Universidad Complutense de Madrid

Facultad de Informática

Proyecto de Sistemas Informáticos 2011/2012

Control de un robot Móvil con visión: Rectificación de imágenes para agricultura de precisión

por

Manuel Sánchez Álvarez

Rafael Vaquero Gil

César Vázquez Montecino

Profesor Director: José Jaime Ruz Ortiz

Madrid, 2012

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Resumen

El tratamiento digital de imágenes en color es un campo en continua evolución. El objetivo de dicho tratamiento puede tener a su vez diferentes motivos: mejorar la propia calidad de la imagen, conseguir una representación más eficiente que permita almacenarla en menor espacio sin una pérdida apreciable de calidad, extraer información relevante de cara a interpretar su contenido y tomar decisiones al respecto, etc. En nuestro caso el procesamiento digital de la imagen está orientado a interpretar la información recibida a través de una cámara de video en tiempo real para detectar un color en particular y sus características (matiz, saturación, brillo).

Como se ha dicho previamente, el procesamiento de imágenes puede tener diferentes utilidades. Por ejemplo la de detectar el porcentaje de un tono de verde concreto en una imagen, puede servir para identificar malas hierbas en un entorno real de manera casi automática. Para un correcto procesamiento de esta imagen es necesario conocer la posición exacta de la lente de la cámara en sus tres ejes (x, y, z) y sus tres ángulos (*pitch*, *yaw* y *roll*) y así, posteriormente poder transformar la imagen en función de dicha posición, empleando para ello algoritmos específicos. Para ello hemos implementado distintas técnicas y algoritmos utilizando el lenguaje de programación C#.

Palabras clave: tratamiento de imágenes, procesamiento de imágenes, visión por ordenador, cámaras, transformaciones aplicadas a imágenes, cálculo de áreas, odometría, C#, HSV, HSB

Abstract

The digital processing of color images is a clearly evolving field. The aim of the processing may have different reasons: to improve the quality of the image itself, achieve a more efficient representation that allows storage in less space without losing much quality, extract relevant information to interpret its content in order to take decisions about it, and so on. In this case, digital image processing aims to interpret the information received through a video camera to detect a particular color and some of its properties (hue, saturation, brightness) in real time.

As mentioned previously, the image processing may have different utilities. That is, for instance, to detect the percentage of a particular shade of green in an image with the aim of identifying bad weeds in a real environment almost automatically. For a proper processing of the image is necessary to know the exact position of the camera lens in three axes (x, y, z) and in three angles (pitch, yaw and roll) to transform the image based on that position, employing specific algorithms. We've implemented different techniques and algorithms using the programming language C#.

Keywords: image processing, image transformations, computer vision, cameras, area calculation, odometry, C#, HSV, HSB

Índice general

Capítulo 1 - Introducción 1

1.1 Motivación del proyecto 1

1.2 Objetivo del proyecto 3

1.3 Problemas a abordar 4

1.3.1 Deformación del espacio por irregularidades del terreno 4

1.3.2 Referenciación de la superficie a tratar mediante segmentación en áreas. 8

1.3.3 Distinción de las distintas componentes del color para el tratamiento de malas hierbas 9

1.3.4 Solapamiento de las áreas de la superficie a tratar 10

1.3.5 Adecuación de la velocidad de refresco del robot a la velocidad de los cálculos realizados sobre las imágenes capturadas. 10

1.4 Dispositivos Hardware. 11

Capítulo 2 - Estado del arte 13

2.1 Evolución del procesamiento de imágenes digitales 13

2.2 Historia de la imagen digital 15

2.3 Rectificación de imágenes 17

2.4 Giróscopo, acelerómetro e inclinómetro 19

2.5 Modelos de color 21

2.6 Visión artificial 24

Capítulo 3 - Diseño técnico del proyecto 26

3.1 Introducción 26

3.2 Organización del diseño 26

3.2.1 Módulos de la arquitectura 26

3.2.2 Diagrama de clases global 30

3.2.3 Detalle de los módulos de la arquitectura desarrollados 31

Capítulo 4 - Servicios y tecnologías utilizadas 38

4.1 Microsoft Visual Studio 38

4.2 PCRemoteAdvance (3.9 - 29/08/2011) 38*

4.3 IP WebCam (1.8.15 - 06/05/2012) 38*

4.4 IMU (Inertial Measurement Unit) 39

4.5 Repositorio 40

Capítulo 5- Manual de Usuario 41

5.1 Requisitos hardware: 41

5.2 Instrucciones de configuración: *41*
5.3 Instrucciones de uso: *45*
5.3.1 Corrección de las irregularidades del terreno *45*
5.3.2 Segmentación de la imagen capturada *49*
5.3.3 Análisis de color de la imagen *51*
5.3.4 Manejo del robot móvil *56*
Capítulo 6 - Proceso de desarrollo **58**
6.1 Scrum *58*
6.2 Versiones de la herramienta *60*
Capítulo 7 - Conclusiones **62**
Apéndice A – Estudio de Modelos de Color **63**
Apéndice B – Estudio del comportamiento del acelerómetro **71**
Bibliografía **75**

Capítulo 1

Introducción

1.1 Motivación del proyecto

La utilidad del procesamiento de imágenes es muy amplia y se puede percibir en distintos ámbitos. Un ejemplo cotidiano es el procesamiento de imágenes, con fines de diagnóstico médico, capaces de facilitar información casi invisible al ojo humano. Otro ejemplo es el procesamiento de imágenes para realizar exámenes en terrenos de cultivo. Estas técnicas se enmarcan dentro de la llamada agricultura de precisión. Algunas utilidades son: detección de distintos tipos de vegetación, detección de elementos extraños o segmentación del terreno en áreas.

La agricultura de precisión no es un concepto nuevo con vistas a generar una nueva agricultura sostenible. Su puesta en marcha comenzó ya en la década de los noventa del siglo pasado, con la emergencia de nuevas tecnologías (GPS, SIG, sensores que estiman ciertas variables en tiempo real, ingeniería satelital muy detallada, etc.) y herramientas matemáticas, como la geoestadística.

Podemos señalar que tal aproximación ha sido más ampliamente aceptada en países del mundo anglosajón, tales como lo son EE.UU. y Australia, en donde ya existen algunos Institutos de investigación en esta materia. Sin embargo, no ha ocurrido lo mismo en Europa y otros continentes.

La obtención de datos relativos al suelo, cultivo y condiciones ambientales en general resulta todavía una labor costosa en tiempo e inversión. Es preciso el desarrollo de sistemas de sensores capaces de generar de forma precisa, rápida y barata la información necesaria. Hasta que no se solucione el problema de la adquisición de datos, la agricultura de precisión no podrá practicarse de forma generalizada.

Durante el curso académico 2011/12 se nos ha otorgado a los integrantes de este grupo la oportunidad de poder ahondar en esta área de conocimiento, gracias al presente proyecto, en el **análisis cromático** de imágenes tomadas frontalmente desde un vehículo terrestre con el consecuente problema de la inestabilidad del suelo y el **descuadre de las imágenes**.

Afrontamos este proyecto como una verdadera ocasión para aplicar los conocimientos adquiridos durante la carrera, dedicándonos para ello al desarrollo de una herramienta capaz de optimizar la complejidad temporal que supone el correcto procesamiento de imágenes de un terreno.

Para este correcto procesamiento será de vital importancia disponer en la herramienta de los procedimientos necesarios, tanto para controlar el robot móvil que capturará la información

del terreno, como para receptar la información obtenida mediante sensores internos ubicados en el propio robot.

Dicha herramienta deberá integrar diversas tecnologías (acelerómetros, cámaras digitales, controlador de robots, etc.), siendo capaz de aprovechar toda la información que se requiere para poder aplicar distintas técnicas de procesado de imágenes

El principal reto de este proyecto es por tanto disponer de todas estas tecnologías trabajando conjuntamente y obtener un resultado aceptable del análisis del terreno en tiempos del orden de magnitud de milésimas.

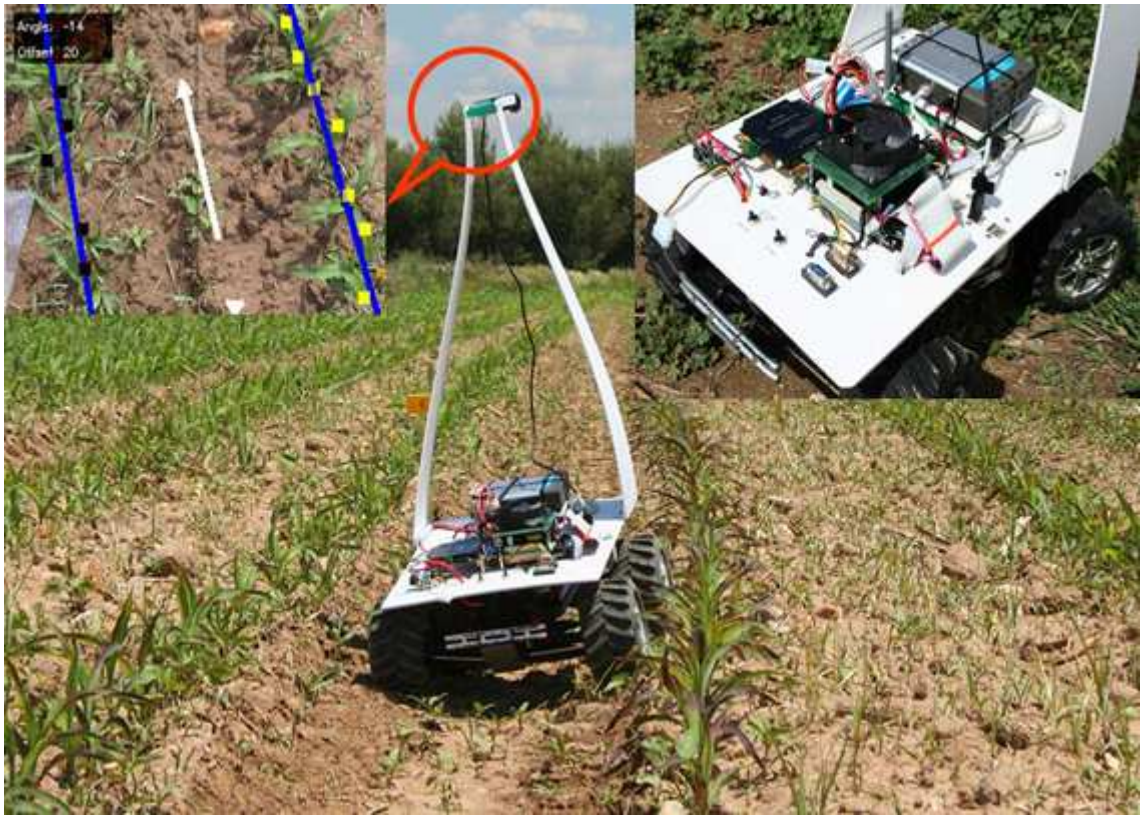

Figura 1 Robot ejemplo del proyecto RHEA

Este estudio podría enmarcarse dentro del proyecto RHEA (*Robot Fleets for Highly Effective Agriculture and Forestry Management*) que se inició en septiembre 2010 y cuyo consorcio está formado por 18 instituciones (organismos públicos de investigación, universidades y empresas) de siete países. Todos ellos poseen un contrastado prestigio internacional en diversas áreas como telecomunicaciones, navegación autónoma, energía, agronomía, teledetección, maquinaria agrícola de precisión, robótica y automatización.

En este proyecto se pretende disminuir hasta en un 75% el uso de agroquímicos en cultivos tanto herbáceos como leñosos así como reducir los costes de producción, haciéndola más sostenible. Todo ello será posible gracias a la labor de una flota de robots terrestres, como el de la Figura 1, y aéreos equipados con sensores de última generación. Entre otras funciones, estos robots estarán diseñados para tratamientos localizados y serán autónomos, heterogéneos y reconfigurables.

1.2 Objetivo del proyecto

El objetivo de este proyecto es diseñar e implementar un sistema capaz de procesar imágenes para la correcta identificación y localización de malas hierbas en un terreno. El sistema es insensible a desviaciones producidas por la irregularidad del terreno, gracias a la información proporcionada por un sensor de tipo acelerómetro. La herramienta creada se ha implementado sobre un robot inalámbrico Surveyor SRV-1.

Para ubicar correctamente las malas hierbas es necesario corregir aquellas desviaciones que pueden producir un error de cálculo, tales como las producidas por los desniveles del terreno. La primera fase del proyecto consistirá en transformar la imagen obtenida a través de una videocámara, mediante la utilización de algoritmos útiles, para corregir todas las variaciones de ángulo posibles (*pitch, jaw, roll*). Dicho sistema se ha diseñado para su integración con una IMU (*Inertial Measurement Unit*), que proporcionará la medida precisada de la aceleración en cada uno de los ejes x, y, z; las cuales servirán para poder calcular posteriormente las desviaciones de los ángulos *pitch* y *roll* en tiempo real. Una vez rectificada la imagen se procederá a analizar el terreno para detectar la posición de las malas hierbas.

Uno de los puntos importantes del proyecto es la correcta colocación del sistema de referencia empleado para tomar las imágenes. De este sistema dependerán las medidas que se le den al actuador encargado de eliminar las malas hierbas. Además, en todas las imágenes de la secuencia de video se debe mantener la correlación entre el píxel analizado y la referencia al punto de la realidad. Cualquier giro desvirtuaría esa relación y asignaría píxeles de la imagen capturada a puntos de la imagen de referencia diferentes y por tanto la información enviada al actuador sería errónea.

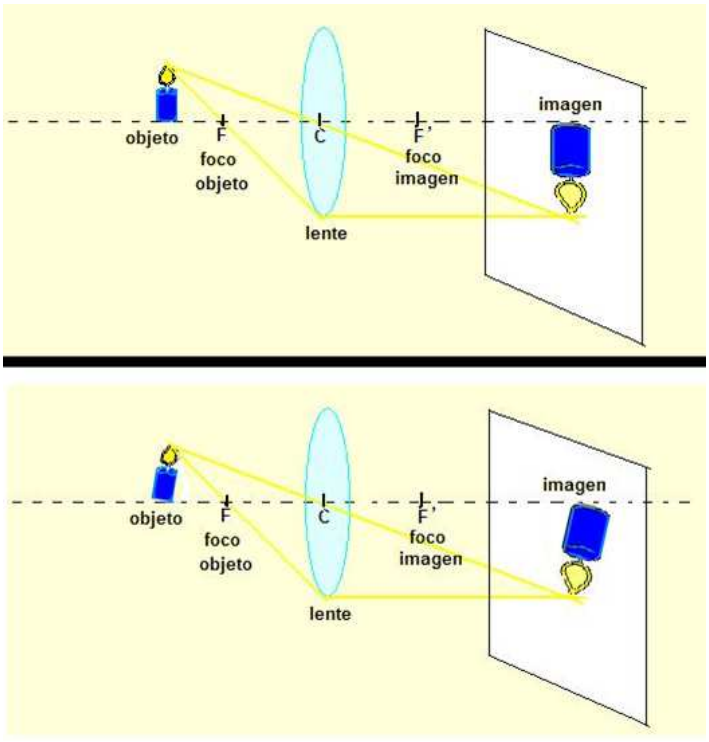

Figura 2. Ejemplo esquemático del problema provocado por las irregularidades del terreno.

Como se puede ver en la Figura 2 la base de la vela ha sido desplazada debido a que el sistema de referencia ha tomado la imagen con un cierto ángulo de cabeceo. Si se tomara una decisión a actuar sobre esa parte de la vela, las distancias serian erróneas, ya que esa parte del objeto se encuentra mal referenciada.

En el siguiente apartado se detallan los tipos de desviaciones que sufre el sistema de referencia y como se corrigen a posteriori.

Para este análisis se han utilizado distintas técnicas de procesamiento de imágenes, tales como: segmentación en áreas, aplicación de transformaciones en función de los valores obtenidos de un acelerómetro, identificación de tonalidades de cualquier color para distintos modelos de color (HSV, RGB, etc.) o algoritmos de optimización temporal.

La aplicación desarrollada se ejecuta en .NET Framework sobre el sistema operativo Windows. Está implementada en el lenguaje de programación C# y se ha desarrollado en el entorno de programación Microsoft Visual Studio.

Para llevar a cabo este proyecto hemos seguido una metodología de desarrollo Scrum (detallado en el capítulo 6: Proceso de desarrollo) debido a la continua evolución de los objetivos según avanzábamos nuestro desarrollo.

En los siguientes apartados veremos toda la evolución del proyecto, así como las fases e hitos alcanzados y los problemas encontrados. Con esto daremos una idea más detallada de la complejidad encontrada al desarrollar una herramienta de este tipo.

1.3 Problemas a abordar

Previamente al detalle de cómo se han alcanzado los requisitos funcionales planteados en una primera fase, se expondrán los problemas que fueron estudiados desde un principio, así como una breve introducción a la resolución final efectuada.

El análisis de imágenes en agricultura de precisión plantea varios problemas iniciales: deformación del espacio por irregularidades del terreno, referenciación de la superficie a tratar mediante segmentación en áreas, distinción de las distintas componentes del color para el tratamiento de malas hierbas, solapamiento de las áreas de la superficie a tratar y adecuación de la velocidad de refresco del robot a la velocidad de los cálculos realizados sobre las imágenes capturadas. En los siguientes apartados se describe como se solucionan estos problemas.

1.3.1 Deformación del espacio por irregularidades del terreno

1.3.1.1 Análisis de las deformaciones sufridas en la imagen

El primero de los problemas analizados es el error producido por la propia irregularidad del terreno donde desarrollará las actividades de análisis el robot

objetivo, y con ello la imposibilidad de conseguir una secuencia de imágenes con la misma perspectiva u orientación.

Este problema perturba la realidad captada por la cámara y puede conllevar a tomar decisiones erróneas sobre puntos del espacio mal referenciados.

En la Figura 3, el plano de enfoque de la cámara se encuentra en paralelo con el del suelo.

Figura 3. Imagen bien referenciada.

Mientras que en la Figura 4 se encuentra girado aproximadamente 15°, por tanto los objetos detectados se posicionarían incorrectamente a la hora del procesado del terreno.

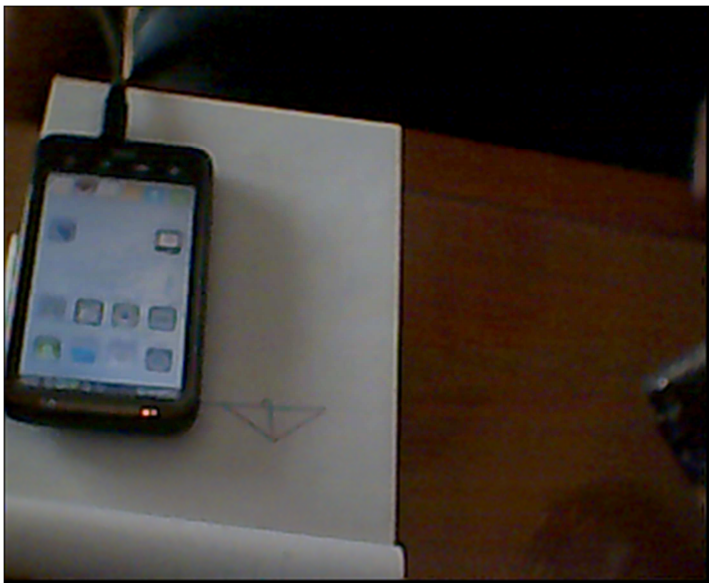

Figura 4. Imagen deformada.

Para solucionar este problema se recurre a un sensor de tipo inclinómetro. Con éste se pueden medir las variaciones de posición del plano del robot, con respecto a una posición de referencia.

Posteriormente se deben aplicar técnicas de rectificación de imágenes para obtener una imagen bien referenciada.

Figura 5. Imagen corregida.

Como se puede apreciar en la Figura 5, parte de la imagen real queda fuera de la modificada. Este error no se puede subsanar, ya que es imposible analizar lo que cae fuera del ángulo de visión. No obstante, dicho problema no será crítico ya que se esta hablando de una secuencia de video en tiempo real. Es decir se deja de analizar una parte de una imagen en una secuencia de entre 30 y 40 *frames* por segundo (limite del ojo humano).

1.3.1.2 Corrección de las deformaciones por irregularidad del terreno

A la hora de capturar imágenes pueden darse dos tipos de perturbaciones, las cuales deberán corregirse para volver a obtener una imagen transformada con mayor parecido a la imagen de referencia. Se tomará la Figura 6 como imagen de referencia para lograr una explicación gráfica de ambos problemas.

Figura 6. Imagen correcta.

Una variación del cabeceo (pitch).

En este problema la imagen se corrige mediante una traslación vertical. Como se puede ver en la Figura 7.

Figura 7. Imagen tomada con un cabeceo de -5.08º.

Una variación en la oscilación (roll)

En este problema la imagen sufre una rotación como la que se puede observar en la Figura 8, calculando los ángulos a corregir mediante una transformación de giro gracias a los valores obtenidos por un acelerómetro (obteniendo los mismos ángulos que obtendríamos utilizando un inclinómetro), y realizando finalmente una translación horizontal.

Figura 8. Imagen tomada con una oscilación de -13º.

Para la corrección de estos defectos, es necesario calcular las distancias que deben trasladarse tanto vertical, como horizontalmente. Para ello se debe realizar una batería de pruebas, calculando la distancia (en píxeles) que debe desplazarse en función del cabeceo y de la oscilación.

Finalmente se llega a una serie de factores de corrección, con los que se construyen dos funciones, una para el ángulo de cabeceo y otra para el ángulo de oscilación. La función de cabeceo relaciona el ángulo con el desplazamiento horizontal y la función de oscilación relaciona el ángulo con el desplazamiento horizontal y vertical.

Todos estos cálculos están influenciados por la distancia de la cámara al suelo. Los cálculos por defecto están realizados con la cámara sobre el robot a una distancia de unos 10 cm sobre el suelo. Todos estos factores de corrección son configurables desde fuera de la aplicación, para que sea valida para cualquier sistema móvil.

1.3.2 Referenciación de la superficie a tratar mediante segmentación en áreas.

Para el análisis del terreno, se divide el espacio visible en parcelas. Esta división se realiza mediante una malla virtual configurable con los parámetros expuestos en la Figura 9.

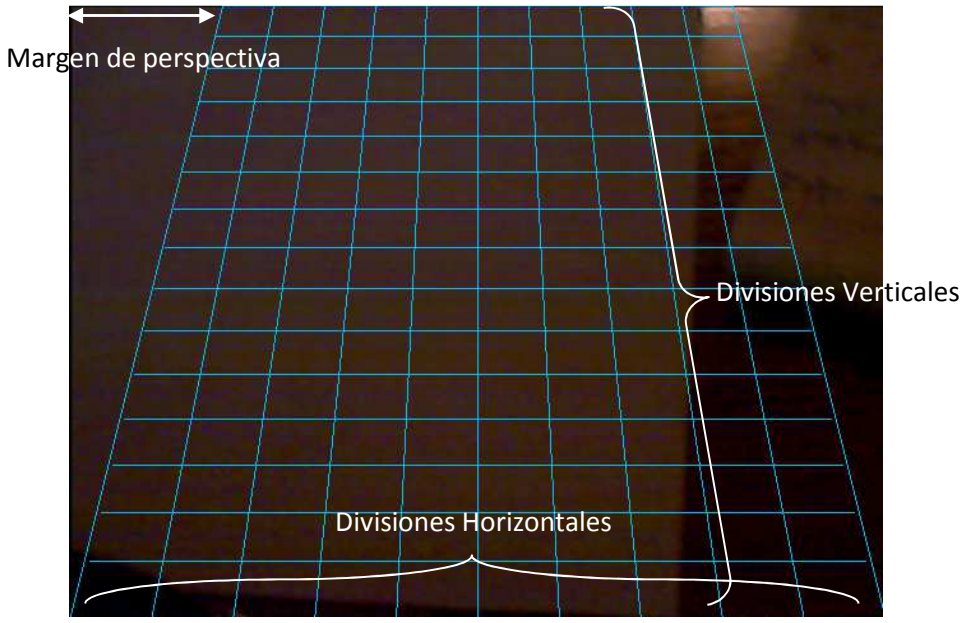

Figura 9. Ejemplo de malla de división del campo visible.

Cada parcela será analizada de manera independiente, extrayendo el número de píxeles de color y decidiendo, según su densidad, si es necesario que el actuador opere o no sobre ella.

1.3.3 Distinción de las distintas componentes del color para el tratamiento de malas hierbas

Para el análisis de color, se han empleado dos técnicas: detección por función de verde vegetal y detección por aproximación a una distribución gaussiana.

1.3.3.1 Detección por función de verde vegetal

Es la técnica más sencilla para la detección de grandes extensiones de verde en espacios abiertos y con luminosidad abundante y natural popularmente conocida como verde vegetal. La función evalúa cada píxel de la siguiente manera: un píxel es verde, cuando el valor numérico de su canal verde multiplicado por dos, restado el canal azul y el canal rojo es mayor que un umbral dado.

$$\boldsymbol{2 \cdot G - R - B} > Umbral$$

1.3.3.2 Detección por aproximación a una distribución gaussiana

La segunda técnica, más compleja, da solución a detecciones más exhaustivas, en espacios con poca luminosidad o con un foco concentrado de luz.

Esta técnica se basa en un sincronismo previo del color a detectar (válido para cualquier color), el cual aproxima la variable a una distribución gaussiana.

La decisión de pertenencia o no se puede realizar para cada uno de los tres canales en cada uno de los dos modelos de color empleados. Es decir se pueden comprobar los

canales RGB (desde uno hasta tres componentes) y HSB (desde uno hasta tres componentes).

1.3.4 Solapamiento de las áreas de la superficie a tratar

Al realizar un análisis en movimiento de cada una de las áreas en las que dividimos la porción de terreno capturada en la imagen, surge un nuevo problema: al desplazarse el robot, podrían referenciarse áreas o porciones de áreas que ya fueron referenciados previamente.

Para solventar este problema será necesario controlar los parámetros de movimiento del robot para ajustarlos al tiempo de cómputo del porcentaje de color en cada pasada del algoritmo de cálculo. Esto se realizará externamente a la herramienta a través de un archivo de configuración que definirá los parámetros del robot, cuyo detalle se explicará más a fondo en los próximos apartados.

1.3.5 Adecuación de la velocidad de refresco del robot a la velocidad de los cálculos realizados sobre las imágenes capturadas.

Análogo al problema anterior, será necesario disponer de una cámara capaz de soportar una frecuencia de refresco mayor al tiempo de cómputo del porcentaje de color en cada pasada del algoritmo de cálculo.

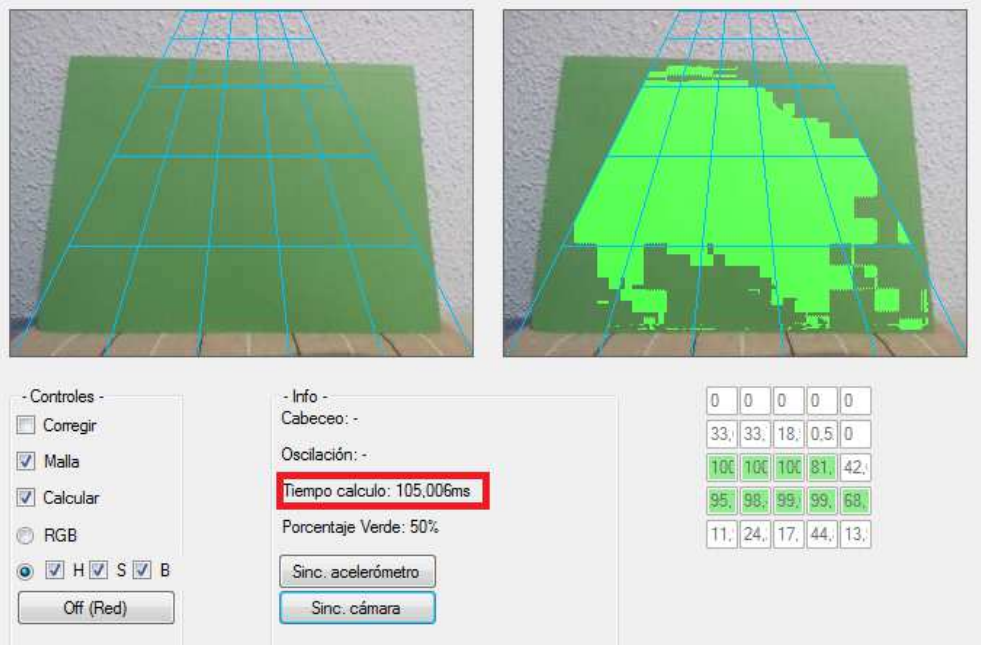

Figura 10. Captura con tiempo de cómputo resaltado.

En nuestro caso, otorgamos una tasa de refresco a la cámara de 20 *frames* por segundo, frecuencia suficiente para soportar tiempos de cálculo mayores de 20ms como el que se puede observar remarcado en la Figura 10.

Todos estos cálculos, deben mantener siempre el compromiso entre precisión de los resultados y coste temporal del procesamiento.

La cota superior de cálculo no debe sobrepasar el tiempo real y ralentizar la cámara y sus fps (*frames* por segundo) por debajo de la apreciación del ojo humano (25 fps aprox.).

1.4 Dispositivos Hardware.

En este proyecto tiene gran peso la componente *hardware*, ya que en este sistema en tiempo real es necesario responder en cuestión de milésimas de segundo, y no todos los dispositivos pueden estar capacitados.

El proyecto tendrá tres dispositivos físicos principales:

1 Computador central que realizara todo el cálculo y la toma de decisiones.

2 Modulo de captura de datos, tanto de imágenes (cámara), como de posición del plano (inclinómetro).

Para esta parte, se utilizará un teléfono móvil con S.O. Android 2.3. El cual enviará todos los datos, tanto imágenes como posición del acelerómetro del móvil al PC (donde emulará a un inclinómetro), mediante conexión TCP o por Bluetooth.

3 Dispositivo de movilidad, el cual desplazará la cámara a lo largo del terreno y realizará su reconocimiento.

Para los *tests* en tiempo real, la aplicación contará con un controlador de robot, de tipo "Surveyor SRV-1" como el que aparece en la Figura 11. Que se conectará mediante Wifi y lo moverá mediante órdenes enviadas desde el PC.

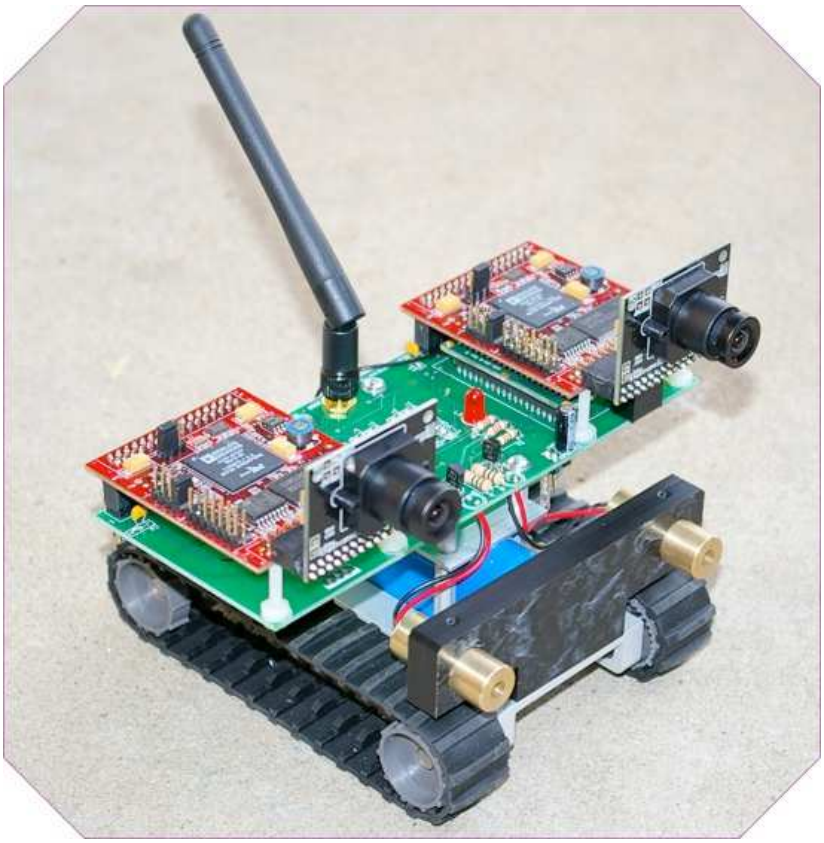

Figura 11. Robot Surveyor SRV-1

Capítulo 2

Estado del arte

2.1 Evolución del procesamiento de imágenes digitales

Una de las primeras aplicaciones datadas de digitalización de imágenes se realizó en el sector periodístico cuando en 1921 se utilizó por vez primera el sistema Bartlane de transmisión de imágenes. Dicho sistema reducía el tiempo recorrido por la imagen a través del Atlántico desde más de una semana a tan solo tres horas aproximadamente. Para lograr esta hazaña se utilizó una técnica en la que inicialmente se codificaba la imagen en 5 niveles de gris como se puede observar en la Figura 12, para posteriormente ser decodificada e impresa mediante un teletipo en el receptor de la transmisión.

Algunos de los problemas iniciales al mejorar la calidad visual de estas imágenes digitales estaban relacionados con los procedimientos de impresión y la distribución de los niveles de intensidad. Los métodos utilizados en la época del sistema Bartlane fueron abandonados en favor de una nueva técnica basada en la reproducción fotográfica.

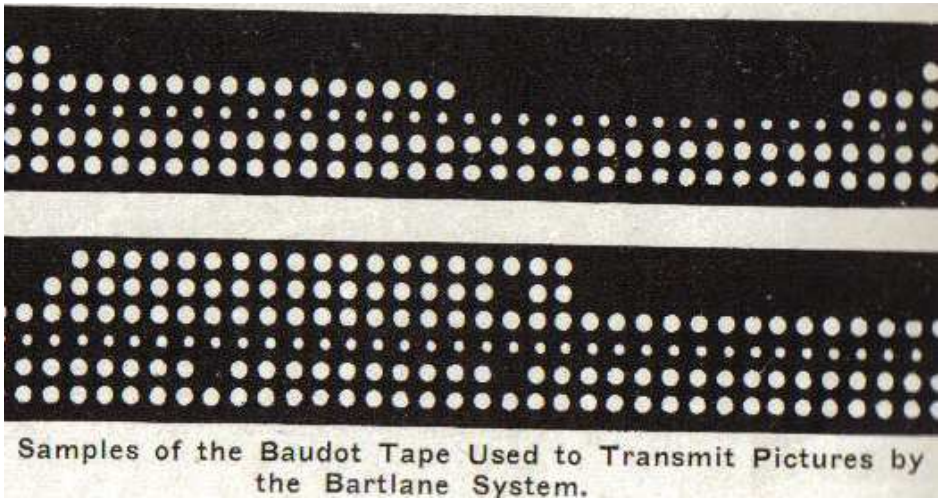

Figura 12. Imagen del código utilizado para transmitir imágenes a través del sistema Bartlane.

Pese a que estos ejemplos han sido citados como imágenes digitales, en realidad no son considerados como tales, ya que para su procesamiento no se utilizó ninguna computadora.

No sería hasta inicios de la década de los 60 cuando aparecieron las primeras computadoras suficientemente potentes como para poder realizar tareas complejas de procesado de imágenes.

Las primeras imágenes procesadas por ordenador fueron aquellas destinadas a corregir distintos tipos de distorsión de las imágenes tomadas por la sonda americana *Ranger* 7,

diseñada en el *Jet Propulsion Laboratory* de Pasadena en 1964 para estudiar la Luna. Un ejemplo de estas imágenes tomadas analógicamente se observa en la Figura 13.

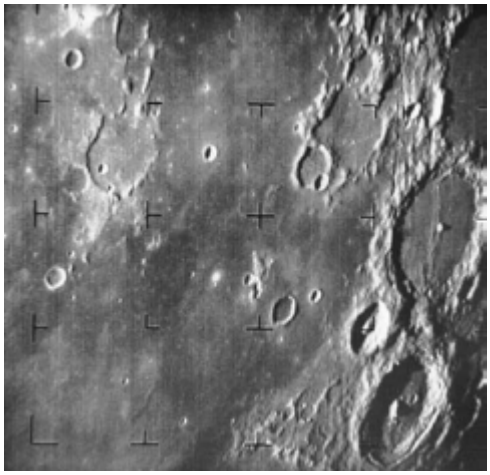

Figura 13. Primera imagen de la Luna realizada por naves americanas (*Ranger* 7)

En paralelo con estas labores de corrección de imágenes, durante las décadas de los 60 y los 70 aparecieron las primeras técnicas de utilidad para la creación de imágenes con fines médicos. El descubrimiento de la técnica de imagen médica denominada tomografía axial computarizada en el año 1967 a manos de Allan McLeod Cormack supuso un gran avance para la medicina. La tomografía consiste en la utilización de una serie de algoritmos que aprovechan los datos tomados por detectores y rayos X para obtener una imagen que representa la sección de un objeto o parte del cuerpo.

Desde la década de 1960 hasta la actualidad, el campo de procesamiento de imágenes ha crecido vertiginosamente. Además de las aplicaciones en medicina y el programa espacial, las técnicas de procesamiento digital se utilizan en una amplia gama de aplicaciones. Se utilizan herramientas informáticas por ejemplo, para mejorar el contraste o la intensidad de un color y de este modo facilitar la interpretación de imágenes utilizadas en la industria y las ciencias biológicas.

Los geógrafos también utilizan técnicas iguales o similares para estudiar los patrones de contaminación detectados por los satélites. En la arqueología, los métodos de procesamiento de imágenes han tenido gran éxito para recomponer imágenes incompletas de objetos antiguos y sin registros disponibles para su restauración, tal como podemos ver en la Figura 14 con la reconstrucción del clásico coliseo romano realizada por Google.

Figura 14. Reconstrucción 3D del coliseo romano.

En física y disciplinas afines, las técnicas de computación rutinariamente mejoran las imágenes de los experimentos en áreas como los plasmas de alta energía y la microscopía electrónica. Del mismo modo aplicaciones exitosas sobre el procesamiento de imágenes se pueden encontrar en la astronomía, la biología, la energía nuclear, la medicina, la aplicación de la ley, la defensa, y la industria.

2.2 Historia de la imagen digital

Anteriormente hemos hecho mención a sistemas que ya eran capaces de capturar imágenes de televisión, convertirlas en una señal eléctrica y guardarlas en soportes magnéticos. Sin embargo, no sería hasta 1969 cuando comenzó realmente la carrera digital.

En este año Willard Boyle y George Smith, investigadores de los Laboratorios Bell situados en Nueva York, diseñan la estructura básica del primer sensor CCD, aunque en principio se plantease como un sistema para el almacenamiento de información. Sería un año más tarde cuando los laboratorios Bell desarrollarían la primera videocámara que implementaba un sensor CCD como sistema para capturar imágenes.

Los sensores CCD (*charge-coupled device*), como el que se puede observar dentro del chip de la Figura 15, son circuitos integrados que contienen un número determinado de condensadores enlazados o acoplados. Cada condensador puede transferir su carga eléctrica a uno o a varios de los condensadores que estén a su lado en el circuito.

Figura 15. Chip contenedor de un sensor CCD.

Su funcionamiento se basa 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 aislados entre sí por una capa de SiO2. De este modo, el CCD se lee línea a línea, aunque existen numerosos diseños diferentes de detectores.

Paralelamente a la investigación de los sensores CCD, un grupo de investigadores liderados por Peter J. W. Noble, Savvas G. Chamberlain y P. K. Weimer, desarrollaron una nueva modalidad de sensor, ampliamente utilizada en la actualidad: el sensor CMOS. En el CMOS, a diferencia del CCD se incorpora un amplificador de la señal eléctrica en cada detector y es común incluir el conversor digital en el propio chip. La ventaja obtenida con esta técnica es que la electrónica puede leer directamente la señal de cada píxel con lo que se soluciona el problema conocido como *blooming*, por el cual cuando los sensores reciben más corriente de la que pueden almacenar se la transmiten al sensor de al lado, el cual sufre el mismo problema, provocando zonas enteras de la imagen quemadas. La desventaja es que entre los receptores de luz (*fotosites*) 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 micro-lentes que a modo de lupa concentran la luz de cada celda en su *fotosite*.

Anteriormente, los sensores CCD eran los líderes en el mercado de las cámaras digitales (sobre todo las de alta calidad), ya que esta tecnología había sido históricamente más ampliamente utilizada en el amplio ámbito de la fotografía digital en comparación con las cámaras CMOS. Es por ello que se trata todavía de un producto más maduro, y tiende a tener una mayor calidad y más píxeles. Por todo esto se puede ver que los sensores CCD son usados en cámaras que tienden a tener una mejor calidad con muchos píxeles y una gran sensibilidad a la luz. Sin embargo, los sensores CMOS están mejorando continuamente y se encuentran en un punto en el cual están por conseguir la misma calidad que los sensores CCD. Por este motivo y por su precio, bastante inferior al de los sensores CCD, la tecnología CMOS se ha convertido en la actual líder del mercado de los sensores para cámaras digitales. Como en tantos otros casos,

no hay una tecnología superior de por sí, sino situaciones en las que cada tecnología es mas adecuada.

Por último, no se podría dejar sin mención en este breve análisis de la historia de las imágenes digitales un hito tan importante como es la captura del color. Y es que todos estos sensores de imagen no son capaces de captar las imágenes en color al ser monocromos, es decir sólo pueden memorizar la intensidad de la luz pero sin obtener ningún registro del color. Para captar la imagen en color es necesario utilizar varios sistemas de filtros de color en cada sensor de la imagen. Uno de los filtros más conocidos es el filtro CFA Bayer, patentado por Bryce Bayer, trabajador de la empresa Kodak, en 1976.

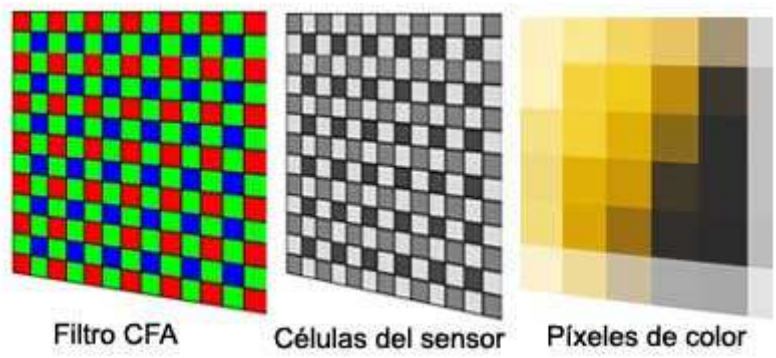

Figura 16. Aplicación del filtro de Bayer sobre las células de un sensor de imagen.

Como observamos en la Figura 16, este sistema cubre más píxeles con filtros verdes que con rojos o azules, debido a que el ojo distingue mejor los detalles finos en la parte central del espectro (verde). Por lo tanto un píxel con un filtro rojo sólo medirá la luz roja y el resto píxeles que forman la imagen, sólo medirán la luz azul o verde. A través de la medición de distintos niveles de brillo de los tres colores primarios, cada grupo de cuatro píxeles aportará los datos de color combinando las lecturas de los filtros colocados sobre los píxeles contiguos.

2.3 Rectificación de imágenes

Durante el proceso de captura de imágenes, estas pueden sufrir ciertas distorsiones geométricas debido a diversos motivos:

- Variaciones de la velocidad del dispositivo de captura de imágenes.
- Movimientos de los ejes del dispositivo de captura de imágenes (cuyos efectos se muestran en la Figura 17:
    - Alabeo u oscilación (*Roll*)
    - Cabeceo (*Pitch*)
    - Guiñada o aleteo (*Yaw*)
- Distorsiones producidas por la lente del dispositivo.

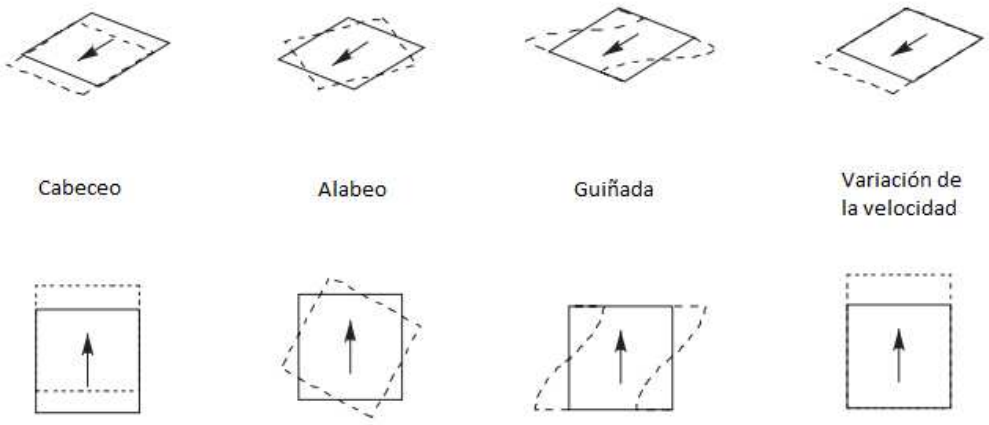

Figura 17. Posibles distorsiones geométricas del robot.

Debido a las características de este proyecto, nos centraremos en detallar la manera sistemática para corregir las distorsiones producidas por el movimiento angular de los ejes, más concretamente los movimientos de alabeo (*roll*) y cabeceo (*pitch*).

Presuponemos por tanto, que el robot móvil ya dispone de los sensores necesarios para la corrección automática para el movimiento de guiñada y que por tanto el robot se moverá continuamente en línea recta. Para la corrección de la rectitud de la dirección se pueden utilizar dos tipos de sensor:

- **Brújula magnética:** Este dispositivo puede proporcionar el ángulo que forma en todo momento el mencionado robot con el norte magnético. Es una medida muy fiable en exteriores, pero no tanto en presencia de entornos metálicos presentes en muchos interiores. En este caso la brújula se verá afectada dando lugar a medidas erróneas.

- **Giróscopo:** La otra posibilidad es la integración de las velocidades angulares proporcionadas por un giróscopo, con el cual se conocen las variaciones con respecto a la orientación en un instante inicial que se debe conocer. Tiene la ventaja de que el ángulo obtenido no se ve afectado por las perturbaciones magnéticas que se producen en las proximidades del sensor en presencia de entornos metálicos.

Para poder corregir los dos movimientos de cabeceo y oscilación es necesario conocer la situación en el momento de la captura de los ejes del dispositivo, así como una posición inicial de referencia. Una vez adquirida esta información se podrán calcular los valores los ángulos producidos por las distorsiones de cabeceo y oscilación:

$$\text{Pitch} = \frac{180 * \text{Atan}\left(\frac{X}{Z}\right)}{\pi} \qquad \text{Roll} = \frac{180 * \text{Atan}\left(\frac{Y}{Z}\right)}{\pi}$$

Una vez obtenidos estos ángulos se realizará una traslación de la imagen en función del valor obtenido por el cabeceo.

Paralelamente se realizará la corrección del ángulo de oscilación o *roll.* Podemos observar otro ejemplo ilustrativo de esta corrección en la Figura 18:

Figura 18. Corrección del ángulo de oscilación.

2.4 Giróscopo, acelerómetro e inclinómetro

Los sensores de orientación son dispositivos que sirven o pueden servir por ejemplo para orientar a un robot en el espacio 3D. Para este propósito existen una gran variedad de sensores y sistemas complejos capaces de determinar la orientación del robot en los tres ejes del espacio 3D. Sin embargo, en este caso, únicamente se hará un análisis de un grupo pequeño de sensores simples, capaces en la mayoría de los casos de medir sólo en una única dimensión. No obstante, con la combinación de varios de ellos se podrá obtener también la orientación en los tres ejes.

Giróscopo.

Los sensores de velocidad angular, denominados giroscopios o giróscopos, son dispositivos utilizados para detectar el ángulo de giro sobre sus ejes. Aunque en un principio eran dispositivos mecánicos (Figura 19), han sido sustituidos por los piezoeléctricos o los ópticos, de los cuales, los láser son los que ofrecen la mayor precisión. Sin embargo sufren el inconveniente de la deriva, es decir, la acumulación de error provocada por la desviación de la

posición fija de referencia inicial provocada, entre otros, por la variación de la gravedad terrestre o por la mala colocación del dispositivo

Figura 19. Giróscopo mecánico.

Acelerómetro.

Los acelerómetros sirven para medir la aceleración lineal de un móvil solidario con ellos en un determinado eje del espacio. Su principal inconveniente es su calibrado, haciendo imposible realizar un cálculo correcto de la velocidad y la posición.

Inclinómetro.

Un inclinómetro es un instrumento utilizado para medir la inclinación del plano con respecto de la horizontal (superficie terrestre). Los inclinómetros son utilizados para medir el ángulo de orientación absoluto respecto a un único eje dentro de un rango predeterminado (como se observa en la Figura 20). Debido a esto, un inclinómetro se adapta mucho mejor al cálculo de la orientación que un giroscopio. Sin embargo, en sistemas que requieren una respuesta inmediata, los giróscopos son más eficaces que los inclinómetros.

Figura 20. Inclinómetro

Estos dispositivos pueden ser sensores de distinta naturaleza: mecánicos, ópticos, capacitivos, piezoresistivos o piezoeléctricos. Sin embargo, actualmente son fácilmente integrables dentro de un sólo chip de silicio gracias a los sistemas microelectromecánicos (MEMS). Esta integración a nivel microscópico provoca que el principio de operación para la determinación de la orientación sea diferente. Los sensores MEMS, están basados en el traspaso térmico por

convección natural, es decir, emplean la transferencia de calor causada por la aceleración para realizar la medida de sus magnitudes.

2.5 Modelos de color

Durante siglos, artistas y filósofos han teorizado que el color es tridimensional. Neurocientíficos contemporáneos han confirmado esta teoría y han descubierto que nuestra sensación del color viene de células nerviosas que envían mensajes al cerebro sobre:

- ✓ El brillo de color
- ✓ Color verde frente a color rojo
- ✓ Color azul frente a color amarillo

Cuando los colores son oscuros o claros, percibimos menos variaciones en su intensidad. Vemos la gama máxima de saturación del color de medios tonos de color. Por eso, muchos modelos de color disminuyen los colores superiores e inferiores y se ensanchan en el medio formando una esfera o bicono.

En la teoría del color, los modelos de color describen matemáticamente cómo pueden ser representados los colores. Un espacio de color es donde los componentes del modelo de color son definidos con precisión, lo que permite a los observadores saber exactamente como se ve cada color.

La representación de la física del espacio de color comenzó con una rueda de dos dimensiones que permitía ver el matiz (rojo, azul, verde, etc.) y el brillo de los diferentes colores. Más tarde, surgió el concepto de colores sólidos. Los colores sólidos son representaciones tridimensionales del espacio de color. Además del matiz y el brillo en el modelo bidimensional, un color sólido muestra degradados de saturación para un matiz particular. La mayoría de los colores sólidos están en la forma de una esfera, pero esto es en gran medida una cuestión de conveniencia. Los colores sólidos pueden tener cualquier forma.

Modelo RGB

A mediados del siglo XIX, Thomas Young y Hermann Helmholtz propusieron una teoría de visión tricromática del color que se convirtió en la base para el modelo de color RGB (rojo, verde, azul). Este es un modelo de color aditivo, en el cual las tres luces de colores se suman para producir diferentes colores.

La intensidad de la luz determina el color percibido. Sin intensidad, cada uno de los tres colores se percibe como negro, mientras que la intensidad completa lleva a la percepción del blanco. Hay diferentes intensidades que producen el matiz de un color, mientras que la diferencia entre la mayor y menor intensidad del color hace que el color resultante sea más o menos saturado.

Las pantallas electrónicas usan el modelo RGB, lo cual significa que los colores no son absolutos, sino que más bien dependen de la sensibilidad y la configuración de cada

dispositivo. Las pantallas de tubos de rayos catódicos, LCD, plasma y pantallas LED usan todas, el modelo RGB, cuya función observamos en la Figura 21.

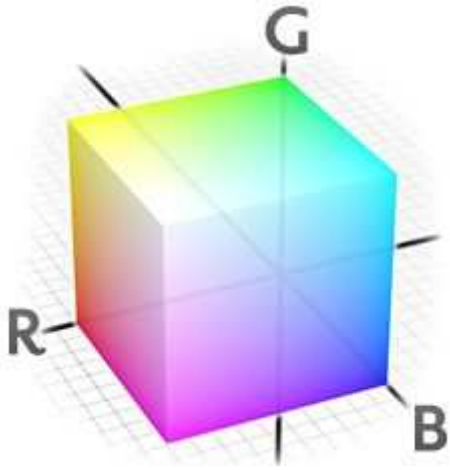

Figura 21. Modelo RGB

El modelo RGB de 24 bits también se utiliza para codificar el color en la informática, donde el valor de cada color se especifica por la intensidad del rojo, verde, y azul, respectivamente. En el diseño de páginas web, hay 216 colores RGB llamados "seguros para web" y representados por valores hexadecimales. Hoy en día, el RGB sigue siendo el modelo de color estándar para la programación HTML, pero la prevalencia de las pantallas de 24 bits permite a más usuarios ver 16.7 millones de colores RGB de código HTML.

Modelo HSV

Representado por primera vez por Alby Smith en 1978, HSV busca representar las relaciones entre los colores, y mejorar el modelo de color RGB. Manteniendo el matiz, la saturación y el valor, HSV representa un color tridimensional tal cual se dispone en la Figura 22. Si se piensa en el HSV como si fuera rueda de queso, el eje central va desde el blanco en la parte superior hacia el negro en la inferior, con otros colores neutrales en el medio. El ángulo del eje representa el matiz, la distancia desde el eje representa la saturación, y la distancia a lo largo del eje representa el valor.

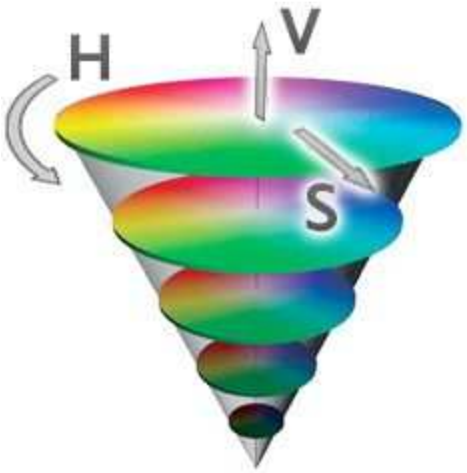

Figura 22. Modelo HSV

Modelo HSL

Como el HSV, HSL fue representado por Alvy Ray Smith y es una representación 3D del color. HSL mantiene el matiz, la saturación, y la luminosidad. El modelo de color HSL tiene claras ventajas respecto al modelo HSV, en el sentido que los componentes de saturación y luminosidad expanden el rango entero de valores.

El modelo de color HSL contiene todos los matices en diferentes niveles de saturación a lo largo de su plano horizontal y con variantes en la intensidad a lo largo de su plano vertical como se observa en la Figura 23.

Por ejemplo, usando el modo "Matiz", se puede posicionar los colores en los lados opuestos del diamante para que se correspondan con los colores complementarios. O se puede disponer los colores así sus matices son ubicados triangularmente, relativos entre sí para un esquema de color tríadico. Y utilizando tres dimensiones cuando se editan los colores o las paletas de colores, se puede entender intuitivamente qué colores son similares, y cuales contrastan.

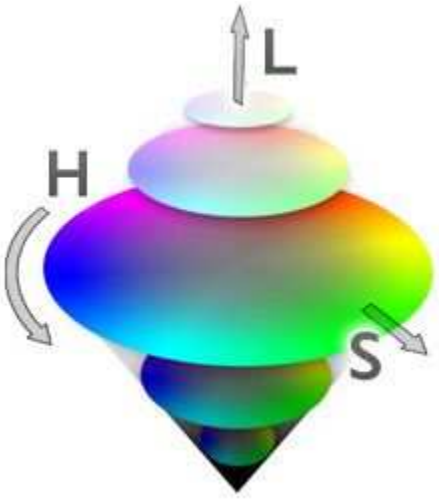

Figura 23. Modelo HSL

En el plano horizontal del ecuador, los matices puros saturados están a lo largo del perímetro ecuatorial. Similar a la rueda tradicional de color y a las representaciones de color esféricas, los matices contrastantes son ubicados opuestos entre sí. A medida que se mueva hacia el centro del disco de color (en el mismo plano) la saturación del color disminuye hacia el centro, donde todos los colores se unen en una único gris. Al moverse verticalmente a lo largo de este centro, el color gradualmente se va aclarando hacia arriba (finalizando en blanco), y oscureciendo hacia abajo (finalizando en negro). Los matices varían en intensidad y saturación a medida que se mueva verticalmente arriba y abajo, o hacia el interior del diamante. Cualquier matiz dado puede variar en saturación moviéndose hacia adentro o en intensidad (tinta) moviéndose verticalmente arriba o abajo.

2.6 Visión artificial

La Visión artificial, también conocida como Visión por Computador o Visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una imagen a través del empleo de diversas acciones (detección y seguimiento de objetos, mapeo de modelos, evaluación de la propia imagen, etc.).

Estos objetivos se consiguen por medio de técnicas como reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de gráficos y otros campos. La visión artificial cognitiva está muy relacionada con la psicología cognitiva y la computación biológica.

De manera natural nuestro mecanismo de visión es estéreo, es decir, somos capaces de apreciar, a través de la visión binocular, las diferentes distancias y volúmenes en el entorno que nos rodea. Nuestros ojos, debido a su separación, obtienen dos imágenes con pequeñas diferencias entre ellas, a lo que denominamos disparidad. Nuestro cerebro procesa las diferencias entre ambas imágenes y las interpreta de forma que percibimos la sensación de profundidad, lejanía o cercanía de los objetos que nos rodean. Este proceso se denomina estereopsis. La distancia interpupilar más habitual es de 65 mm, pero puede variar desde los 45 a los 75 mm.

En la estereopsis intervienen diversos mecanismos. Cuando observamos objetos muy lejanos, los ejes ópticos de nuestros ojos son paralelos. Cuando observamos un objeto cercano, nuestros ojos giran para que los ejes ópticos estén alineados sobre él, es decir, convergen. A su vez se produce la acomodación o enfoque para ver nítidamente el objeto. Este proceso conjunto se llama fusión. No todo el mundo tiene la misma capacidad de fusionar un par de imágenes en una sola tridimensional. Alrededor de un 5% de la población tiene problemas de fusión. La agudeza estereoscópica es la capacidad de discernir, mediante la estereopsis, detalles situados en planos diferentes y a una distancia mínima. Hay una distancia límite a partir de la cual no somos capaces de apreciar la separación de planos, y que varía de unas persona a otras. Así, la distancia límite a la que dejamos de percibir la sensación estereoscópica puede variar desde unos 60 metros hasta cientos de metros.

Un factor que interviene directamente en esta capacidad es la separación interocular. A mayor separación entre los ojos, mayor es la distancia a la que apreciamos el efecto de relieve. Esto se aplica por ejemplo en los prismáticos, en los que, mediante prismas, se consigue una separación interocular efectiva mayor que la normal, con lo que se consigue apreciar en relieve objetos distantes que en condiciones normales no seríamos capaces de separar del entorno. También se aplica en la fotografía aérea, en la que se obtienen pares estereoscópicos con separaciones de cientos de metros y en los que es posible apreciar claramente el relieve del terreno, lo que con la visión normal y desde gran altura sería imposible. El efecto obtenido con una separación interocular mayor que la habitual es el de que los objetos parecen más pequeños de lo normal (liliputismo), y la técnica se denomina hiperestereoscopia.

El efecto contrario se consigue con la hipoestereoscopia, es decir, con la reducción de la distancia interocular, imprescindible para obtener imágenes estereoscópicas de pequeños objetos (macrofotografías), o incluso obtenidas por medio de microscopios.

Capítulo 3

Diseño técnico del proyecto

3.1 Introducción

En este apartado se comenzará explicando cual es la organización o arquitectura de los módulos del sistema desarrollado. Se empezará introduciendo la estructura general del aplicativo para posteriormente centrarse en cada una de las clases desarrolladas.

3.2 Organización del diseño

El patrón de diseño escogido para el diseño de la arquitectura de nuestra aplicación es el popular patrón MVC, siglas de modelo-vista-controlador (o en inglés, *model-view-controller*), el cual es uno de los patrones más utilizados en la arquitectura de *software*.

El patrón MVC es un patrón de arquitectura de *software* encargado de separar la lógica de negocio de la interfaz del usuario, facilitando la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite "no mezclar lenguajes de programación en el mismo código".

MVC divide las aplicaciones en tres niveles de abstracción:

•**Modelo**: Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedido por el controlador para añadir, eliminar, consultar o actualizar datos.

•**Vista**: Es la parte ocupada de presentar el modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.

•**Controlador**: Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

3.2.1 Módulos de la arquitectura

Por funcionalidad de clases, la arquitectura de divide en siete módulos. Dos de ellos son equivalentes a los de la división del modelo vista controlador, la vista y el controlador. El modelo se divide en cinco submódulos de la siguiente manera:

- *Cam Manager* – Es el encargado de manejar la cámara. Tiene el método para encenderla, para apagarla y un evento que se dispara cuando hay una actualización de imagen. Tiene los siguientes parámetro de configuración:
  - ✓ *Image width*

- ✓ *Image height*
- ✓ *Capture period*

Este módulo es abstracto y tiene dos implementaciones concretas de dos modelos de cámara que internamente funcionan de forma diferente.

- o *Usb Cam Manager* – Maneja una cámara conectada por USB.
- o *Ip Cam Manager* – Maneja una cámara conectada por TCP/IP

- *Accelerometer Manager* – Este módulo se encarga de manejar el acelerómetro. Su funcionamiento es muy parecido al módulo *Cam Manager*. Tiene un método de inicio y de parada y un evento que se dispara cada vez que se actualizan las tres componente de la aceleración (X, Y, Z).

- *Robot Manager* – Este módulo se encarga de la conexión con el robot. Tiene un método para conectar y otro para desconectar. También proporciona una API para el desplazamiento adelante, atrás, izquierda y derecha, con la velocidad deseada y con un intervalo concreto.
  Sus parámetros de configuración son los siguientes:
  - ✓ *Default IP* – IP por defecto
  - ✓ *Normal period* – Periodo de refresco de la orden adelante y atrás
  - ✓ *Normal speed* – Velocidad de las ruedas en movimiento adelante y atrás
  - ✓ *Turn period* – Periodo de refresco del robot en los movimientos izquierda y derecha
  - ✓ *Turn speed* – Velocidad de las ruedas en los movimiento izquierda y derecha.

  A la hora de girar el robot realiza un giro con velocidad *Turn speed* y durante un tiempo de *Turn period*, para luego volver a su estado normal.

  Sin embargo al hacer un movimiento de adelante o atrás, si la petición es contraria al estado actual, el robot se para, si no su velocidad se incrementará la cantidad indicada en *Normal speed*.

- *Image Editor* – Este módulo es el motor de cálculo de la aplicación. Trabaja sobre dos sistemas de colores RGB y HSB. Contiene los siguientes algoritmos de cálculo.
  - o *Calculate Inecuations* – Calcula las inecuaciones que dividen la imagen en parcelas en función de los parámetros nº de divisiones verticales, nº de divisiones horizontales y margen de perspectiva.

    La distancia entre las líneas horizontales es proporcional a la reducción de longitud. Con esto se consigue el efecto de profundidad según las líneas del terreno. Podemos observar el resultado que obtenemos sobre la imagen en la Figura 24.

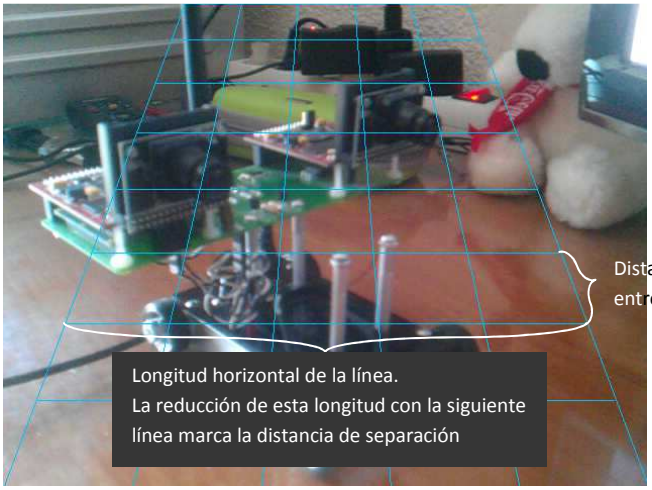

Figura 24. Detalle de proporciones de la maya

- *Calculate mask* – Después del calculo de las inecuaciones, se calcula una mascara de traducción de puntos. Se construye una tabla que dado un píxel (X, Y) te dice si está dentro o no de la maya y en qué fila y columna se encuentra.

mask(x, y) = (-1, -1)	Si el píxel está fuera de la maya
mask(x, y) = (f, c)	Donde f y c son la fila y la columna de la parcela que contiene el píxel

- *Syncronize Color* – Este método sincroniza el color y calcula el modelo gaussiano a seguir para cada uno de los tres canales y para cada uno de los dos modelos usados (RGB y HSB).

  Selecciona una serie de píxeles con la llamada a la función "*Select Cental Form*" (se explicará más adelante) y calcula la media y la desviación típica de las siguientes variables:

  - ✓ Canal Rojo
  - ✓ Canal Verde
  - ✓ Canal Azul

  } RGB

  - ✓ Canal Tono
  - ✓ Canal Saturación
  - ✓ Canal Brillo

  } HSB

- *Select Cental Form* – Dada una imagen selecciona todos los puntos cuya diferencia con el punto central es menor que un umbral. Esta diferencia se evalúa canal a canal para un modelo de color en concreto, en este caso RGB o HSB

- *Calculate Color Index* – Calcula en índice de verde de cada parcela. Para ello por cada píxel decidirá si es verde o no (ya sea por formula de verde o por distribución de Gauss).
  Cada parcela tiene un contador de píxeles totales y de píxeles color. El porcentaje de color, en tanto por uno, de la parcela será el nº de píxeles de color dividido entre el nº de píxeles totales

- *Field Manager* – Este módulo es el encargado de guardar la información relativa al análisis de las imágenes para posteriormente reconstruir el campo virtual. El campo se recompone como la superposición de la primera línea de la maya con un período de anexado igual al refresco de la cámara.

  Para evitar solapamientos, la velocidad de movimiento debe ser el tamaño real de una parcela, dividido entre el periodo de refresco de la cámara.

$$Velocidad = \frac{Tamaño\ de\ parcela}{Periodo\ de\ cámara}$$

3.2.2 Diagrama de clases global

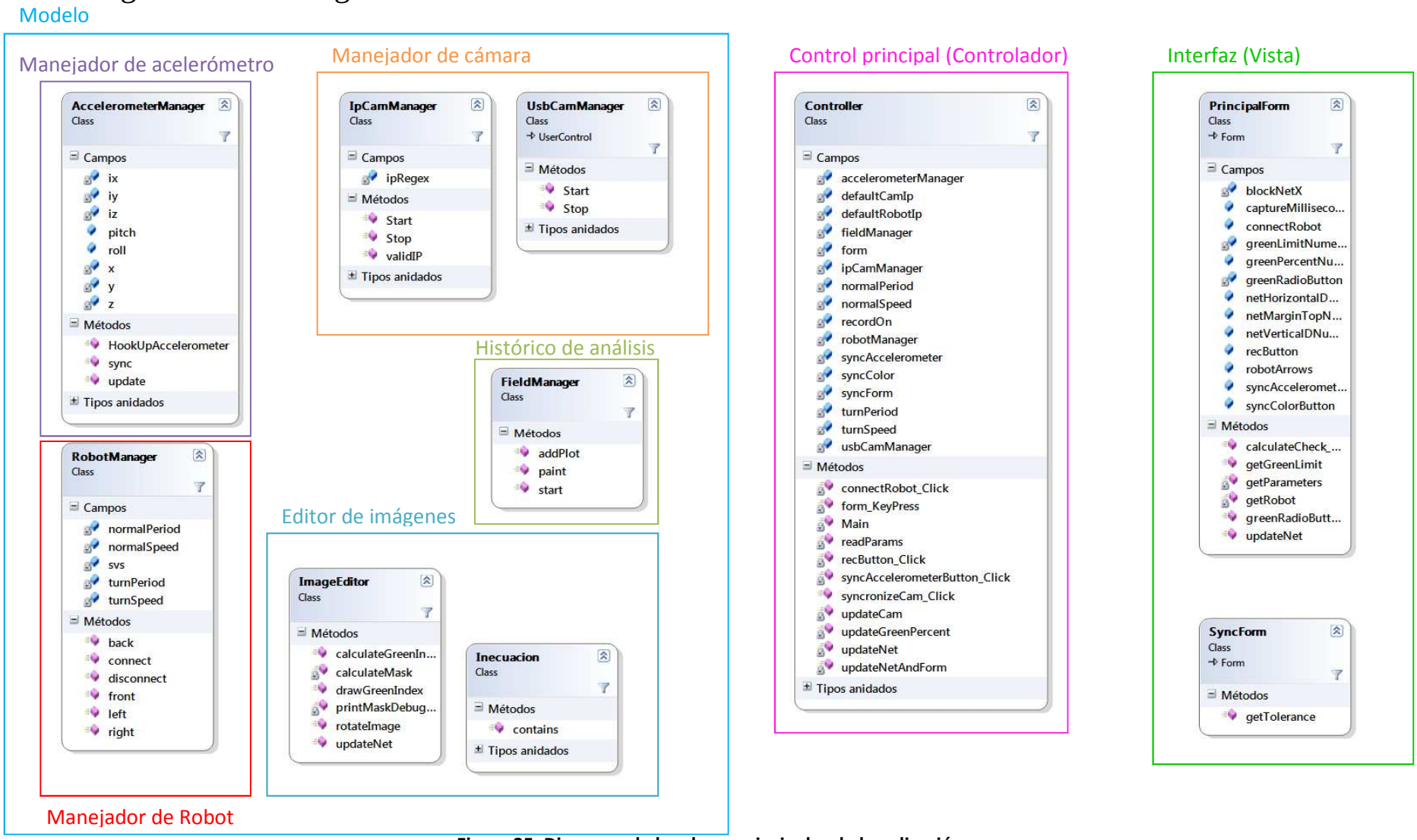

Figura 25. Diagrama de las clases principales de la aplicación

3.2.3 Detalle de los módulos de la arquitectura desarrollados

A continuación detallaremos cada una de las clases que se han presentado en la Figura 25 como componentes de la arquitectura global del sistema, así como la funcionalidad de sus variables y métodos más significativos.

Clases del nivel de Modelo

AccelerometerManager:

Esta clase se encarga del manejo del acelerómetro. Guarda información de los valores actuales (x, y, z) del acelerómetro.

Además permite la puesta a cero en cualquier posición, para ello guarda unos valores de referencia: ix, iy e iz (visibles en la Figura 26).

Delega un evento en el controlador cada vez que se actualizan los datos del acelerómetro.

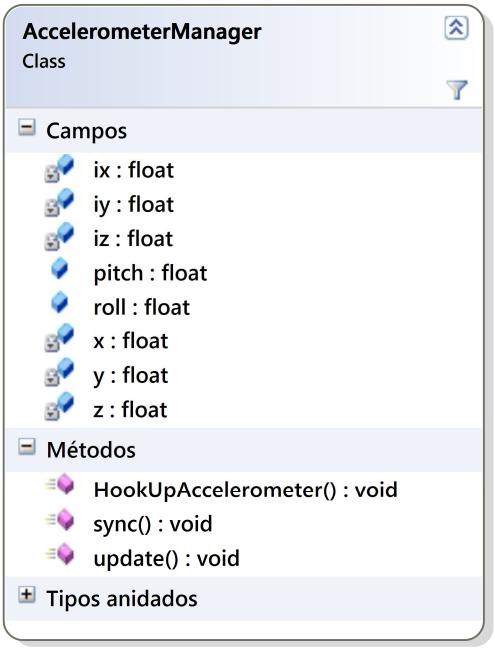

Figura 26. Detalle de la clase AccelerometerManager.

UsbCamManager:

Se encarga de manejar una cámara conectada por USB, mediante los métodos *Start()* y *Stop()* que observamos en la Figura 27.

Delega un evento en el controlador cada vez que se actualizan la imagen actual de la cámara.

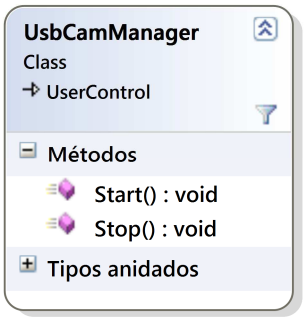

Figura 27. Detalle de la clase UsbCamManager.

IpCamManager:

Se encarga de manejar una cámara conectada por TCP/IP.

Delega un evento en el controlador cada vez que se actualiza la imagen actual de la cámara. Podemos observar en la Figura 28 cómo utiliza el mismo nombre para los métodos de puesta en marcha y apagado de la cámara que la clase anterior. Así como dispone de un método y un campo específicos para comprobar IPs válidas y almacenarlas.

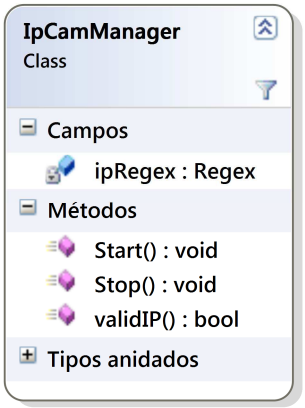

Figura 28. Detalle de la clase IpCamManager.

ImageEditor:

Esta clase es el motor de cálculo de la aplicación. Contiene numerosos campos y métodos como se observa en la Figura 29, los más importantes son:

- Factores de corrección empíricos
    - *angleCorrectionFactor*
    - *rollXCorrectionFactor*
    - *rollYCorrectionFactor*
    - *pitchCorrectionFactor*
- Red de inecuaciones para la división del campo visible en parcelas
    - hI(*Horizontal inecuations*) – Divisiones horizontales del campo
    - vI(*Vertical inecuations*) – Divisiones verticales del campo
    - mask – Máscara que mapea píxeles con parcelas del campo visible
- Valores resultados del sincronismo de colores:
    - rgbMean – Media aritmética de los tres canales RGB
    - rgbDeviation – Desviación típica de los tres canales RGB
    - hsbMean – Media aritmética de los tres canales HSB
    - hsbDeviation – Desviación típica de los tres canales HSB

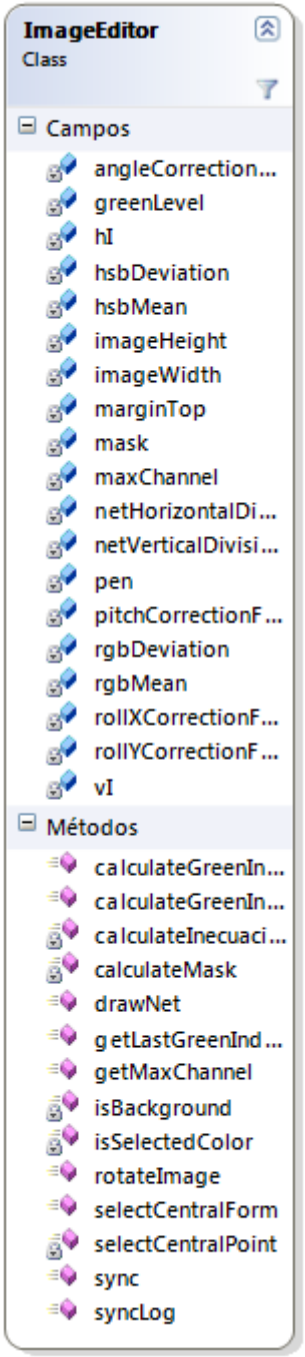

Figura 29. Detalle de la clase ImageEditor.

Inecuación:

Clase que contiene el cálculo de una inecuación. Como se observa en la Figura 30 dispone de un método denominado *contains* para poder determinar con él a qué parcela pertenece cada píxel.

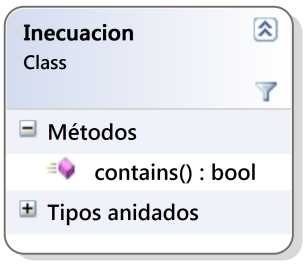

Figura 30. Detalle de la clase Inecuación.

Robot Manager:

Clase que contiene los métodos de control del robot. Los campos que se observan en la Figura 31 serán tratados en detalle en el de Manual de Usuario de la aplicación.

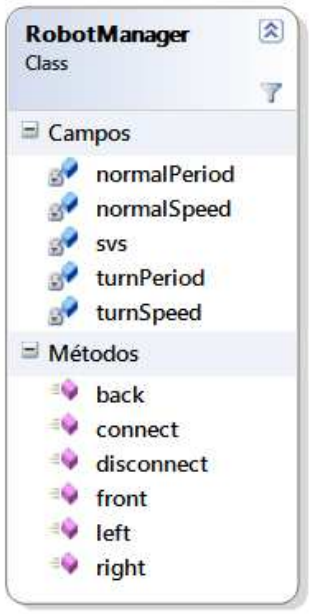

Figura 31. Detalle de Clase RobotManager

Field Manager:

Clase para el almacenamiento de los datos procesados y la posterior representación del campo virtual gracias al método *addPlot* que se observa en la Figura 32.

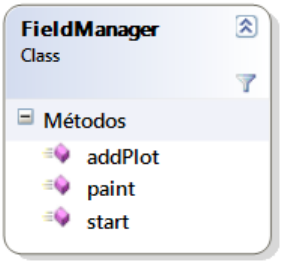

Figura 32. Detalle de Clase FieldManager

Clases del nivel de Vista

PrincipalForm:

Formulario principal que supondrá la interfaz de visión de todas las opciones del proyecto. Se pueden observar dos componentes del mismo en la Figura 33 utilizados para la puesta a 0 de los datos del acelerómetro y para la captura de las componentes de un nuevo color como referencia.

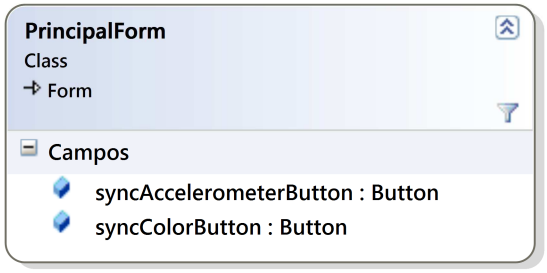

Figura 33. Detalle de la clase PrincipalForm.

SyncForm:

Formulario de sincronismo de color. En él se podrá indicar la tolerancia al mismo para posteriormente poderse obtener mediante el método *getTolerance()* que observamos en la Figura 34

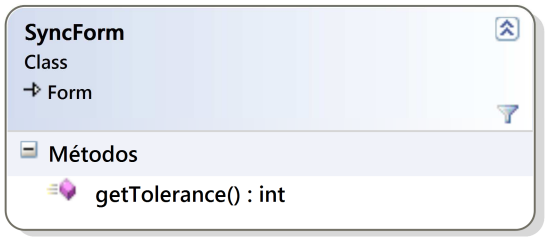

Figura 34. Detalle de la clase SyncForm.

Clases del nivel de Controlador

Controller:

Clase cuyos campos y métodos se observan en la Figura 35, encargada de interconectar los demás módulos: por un lado la parte del acelerómetro, por otro la parte de cámara, tanto por USB como por TCP, y por ultimo la interfaz.

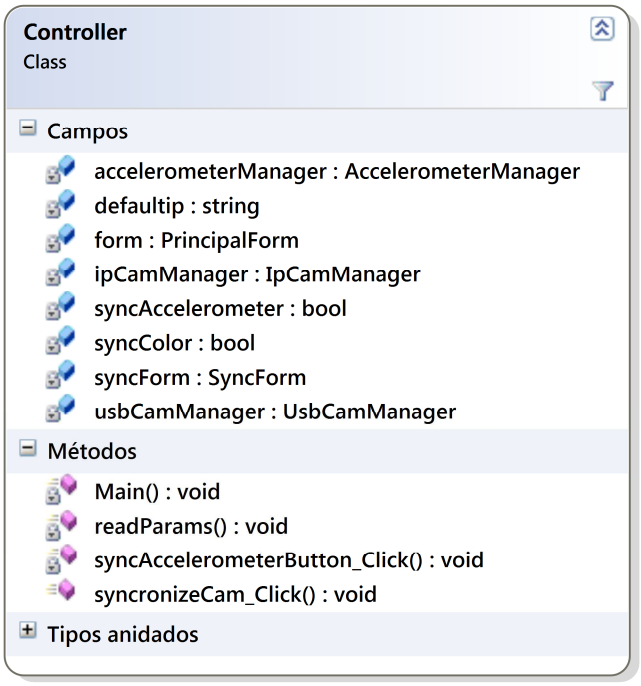

Figura 35. Detalle de la clase Controller.

Capítulo 4

Servicios y tecnologías utilizadas

4.1 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

4.2 PCRemoteAdvance (3.9 - 29/08/2011)*

PCRemote es una aplicación libre para Android desarrollada por FreeVanx. Permite obtener información de diversos sensores siempre y cuando estén disponibles. Su versión *Advance* permite obtener las siguientes funciones entre otras:

- ✓ Mapear la localización del dispositivo vía GPS o Wifi.
- ✓ Obtener los parámetros del acelerómetro del dispositivo.
- ✓ Obtener los parámetros de la brújula magnética del dispositivo.
- ✓ Obtener los parámetros del giroscopio del dispositivo.
- ✓ Obtener los parámetros del sensor de luz del dispositivo.

** Versión y fecha de publicación incluida en la última release del proyecto.*

4.3 IP WebCam (1.8.15 - 06/05/2012)*

IP Webcam es una aplicación para Android desarrollada por Pas. Permite la conexión, en tiempo real, con la cámara del terminal.
El envío de la señal se hace mediante el protocolo TCP/IP y se accede mediante la dirección asignada al terminal en la red.

La aplicación permite las siguientes configuraciones:

- Resolución configurable (480x320 – 960x720)
- Puerto utilizado para el envío
- *Login/password* para proteger.
- Flash LED

** Versión y fecha de publicación incluida en la última release del proyecto.*

4.4 IMU (*Inertial Measurement Unit*)

Figura 36. Acelerómetro externo

Una IMU (*inertial measurement unit*) o unidad de medición Inercial, es un dispositivo electrónico como el que se puede observar en la Figura 36, el cual mide e informa acerca de la velocidad, orientación y Fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giroscopios.

La tarjeta IMU es el componente principal de los sistemas de navegación inercial, usados en aviones, barcos, etc., su gran ventaja es la medición casi instantánea en la aceleración y el giro, en sus 2 o 3 ejes, aceleración en el eje x, y, z además de *pitch, roll* y *yaw*.

La desventaja de la misma, es que las tarjetas IMU acumulan un error conocido como ABBE, que en palabras sencillas, si la IMU encuentra un error por pequeño que sea, se irá acumulando poco a poco des calibrando a la tarjeta en si, Esto lleva a una 'deriva', o a una diferencia que aumenta siempre entre donde el sistema piensa que se encuentra localizado y la posición real.

Debido a esto generalmente los sistemas de navegación cuentan con un sistema de auto calibración que les permite compensar la posición del dispositivo o las IMU actúan como apoyo a sistemas de navegación global como los GPS.

4.5 Repositorio

Para el mantenimiento del proyecto se ha elegido utilizar la herramienta Subversion, la cual es una herramienta de libre distribución que permite el control de versiones de elementos *software*. Dicho sistema facilita el control de versiones de los distintos elementos *software* seleccionados por los miembros del proyecto. Está especialmente orientado a elementos conformados por texto, sobre los que permite realizar un buen manejo, control y gestión de sus modificaciones.

En concreto el proyecto se ha alojado en el servidor de control de versiones http://xp-dev.com.
Para el tráfico con el servidor, se ha empleado el cliente *Open Source* "Tortoise SVN" de Tigris.

Capítulo 5

Manual de Usuario

Para poder ejecutar la aplicación creada correctamente es necesario poseer ciertos dispositivos específicos, así como seguir una serie de instrucciones, para poder así inicializar todos los elementos necesarios y disponer de todas las funcionalidades de la herramienta principal. A continuación enumeraremos los requisitos mínimos que necesita nuestra aplicación para poder ser disfrutada en su totalidad.

5.1 Requisitos hardware:

- Dispositivo móvil:
  - Deberá se compatible con la versión 2.1 o superiores del SO Android.
  - Disponer de conexión 3G, WiFi o Bluetooth.
  - Disponer de acelerómetro.
  - Disponer de cámara (opcional).
- Dispositivo fijo (PC):
  - Deberá poder soportar el SO Windows XP u otros superiores (32 o 64 bits).
  - Disponer de cámara (opcional si la tiene ya el dispositivo móvil).

5.2 Instrucciones de configuración:

1. Deberá descargarse e instalarse la aplicación PC Remote Advance para el dispositivo Android desde la tienda de Android/Google Play (https://play.google.com/store/apps/details?id=org.linknet.pcremote&feature=search_result)

2. (Opcional) Descargarse e instalar la aplicación IP Webcam al dispositivo móvil. https://play.google.com/store/apps/details?id=com.pas.webcam&feature=search_result

3. Descargar el servidor remoto en el PC según la arquitectura del mismo, desde:

   a. Windows XP y superiores (32 bits): http://feishare.com/file/remotesensors/RemoteSensors_setup_32bit.exe

   b. Windows XP y superiores (64 bits): http://feishare.com/file/remotesensors/RemoteSensors_setup_x64_beta.exe

4. Instalar el servidor remoto en el PC. (Video explicativo: http://www.youtube.com/watch?v=U0jLu7a-PFI).

5. Conectar el dispositivo móvil y el PC a través de la misma red 3G, WiFi o Bluetooth.

6. (Opcional) Si se cumplió el paso 2 de estas instrucciones para aprovechar la cámara del dispositivo móvil se deberá iniciar la aplicación y seguir los siguientes pasos detallados:

    a. Configurar la resolución de la cámara a 320x240 (o el que se desee) y la orientación a apaisada (*Landscape*) en la sección de preferencias de video de la cual se ha hecho un recorte en la Figura 37.

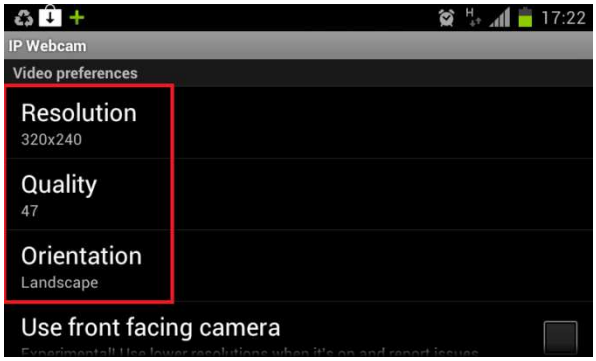

Figura 37. Captura IP Webcam

    b. Empezar a enviar imágenes -> *Start server*

    c. Guardar la IP y el puerto que nos salen por pantalla sobre el área marcada en la Figura 38 para introducirlos posteriormente en la aplicación.

Figura 38. Captura IP Webcam

    d. Poner la aplicación de envío de imágenes en *background* bien mediante algún atajo directo del móvil o pulsando *Actions -> Run in background* tal cual se remarca en la Figura 39.

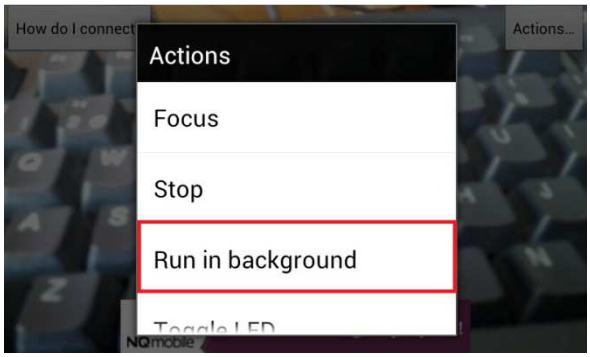

Figura 39. Captura IP Webcam

7. Ejecutar la aplicación PCRemote Server instalada previamente en nuestro PC a través del ejecutable generado tras la instalación: PCRemote.exe.

8. Iniciar la aplicación PCRemote Advance en nuestro móvil. Para ello habrá que:

a. Pulsar *Discovery Server* (remarcado en la Figura 40) e introducir bien la IP privada del PC (WiFi o 3G) al que se quiere conectar o probar la conexión automática (BlueTooth).

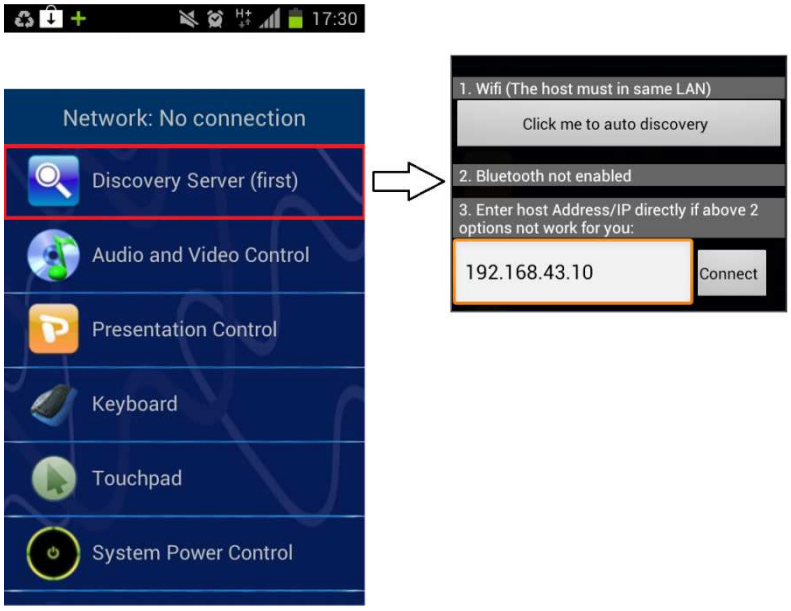

Figura 40. Captura PCRemote Advance

b. Pulsar en Sensors to PC, tras lo cual nos aparecerá una pantalla como la de la Figura 41:

Figura 41. Captura PCRemote Advance

9. Por último debemos ejecutar la aplicación desarrollada: Webcam_Capture.exe, nos aparecerá una ventana como la de la Figura 42.

   a. Una vez abierta se deberían reflejar los datos del acelerómetro que ya se encuentran enviando. Se puede activar y desactivar desde el propio dispositivo móvil.

   b. Si pulsamos el botón On que nos dará la opción de utilizar la cámara del móvil o una cámara USB.

      i. Si escogemos la cámara USB debemos asegurarnos que esta estaba previamente configurada.

      ii. Si escogemos la cámara del móvil nos pedirá la dirección IP y el puerto por el que está enviando la cámara del móvil y que, recordemos, guardamos anteriormente.

   c. (Opcional) Podemos sincronizar el acelerómetro para reiniciar a 0 todas las coordenadas, según la posición actual del acelerómetro.

10. Una vez llegado a este punto podremos elegir opcionalmente las distintas funcionalidades del aplicativo que detallaremos a continuación.

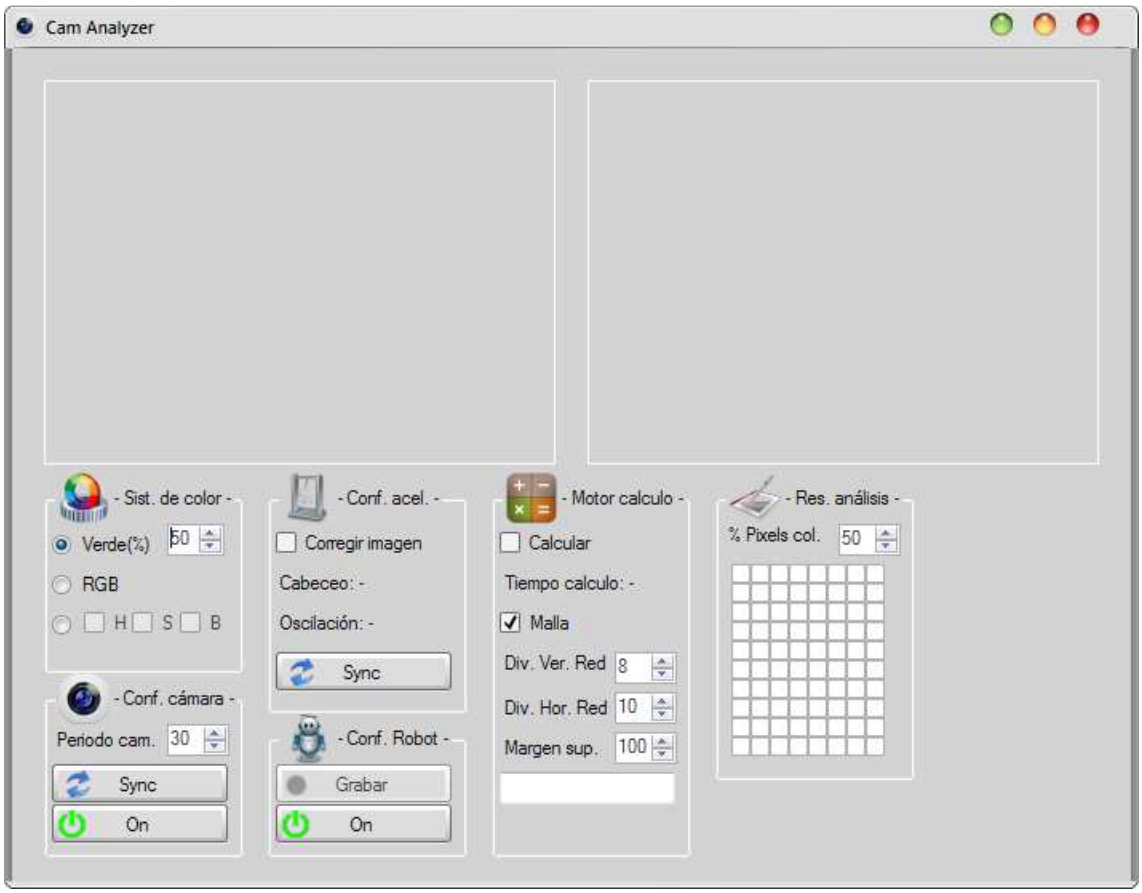

Figura 42. Estado inicial de la herramienta

5.3 Instrucciones de uso:

A continuación se expondrá una lista con todas las funcionalidades que ofrece al usuario la herramienta desarrollada, así como la utilidad de los distintos parámetros de entrada de la interfaz y del archivo de configuración: para cada funcionalidad.

5.3.1 Corrección de las irregularidades del terreno

Prerrequisitos: Tener configurados y en ejecución la cámara (USB o IP) y el dispositivo con acelerómetro.

Dentro del archivo de configuración config.xml de nuestra herramienta se podrán variar ciertos parámetros relacionados directamente con esta funcionalidad.

```
<config>
    <image  width="320"
            height="240"/>

    <cam    captureMilliseconds="30"
            defaultIp="192.168.1.35:8080"
            login=""
            pass=""/>

    <net    netVerticalDivisions="8"
            netHorizontalDivisions="10"
            netMarginTop="100"/>

    <imageEditor greenPercent="50"/>

    <accelerometer  pitchCorrectionFactor="-7,3"
                    rollXCorrectionFactor="-0,79"
                    rollYCorrectionFactor="1,19"
                    angleCorrectionFactor="-0,27"/>

    <robot  defaultIp = "169.254.0.10"
            normalPeriod = "1"
            normalSpeed = "5"
            turnPeriod = "2"
            turnSpeed = "100"/>
</config>
```

Figura 43. Parámetros del archivo de configuración significativos para la corrección de irregularidades.

Como se observa en la Figura 43, desde este archivo se pueden modificar parámetros para:

- **Formato de la imagen**. Se puede cambiar la resolución de la interfaz para ajustar a ella las imágenes capturadas y enviadas por la cámara. La resolución por defecto será de 320x240 píxeles.

- **Parámetros de la cámara**. Se podrá modificar desde este archivo de configuración la frecuencia de refresco de la cámara, así como la dirección IP por defecto de ésta. Estos parámetros se podrán modificar previamente desde la propia herramienta.

- **Parámetros de corrección del acelerómetro**. Estas variables hacen referencia a los factores a os que ya se hizo referencia en el primer apartado de esta memoria. Los valores que aparecen en la captura han sido tomados para una altura de cámara de 10 centímetros aproximadamente. La función de cada uno de estos factores es:

    - *pitchCorrectionFactor.* Factor de desplazamiento aplicable a la transformación de traslación motivada por el problema de cabeceo.

    - *rollXCorrectionFactor.* Factor de desplazamiento horizontal aplicable a la transformación de giro motivada por el problema de oscilación.

- *rollYCorrectionFactor.* Factor de desplazamiento vertical aplicable a la transformación de giro motivada por el problema de oscilación.
- *angleCorrectionFactor.* Factor que multiplicado por el valor de oscilación nos da el ángulo de giro para la corrección.

A continuación se indicará cuales son los pasos a seguir dentro de la herramienta para aplicar la corrección de las desviaciones de los ejes producidas por la irregularidad del terreno.

1. Comprobar que los datos del acelerómetro se están recibiendo correctamente. Se podrá observar si el acelerómetro está funcionando correctamente si en el apartado "Conf. acel." de la interfaz se obtienen con alta frecuencia valores nuevos de las variables de oscilación y cabeceo (Figura 44).

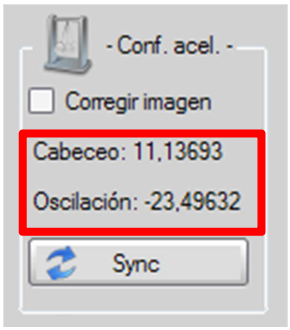

Figura 44. Valores recibidos desde el acelerómetro.

2. A continuación se procederá a conectar una cámara, tal cual se indicó en el apartado de instrucciones de configuración y como se explica gráficamente en la Figura 45. Como se explicó antes, se podrá volver a introducir una nueva IP correspondiente a la cámara si fuese necesario.

Figura 45. Secuencia de activación de la cámara IP.

3. Se podrá modificar la frecuencia de refresco de la cámara desde la sección "Conf. cámara" cambiando el valor del área marcada de la Figura 46. Al igual que se hiciese desde el archivo de configuración.

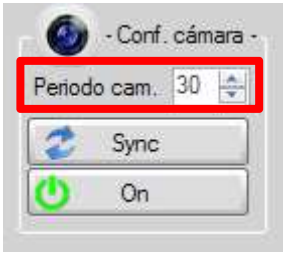

Figura 46. Opción de modificación del refresco de la cámara.

4. Opcionalmente podremos actualizar los ejes de referencia del acelerómetro a la posición actual del dispositivo, por lo que los valores de cabeceo y oscilación quedarán reseteados a 0. Para ello pulsaremos el botón "Sinc. acelerómetro" remarcado en la Figura 47.

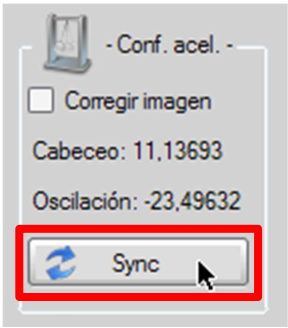

Figura 47. Botón de sincronización de los ejes con la posición actual del acelerómetro.

5. Una vez explicadas todas las opciones iniciales para esta funcionalidad procederemos a la acción de rectificación de la imagen marcando el *checkbox* "Corregir" quedando como en la Figura 48.

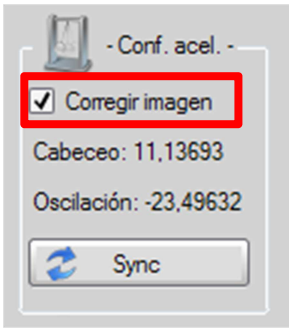

Figura 48. *Checkbox* "Corregir" activado.

6. Tras este último paso observaremos como a la segunda imagen de la interfaz se le aplican las transformaciones necesarias para ser rectificada, tal cual sucede en la Figura 49.

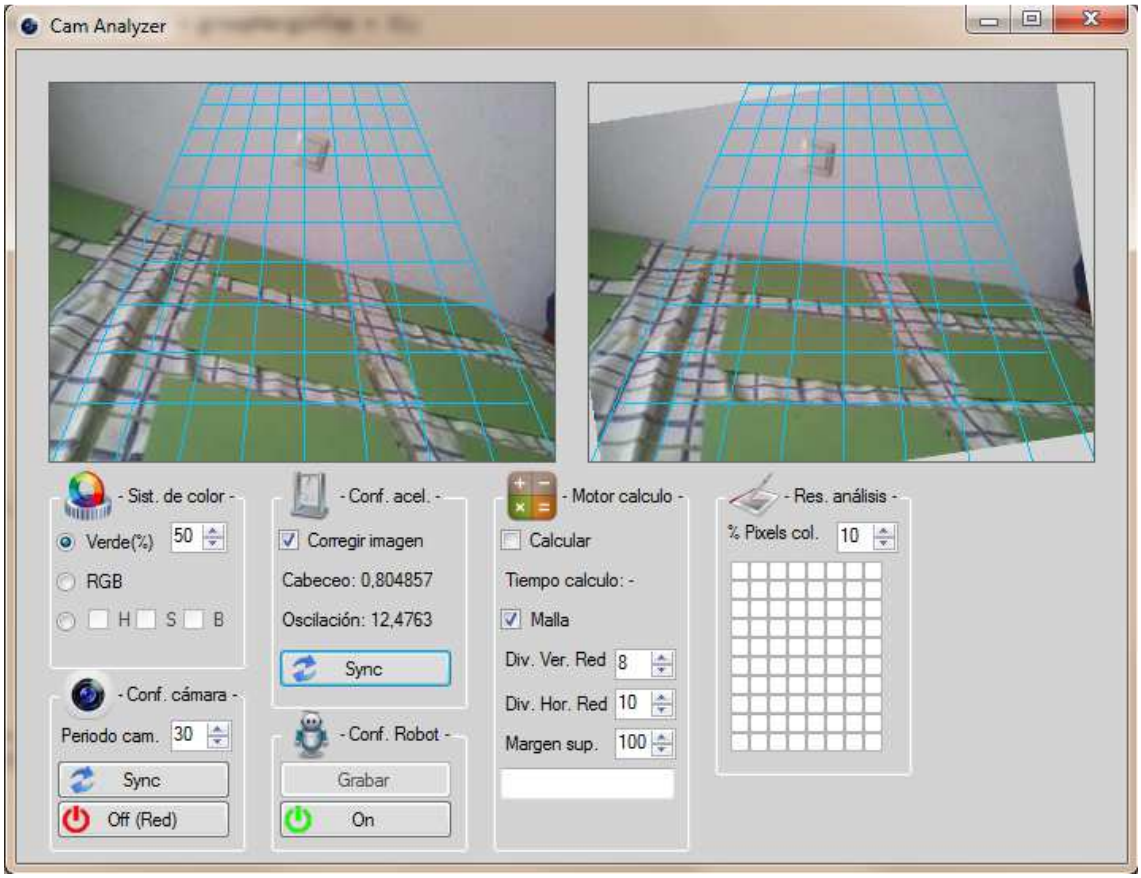

Figura 49. Transformación de una imagen para la corrección de la irregularidad del terreno.

5.3.2 Segmentación de la imagen capturada

Prerrequisitos: Tener configurada y en ejecución la cámara (USB o IP).

Los parámetros modificables para esta funcionalidad dentro del archivo de configuración XML se detallan a continuación en la Figura 50:

```
<net    netVerticalDivisions="8"
        netHorizontalDivisions="10"
        netMarginTop="100"/>
```

Figura 50. Parámetros del archivo de configuración significativos para la referenciación del terreno por parcelas.

- *netVerticalDivisions*. Valor correspondiente al número de divisiones verticales en las que se dividirá la malla que cubre la imagen del terreno.
- *netHorizontalDivisions*. Valor correspondiente al número de divisiones horizontales en las que se dividirá la malla que cubre la imagen del terreno.
- netMarginTop. Valor correspondiente a la distancia en píxeles entre los extremos superiores de la malla y los extremos superiores correspondientes de la captura. Esta

distancia, denominada también margen de captura, servirá para ajustarse a la parte de la imagen que deberá ser analizada.

A continuación se expondrán los pasos a seguir dentro de la herramienta aplicar la segmentación en áreas sobre la imagen capturada y opcionalmente corregida previamente. Para esta secuencia de pasos se partirá de la premisa de que la cámara ya ha sido previamente conectada y activada.

1. La activación de la malla que segmenta la imagen capturada se realizará por defecto una vez que se inicia la herramienta. Para desactivarla bastará con deseleccionar el *checkbox* con la etiqueta "Malla" marcado en la Figura 51.

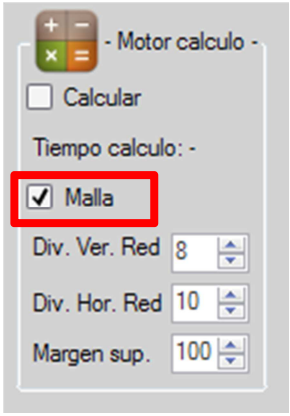

Figura 51. *Checkbox* para activar/desactivar la malla sobre la imagen.

2. Observaremos que ambas imágenes de nuestra interfaz: con corrección y sin corrección son modificadas con la superposición de una malla sobre ellas, tal cual se observa en la Figura 52.

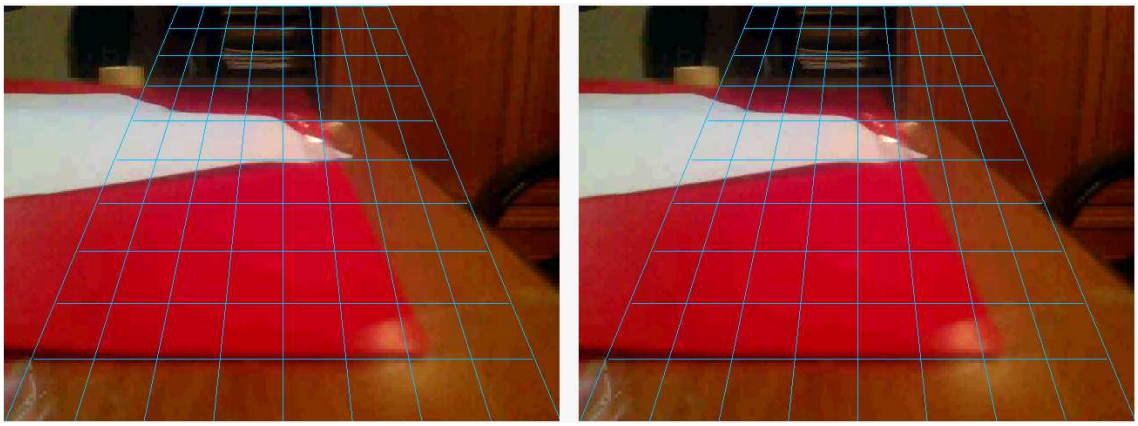

Figura 52. Malla superpuesta sobre la imagen capturada/corregida.

3. Cabe decir, que los parámetros que fueron ya definidos en el archivo de configuración, también son modificables desde la interfaz desde la sección "Motor calculo". Los parámetros modificables se remarcan en la Figura 53.

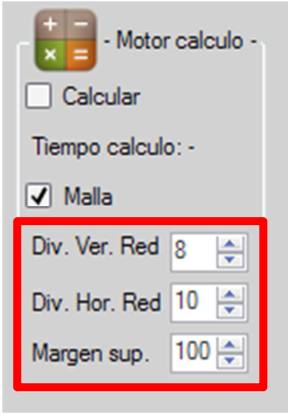

Figura 53. Parámetros de la malla modificables desde la interfaz.

5.3.3 Análisis de color de la imagen

Prerrequisitos: Tener configurada y en ejecución la cámara (USB o IP). Tener activada la malla.

Analizaremos, como siempre, el único parámetro que se puede modificar para esta funcionalidad desde el archivo config.xml: *greenPercent.* Este parámetro, mostrado en la Figura 54, nos apostará el porcentaje de color (verde) mínimo de una parcela para que esta sea considerada como una parcela verde.

```
<imageEditor greenPercent="10"/>
```

Figura 54. Parámetros del archivo de configuración significativos para el análisis de color de la imagen.

Una vez situados en la interfaz de nuestra herramienta, y con la premisa de que se cumplan todos los prerrequisitos previamente listados, se indicará cuál es la secuencia lógica de pasos a seguir dentro de la herramienta para aplicar el análisis de color sobre cada una de las parcelas en la que se divide la imagen.

1. Se elegirá el modelo de color sobre el que se analizará cada píxel, en función de las condiciones ambientales o de la predilección del usuario a través del *radiobutton* remarcado en la Figura 55. Se puede observar en el Apéndice A – Estudio de Modelos de Color de este documento, las pruebas que fueron realizadas en una versión temprana de esta herramienta para analizar el comportamiento de los distintos modelos de color.

Figura 55. Modelos de color disponibles para analizar la imagen.

Cada una de las opciones de este *RadioButton* corresponde a uno de los siguientes modelos de color:

- **Verde (%).** Esta opción no se trata de la aplicación estricta de un modelo de color, sino que corresponde a la admisión de un píxel como verde si cumple la fórmula de la función de verde vegetal que ya se contempló en el primer apartado de esta memoria:

$$\mathbf{2}.\boldsymbol{G} - \boldsymbol{R} - \boldsymbol{B} > Umbral$$

  El valor modificable en la interfaz corresponde al valor umbral sobre el que se considerará o desestimará un píxel como verde.

- **RGB.** Si se selecciona este modelo de color se realizará la detección de un color mediante el cálculo de la aproximación gaussiana de las componentes de cada píxel al valor sincronizado (si no hay sincronización, será a las componentes RGB [0, 0 ,0]). Posteriormente se determinara si el pixel esta dentro de la distribución normal, es decir esta en la media aritmética $\pm$ desviación típica. Se comprueba cada canal por separado tanto RGB como HSB

$$Pixel.R > Media(R) - DesviacionTipica(R)$$

$$/\backslash$$

$$Pixel.R < Media(R) + DesviacionTipica(R)$$

  Donde cada una de las tres componentes (R, G y B) en este caso deben cumplir la condición expuesta para ser incluidas dentro del rango de admisión. Este rango estará determinado por el valor Tolerancia, inicializado por defecto en nuestra herramienta a un valor del 10%.

- **HSB.** Análogo al modelo RGB, si se selecciona se realizará la determinación de validez de un píxel el mediante la misma fórmula. Sin embargo, esta se realizará, lógicamente, con las componentes (H, S y B) de la imagen. Además en este caso podemos elegir omitir la consideración de cada una de las componentes de este modelo. El motivo de esto es la mejoría de los resultados obtenidos al eliminar la componente B (*Brightness)* en cada una de las pruebas de color realizadas.

2. Opcionalmente, se podrá realizar una elección de un píxel de referencia nuevo para el cálculo de la desviación producida para cada componente del modelo. Asimismo, se podrá indicar un nuevo valor de tolerancia (%), el cuál supondrá el nuevo límite de admisión de un píxel como cercano al píxel referencia. Para ello se deberá acceder a esta funcionalidad a partir del botón “Sinc. Cámara” situado en la sección “Info” tal cual se muestra en la Figura 56.

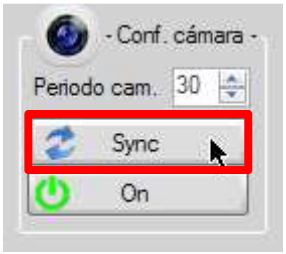

Figura 56. Botón para sincronizar un nuevo píxel de referencia.

Tras esta acción se nos abrirá una nueva ventana donde podremos modificar los valores que se han expuesto previamente. En esta misma ventana, que tendrá el aspecto mostrado en la Figura 57, se podrá un observar una previsualización de los píxeles que serían admitidos.

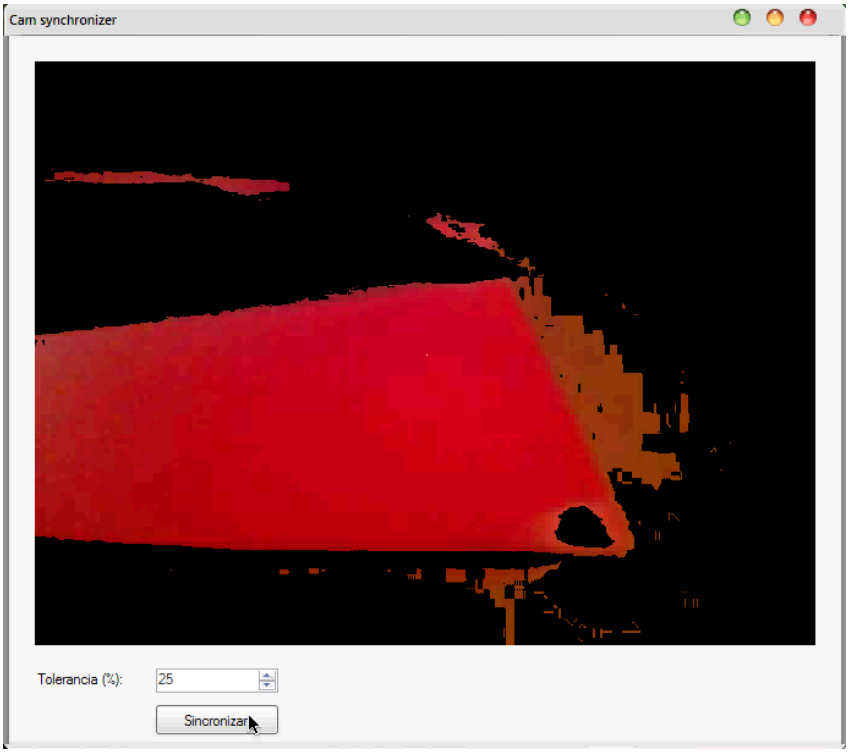

Figura 57. Pantalla de sincronización de un nuevo píxel de referencia.

La admisión de esos pixeles se hace en función de su proximidad en tono al punto central. Es decir se mira si la diferencia de cada canal, tanto en RGB como en HSB, es menor que un porcentaje de tolerancia.

$$|\boldsymbol{Pixel.R} - \boldsymbol{Referencia.R}| > \frac{ComponenteR.MáximoValor * Tolerancia}{100}$$

$$|\boldsymbol{Pixel.G} - \boldsymbol{Referencia.G}| > \frac{ComponenteG.MáximoValor * Tolerancia}{100}$$

$$|\boldsymbol{Pixel.B} - \boldsymbol{Referencia.B}| > \frac{ComponenteB.MáximoValor * Tolerancia}{100}$$

3. Podremos ajustar a través de nuestra interfaz el valor del parámetro que ya fue inicializado desde el archivo de configuración a través del valor remarcado en la Figura 58.

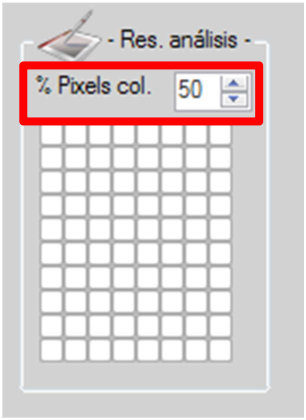

Figura 58. Parámetro para el tratamiento del color modificable desde la interfaz.

4. Tras estas acciones previas, se activará el cálculo de color de la imagen capturada a través del *checkbox* "Calcular" (marcado en la Figura 59) y se mostrara el tiempo necesario para dicho cálculo.

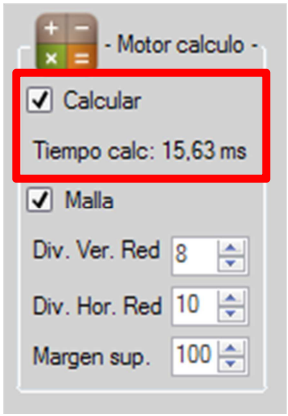

Figura 59. *Checkbox* "Calcular" activado.

Una vez seleccionada esta opción, observaremos como la imagen capturada (y rectificada) de nuestra interfaz es complementada con el porcentaje de píxeles admitidos por cada parcela de la malla (Figura 60). También, se podrán observar las parcelas admitidas a través de una matriz gráfica que reflejará esta información como se muestra en la Figura 61 a modo de ejemplo.

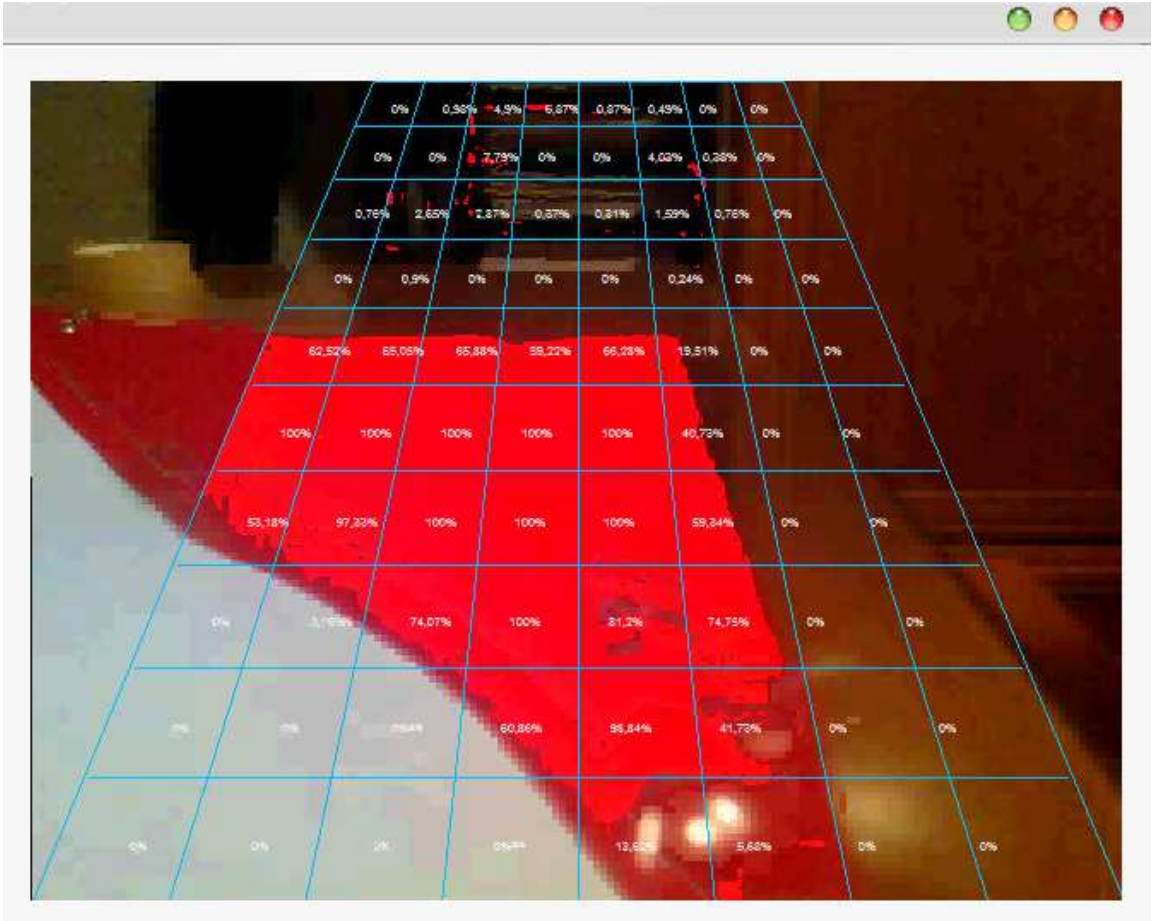

Figura 60. Recorte de la imagen analizada por la herramienta.

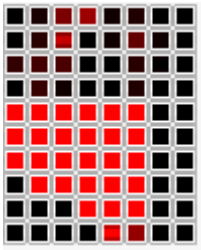

Figura 61. Matriz gráfica.

5. Por último, podremos capturar la información obtenida de este análisis en un período de tiempo. Para ello pulsaremos el botón “Grabar” (mostrado en la Figura 62).

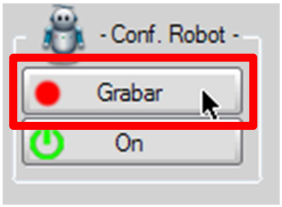

Figura 62. Botón "Grabar" para registrar los valores obtenidos.

Esta opción nos permite registrar a determinada frecuencia la información de la primera fila (fila inferior) de celdas analizada. Esta información es útil si es aplicada a la par que el robot cuyo manejo se expondrá en el siguiente apartado.

6. Para obtener los resultados almacenados de la captura de la información extraída del análisis de color, deberemos pulsar el mismo botón sobre el que ahora aparece escrita la palabra “Parar”. Una vez pulsado, aparecerá una nueva ventana con la información que se ha sido registrada hasta ese momento con el mismo aspecto que el recorte mostrado en la Figura 63.

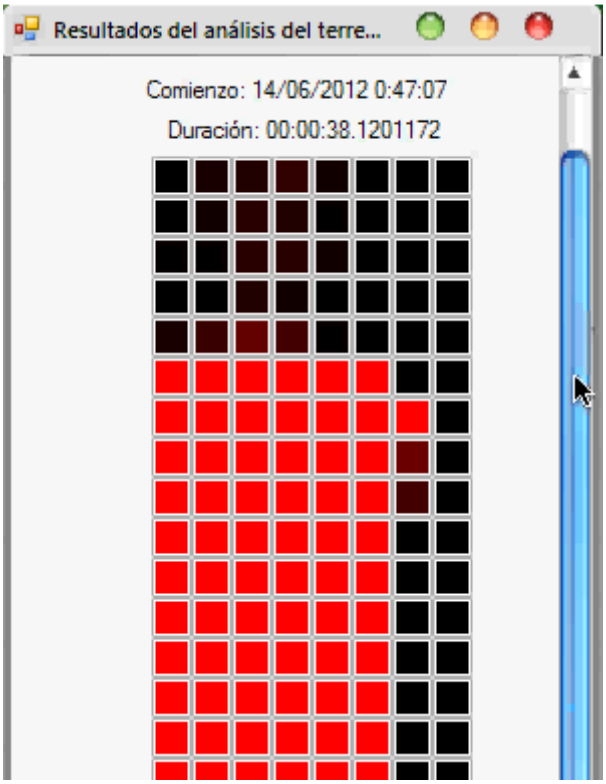

Figura 63. Recorte de la información obtenida tras una prueba con robot.

5.3.4 Manejo del robot móvil

Prerrequisitos: Disponer en la misma red el robot móvil y el computador donde se ejecuta la herramienta.

Esta funcionalidad también dispone de parámetros configurables previamente a la ejecución de la herramienta. El fragmento del archivo de configuración editable para configurar dichos parámetros se muestra en la Figura 64.

```
<robot  defaultIp = "169.254.0.10"
        normalPeriod = "1"
        normalSpeed = "5"
        turnPeriod = "2"
        turnSpeed = "100"/>
```

Figura 64. Parámetros del archivo de configuración significativos para el manejo del *robot* móvil.

Entre estos parámetros podemos encontrar:

- *defaultIp* – Será la IP que se utilizará por defecto para la conexión del robot con la herramienta.
- *normalPeriod* – Período entre dos señales enviadas al robot cuando este se encuentra en movimiento recto.
- *normalSpeed* – Velocidad del robot en movimiento recto.
- *turnPeriod* – Período entre dos señales enviadas al robot cuando este se encuentra en un movimiento de giro.
- *turnSpeed* - Velocidad del robot en giro.

Una vez situados sobre la interfaz de nuestra herramienta, la única acción concerniente a esta funcionalidad reside en el botón "On" (remarcado en la Figura 65).

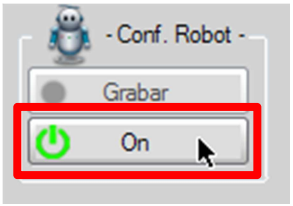

Figura 65. Botón "On" para empezar la comunicación con el robot.

Podremos introducir la IP y el puerto para la comunicación del Robot desde nuestra interfaz, bien a través del área de texto situado encima del botón o a través del mensaje de diálogo que nos aparecerá tras pulsarlo.

Una vez que el robot se haya conectado correctamente, podremos controlarlo de forma teledirigida mediante las fechas del teclado.

Capítulo 6

Proceso de desarrollo

6.1 Scrum

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Onaka a mediados de los 80.

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; preocupaciones frecuentes en el desarrollo de determinados sistemas de *software*.

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Scrum es una metodología ágil, y como tal:

- ✓ Es un modo de desarrollo de carácter adaptable más que predictivo.
- ✓ Orientado a las personas más que a los procesos.
- ✓ Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (normalmente de 30 días).

Cada uno de estos periodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto. Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el abajo realizado el día anterior y el previsto para el día siguiente. En nuestro caso estas reuniones son las acontecidas cada dos semanas con el tutor del proyecto.

Scrum controla de forma empírica y adaptable la evolución del proyecto, empleando las siguientes prácticas de la gestión ágil:

Revisión de las Iteraciones. Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.

Desarrollo incremental. Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

Desarrollo evolutivo. Los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos. Intentar predecir en las fases iniciales cómo será el producto final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.

Para qué predecir los estados finales de la arquitectura o del diseño si van a estar cambiando. En Scrum se toma a la inestabilidad como una premisa, y se adoptan técnicas de trabajo para permitir esa evolución sin degradar la calidad de la arquitectura que se irá generando durante el desarrollo.

El desarrollo Scrum va generando el diseño y la arquitectura final de forma evolutiva durante todo el proyecto. No los considera como productos que deban realizarse en la primera "fase" del proyecto. (El desarrollo ágil no es un desarrollo en fases)

Auto-organización. Durante el desarrollo de un proyecto son muchos los factores impredecibles que surgen en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.

En Scrum los equipos son auto-organizados (no auto-dirigidos), con margen de decisión suficiente para tomar las decisiones que consideren oportunas.

Colaboración. Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto-organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

Las reuniones

Scrum denomina "*sprint*" a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días.

- ✓ Planificación de *sprint*: Jornada de trabajo previa al inicio de cada *sprint* en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en esa iteración.
- ✓ Reunión diaria: Breve revisión del equipo del trabajo realizado hasta la fecha y la previsión para el día siguiente.
- ✓ Revisión de *sprint*: Análisis y revisión del incremento generado.

6.2 Versiones de la herramienta

El proyecto se ha desarrollado en 8 meses (desde el 22/10/2011 hasta el 19/05/2012 aprox.)

El repositorio SVN ha recibido aproximadamente 150 revisiones repartidas en el tiempo según se muestra en la Figura 66.

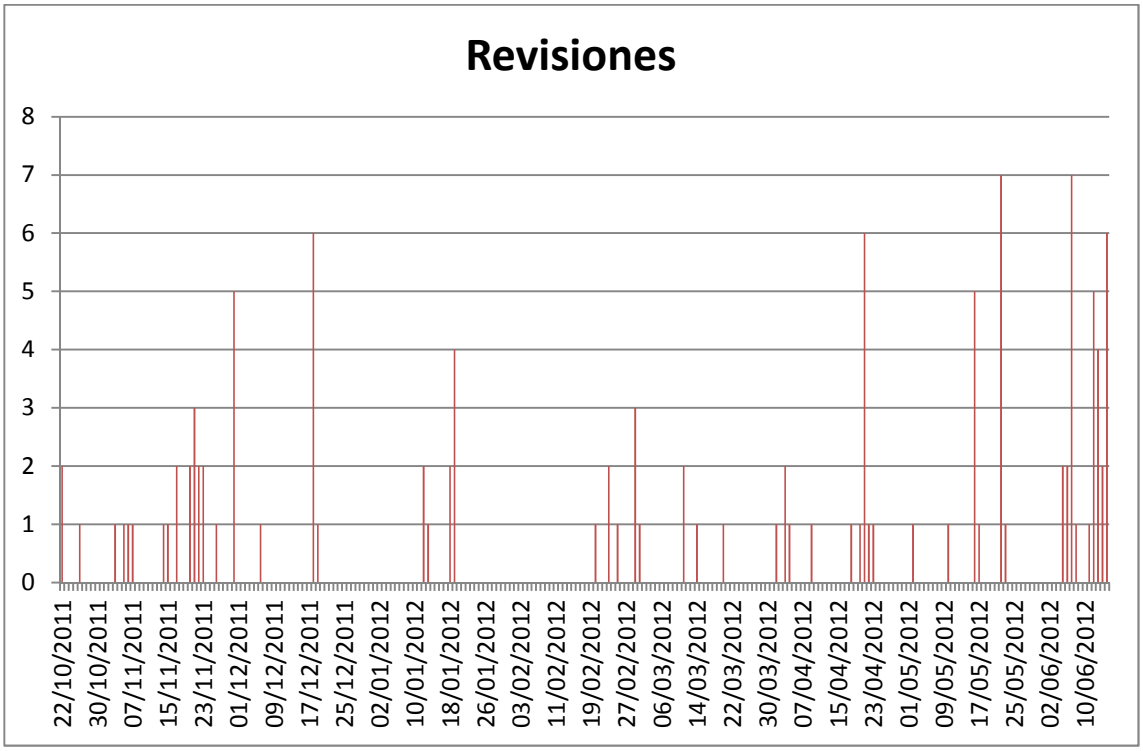

Figura 66. Revisiones realizadas en el proyecto

Hitos del proyecto:

- 06/11/2011 Aplicación carga imágenes de muestra fijas
- 07/11/2011 Añadido método de giro para rotar imágenes
- 20/11/2011 Añadida la opción de superponer imágenes para comparar
- 21/11/2011 Corrección de posiciones empíricamente
- 26/11/2011 Generación de malla de división del campo
- 30/11/2011 Añadido el cálculo del índice de verde mediante inecuaciones
- 12/01/2012 Añadido el módulo de conexión con acelerómetro Android
- 20/02/2012 Incluido el gráfico resultado, con el cálculo en tiempo real.
- 29/02/2012 Añadido el método de cálculo en bajo nivel (es 10 veces más rápido que el de alto nivel).
- 11/03/2012 Cambio de color de píxeles en tiempo real.

- 03/04/2012 Añadida la opción de sincronizar colores como método de detección
- 20/04/2012 Añadido el Manager de cámara por TCP/IP.
- 21/04/2012 Añadida la opción de sincronizar el acelerómetro, para colocar el móvil en cualquier posición
- 05/02/2012 Redacción del manual para instalarlo.
- 16/05/2012 Pruebas definitivas de color indoor.
- 06/06/2012 Mejoras de rendimiento en las pruebas en dinámico
- 07/06/2012 **Prueba en tiempo real realizada correctamente**
- 12/06/2012 Añadida la parte de control del robot a la aplicación principal
- 14/06/2012 Añadido vídeo de prueba 1
- 15/06/2012 Cambios en la interfaz para ser más amigable

Capítulo 7

Conclusiones

Tras la gran cantidad de pruebas, revisiones, análisis de los resultados y el *know-how* adquirido a lo largo de todo el desarrollo de nuestro proyecto, presentaremos este apartado como una explicación de los objetivos inicios alcanzados, así como una valoración de los resultados obtenidos finalmente por nuestra plataforma.

Gracias a esta herramienta hemos logrado conocer un nuevo lenguaje de programación: C#, utilizando además un nuevo entorno de desarrollo: Microsoft Visual Studio, integrando nuevas tecnologías desconocidas para nosotros y obteniendo a lo largo de 9 meses numerosas mejoras técnicas para la versión definitiva de la herramienta.

La mayor dificultad de este proyecto ha consistido en lograr acoplar distintas tecnologías trabajando conjuntamente para obtener información de análisis de un terreno en tiempo real con la calidad suficiente para que el actuador que utilice dicha información pueda trabajar de manera eficaz.

Los resultados obtenidos por la herramienta durante las pruebas realizadas sobre la misma, nos han demostrado que las condiciones de luz del entorno donde trabajará el robot que soportará nuestra herramienta son un factor determinante para la eficacia de la misma. Obteniendo la conclusión de que la utilización del filtrado de píxeles mediante la función de verde natural es favorable en espacios abiertos mientras que el empleo de los modelos RGB y HS funcionan mejor en espacios cerrados o con control de la luz ambiente.

..

Apéndice A – Estudio de Modelos de Color

Paralelamente al desarrollo de la herramienta en la que se centra la atención de esta memoria, fue realizado un estudio sobre la conveniencia de los distintos modelos de color para el objetivo propuesto de aplicación del análisis de color sobre terrenos predispuestos para la agricultura de precisión.

Para analizar el comportamiento de dichos modelos, fueron realizadas distintas pruebas con distintos colores, intentando que se asemejaran a aquellos colores que aparecerían frecuentemente en las imágenes que se capturarían en un terreno real.

Asimismo, se probó el efecto de distintas tolerancias a la hora de analizar los píxeles de la imagen. A continuación se muestran desde la Figura 67 a la Figura 74, los resultados obtenidos con distintos colores para cada uno de los modelos de color probados.

Modelo RGB – Tolerancia 10%

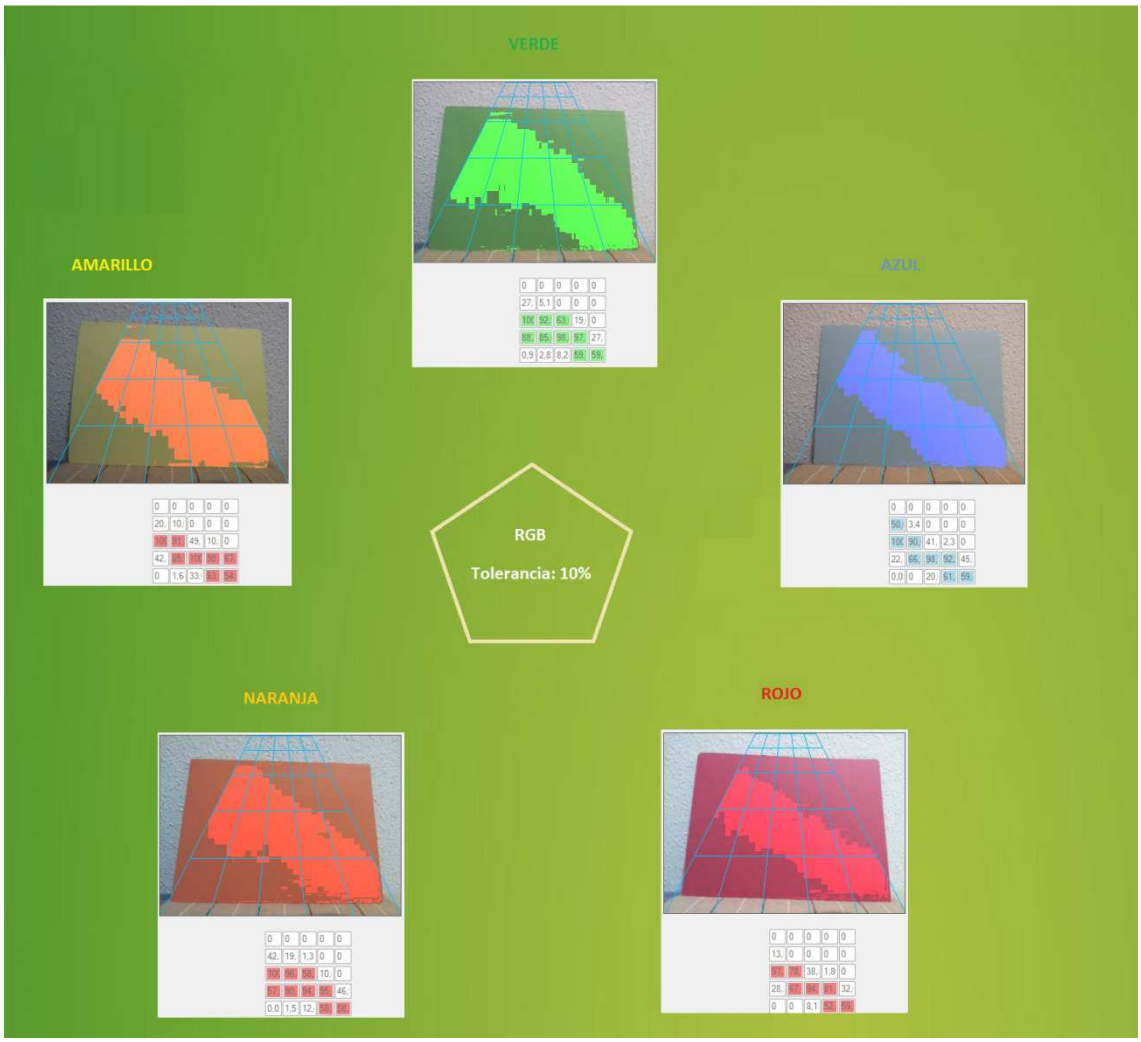

Figura 67. Capturas de pruebas de color RGB (Tolerancia 10%)

Modelo RGB – Tolerancia 50%

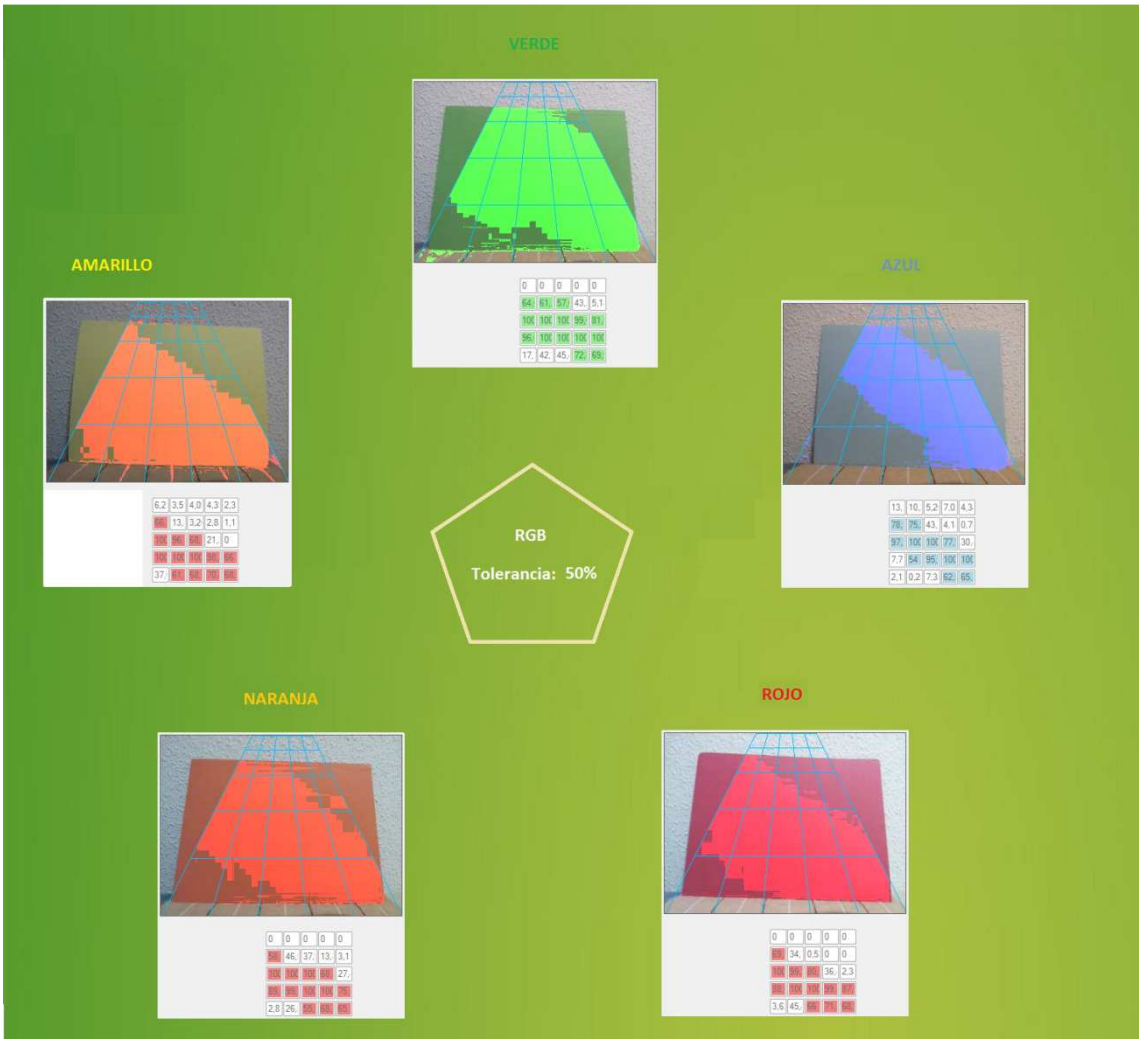

Figura 68. Capturas de pruebas de color RGB (Tolerancia 50%)

Modelo HSB – Tolerancia 10%

Figura 69. Capturas de pruebas de color HSB (Tolerancia 10%)

Modelo HSB – Tolerancia 50%

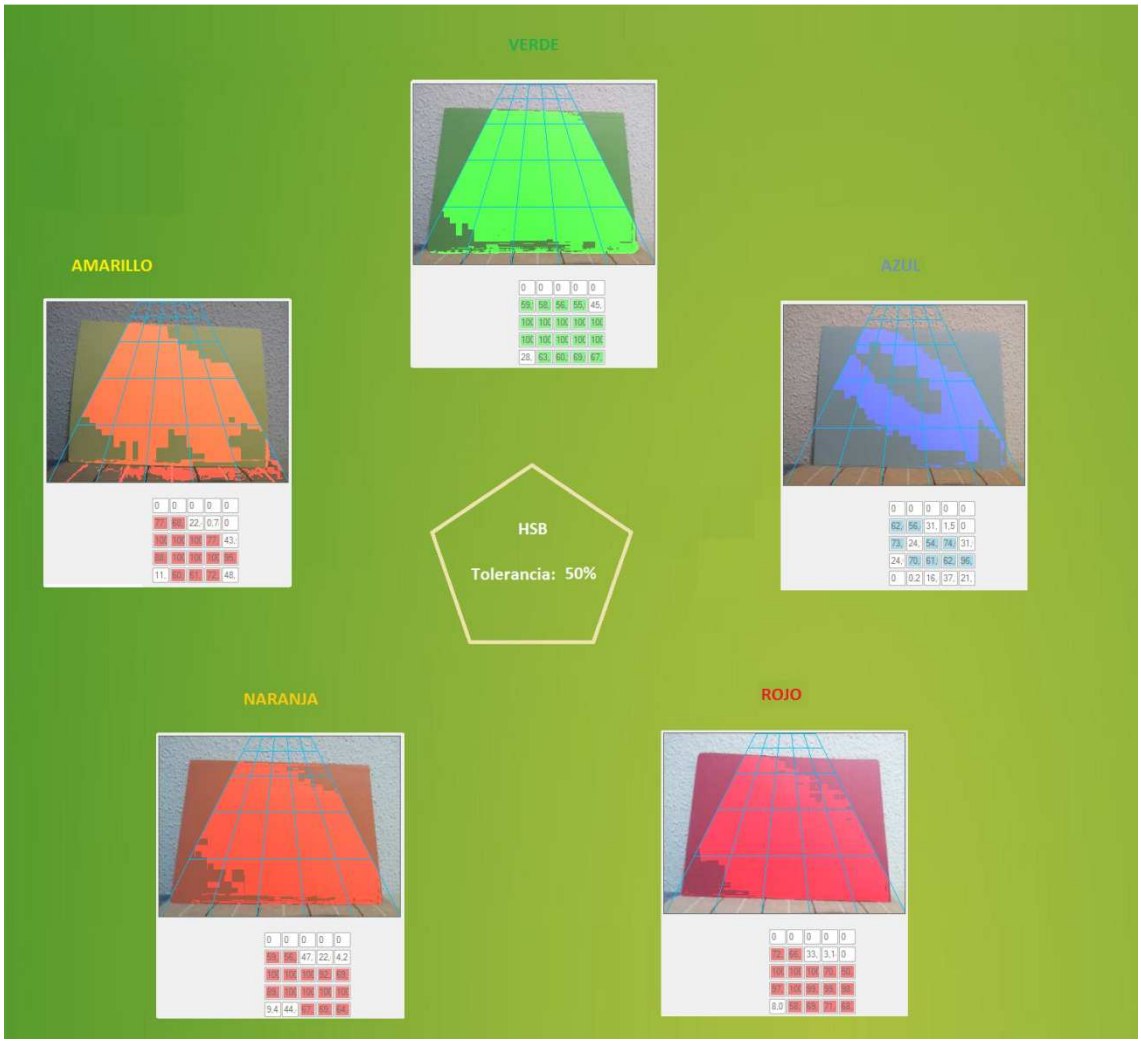

Figura 70. Capturas de pruebas de color HSB (Tolerancia 50%)

Modelo HS – Tolerancia 10%

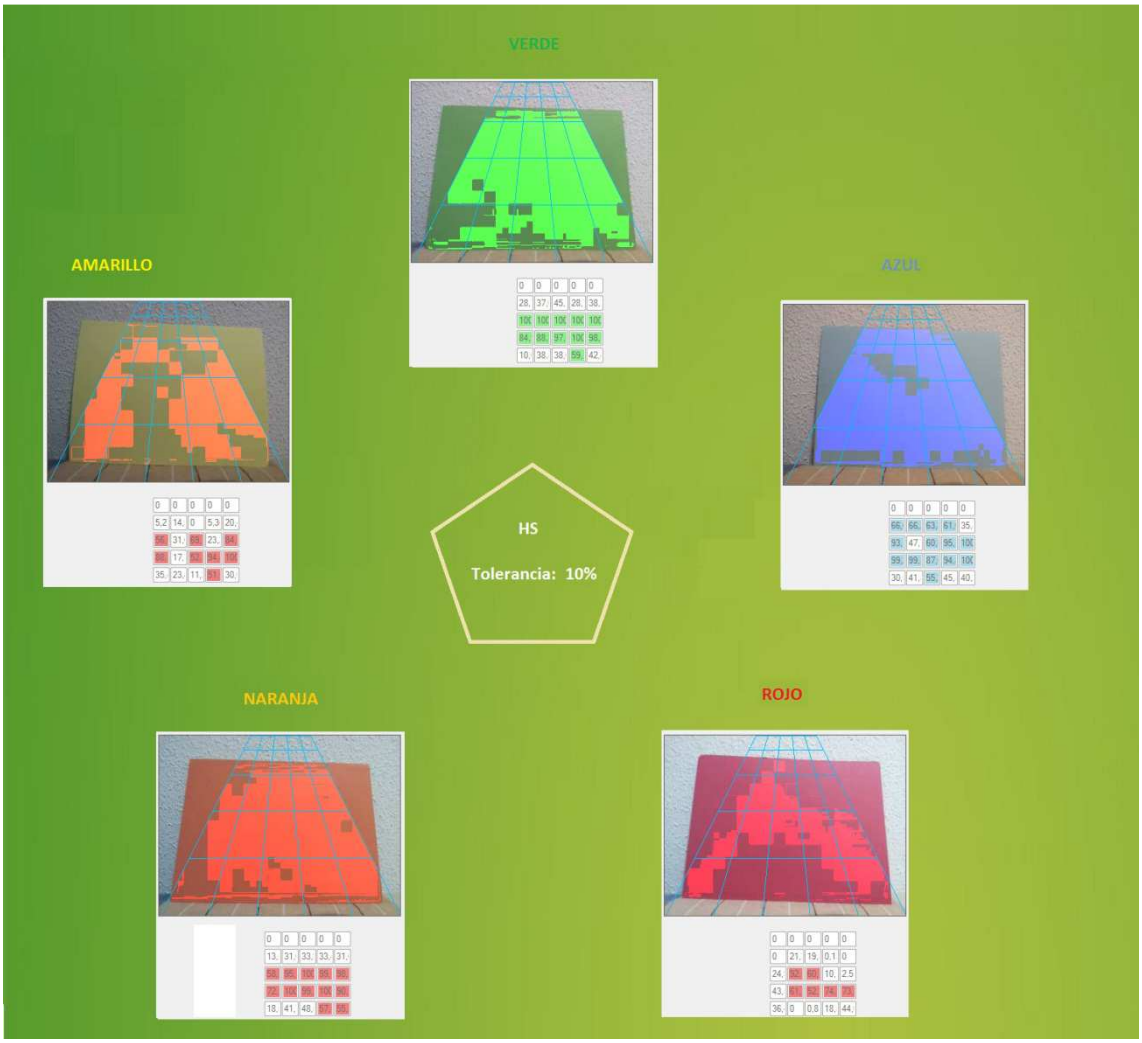

Figura 71. Capturas de pruebas de color HS (Tolerancia 10%)

Modelo HS – Tolerancia 50%

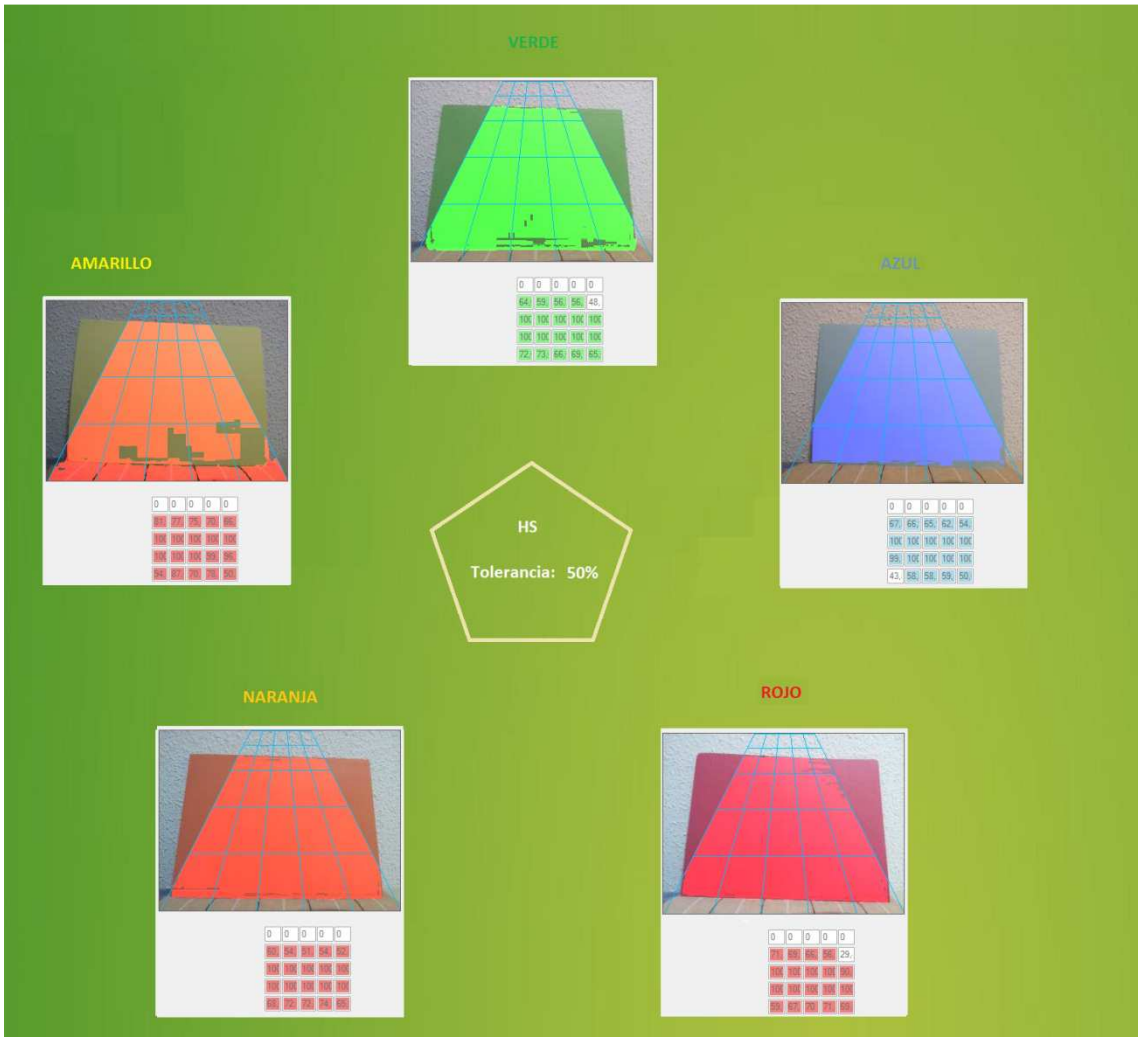

Figura 72. Capturas de pruebas de color HS (Tolerancia 50%)

Modelo H – Tolerancia 10%

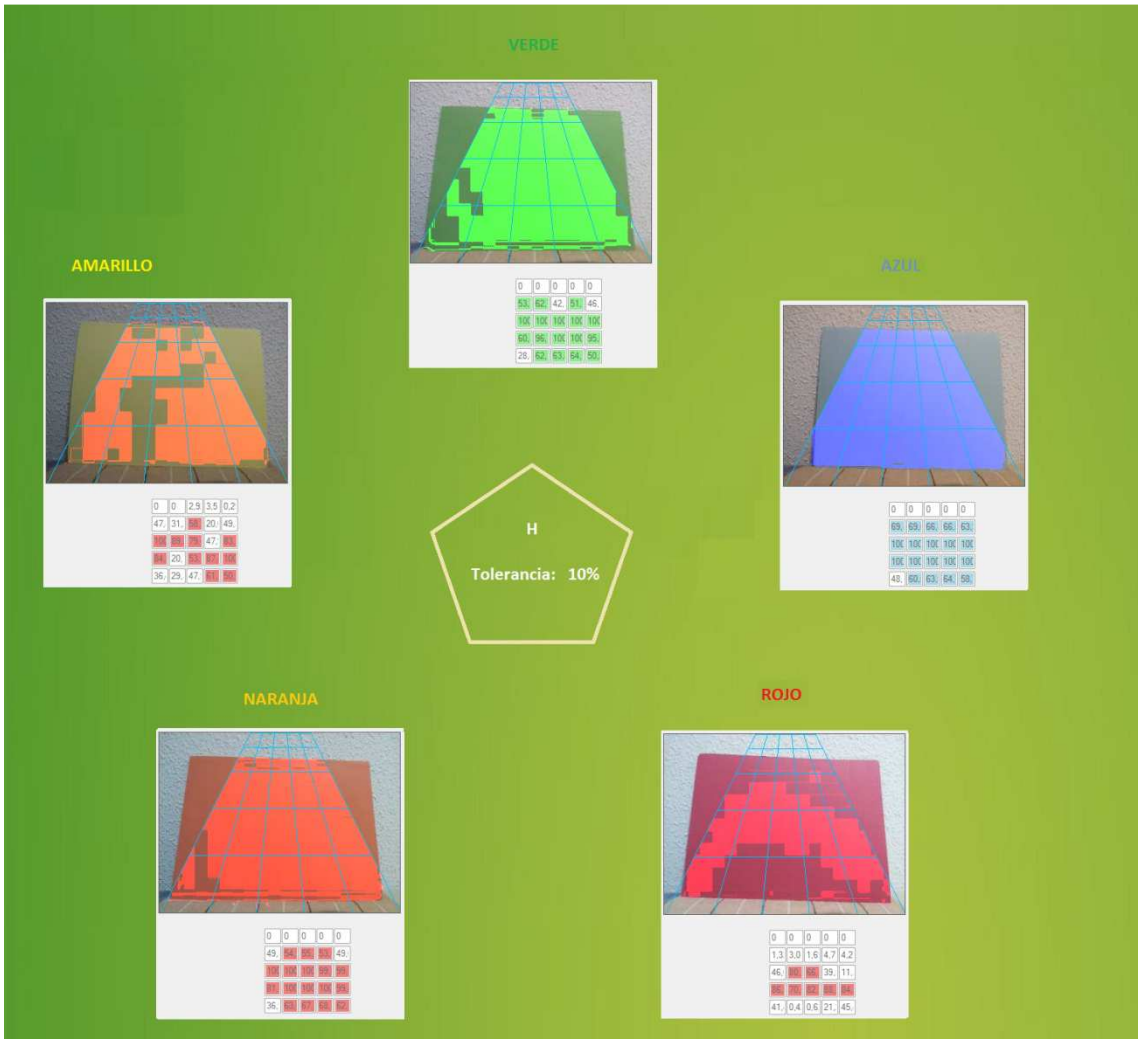

Figura 73. Capturas de pruebas de color H (Tolerancia 10%)

Modelo H – Tolerancia 50%

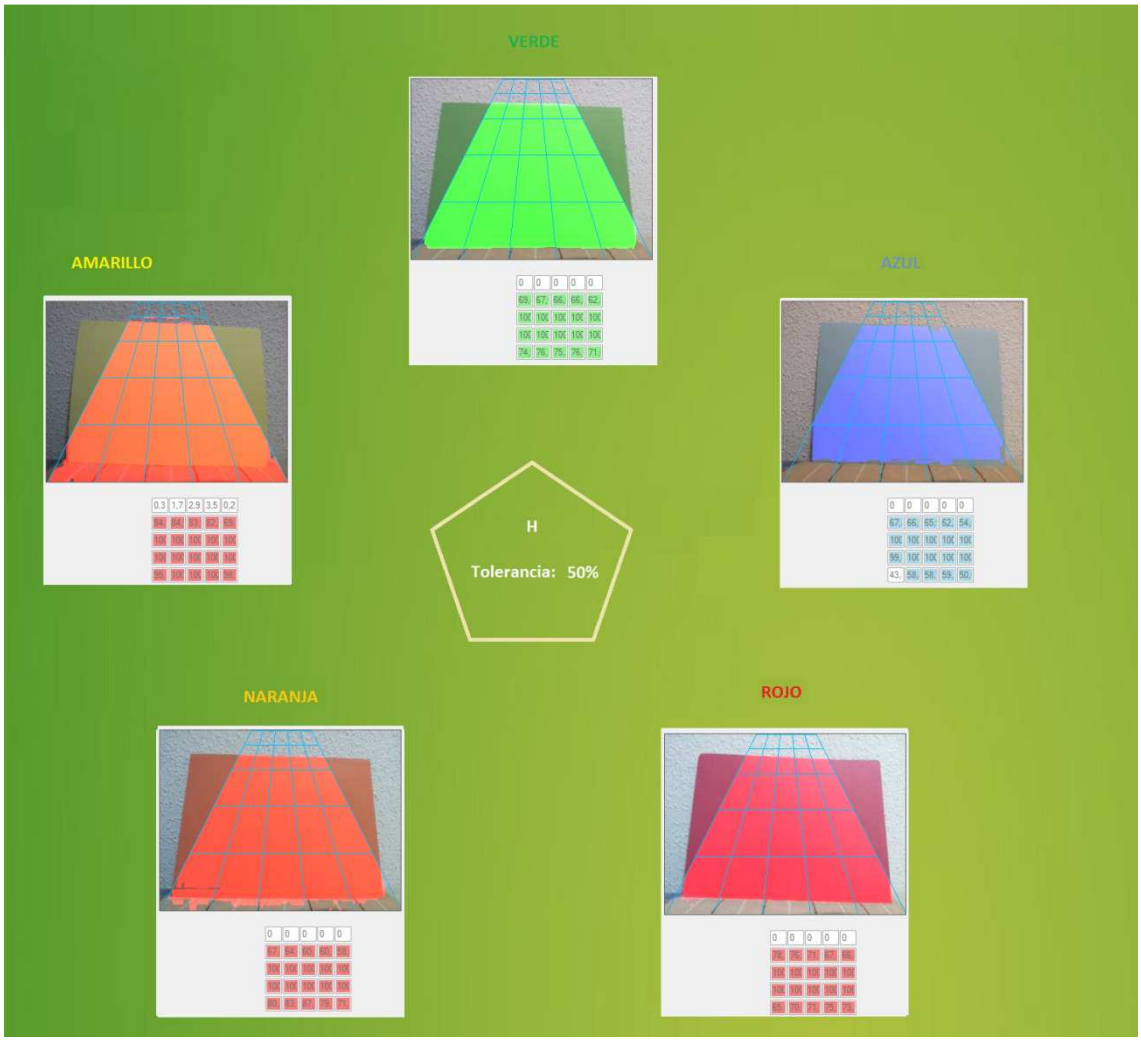

Figura 74. Capturas de pruebas de color H (Tolerancia 50%)

Conclusiones obtenidas del estudio:

1. Para detectar un único color plano la mejor opción es comparar únicamente con la componente H (*Hue*) del modelo HSB o incluso con el S (*Saturation*) conjuntamente.
2. Para detectar un único color pero con valores más exactos de sus componentes, la opción mejor es comparar con el modelo RGB.
3. No existe una gran diferencia en los resultados obtenidos para los distintos colores, salvo en alguna ocasión especial (p. e. azul en cada uno de los modelos heredados del HSB).
4. La mayor diferencia obtenida de unos colores a otros es debida a los distintos tipos de brillo que existía en cada instante del estudio.

Apéndice B – Estudio del comportamiento del acelerómetro

A lo largo del desarrollo de esta herramienta se han realizado numerosas pruebas para comprobar el correcto funcionamiento de la funcionalidad de corrección de irregularidades del terreno, mediante la utilización de los valores obtenidos de un acelerómetro. A continuación expondremos una breve prueba de como se corrigen los distintos tipos de movimientos en los ejes del supuesto robot móvil.

Primero se deberá tomar una imagen de referencia y sincronizaremos el acelerómetro con respecto a esta imagen:

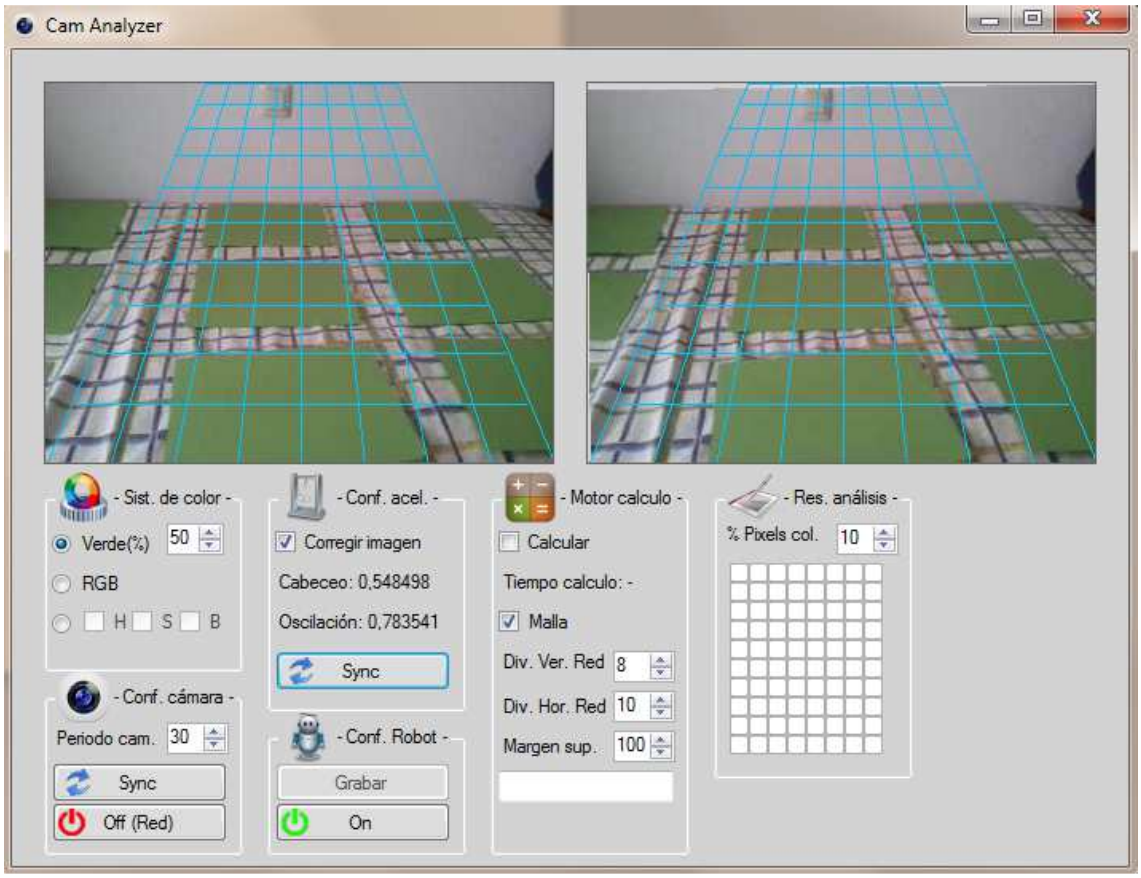

Figura 75. Valores del acelerómetro aproximadamente a 0 tras la sincronización.

Una vez que se ha tomado esta imagen de referencia procederemos a girar la cámara para ver como se corrigen estas deformaciones intentando obtener de nuevo la imagen original expuesta en la Figura 75

Primero realizaremos un movimiento de cabeceo, aplicando un movimiento en el eje horizontal (eje paralelo a la imagen) imaginario sobre el que girará nuestra cámara. Los resultados obtenidos son:

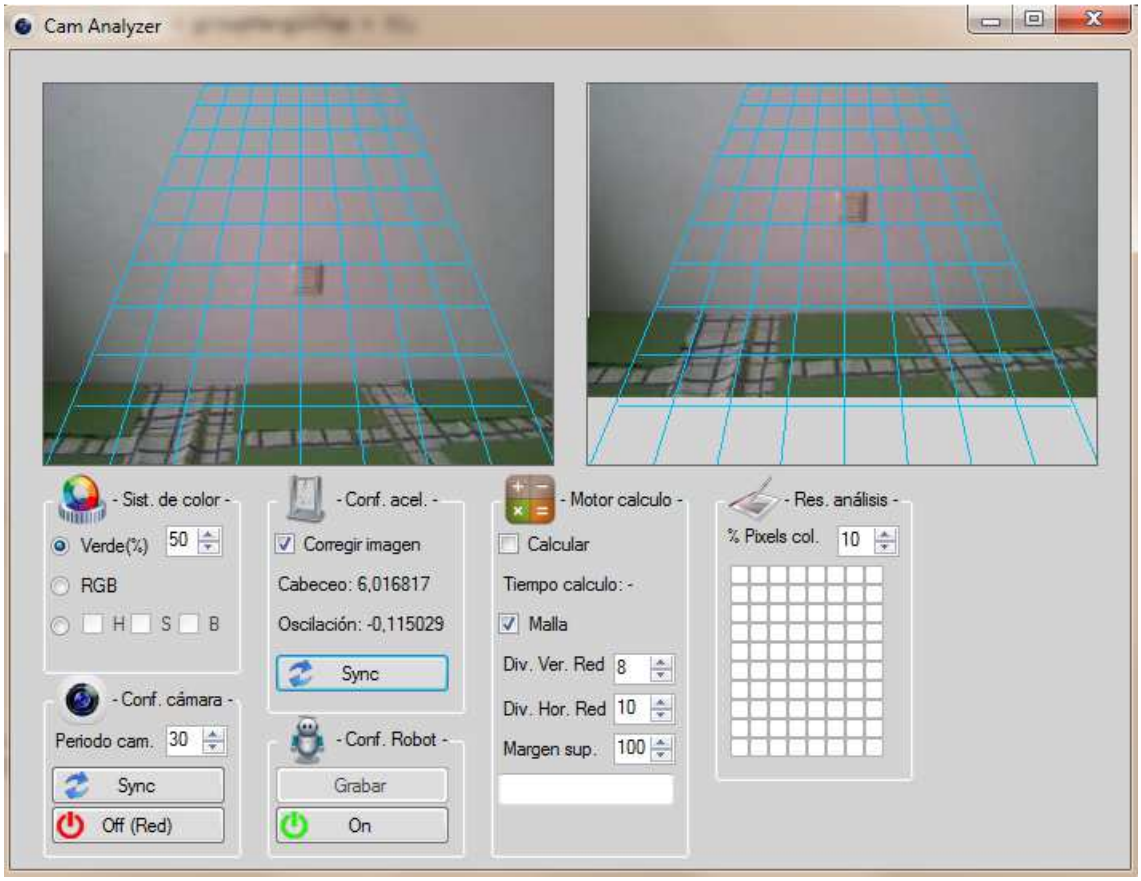

Figura 76. Transformación de la imagen en la ventana de la derecha, tras la corrección de un movimiento de cabeceo.

Observamos en la Figura 76 cómo se produce una ligera desviación, producida por la importancia de los factores de corrección de cabeceo, los cuales están configurados para una altura de 10 cms. de la cámara. En esta imagen vemos como la altura de la cámara también se ha elevado produciendo un error previsto. Sin embargo, se puede observar como se ha transformado levemente la imagen desplazándose esta verticalmente.

A continuación realizaremos un movimiento de oscilación sobre la cámara, aplicando un movimiento en el eje horizontal (eje perpendicular a la imagen) imaginario sobre el que girará nuestra cámara. Los resultados obtenidos son:

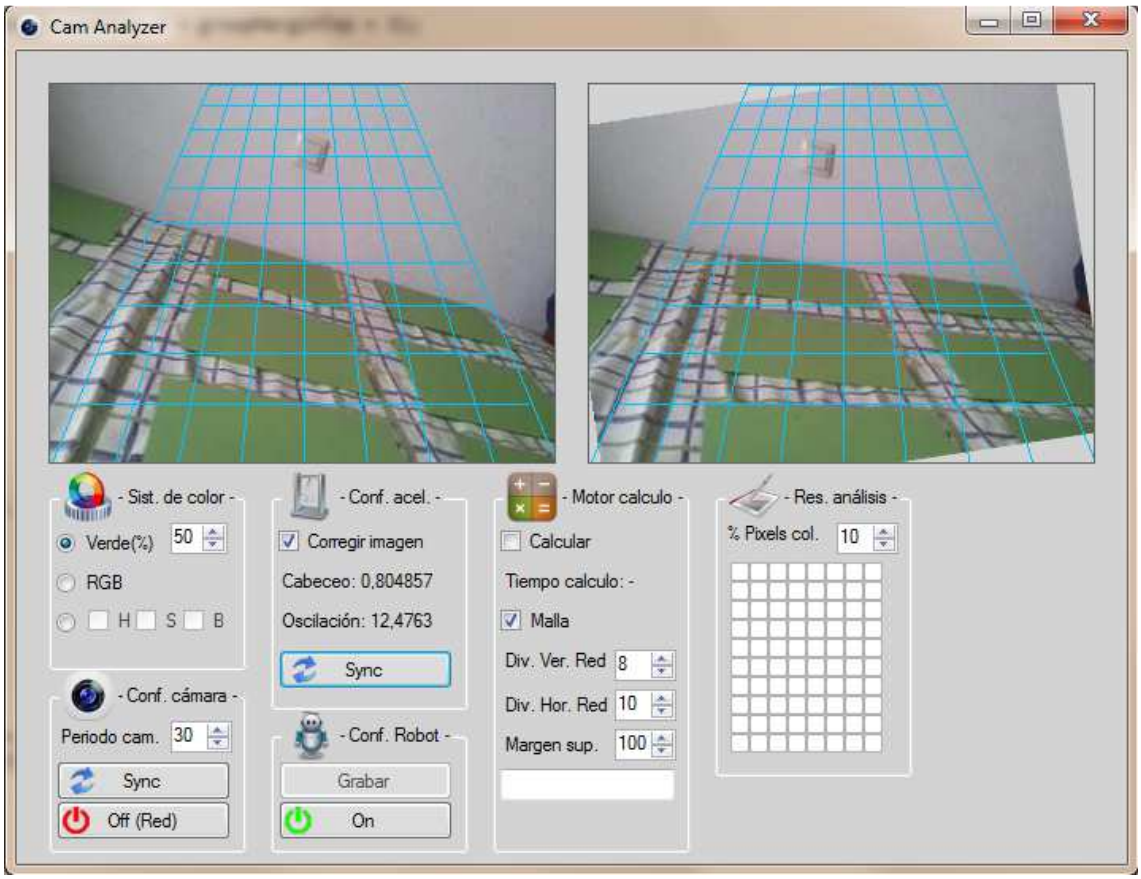

Figura 77. Transformación de la imagen en la ventana de la derecha, tras la corrección de un movimiento de oscilación.

Observamos cómo en esta corrección mostrada en la Figura 77, al haberse realizado a una altura más aproximada a los 10 cms. para los que fueron calculados los factores de corrección, la imagen transformada se asemeja más a la que se tomo inicialmente como referencia en la Figura 75.

Por último, aplicaremos los dos movimientos a la vez: cabeceo y oscilación.

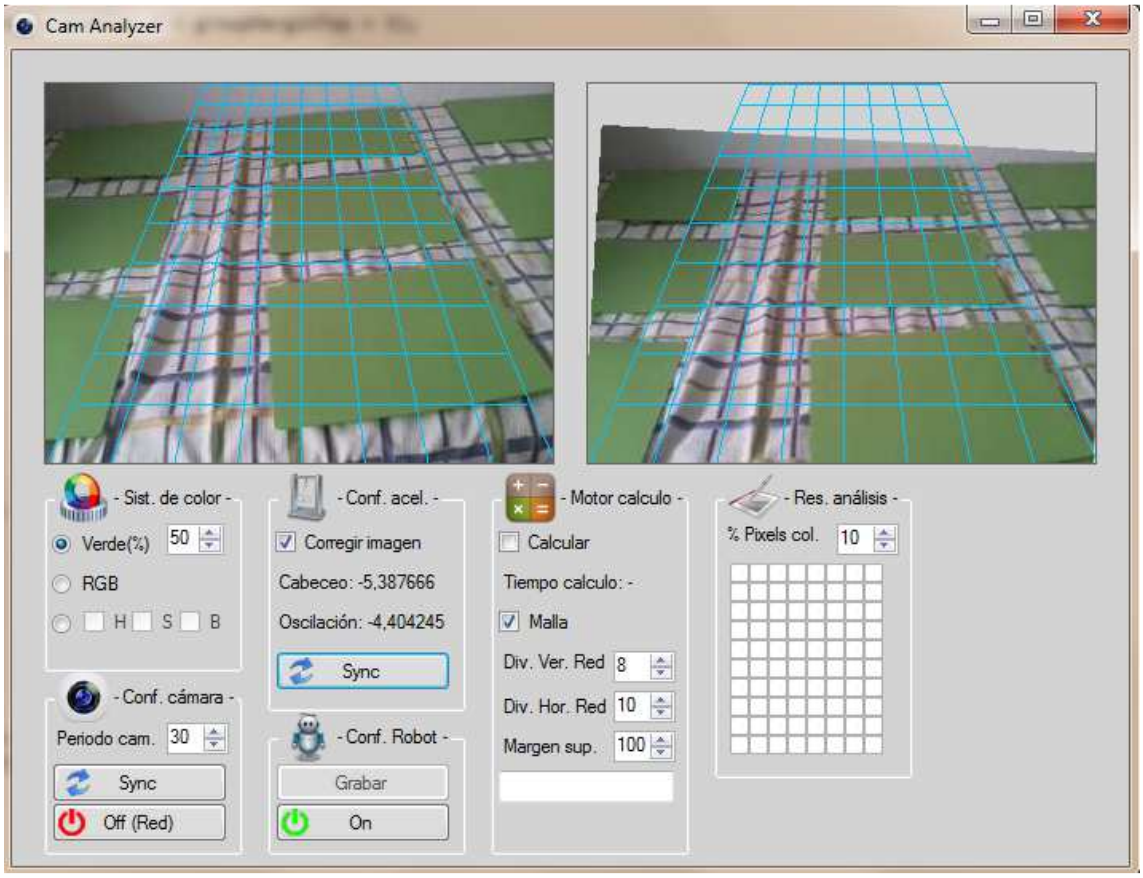

Figura 78. Transformación de la imagen en la ventana de la derecha, tras la corrección de un movimiento de cabeceo y oscilación.

En la Figura 78 se ve claramente como obtenemos el mejor resultado de los realizados ya que la altura de la imagen de referencia y la altura con la que se ha tomado esta imagen son las más similares.

Observamos de nuevo como se ratifica que el principal problema a la hora de corregir una imagen será la altura de la cámara, movimiento sobre un eje que se controla mediante la modificación de los parámetros de corrección existentes en el archivo XML de configuración de la herramienta.

Bibliografía

"Visión por computador: imágenes digitales y aplicaciones" Gonzalo Pajares Martinsanz, Jesús Manuel de la Cruz García

"Electronic Imaging Technology" Edward R. Dougherty

"Astrofotografía con cámaras digitales" Michael A. Covington

"Science for the Curious Photographer: An Introduction to the Science of Photography" Charles S. Johnson

"Correction for Camera Roll in a Perspectively Distorted Image: Cases for 2 and 3 point perspectives" Avinash N., Murali S.

"Fotogrametría Moderna: Analítica Y Digital" José Luis Lerma García

"Cámaras digitales y fotografía: artículos y reportajes" Iker Morán

"How the New Technology Works: A Guide to High-Tech Concepts" Robert J. Cone, Patricia L. Barnes-Svarney

"Multidimensional Signal, Image, and Video Processing and Coding" John W. Woods

"http://www.colorotate.org" portal Web creado por el

"Strapdown Inertial Navigation Technology" David H. Titterton, John L. Weston

"Embedded Robotics: Mobile Robot Design and Applications With Embedded Systems" Thomas Bräunl

Figuras extraídas de Internet:

Figura 1 (http://www.car.upm-csic.es/gpa/)

Figura 12 (http://www.hffax.de/history/html/bartlane.html)

Figura 13 (http://en.wikipedia.org/wiki/File:Ranger7_PIA02975.jpg)

Figura 14 (http://earth.google.es/rome/)

Figura 15, Figura 16 (http://www.fotomanel.com/20_Equipo/Sensores/Sensores.html)

Figura 19 (http://gyroscope.com/d.asp?product=SUPER2)

Figura 20 (http://www.directindustry.es/prod/r-b/inclinometros-analogicas-59201-384415.html)

Figura 21,Figura 22,Figura 23 (http://www.colorotate.org/)

Figura 25 - Figura 35: creadas con Microsoft Visual Studio

Figura 36 (http://www.globalspec.com/supplier/profile/MEMSense)