	Simon Nash
1988	SPARK	Bernard A. Carré
1989	Turbo Pascal OOP	Borland
1989	Standard C89/90	ANSI X3.159-1989 (adopted by ISO in 1990)
1989	Modula-3	Cardeli, y otros
1989	Oberon	Niklaus Wirth
1989	VisSim	Peter A. Darnell

Cuadro B.3: Cronología Lenguajes de Programación (1980-1989)

Año	Lenguaje de Programación	Desarrollador/Creador
1990	Object Oberon	Niklaus Wirth
1990	J	Kenneth Iverson, Roger Hui (Iverson Software)
1990	Haskell	
1990	1990 MUMPS	
1991	Fortran 90	
1991	Oberon-2	Niklaus Wirth
1991	Python	Van Rossum
1991	Q	
1991	Visual Basic	Alan Cooper (Microsoft)
1992	SQL-92	
1992	Borland Pascal	
1993?	Z Shell (zsh)	
1993?	Self	Sun Microsystems Inc.
1993	FALSE	Oortmerrsen
1993	Brainfuck	Urban Müller
1993	Revolution Transcript	
1993	AppleScript	Apple
1993	K	Whitney
1993	Ruby	
1993	Lua	Roberto Ierusalimsch (Tecgraf, PUC-Rio)
1993	ZPL	Chamberlain (Universidad de Washington)
1994	Dylan	Apple Computer
1995	Ada 95	ISO
1995	Delphi	Anders Hejlsberg en Borland
1995	ColdFusion	Allaire
1995	Java	James Gosling (Sun Microsystems)
1995	1995 MUMPS	
1995?	LiveScript	Brendan Eich (Netscape)
1996	Fortran 95	
1996	Perl Data Language (PDL)	Karl Glazebrook, Jarle Brinchmann,
		Tuomas Lukka, y Christian Soeller
1996	NetREXX	Cowlishaw
1997?	JavaScript	Brendan Eich (Netscape)
1997	SML 97	
1997	PHP	
1997	Pico	Universidad Libre de Bruselas
1997	Squeak Smalltalk	Alan Kay (Apple Computer)
1997?	ECMAScript	ECMA TC39-TG1
1998	Standard C++	ANSI/ISO Standard C++
1998	Erlang	Open Source Erlang (Ericsson)
1999	Standard C99	ISO/IEC 9899:1999

Cuadro B.4: Cronología Lenguajes de Programación (1990-1999)

Año	Lenguaje de Programación	Desarrollador/Creador
2000	Join Java	G Stewart von Itzstein
2000	Joy	2000
2000	XL	Christophe de Dinechin
2000	C#	Anders Hejlsberg, Microsoft (ECMA)
2000	Ferite	Chris Ross
2001	AspectJ	Xerox PARC
2001	Processing	Casey Reas y Ben Fry
2001	Visual Basic .NET	Microsoft
2002	Io	Steve Dekorte
2003	Nemerle	Universidad de Wrocław
2003	Factor	Slava Pestov
2003	Scala	Martin Odersky
2003	Squirrel	Alberto Demichelis
2004	Subtext	Jonathan Edwards
2004	Alma-0	Krzysztof Apt (Centrum Wiskunde & Informatica)
2004	Boo	Rodrigo B. de Oliveira
2004	Groovy	James Strachan
2004	Little b	Aneil Mallavarapu (Harvard Medical School)
2005	F#	Don Syme (Microsoft Research)
2005	Seed7	Thomas Mertes
2006	Links	Philip Wadler (Universidad de Edinburgo)
2006	Cobra	ChuckEsterbrook
2006	Windows PowerShell	Microsoft
2006	OptimJ	Ateji
2007	Ada 2005	Ada Rapporteur Group
2007	Fantom	Brian Frank y Andy Frank
2007	Vala	GNOME
2007	Clojure	Rich Hickey
2007	Fortress	Guy Steele
2007	Oberon-07	Wirth
2008	Genie	Jamie McCracken
2008	Pure	Albert Gräf
2009	Go	Google
2009	CoffeeScript	Jeremy Ashkenas
2010	Fancy	Christopher Bertels
2010	Grace	James Noble, Kim Bruce y Andrew Black
2010	Rust	Graydon Hoare (Mozilla)
2010	Kotlin	JetBrains
2011	Dart	Google
2011	Ceylon	Gavin King (Red Hat)

Cuadro B.5: Cronología Lenguajes de Programación (2000-2011)

Bibliografía

[1] F. Adolf; "How-to build a cascade of boosted classifiers based on Haar-like features"; OpenCV's Rapid Object Detection; Sept. 2003.

[2] I. Blasco; "Algoritmos de detección de imágenes ruidosas y duplicadas"; Proyecto Fin de Carrera, Universidad Autónoma de Madrid; Dic. 2008.

[3] M.A. Sotelo, F.J. Rodríguez, L. Magdalena, L. Bergasa y L. Boquete; "A Color Vision-Based Lane Tracking System for Autonomous Driving on Unmarked Roads"; Autonomous Robots Vol. 16, Pags. 95–116; 2004.

[4] M. Euston, P. Coote, R. Mahony, J. Kim y T. Hamel, "A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV".

[5] R. Mahony, S.H. Cha y T. Hamel; "A coupled estimation and control analysis for attitude stabilisation of mini aerial vehicles"; Nov. 2006.

[6] N. Guenard, T. Hamel y R. Mahony; "A Practical Visual Servo Control for an Unmanned Aerial Vehicle"; Jun. 2010.

[7] S. Hutchsinson, G. Hager, P. Corke; "A tutorial on Visual Servo Control"; IEEE Transactions on Robotics and Automation, VOl. 12, Nº 5, Pags. 651-670; Oct. 1996.

[8] S. Fürst, E.D. Dickmanns; "A vision based navigation system for autonomous aircraft", Robotics and Autonomous Systems, Vol. 28,Pags. 173-184; 1999.

[9] C.S. Sharp, O. Shakernia y S.S. Sastry; "A Vision System for Landing an Unmanned Aerial Vehicle"; Proceedings of the 2001 IEEE International Conference on Robotics and Automation; Seul; May 2001.

[10] S. Huh, D.H. Shim; "A Vision-Based Automatic Landing Method for Fixed-Wing UAVs"; Journal of Intelligent and Robotic Systems, Vol. 57, Pags. 217–231; 2010.

[11] N.R. Gans, G. Hub, J. Shen Y. Zhang y W.E. Dixon; "Adaptive visual servo control to simultaneously stabilize image and pose error"; Mechatronics; 2011.

[12] K. Valavanis; "Advances in unmanned aerial vehicles, State of the Art and the road to Autonomy"; Springer; 2007.

[13] K. Terzidis; "Algorithms for visual desing using the processing language"; Wiley Publishing Inc.; 2009.

[14] E. Pastor, C. Barrado, P. Royo, J. Lopez y E. Santamaria; "An Architecture for the Development of Complex UAS Missions"; 2009.

[15] E. Pastor, C. Barrado, P. Royo, J. Lopez, X. Prats y E. Santamaria; "An Architecture for the Seamless Integration of UAS Remote Sensing Missions"; 2009.

[16] T.W. McLain, R.W. Beard, D. Blake Barber y N.B. Knoebel; "An Overview of MAV Research at Brigham Young University"; NATO UNCLASSIFIED; 2007.

[17] J. García de Jalón, J.I. Rodríguez, J.M. Sarriegui y A. Brazález; "Aprenda C++ como si estuviera en primero"; Escuela Superior de Ingenieros Industriales de San Sebastián, Universidad de Navarra; Abr. 1998.

[18] Anónimo, "Aprenda Qt4 desde hoy mismo"; Versión de distribución libre; Oct. 2010.

[19] Roke Manor Research Ltd.; "Autoland"; 2010.

[20] Roke Manor Research Ltd.; "Autolanding UAVs: Vision for the future"; 2010.

[21] L. Coutard, F. Chaumette y J.M. Pflimlin; "Automatic landing on aircraft carrier by visual servoing"; 2011.

[22] K.E. Wenzel, A. Masselli y A. Zell; "Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle"; 2011.

[23] W.E. Green y P.Y. Oh; "Autonomous Hovering of a Fixed-Wing Micro Air Vehicle"; Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida; May. 2006.

[24] S. Rathinam, P. Almeida, Z. Kim, S. Jackson, A. Tinka, W. Grossman y R. Sengupta; "Autonomous Searching and Tracking of a River using an UAV"; Proceedings of the 2007 American Control Conference, Marriott Marquis Hotel at Times Square, New York City; Jul. 2007.

[25] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain y M.A. Goodrich; "Autonomous Vehicle Technologies for Small Fixed-Wing UAVs"; Journal of aerospace computing, information, and communication, Vol. 2; Ene. 2005.

[26] T. Templeton, D.H. Shim, C. Geyer y S.S. Sastry; "Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft"; 2007.

[27] L. Mejías, J.F. Correa, I. Mondragón y P. Campoy; "COLIBRI: A vision-Guided UAV for Surveillance and Visual Inspection"; 2007 IEEE International Conference on Robotics and Automation, Roma; Abr. 2007.

[28] G. Baldwin, R. Mahony, J. Trumpf, T.Hamel y T. Cheviron; "Complementary filter design on the Special Euclidean group SE(3)".

[29] R. Szeliski; "Computer Vision: Algorithms and Applications"; Dic. 2009.

[30] P. Campoy, J.F. Correa, I. Mondragón, C. Martínez, M. Olivares, L. Mejías y J. Artieda; "Computer Vision onboard UAVs for civilian tasks"; 2008.

[31] L.O. Mejías; "Control Visual de un Vehículo Áereo Autónomo Usando Detección y Seguimiento de Características en Espacios Exteriores"; Tesis Doctoral, Universidad Politécnica de Madrid, Escuela Técnica Superior De Ingenieros Industriales; 2006.

[32] C. Ferraz; "Design of take-off and landing operational procedures for unmanned aerial vehicles"; Proyecto Fin de Carrera, Universitat Politécnica de Catalunya, Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació; Jul. 2009.

[33] J. Kim, D.W. Lee, K. Cho, S. Jo, J. Kim, C. Min, D. Han y S. Cho; "Development of an electro-optical system for small UAV"; Aerospace Science and Technology, Vol. 14, Pags. 505–511; Mar. 2010.

[34] R. Mahony; "Dynamic Image Based Visual Servo Control: Applications to Aerial Robotic Vehicles"; The Australian University; Nov. 2005.

[35] M. Hwangbo, J.Kuffner y T. Kanade; "Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs"; 2007.

[36] D.I.B. Randeniya, M. Gunaratne y S. Sarkar; "Fusion of Vision Inertial Data for Automatic Georeferencing"; Ago. 2008.

[37] R.M. Stallman y GCC Developer Community; "Using the GNU Compiler Collection"; GNU Press; Oct. 2003.

[38] M. Bertozzi y A. Broggi; "GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection"; IEEE Transactions on image processing, VOL. 7, Nº. 1, Pags. 62-81; Ene. 1998.

[39] T. Hamel y R. Mahony; "Image based visual servo control for a class of aerial robotic systems"; Automatica, Vol. 43, Pags. 1975–1983; 2007.

[40] R. Mahony y T. Hamel; "Image-Based Visual Servo Control of Aerial Robotic Systems Using Linear Image Features"; IEEE Transactions on Robotics, Vol. 21, Nº. 2, Pags. 227-239; Abr. 2005.

[41] J.R. Azinheira y P. Rives; "Image-Based Visual Servoing for vanishing features and Ground Lines tracking: application to a UAV automatic landing"; 2008.

[42] R. Mahony, A. Tayebi y T. Hamel; "Introduction to the special issue on aerial robotics" ;Control Engineering Practice, Vol. 18, Pags. 677–678; 2010.

[43] N. Correll, G. Sempo, Y. Lopez de Meneses, J. Halloy, J.L. Deneubourg y A. Martinoli; "SwisTrack: A Tracking Tool for Multi-Unit Robotic and Biological Systems"; 2006.

[44] O. Shakernia, Y. Ma, T.J. Koo y S. Sastry; "Landing an Unmanned air vehicle vision based motion estimation and nonlinear control"; Asian Journal of Control, Vol. 1, Nº. 3, Pags. 128-145; Sep. 1999.

[45] G. Bradski y A. Kaehler; "Learning OpenCV"; O'Reilly; 2008.

[46] J. Egbert y R.W. Beard; "Low-altitude road following using strap-down cameras on miniature air vehicles"; Mechatronics, Vol. 21, Pags. 831–843; 2011.

[47] D. Cook; "SwisTrack Manual"; Mar. 2010.

[48] NC State University Aerial Robotics Club; "ARCWulf Unmanned Aerial System"; May. 2011.

[49] R. Mahony, T. Hamel y J.M. Pflimlin; "Nonlinear Complementary Filters on the Special Orthogonal Group"; IEEE Transactions on Automatic Control, Vol. 53, Nº. 5, Pags. 1203-1218; Jun. 2008.

[50] J. Neira y J.D. Tardós; "Curso de Visión por Computador"; Universidad de Zaragoza.

[51] R.M. Caraballo; "Desarrollo de software para la ayuda en el aterrizaje autónomo mediante visión por computador"; Proyecto Fin de Carrera, Universidad de Sevilla, Ingeniería de Telecomunicaciones; Feb. 2010.

[52] F. Le Bras, T. Hamel, C. Barat y R. Mahony; "Nonlinear Image-Based Visual Servo controller for automatic landing guidance of a fixed-wing Aircraft"; 2009.

[53] R. Hewitt; "Seeing with OpenCV, A computer-Vision Library. Part 1"; Servo, Pags. 62-66; Ene. 2007.

[54] R. Hewitt; "Seeing with OpenCV, Finding faces in Images. Part 2"; Servo, Pags. 48-52; Feb. 2007.

[55] R. Hewitt; "Seeing with OpenCV, Follow that Face!. Part 3"; Servo, Pags. 36-40; Mar. 2007.

[56] R. Hewitt; "Seeing with OpenCV, Face recognition with eigenface. Part 4"; Servo, Pags. 36-39; Abr. 2007.

[57] R. Hewitt; "Seeing with OpenCV, Implementing eigenface. Part 5"; Servo, Pags. 44-50; May. 2007.

[58] E. Menegatti y T. Pajdla; "Omnidirectional robot vision "; Robotics and Autonomous Systems, Vol. 58, Pags. 745-746; 2010.

[59] I.F. Mondragón, P. Campoy, C. Martinez y M. Olivares; "Omnidirectional vision applied to Unmanned Aerial Vehicles (UAVs) attitude and heading estimation"; Robotics and Autonomous Systems, Vol. 58, Pags. 809-819; 2010.

[60] C. Martínez, I.F. Mondragón, M.A. Olivares-Méndez y P. Campoy;"On-board and Ground Visual Pose Estimation Techniques for UAV Control"; Journal of Intelligent and Robotic Systems Vol. 61, Pags. 301–320; 2011.

[61] I.F. Mondragón; "On-board visual control algorithms for Unmanned Aerial Vehicles";Tesis Doctoral, Universidad Politécnica de Madrid, Escuela Técnica Superior De Ingenieros Industriales; 2011.

[62] R. Laganiére; "OpenCV 2 Computer Vision Application Programming Cookbook"; Packt Publishing Ltd.; May. 2011.

[63] F. Kendoul,I. Fantoni y K. Nonami; "Optic Flow-Based Vision System for Autonomous 3D Localization and Control of Small Aerial Vehicles"; Robotics and Autonomous Systems, Vol. 57, Nº 6-7, Pags. 591-602; 2011.

[64] A. Beyeler, J.C. Zufferey y D. Floreano; "optiPilot: control of take-off and landing using optic flow"; 2009. VV.AA.; "Paparazzi User's Manual"; Ecole Nationale de l'Aviation Civile, Toulouse; Oct. 2007.

[65] L. Ibáñez, W. Schroeder, L. Ng, J. Cates y Insight Software Consortium; "The ITK Software Guide"; Ago. 2003.

[66] Dr. W.J. Crowther; "Perched Landing and Takeoff for Fixed Wing UAVs"; NATO UNCLASSIFEID; 2000.

[67] F. Zurbriggen; "PixHawk - Dynamic Model and Sensor Fusion"; Feb. 2010.

[68] VV.AA. "CodeBlocks 10.05 manual Version 1.1"; GNU Free Documentation License.

[69] F.J. Ceballos; "C/C++ : curso de programación "; RA-MA; 2007.

[70] H.M. Deitel; "Cómo programar en C/C++ y Java"; Pearson Educación de México; 2004.

[71] B. Thuilot, P. Martinet, L. Cordesses y J. Gallice; "Position based visual servoing : keeping the object in the field of vision"; Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC; May. 2002.

[72] X. Nus; "Projecte ICARUS – Sistema de navegació automàtic"; Proyecto Fin de Carrera, Universitat Politécnica de Catalunya, Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació; Feb. 2006.

[73] A. Miranda Neto, A. Corrêa Victorino, I. Fantoni y D.E. Zampieri; "Robust Horizon Finding Algorithm for Real-Time Autonomous Navigation based on Monocular Vision"; 14th International IEEE Conference on Intelligent Transportation Systems, Washington DC; ITSC 2011.

[74] L. Reguera; "Video Tracking"; Visión Computacional, Ingeniería Industrial, Universidad de León; Abr. 2010.

[75] M. Quigley, M.A. Goodrich y R.W. Beard; "Semi-Autonomous Human-UAV Interfaces for Fixed-Wing Mini-UAVs"; 2004.

[76] R. Feliu; "Sistema de finalització de vol per plataformes UAV"; Proyecto Fin de Carrera, Universitat Politécnica de Catalunya, Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació; Sep. 2009.

[77] J.M. Batlle; "Sistemes d'adquisició d'imatges aèries"; Proyecto Fin de Carrera, Universitat Politécnica de Catalunya, Enginyeria Tècnica de Telecomunicació, especialitat en Sistemes de Telecomunicació; Ene. 2008.

[78] S.S. Mehta, V. Jayaraman, T.F. Burks y W.E. Dixon; "Teach by zooming: A unified approach to visual servo control"; Mechatronics; 2011.

[79] K. Novins y B. McCane; "Teaching Computer Vision using Primary Source Material"; 2000.

[80] K. Kluge y C. Thorpe; "The YARF System for Vision-Based Road Following"; Mathematical Computer Modelling, Vol 2, Nº 4-7, Pags. 213-233; 1995.

[81] P.J. Garcia-Pardo, G.S. Sukhatme y J.F. Montgomery; "Towards vision-based safe landing for an autonomous helicopter"; 2001.

[82] R. Cañuelo; "Technical report nº 3: Instalación de Linux para ARM en sistemas empotrados"; Proyecto Fin de Carrera, Universidad de Granada; 2010.

[83] R. Igual y C. Medrano; "Tutorial de OpenCV"; GNU Free Documentation License; 2007.

[84] P. Martinet y E. Cervera; "Tutorial on Advanced Visual Servoing"; IEEE/RSJ IROS'04, Sendai; Sep. 2004.

[85] F. Chaumette; "Tutorial on Advanced Visual Servoing"; IEEE/RSJ IROS'04, Sendai; Sep. 2004.

[86] E. Malis; "Tutorial on Advanced Visual Servoing"; IEEE/RSJ IROS'04, Sendai; Sep. 2004.

[87] P. Martinet; "Tutorial on Advanced Visual Servoing"; IEEE/RSJ IROS'04, Sendai; Sep. 2004.

[88] S. Thurrowgood, R.J.D. Moore, D. Bland, D. Soccol y M.V. Srinivasan; "UAV Attitude Control using the Visual Horizon".

[89] W. Bath y J. Paxman; "UAV Localisation & Control Through Computer Vision".

[90] I. Mondragon, M.A. Olivares, P. Campoy, C. Martinez y L. Mejias; "Unmanned Aerial Vehicles UAVs attitude, height, motion estimation and control using visual systems"; Autonomous Robots; 2010.

[91] J.D. Blom; "Unmanned Aerial Systems: A Historical Perspective"; Combat Studies Institue Press; 2010.

[92] P. Baker y B. Kamgar-Parsi; "Using shorelines for autonomous air vehicle guidance";Computer Vision and Image Understanding, Vol. 114, Pags. 723–729; 2010.

[93] B. Sinopoli, M. Micheli, G. Donato y T.J. Koo; "Vision Based Navigation for an Unmanned Aerial Vehicle";Proceedings of the IEEE International Conference on Robotics an Automation, Seul; May. 2001.

[94] O. Shakernia y Y. Ma; "Vision Guided Landing of an Unmanned Air Vehicle"; Proceedings of the 38th Conference on Decision and Control; Dic. 1999.

[95] J.F. Vélez, A.B. Moreno, A. Sánchez y J.L. Estebán; "Vision Por Computador"; 2003.

[96] A. Hernández; "Visión Artificial".

[97] T.F. Gonçalves, J.R. Azinheira y P. Rives; "Vision-based Autonomous approach and landing for an aircraft using a direct visual tracking method"; 2009.

[98] S. Saripalli, J.F. Montgomery y G.S. Sukhatme; "Vision-based Autonomous Landing of an Unmanned Aerial Vehicle"; 2002.

[99] R. Cunha, C. Silvestre, J. Hespanha y A. Pedro Aguiar; "Vision-based control for rigid body stabilization"; Automatica, Vol. 47, Pags. 1020–1027; 2011.

[100] J. Courbon, Y. Mezouar, N. Guénard y P. Martinet; "Vision-based navigation of unmanned aerial vehicles"; Control Engineering Practice, Vol. 18, Pags. 789–799; 2010.

[101] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial y R. Sengupta; "Vision-Based Road-Following Using a Small Autonomous Aircraft"; 2004.

[102] J. Shang y Z. Shi; "Vision-based Runway Recognition for UAV Autonomous Landing"; International Journal of Computer Science and Network Security, Vol.7 Nº.3, Pags. 112-117; Mar. 2007

[103] J.R. Azinheira y P. Rives; "Visual Auto-landing of an Autonomous Aircraft"; INRIA Rapport de Recherche Nº 4606; Nov. 2002.

[104] L. Coutard y F. Chaumette; "Visual detection and 3D model-based tracking for landing on an aircraft carrier"; 2011.

[105] F. Chaumette y S. Hutchinson; "Visual Servo Control Part I: Basic Approaches"; IEEE Robotics and Automation Magazine, Vol. 13, Nº 4, Pags. 82-90; 2006.

[106] F. Chaumette y S. Hutchinson; "Visual Servo Control Part II: Advanced Approaches"; IEEE Robotics and Automation Magazine, Vol. 14, Nº 1, Pags. 109-118; 2007.

[107] O. Bourquardez y F. Chaumette; "Visual Servoing of an Airplane for Alignment with respect to a Runway"; 2007.

[108] O. Bourquardez y F. Chaumette; "Visual Servoing of an Airplane for Auto-Landing"; 2007.

[109] R.F. Fung; "Visual Servoing"; Intech; 2010.

[110] S. Saripalli, J.F. Montgomery y G.S. Sukhatme; "Visually-Guided Landing of an Unmanned Aerial Vehicle" ; IEE Transactions on Robotics and Automation; Vol. 19; Nº 3; Pags. 371-381; Jun. 2003.

[111] B. Stroustrup; "The C++ Programming Language"; AT&T LAbs; Addison-Wesley Publishing Company; Jun. 1997.

[Web01] http://opencv.itseez.com/

[Web02] http://opencv.willowgarage.com/wiki/

[Web03] http://www.softintegration.com/products/thirdparty/opencv/

[Web04] http://www.ros.org/wiki/ROS http://www.computervisiononline.com/software

[Web05] http://www.tina-vision.net/

[Web06] http://www.simplecv.org/

[Web07] http://www.vtk.org/VTK/resources/relatedsoftware.html

[Web08] http://www.emgu.com/wiki/index.php/Main_Page

[Web09] http://www.roborealm.com/links/vision_software.php

[Web10] http://webdiis.unizar.es/ %7Eneira/vision.html

[Web11] http://www.dia.fi.upm.es/~lbaumela/doctorado/

[Web12] http://dis.um.es/ %7Eginesgm/pivc.html

[Web13] http://beagleboard.org/

[Web14] https://pixhawk.ethz.ch/

[Web15] http://www.arm.com/products/processors/cortex-r/index.php

[Web16] http://www.raspberrypi.org/

[Web17] https://www.gumstix.com/store/product_info.php?products_id=254

[Web18] https://www.gumstix.com/store/product_info.php?products_id=230

[Web19] https://www.gumstix.com/store/product_info.php?products_id=268

[Web20] http://www.ids-imaging.de/

[Web21] http://www.ptgrey.com/products/chameleon/chameleon_usb_camera.asp

[Web21] http://www.arm.com/products/processors/index.php

[Web22] http://opencv.willowgarage.com/documentation/python/cookbook.html

[Web23] http://www.laganiere.name/

[Web24] http://www.alereimondo.com.ar/OpenCV

[Web25] http://ocw.unizar.es/ocw/ensenanzas-tecnicas/vision-por-ordenador/material-de-clase.html/

[Web26] http://homepages.inf.ed.ac.uk/rbf/CVonline/

[Web27] http://nashruddin.com/tag/opencv/3/

[Web28] http://staff.qut.edu.au/staff/corkep/

[Web29] http://roboticsconference.org/pmwiki/index.php

[Web30] http://www.vision4uav.com/

[Web31] http://www.icarus.upc.edu/

[Web32] http://www.irisa.fr/lagadic/welcome.html

[Web33] http://sopa.dis.ulpgc.es/so/cpp/intro_c/

[Web34] http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

[Web35] http://www.pages.drexel.edu/~nk752/tutorials.html

[Web36] http://www.programacionfacil.com/cpp/start

[Web37] http://copterix.perso.rezel.net/

[Web38] http://en.wikipedia.org/

[Web39] http://es.wikipedia.org/

[Web40] http://paparazzi.enac.fr/

[Web41] http://pandaboard.org/content/platform

[Web42] http://www.linaro.org/

[Web43] http://www.origenboard.org/Store

[Web44] http://wiki.gumstix.org/index.php?title=Category:How_to_-_linux

[Web45] http://wiki.gumstix.org/index.php?title=Installing_Ubuntu_10.04_on_Gumstix_Overo

[Web46] http://www.aishack.in/topics/tutorials/vision/

[Web47] http://blog.damiles.com/2010/01/segmentation-object-detection-by-color/

[Web48] http://gumstix.org/

[Web49] http://wiki.gumstix.org/index.php?title=Category:Projects_-_robotics

[Web50] http://www.gumstix.org/hardware-design/overo-coms/75-overview-and-roadmap/67-overo-com-overview.html

[Web51] https://pixhawk.ethz.ch/electronics/camera

[Web52] https://www.ridgerun.com/developer/wiki/index.php/Gumstix_Overo_OMAP35x

[Web53] http://www.gumstix.org/create-a-bootable-microsd-card.html

[Web54] http://www.angstrom-distribution.org/

[Web55] http://gumstix.org/connect-to-my-gumstix-system.html

[Web56] http://narcissus.angstrom-distribution.org/

[Web57] gumstix.8.n6.nabble.com/

[Web58] http://gumstix.org/get-started/how-to.html

[Web59] http://forja.rediris.es/frs/download.php/1637/guia_beagle.pdf

[Web60] http://www.sciencedirect.com/

[Web61] http://www.elsevier.com/

[Web62] https://wiki.ubuntu.com/ARM/RootfsFromScratch

[Web63] http://www.codeblocks.org/

[Web64] http://paparazzi.enac.fr/wiki/Overview#System_Architecture

[Web65] http://paparazzi.enac.fr/wiki/DevGuide/Server_GCS_com

[Web66] http://paparazzi.enac.fr/wiki/Ivy