DETECCIÓN Y EVITACIÓN DE OBJETOS FLOTANTES EN EL CONTEXTO DE UN VEHÍCULO AUTÓNOMO DE SUPERFICIE.

ALUMNA: LAURA DEL PINO DÍAZ
TUTOR : JORGE CABRERA GAMEZ

TRABAJO DE FINAL DE GRADO

CURSO 2015/16

[Página intencionadamente dejada en blanco]

AGRADECIMIENTOS

Me gustaría comenzar esta memoria agradeciendo a todas aquellas personas que de forma directa o indirecta han colaborado en la realización de este proyecto.

Empezando por el profesor José Daniel Hernández por habernos llevado de visita al Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) sin la cual no hubiese conocido a mi tutor Jorge Cabrera Gámez. A éste mismo por su apoyo, asesoramiento y paciencia a lo largo del desarrollo del Trabajo Final de Grado (TFG). A Antonio Carlos Domínguez Brito por prestarme parte del material utilizado para el desarrollo del TFG.

También quiero agradecerle a mi familia por el apoyo y el consejo que me han prestado durante estos meses. Y por último y no por ello menos importante a mis amigos, compañeros de clase y de laboratorio por estar siempre ahí sacándome de la caja del problema para mostrarme que la solución de mis problemas está delante de mis narices.

A todos muchas gracias.

[Página intencionadamente dejada en blanco]

RESUMEN

El Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería y en especial la División de Robótica y Oceanografía Computacional está desarrollando un velero autónomo de superficie que requiere de un sistema de visión por computador para la detección y evasión de obstáculos.

Dicho sistema se ha desarrollado desde cero sobre una Raspberry Pi en la que se ha desarrollado un servicio para la captura de imágenes, así como un servidor web que permita la modificación de la configuración de la cámara a través de una página web. Una vez completada dicha infraestructura se tomarán las fotografías que conformarán el conjunto de entrenamiento para el sistema de visión por computador y se desarrollará este último. Como último paso se han integrado los resultados del sistema de visión por computador con el sistema del control del vehículo modificando el rumbo cuando se detecte un obstáculo.

[Página intencionadamente dejada en blanco]

ABSTRACT

University Institute of Intelligent Systems and Numerical Applications in Engineering and, in special, the Division of Robotics and Computational Oceanographic is developing an autonomous sailing ship which requires a computer vision system for the detection and avoidance of obstacles.

That system will be developed from scratch on a Raspberry Pi. A service for taking photos will be programmed, as well as a web service which allows modifications of the camera configuration from a web environment. Once this phase is completed, an image database is obtained using the previous system. This dataset will be used by the Computer Vision System to detect the possible obstacles. The last step in this process is the integration of the computer vision system with the control system to correct the course when an obstacle is detected.

[Página intencionadamente dejada en blanco]

Tabla de contenido

1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

Desde principios de 2008 la División de Robótica y Oceanografía Computacional de Instituto de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) de la ULPGC desarrolla activamente una línea de desarrollo e investigación en vehículos autónomos marinos, tales como planeadores submarinos (gliders), AUV y veleros autónomos. En esta línea de trabajo ha dado lugar a una amplia serie de artículos, la participación en la primera travesía del Atlántico Norte por un glider submarino, varios premios en concursos internacionales y recientemente a un contrato de cesión de tecnología con la empresa Marina Signals S.L

La presente propuesta se ubica en esta línea de trabajo y aborda la integración de un sistema de visión multipropósito aplicable en el contexto de un vehículo autónomo de superficie (ASV), si bien este proyecto, la plataforma experimental será un velero autónomo de 2m de eslora. La percepción del entorno aéreo más inmediato es un déficit actual de gran parte de los diseños de ASVs que bien por envergadura, bien por restricciones energéticas o de coste no pueden integrar un sistema electro-óptico o de radar para tal fin. Los problemas a desarrollar son la detección de objetos flotantes próximos al ASV a partir de una imagen estática. A partir de este sistema se dotará al ASV de un sistema de control que evite los obstáculos una vez haya detectado los obstáculos.

1.2 ESTADO ACTUAL

La detección de objetos flotantes en el mar es un problema que, a pesar de su aparente simplicidad, entraña una gran dificultad por varios motivos. Una parte, es necesario operar en un rango muy amplio de condiciones de mar y de iluminación; de otra, los objetos pueden no ser constantemente visibles, en particular si están parcialmente sumergidos. Por todo ello, en la actualidad el mejor sensor más usado y polivalente para la detección de objetos en el mar sigue siendo el ojo humano, puesto que tiene en cuenta varios factores como el movimiento de los objetos en la escena que es distinto del movimiento del oleaje, los objetos flotantes tienen un comportamiento distinto ante los rayos del sol que la superficie del mar y todas estas consideraciones las toma en un intervalo de tiempo reducido y con menos recursos que los algoritmos más avanzados que existen en la actualidad.

La gran mayoría de los algoritmos desarrollados usan una secuencia de imágenes para el aprendizaje de los clasificadores que discriminan entre lo que es el fondo y el frente apoyados mayoritariamente en algún tipo de cámara térmica, multiespectral o hiperespectral para disponer de datos que permitan hacer la discriminación más directa o al menos reducir la búsqueda a un área más reducida. Un barrido de todo el conjunto de sensores que se pueden utilizar para facilitar esta tarea puede consultarse en [1].

Aunque los algoritmos que se usan en la gran mayoría de los casos están basados en secuencias de imágenes también existen algoritmos orientados al nivel de píxel de la imagen recogidos en [2] donde clasifican los existentes métodos en tres: básicos, paramétricos y basados en muestras. El sistema que se desarrolla en este trabajo está dentro de los básicos puesto que toma descriptores de las ventanas en las que se subdivide la imagen para determinar la ausencia de objetos.

Un sistema similar al descrito en [3] creando unos pequeños vehículos autónomos controlados con Arduino y cuyo sistema de visión por computador se desarrollaba dentro de un Smartphone para detectar y esquivar los obstáculos en el entorno de un río usando técnicas de flujo óptico. Tanto en este entorno de desarrollo como en el ámbito marino en el que se trabaja en este trabajo nos encontramos con problemas como las ondulaciones de la superficie del agua, los cambios graduales de iluminación por el transcurso del día, los reflejos de la orografía en la superficie y los reflejos generados por la propia luz solar.

El único sistema que afirma haber conseguido poder detectar obstáculos en el mar está patentado por la empresa Utopia Compression Corporation bajo el nombre de *SeaProtector* y es utilizado por las Fuerzas Marítimas Estadounidenses, y utiliza múltiples tipos de sensores para la fácil discriminación de los vehículos marítimos y posibles obstáculos.

1.3 OBJETIVOS

La lista completa de objetivos a desarrollar en este Trabajo de Final de Grado son los siguientes:

1. Desarrollo de la infraestructura básica.
    1. Configuración básica de la Raspberry Pi [4]: interfaces e instalación de la librería OpenCV para visión por computador.
    2. Captura de imágenes utilizando la cámara de la Raspberry Pi.
    3. Desarrollo de un servidor web para la monitorización de la captura de imágenes durante la captura de las mismas.
    4. Construcción de un contenedor que proteja la Raspberry Pi.
2. Desarrollo del sistema de visión por computador para la detección de obstáculos flotantes.
    1. Obtención del conjunto de imágenes.
    2. Estructura del algoritmo básico para la segmentación imagen-fondo.
3. Aplicación del sistema de detección de obstáculos flotantes a la evitación de los mismos.
    1. Desarrollo de una estrategia de control para la evitación de obstáculos teniendo en cuenta la información obtenida del sistema de visión y la información obtenida por otros sensores.
4. Documentación y defensa
    1. Elaboración de la presente memoria, así como de la presentación de soporte para la defensa en público.

1.4 JUSTIFICACIÓN DE LAS COMPETENCIAS ESPECÍFICAS CUBIERTAS

La relación de competencias específicas cubiertas y cómo se han cubierto en este TFG se presenta en la siguiente tabla:

Competencia	Definición de la competencia	Tarea asociada
FB05	Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de la programación y su aplicación para la resolución de problemas propios de la ingeniería.	Intercomunicación de los distintos componentes del entorno de desarrollo: Raspberry Pi, Macbook Pro.
CII010	Conocimiento de las características, funcionalidades y estructuras de los sistemas operativos y diseñar e implementar aplicaciones basadas en sus servicios.	Desarrollo de una página web sobre el servidor Apache. Implementación de un programa que se ejecuta al inicio del sistema.
CII011	Conocimiento y aplicación de las características, funcionalidades y estructura de los sistemas distribuidos, las redes de computadores e internet y diseñar e implementar aplicaciones basadas en ellas.	Desarrollo de una página web que usa distintas tecnologías para su correcto funcionamiento (PHP,HTML, CSS, Javascript, AJAX,JSON). Comunicación de procesos mediante sockets.
CII014	Conocimientos y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.	Desarrollo de una aplicación que realiza una escucha del socket a la vez que toma imágenes de la cámara.
CII015	Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica.	Arquitectura de agente para la implementación del control del velero.
CP04	Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.	Diseño del sistema de visión por computador.
CP05	Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para	Diseño del sistema de visión por computador.

	la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.	
CP07	Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilice, incluyendo las dedicadas a la extracción automática de información y conocimiento a partir de grandes volúmenes de datos.	Propuesta de sistema de aprendizaje para la correcta umbralización del sistema de visión

1.5 APORTACIONES

El desarrollo del sistema de visión por computador y el correspondiente sistema de control aportan un primer avance a la línea de desarrollo e investigación de vehículos autónomos de superficie dándoles a éstos la capacidad de evitar obstáculos.

Dentro de esta línea este trabajo de fin de grado contribuye en los siguientes aspectos:

- Facilitación del desarrollo de aplicaciones. Al hacer uso de un compilador cruzado que permite el desarrollo de las aplicaciones de control en entornos de desarrollo de sistemas con interfaces gráficas avanzadas que abstraen al programador de los detalles a bajo nivel que consumen tiempo de desarrollo.
- El acceso a la cámara se hace con un código más limpio y entendible. Al hacer uso de la librería OMXCam.
- El desarrollo de un servidor web para la captura de imágenes y la configuración de los parámetros de captura con una interfaz amigable.
- La evasión de obstáculos se apoya en el sistema de visión por computador que es el sistema encargado de detectar los obstáculos.

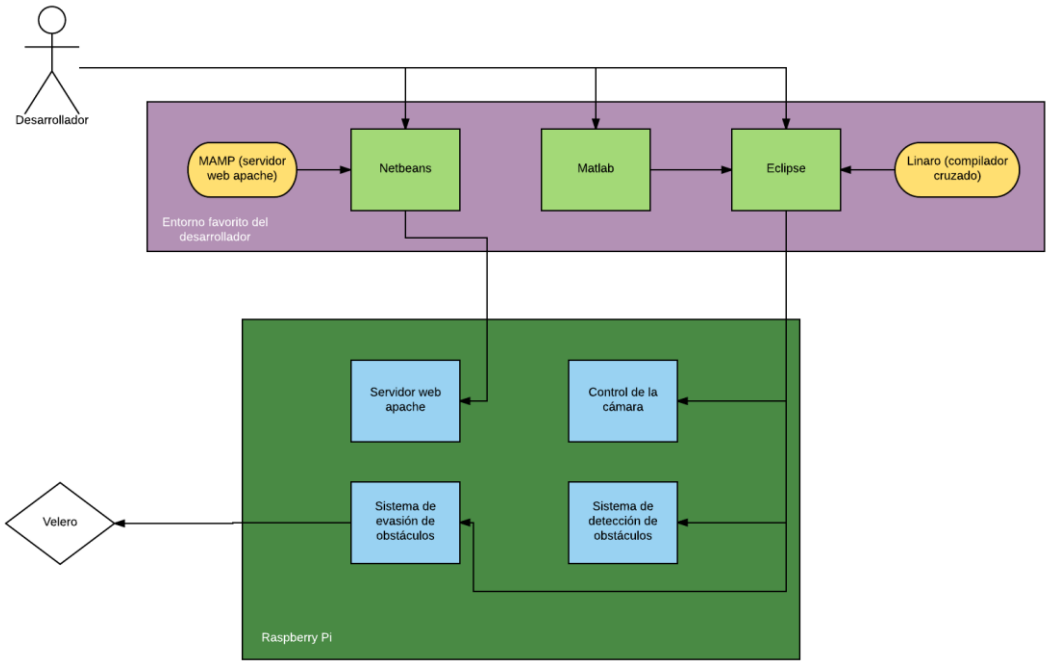


Ilustración 1: Flujo de desarrollo

1.6 METODOLOGÍA DE TRABAJO

1.6.1 MATERIALES PARA EL DESARROLLO

Para alcanzar el objetivo de detectar obstáculos para su posterior evitación de objetos flotantes en ambientes marinos utilizamos los siguientes medios:

- Raspberry Pi 2 B: Para el procesamiento de las imágenes.
  - Posee:
    - Procesador de cuatro núcleos a 0,9GHz con arquitectura ARM.
    - 1 GB de memoria RAM.
    - Salida de pantalla HDMI.
    - 4 puertos USB.
    - 1 puerto Ethernet para conexión a Internet por cable.
    - 1 puerto CSI para la conexión de la cámara diseñada específicamente para este dispositivo.
    - Puerto para microSD desde donde se leerá el sistema operativo.
    - Núcleo gráfico IV 3D VideoCore.
    - Entrada de corriente mini USB tipo B.
  - Extendido con:
    - Antena WiFi USB.
    - Camera sin filtro de corte infrarrojos Raspberry Cam NoIR.
  - Ejecuta el sistema operativo Raspbian Jessie Minimal desarrollado en específico para este tipo de sistemas basado en Debian que es una distribución de Linux.
- Lipo Rider Pro

    - Posee:
        - Entrada para batería de 3,7 V
        - Entrada para panel solar
        - Entrada para miniUSB tipo A para carga de la batería
        - Salida USB tipo A.
    - Extendido con:
        - Batería de 3,7V 6000mAh
- Portátil Macbook Pro:
    - Posee:
        - Procesador de dos núcleos Intel Core i5. (2,7GHz)
        - 8 GB de memoria RAM.
        - Interfaz WiFi, Bluetooth, FireWire para conexiones externas
    - Ejecuta el sistema operativo Mac OSX "El Capitan".
    - Posee las siguientes aplicaciones destacables:
        - Compilador cruzado Linaro (versión no oficial para Mac OSX).
        - Entorno de desarrollo integrado Eclipse.
        - CMake para la compilación cruzada de librerías.
        - Netbeans como entorno de desarrollo integrado para programación web.
        - Servidor web MAMP.
        - MatLab 2014b.
- Smartphone BQ Aquaris E5
    - Permite generar una red Wifi local perfecta para comunicar ambos dispositivos.

1.6.2 ENTORNO DE DESARROLLO

El entorno de trabajo consiste en conectar los tres componentes principales, presentados en el apartado anterior (Raspberry Pi, Macbook Pro y Aquaris E5) mediante la creación de una red Wifi generada por el smartphone a la que se conectan los otros dos componentes, permitiendo así que exista una comunicación entre ellos.

No se ha conectado el portátil directamente a la Raspberry Pi porque utilizando el Smartphone se puede transmitir los ejecutables del portátil a la Raspberry Pi haciendo uso de la red creada por el Smartphone y permite prescindir del portátil a la hora de visualizar las imágenes en la playa ya que se puede acceder directamente desde el Smartphone manteniendo así el portátil seguro del salitre.

El desarrollo de los programas involucrados en este trabajo se realiza en el Macbook Pro dentro del entorno de desarrollo integrado Eclipse. Éste permite la fácil integración con el compilador cruzado Linaro para la producción de ejecutables preparados para la arquitectura ARM propia de la Raspberry Pi.

Dichos ejecutables se transmiten mediante una conexión segura a través de nuestra red WiFi particular y se ejecutan en la Raspberry Pi como si hubiesen sido desarrollados en este sistema.

1.6.3 METODOLOGÍA DE DESARROLLO

La metodología de desarrollo elegida para este proyecto es el Proceso Unificado de Desarrollo de software, caracterizado por estar dirigido por los casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

Los casos de uso se determinaban en reuniones semanales y se daba un plazo de una semana para el desarrollo de los mismos. En estas mismas reuniones se validaban si el desarrollo se hacía de forma efectiva.

La temporización estimada de cada uno de los puntos mostrados en la sección de OBJETIVOS se muestra a continuación:

Configuración básica de la Raspberry Pi: interfaces e instalación de la librería OpenCV para visión por computador.	25 horas
Desarrollo del servicio de captura de imágenes.	35 horas
Desarrollo de un servidor web para la monitorización de la captura de las imágenes mientras se obtienen.	80 horas
Integración del servidor web con el servicio de captura de imágenes.	20 horas
Construcción de un contenedor que proteja la Raspberry Pi.	10 horas
Obtención del conjunto de imágenes.	5 horas
Estructura del algoritmo básico para la segmentación fondo-frente.	50 horas
Desarrollo de una estrategia de control para la evitación de obstáculos teniendo en cuenta la información obtenida del sistema de visión y la información obtenida por otros sensores.	40 horas
Documentación y defensa	35 horas

2 DESARROLLO DE LA INFRAESTRUCTURA BÁSICA

2.1 DEFINICIÓN DE INFRAESTRUCTURA BÁSICA

El sistema de detección de obstáculos debe ser capaz de adaptarse a las condiciones del medio en las que se encuentra. Para comprobar que el algoritmo es capaz de realizar esta tarea es necesario disponer de un conjunto de imágenes con situaciones diversas a las que podría enfrentarse.

Si identificamos los diferentes tipos de imágenes que podemos recoger tenemos tres casos principales:

- La superficie del mar únicamente.
- La superficie del mar con el horizonte y el cielo.
- La superficie del mar con el horizonte, el cielo y un perfil de la costa.

En nuestro caso nos centraremos únicamente en el primer caso. En ausencia de un conjunto de imágenes con estas características desarrollaremos una infraestructura que nos permita llevar la Raspberry Pi al entorno acuático similar a las condiciones que se tendrá que enfrentar cuando el sistema se integre en el vehículo autónomo de superficie. Para obtener dicho conjunto de imágenes nos vemos en la necesidad de:

1. Visualizar las imágenes mientras se capturan.
2. Ser capaces de modificar los parámetros de la cámara mientras se realiza la captura de las imágenes.
3. Proteger la Raspberry Pi para que no sea dañada por el agua o el salitre.

Para lograrlo definimos como infraestructura básica para este proyecto el desarrollo de un servicio web que permita a un conjunto de usuario visualizar las imágenes obtenidas por la cámara mientras ésta las captura. Limitando el número de usuarios que pueden modificar los parámetros de la cámara a uno. Siendo éste el primero que pida los permisos de superusuario. También forma parte de esta infraestructura básica la construcción de una carcasa externa para proteger la Raspberry Pi del salitre.

2.2 DESARROLLO DEL SERVICIO WEB PARA EL CONTROL DE LA CÁMARA

Con el objetivo de tomar las imágenes que formarán parte del conjunto de entrenamiento, y también para permitir el ajuste de los parámetros de la cámara, se ha desarrollado una página web en la Raspberry Pi que permita el acceso a cualquier dispositivo de cualquier plataforma que esté en dentro de su red.

2.2.1 TIPOS DE USUARIOS

Para este servidor existen dos tipos de usuarios: el superusuario y usuarios normales. El superusuario es el primer usuario que pide permiso para acceder a la configuración de la cámara, dejando así como usuarios normales todos los demás que intenten acceder a la configuración de la cámara.

2.2.2 CASOS DE USO

Las acciones que pueden realizar ambos tipos de usuarios son las siguientes:

- Ver la imagen que ha capturado la cámara.
- Pedir privilegios de superusuario para modificar la cámara.

Las acciones que pueden realizar únicamente el superusuario son las siguientes:

- Cambiar el estado de la cámara de apagado a encendido o viceversa.
- Apagar por completo la Raspberry Pi.
- Cargar la configuración por defecto de la cámara. [1]
- Proponer una configuración para ser enviada a la Raspberry Pi.
- Enviar la configuración a la Raspberry Pi.
- Seleccionar entre las resoluciones propuestas en la pestaña para la propuesta de la configuración.

Los parámetros de configuración que se pueden ajustar desde la página web se muestran en la Tabla 1:

Tabla 1: Parámetros configurables de la cámara desde la web con sus rangos y su valor por defecto.

Parámetro	Rango	Valor por defecto
Ancho	[16,2592]	2592
Alto	[16,1944]	1944
Nitidez	[-100,100]	0
Contraste	[-100,100]	0
Brillo	[0,100]	50
Saturación	[-100,100]	0
Tiempo de exposición	[0, 6000]	0
ISO	{100,200,400,800}	Automático
Balance de blancos: ganancia azul	[0, ∞) acotado a [0, 3000]	100
Balance de blancos ganancia roja	[0, ∞) acotado a [0, 3000]	100

Todos ellos se determinan en un *slider* o barra deslizante que permite seleccionar los valores en una escala continua de números enteros. Al ser la escala de los parámetros ancho y alto muy amplia, dificultando así el ajuste con precisión de estos parámetros. Se aporta una pestaña con algunos de los valores de resolución más comunes y otras presentes en las especificaciones del sensor. Todos ellos los podemos ver en la Tabla 2 .

[1] Para saber más de las características de la cámara consultar APÉNDICE A: MÓDULO DE LA CÁMARA .

Tabla 2: Resoluciones facilitadas en la pestaña de la página web.

Resolución	Dimensiones (en píxeles)	Observaciones
VGA	640x480	Presente en la especificación del sensor. Ratio ancho/alto = 1,3.
XGA	1024x768	Ratio ancho/alto = 1,3.
Widescreen	1600x1200	Ratio ancho/alto = 1,3.
720p	1280x720	Presente en la especificación del sensor.
980p	1280x980	Presente en la especificación del sensor.
1080p	1080x720	Presente en la especificación del sensor.
QSXGA	2590x1944	Por defecto. Se especifica en la web al presionar el botón para cargar la configuración por defecto. Presente en la especificación del sensor.

2.2.3 LA INTERFAZ GRÁFICA

El diseño de la página web es el que se muestra a continuación en la Ilustración 2 y la Ilustración 3 .

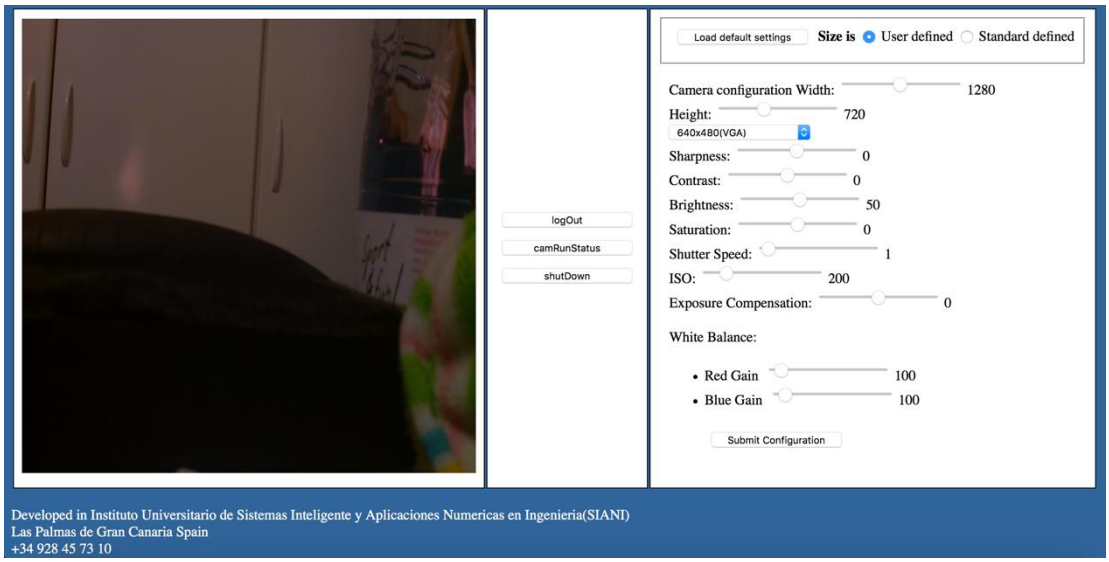


Ilustración 2:Vista de la página web de la Raspberry Pi vista desde el Macbook Pro como superusuario con una imagen de prueba.

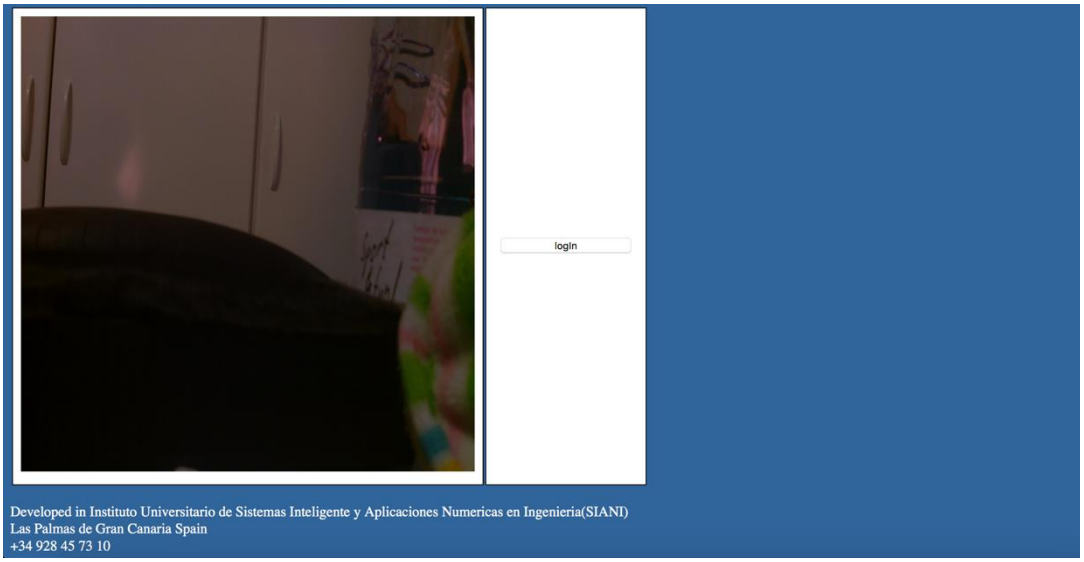


Ilustración 3: Vista de la página web de la Raspberry Pi vista desde el Macbook Pro como usuario normal con una imagen de prueba.

Se ha optado por un diseño en horizontal que permita tener toda la información en el sentido de la lectura. De forma que la información que recibe el usuario está situado a la izquierda, siendo así el primer elemento que procesa el usuario, y las acciones a la derecha de forma que sigue el flujo natural de "recibir información" seguido de "tomar decisiones al respecto".

Los botones de la franja interior están colocados de forma que el botón de "logIn/logOut" que es el que primero se usa está en la parte superior, seguido de los botones del superusuario: el botón que controla en encendido y apagado de la cámara y el botón que apaga la Raspberry Pi.

En la franja de la parte derecha de la Ilustración 2 se encuentra el panel que contiene los deslizadores correspondientes a los parámetros presentados en la Tabla 1. De esta franja solamente cabe destacar el bloque superior en el que se encuentra el botón que carga los valores por defecto en la web del cliente, proponiéndola para su envío, y la opción de seleccionar de donde viene la información de las dimensiones de la imagen: con el parámetro "User defined" la información se tomará de los deslizadores de ancho y alto mientras que con la opción "Standard defined" se tomarán los valores de la pestaña que fueron presentados en la Tabla 2 .

2.2.4 EL SERVIDOR

La página web de la Raspberry Pi se construye sobre el servidor Apache. Este servidor no está instalado por defecto en la versión mínima del sistema operativo que ejecuta la Raspberry Pi sino que tiene que ser instalado y complementado con el módulo del intérprete de PHP 5 para ello se usa el comando '`sudo apt-get install apache2 php5`'. Una vez completada la instalación se podrán colocar los ficheros del servidor web en la ruta `/var/www/html` y se modifican los permisos para que el grupo propietario sea el correspondiente con www-data y su usuario propietario sea www-data que se trata del usuario que controla el servidor Apache.

El conjunto de ficheros que se encuentran en el servidor web se lista a continuación:

- Ficheros PHP:
  - downgradeStatus.php : cambia el estatus del usuario de superusuario a usuario corriente.
  - index.php : genera la página principal.
  - lib_php.php : librería de funciones para generar la página principal dinámicamente.
  - lib.php : librería de funciones para generar el contenido estático (cabeceras HTML) de la página principal.
  - upgradeStatus.php : cambia el estatus del usuario de usuario común a usuario corriente siempre que no haya ningún otro usuario.
  - postHandler.php : recibe la información de la cámara del superusuario y se la retransmite al control de la cámara.
- Ficheros Javascript
  - Jquery.js : framework de Javascript.
  - Script.js : conjunto de funciones para la modificación de la página web, sin la intervención del servidor.
- Fichero CSS
  - wide_screen.css : hoja de estilo para la página web.
- Fichero TXT
  - cache.txt : fichero donde se almacena la dirección ip del usuario para saber si existe o no un superusuario.

El motivo por el que es necesario un fichero de text (.txt) es porque no existe una memoria que esté activa desde el momento en el que se inicia el intérprete PHP hasta que éste se desactiva.

El flujo de interacción entre los ficheros presentados anteriormente se muestra en la Ilustración 4

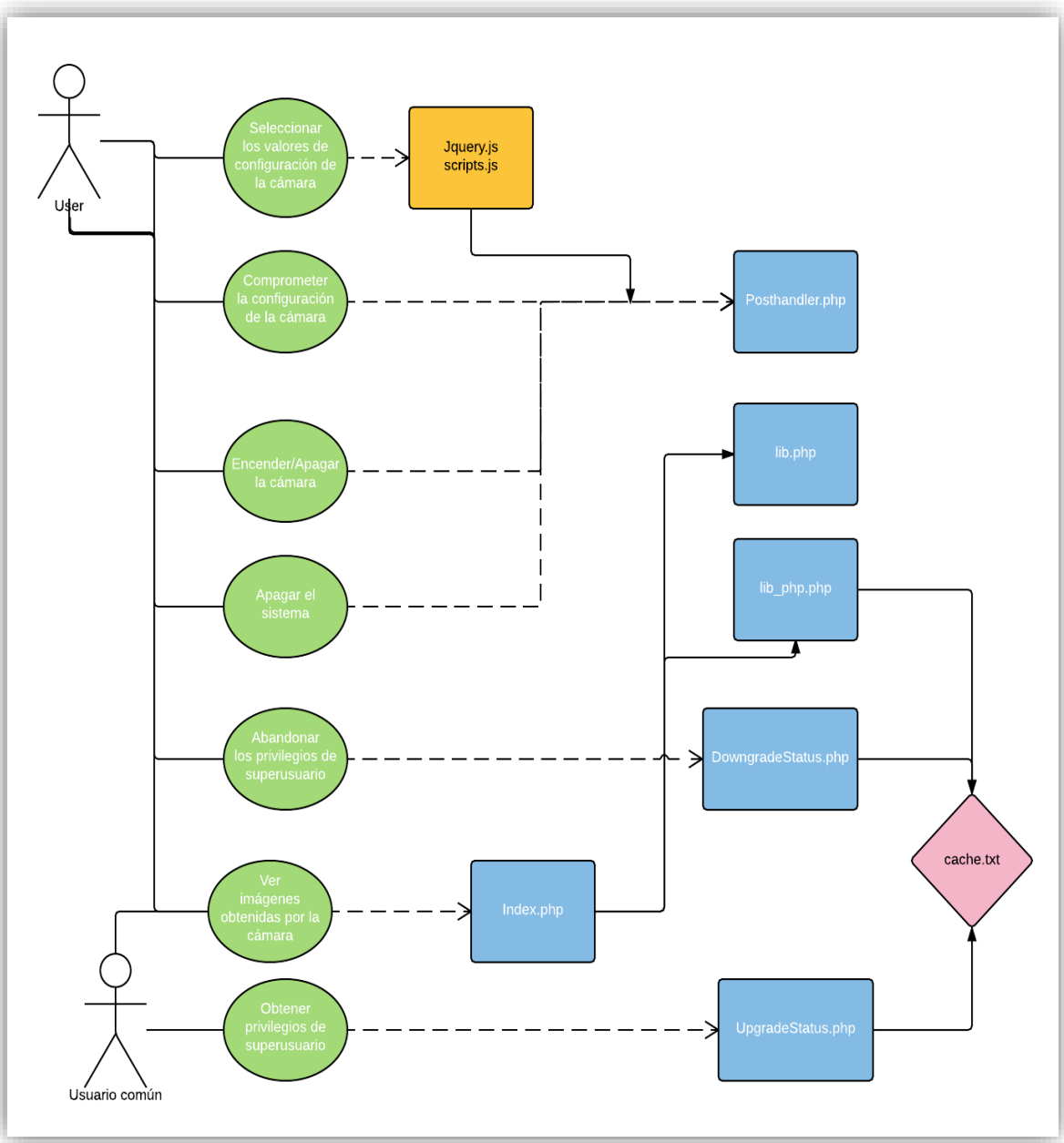


Ilustración 4: Diagrama de casos de uso interactuando con los ficheros del servidor.

2.2.5 COMUNICACIÓN ENTRE EL SERVIDOR WEB Y EL CONTROL DE LA CÁMARA

La comunicación entre el servidor y el control de la cámara se realiza mediante el envío de objetos JSON a través de sockets. Cada vez que el superusuario quiera apagar o encender la cámara, cambiar la configuración de la misma o apagar la Raspberry Pi se abrirá un socket de comunicación desde el intérprete de PHP que trabaja en el puerto 80 al control de la cámara que estará escuchando en el puerto 3490.

El tipo de conexión será orientado a conexión bajo el protocolo TCP que nos asegura la recepción de lo paquetes correctos y en orden.

Los tipos de objetos JSON que puede transmitirse al control de la cámara a través del script de PHP 'postHandler.php' se representan en la Tabla 3.

	Cambio de estado de la cámara	**Apagado del sistema**	**Cambio de configuración de la cámara**
Control	'camStatus'	'shutDown'	'camSettings'
Width	-	-	Valor entero
Height	-	-	Valor entero
Sharpness	-	-	Valor entero
Contrast	-	-	Valor entero
Brightness	-	-	Valor entero
Saturation	-	-	Valor entero
ShutterSpeed	-	-	Valor entero
ISO	-	-	Valor entero
ExposureCompensation	-	-	Valor entero
redGain	-	-	Valor entero
blueGain	-	-	Valor entero

Tabla 3: Tipos de objetos JSON

2.2.6 DESARROLLO DEL CONTROL DE LA CÁMARA

El objetivo de acceder a la cámara es obtener una imagen por segundo con la configuración que se decida desde el cliente web, para ello se ha desarrollado un programa concurrente que tiene las siguientes funcionalidades:

- Escuchar en el puerto 3490 a la espera de objetos JSON para la modificación de los parámetros de la cámara.
- Tomar una imagen desde la cámara y guardarla en una matriz de OpenCV.
- Grabar la matriz de OpenCV en la memoria y en el servidor web.
- Apagar el sistema operativo.
- Apagar o encender la cámara.

La concurrencia se realiza mediante la creación de dos hilos de ejecución uno para la escucha desde el puerto y otro para la captura de las imágenes.

La región crítica que comparten ambos procesos es la estructura que contiene todos los parámetros de configuración de la cámara cuyo acceso está controlado por un mutex.

Otra variable de control de concurrencia que aparece en este programa es la variable condición que controla el encendido y apagado de la cámara. Esta variable pone en reposo el hilo de la cámara al comienzo de la ejecución (gracias a esto la cámara comienza apagada) y no es hasta que el hilo que escucha de la red recibe la orden de cambiar el estado de la cámara que la cámara no se enciende.

Los dos hilos están desvinculados del proceso principal que los generó, por lo que al morir este proceso principal ambos hilos siguen ejecutándose en el sistema. Para el sistema operativo este estado en el que los hilos del proceso siguen vivos pero el programa padre no lo está se etiqueta como *defunct* .

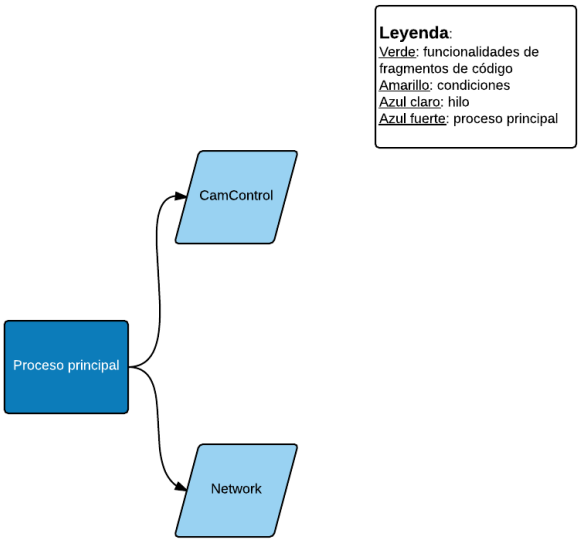


Ilustración 5:Creación de hilos

Las tareas que realiza el hilo del control de la cámara son las que se muestran en la ilustración 5.

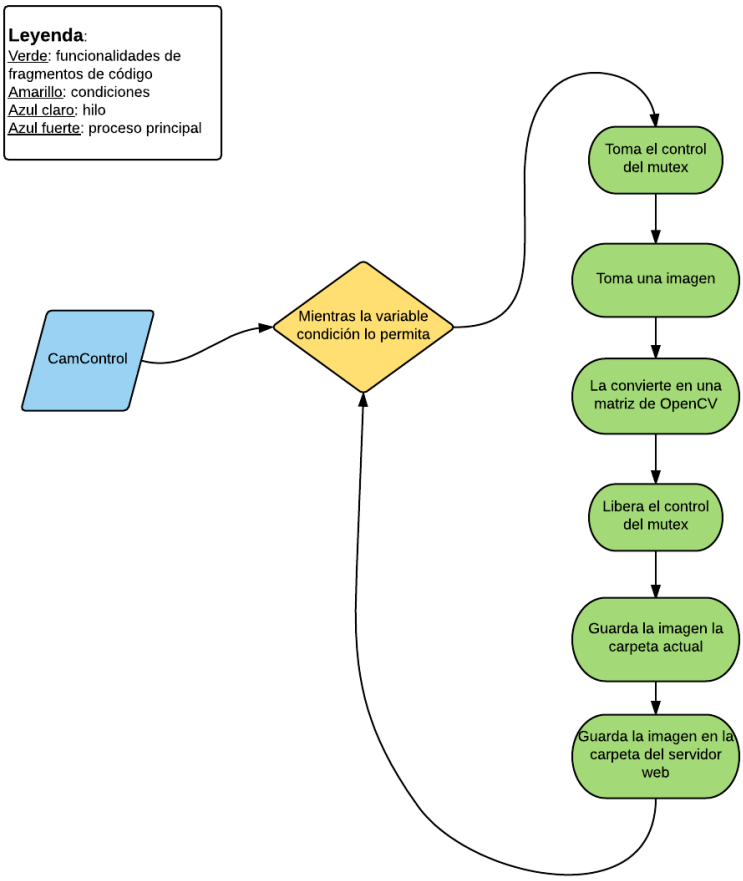


Ilustración 6: Funcionamiento del control de la cámara

Las imágenes obtenidas por el control de la cámara tienen el formato .jpg para aprovechar que este sistema de codificación está acelerado por hardware, al igual que el formato de compresión de video H.264. Una vez tenemos la información de la imagen en el buffer, realizamos la conversión a la matriz de la librería OpenCV con el fin de aprovechar la función de esta misma librería de guardado de matrices como imágenes en el sistema de ficheros.

Las imágenes guardadas tendrán como título una cadena que tiene la siguiente estructura: <Mes>_<Día><hora>:<minuto><segundo>.jpg . De esta forma, el control de la cámara evita sobrescribir imágenes.

Por otro lado al guardar la imagen en el servidor web es necesario sobreescribir la imagen “img.jpg” dado que el navegador del cliente solo conoce la localización de ésta y no la ruta ni el nombre a ninguna otra imagen.

Mientras que las tareas del hilo que escucha a través del puerto realiza las tareas que recoge la ilustración 6.

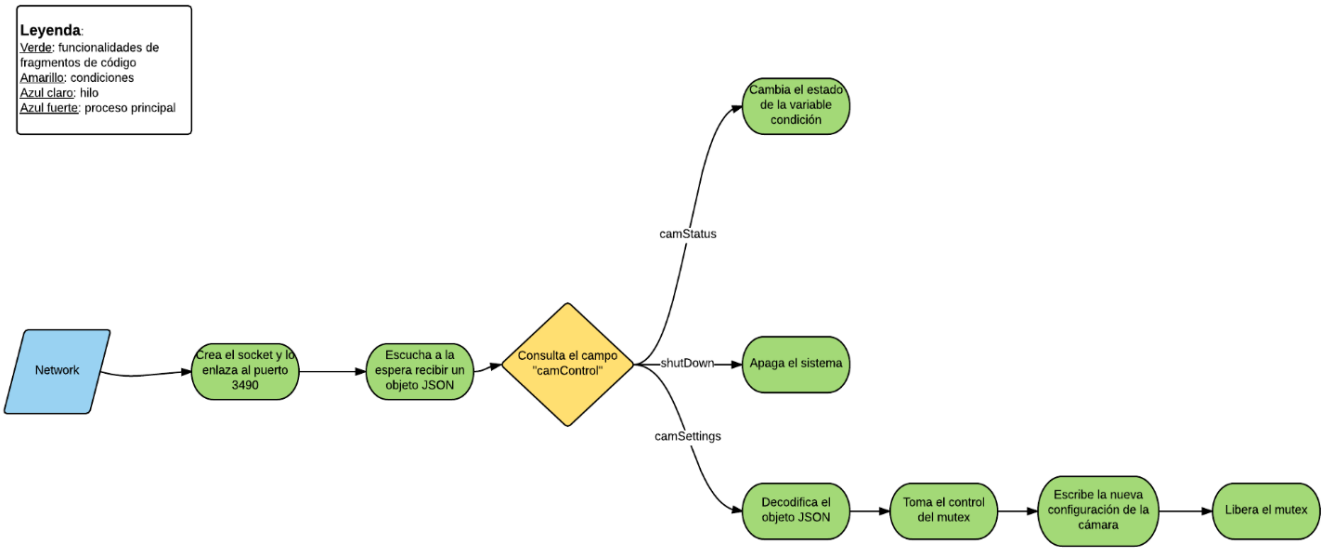


Ilustración 7: Funcionamiento del hilo conectado a la red.

La interpretación del objeto JSON que se obtiene a través del socket se realiza utilizando la librería "jansson" para C++ que nos permite saber si el objeto está léxica y sintácticamente correcto antes de modificar ningún parámetro de la configuración de la cámara, y una vez sabemos que está correcto modificamos los parámetros uno por uno.

El primer parámetro que se obtiene es *camControl* que nos permite cambiar el flujo del programa para apagar o encender la cámara, apagar el sistema o modificar la configuración de la cámara.

En el caso de que desee modificar la configuración de la cámara hay que realizar una conversión de la cadena de caracteres del objeto JSON al tipo entero con el que la estructura de configuración de la cámara trabaja.

2.3 ELABORACIÓN DE UNA CARCASA PARA LA PROTECCIÓN DEL SISTEMA

Una vez definida la infraestructura software necesaria para adquirir una base de imágenes sobre las que basar el desarrollo de los algoritmos de detección, se procedió a construir un contenedor para proteger el sistema de adquisición de salpicaduras y del spray marino durante la toma de imágenes.

Lo primero que haremos es abrir en uno de los laterales un agujero que será por donde se meterá la óptica de la cámara para la captura de las imágenes y le colocaremos el cristal por la parte externa del tupper con cinta adhesiva de butilo, utilizada en el campo de la robótica submarina y para evitar derrames causados por la exposición al sol de dicha cinta se protegen las terminaciones con cinta aislante.

Ilustración 8: Carcasa.

Tras ello colocamos recortamos el metacrilato para que coincida con la base interior del tupper y suavizamos los bordes con una lija de mano. Una vez tenemos el metacrilato listo colocamos los componentes en la posición que van a ocupar una vez montados y marcamos las posiciones sobre el metacrilato. Perforamos dichas posiciones y colocamos los tornillos y los separadores para colocar los componentes en las posiciones deseadas. De esta forma obtenemos una primera versión como se muestra en la Ilustración 9.

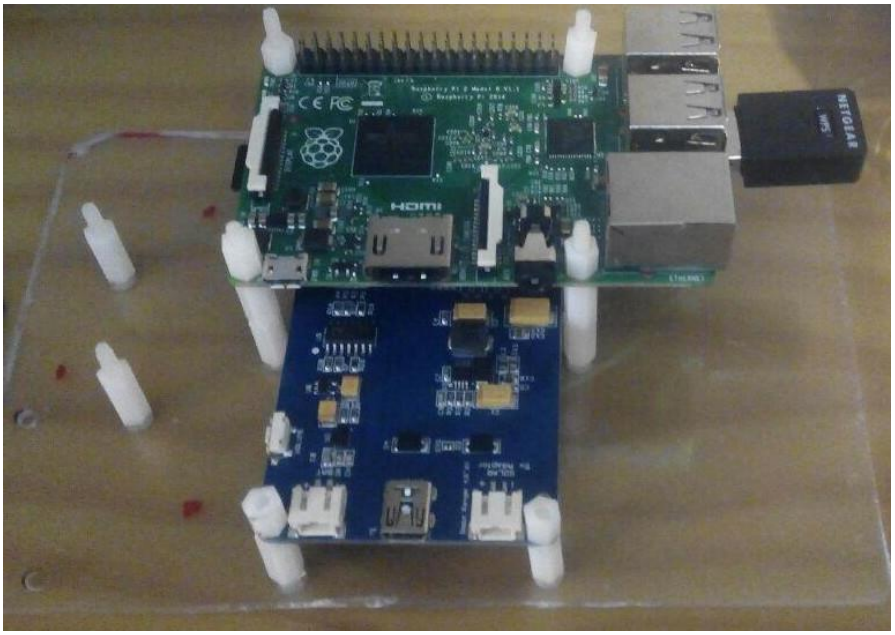


Ilustración 9: Montaje en la fase 1.

Para alimentar a la Raspberry Pi, conectaremos una batería de 6000mAh y 3,7V a la Lipo Rider Pro (placa azul de la Ilustración 9: Montaje en la fase 1.) para que ésta suministre 5V a la entrada de corriente de la Raspberry Pi mediante el puerto mini USB.

Ilustración 10: Sistema montado en la Raspberry Pi

Además le conectaremos la cámara al puerto CSI de la Raspberry Pi, que para facilitar su anclaje a la carcasa se la ha enfundado con una carcasa para adaptador de tarjetas microSD al que se le ha elaborado dos aberturas, una para pasar el cable de conexión y el segundo para sacar el objetivo con la lente. Tal y como se muestra en las Ilustración 11Ilustración 1

Ilustración 11: Cámara parte delantera (izquierda) y delantera (derecha)

La visualización de las imágenes mientras éstas se capturan es posible gracias a la colocación de un segundo recorte de metacrilato sobre la Raspberri Pi que permite, al colocar una tira de velcro que permite la sujeción del Smartphone mostrando la

pantalla a través de la tapa superior del tupper tal y como se muestra en la Ilustración 12.

Ilustración 12: Web de la Raspberry Pi vista en el smartphone a través del contenedor.

Con el sistema ya construido lo someteremos a unas pruebas antes de llevarlo al mar para tomar las fotografías.

2.4 PRUEBAS

Las pruebas que realizaremos son las siguientes:

- Tomar fotos en el exterior.
- Comprobar que se toma una foto por segundo.
- Calcular aproximadamente la duración de la batería.

2.4.1 TOMA DE FOTOS EN EXTERIORES

Al tomar las fotos en el exterior, usando la configuración por defecto de la librería de la cámara, toma las fotos con tonalidad verde debido a que la configuración por defecto de la cámara tiene como velocidad del obturador 0, cuyo significado en el manual es velocidad automática, pero ante fotos en el exterior tiene tendencia a entrar en modo nocturno.

En este modo el obturador tiene una velocidad muy baja, es decir, está mucho tiempo abierto y además solo toma la información del canal verde, dándole así la tonalidad verde a las fotos.

La corrección de este fallo pasa por realizar un programa que toma una imagen variando la velocidad del obturador. Algunas de las obtenidas se muestran en la Ilustración 13, la Ilustración 14, la Ilustración 15, la Ilustración 16, la Ilustración 17 y la Ilustración 18.

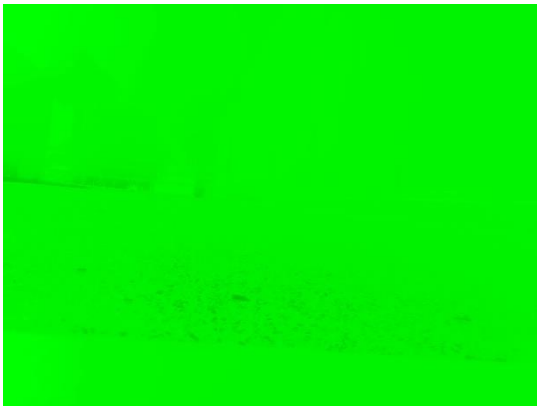


Ilustración 13: Imagen tomada con el obturador abierto 0ms.(Entrada automática en modo nocturno)

Ilustración 14: Imagen tomada con el obturador abierto 100ms.

Ilustración 15: Imagen tomada con el obturador abierto 200ms.

Ilustración 16: Imagen tomada con el obturador abierto 300ms

Ilustración 17:Iimagen tomada con el obturador abierto 900ms

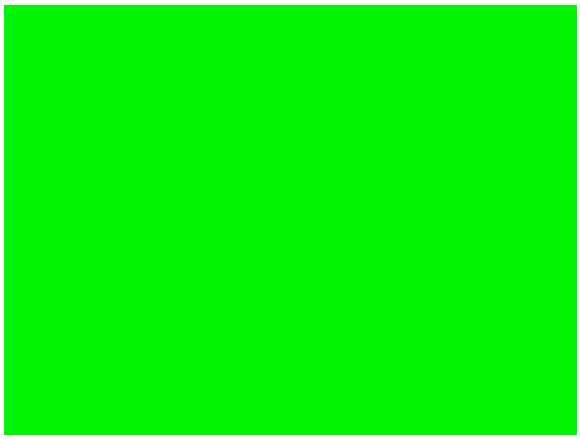


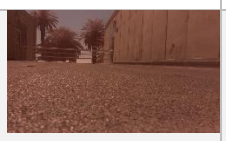
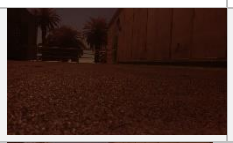
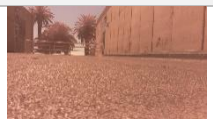
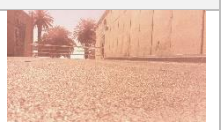
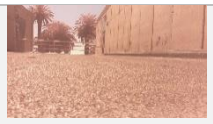
Ilustración 18: Imagen tomada con el obturador abierto 4200ms (modo nocturno)

Como se puede comprobar a partir de 900ms de exposición las imágenes empiezan a perder color y no nos interesan al igual que no nos interesan aquellas en modo nocturno, es por ello que preferimos aquellas imágenes que se obtienen con un tiempo de exposición de más de 1 ms a 300ms.

Sin embargo, si planteamos un escenario donde tenemos un objeto que se mueve a 0.5m/s * 0.3s = 0.15m , el objeto se habrá movido 15 cm antes de que se cierre el obturador es por ello que tenemos que buscar otra forma de obtener las imágenes con tanto color con menos tiempo de exposición. Para ello haremos una prueba muy similar a la anterior donde no solo tocamos el tiempo de exposición sino

también la sensibilidad ISO que nos permite equilibrar la cantidad de luz. Los resultados de esta segunda prueba los encontramos en las ilustraciones de la Tabla 4:

Tabla 4:Comparación de las imágenes obtenidas respondiendo a su valor ISO y a su tiempo de exposición.

Tiempo de exposición	ISO 100	ISO 200	ISO 400	ISO 800
10ms				
20ms				
30ms				
80ms				
100ms				

A la vista de las imágenes parece que tienen mejor calidad aquella con 80ms y 200 de ISO así como aquella con 30ms y 400 de ISO. Escogemos aquella que tiene menor tiempo de exposición para evitar que los objetos en la escena puedan moverse una distancia considerable durante el tiempo en que se tarda en tomar una fotografía.

Al establecer estos nuevos parámetros como configuración inicial podemos proceder a realizar las fotos en el exterior.

2.4.2 COMPROBACIÓN DEL INTERVALO DE CAPTURA

El objetivo es tomar una fotografía por cada segundo una vez el usuario haya dado la orden de encender la cámara, esto lo comprobamos añadiendo código que nos permita saber cuánto ha tardado en ejecutar cada uno de los pasos y obtenemos que para la resolución definida por defecto el sistema tarda aproximadamente unos 6 segundos. Al ser este dato inesperado tomamos los datos de tiempo de ejecución para cada una de las resoluciones que hemos definido en la pestaña de la interfaz gráfica, obteniendo así los siguientes resultados:

Tabla 5: Tiempo de ejecución por paso, para la resolución 2590x1944

2590x 1944	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Tomar la foto	1,43	1,39	1,47	1,42	1,43	1,428
Convertir buffer en matriz	1,53	1,53	1,47	1,48	1,49	1,5
Escritura en disco 1	1,72	1,67	1,67	1,69	1,67	1,684
Escritura en disco 2	1,66	1,66	1,66	1,68	1,67	1,666
Suma	6,34	6,25	6,27	6,27	6,26	6,278

Tabla 6: Tiempo de ejecución por paso, para la resolución 1080p

1080x720 (1080p)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Tomar la foto	0,4	0,38	0,38	0,39	0,38	0,386
Convertir buffer en matriz	0,29	0,3	0,3	0,31	0,31	0,302
Escritura en disco 1	0,33	0,34	0,33	0,33	0,33	0,332
Escritura en disco 2	0,33	0,33	0,33	0,33	0,33	0,33
Suma	1,35	1,35	1,34	1,36	1,35	1,35

Tabla 7: Tiempo de ejecución por paso para la resolución 980p

1280x980 (980p)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Medi a por paso
Tomar foto	0,65	0,64	0,62	0,62	0,63	0,632
Buffer a matriz	0,44	0,44	0,44	0,44	0,44	0,44
Escritura en disco	0,5	0,5	0,5	0,5	0,5	0,5
Escritura en disco 2	0,5	0,5	0,5	0,5	0,53	0,506
Suma	2,09	2,08	2,06	2,06	2,1	2,078

Tabla 8: Tiempo de ejecución por paso para la resolución 720p

1280x720 (720p)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Toma foto	0,41	0,4	0,4	0,4	0,42	0,406
Buffer a matriz	0,3	0,31	0,3	0,31	0,31	0,306
Escritura en disco	0,34	0,34	0,34	0,34	0,34	0,34
Escritura en disco 2	0,34	0,34	0,34	0,34	0,34	0,34
Suma	1,39	1,39	1,38	1,39	1,41	1,392

Tabla 9:Tiempo de ejecución por paso para la resolución widescreen

1600x1200 (wide-screen)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Toma foto	0,96	0,94	0,98	0,93	0,98	0,958
Buffer a matriz	0,66	0,66	0,66	0,66	0,66	0,66
Escritura en disco	0,77	0,77	0,78	0,77	0,77	0,772
Escritura en disco 2	0,86	0,77	0,82	0,76	0,77	0,796
Suma	3,25	3,14	3,24	3,12	3,18	3,186

Tabla 10: Tiempo de ejecución por paso para la resolución XGA.

1024x768 (XGA)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Tomar foto	0,35	0,34	0,33	0,34	0,33	0,338
Buffer a matriz	0,26	0,25	0,26	0,26	0,26	0,258
Escritura en disco	0,29	0,29	0,28	0,29	0,29	0,288
Escritura en disco	0,28	0,28	0,28	0,28	0,28	0,28
Suma	1,18	1,16	1,15	1,17	1,16	1,164

Tabla 11: Tiempo de ejecución por paso para la resolución VGA

640x480 (VGA)	t(s) para la primera imagen	t(s) para la segunda imagen	t(s) para la tercera imagen	t(s) para la cuarta imagen	t(s) para la quinta imagen	Media por paso
Tomar foto	0,15	0,14	0,15	0,15	0,14	0,146
Buffer a matriz	0,1	0,1	0,1	0,1	0,1	0,1
Escritura en disco	0,11	0,1	0,11	0,1	0,1	0,104
Escritura en disco	0,11	0,1	0,1	0,11	0,11	0,106
Suma	0,47	0,44	0,46	0,46	0,45	0,456

Como es de esperar el tiempo de ejecución disminuye al disminuir la resolución puesto que se reduce el número de bytes a transmitir. Lo que llama la atención es que el tiempo de ejecución de la toma de la fotografía y el guardado en el sistema de ficheros es comparable cuando la toma de la fotografía debería ser mucho más rápida.

Con esta incógnita me puse en contacto con el desarrollador de la librería con la que se captura la imagen (OMXCam), Gabriel Llamas, quien me explicó que las operaciones con la cámara por lo general se suelen realizar en la GPU, pero con su librería se realizan en la CPU puesto que la documentación para implementar operaciones en la GPU de la Raspberry Pi no está liberado. Otro punto sobre el que arrojó luz Gabriel Llamas, es que la cámara, además de tomar la imagen, realiza operaciones como el balanceo de blanco que pueden producir que tarde más tiempo la captura de la fotografía.

Sabiendo esto se opta por modificar la resolución por defecto de la cámara a la resolución 720p que da una imagen mayor y es más interesante para las fases de desarrollo en las que el sistema de visión por computador está aún en fase experimental. En la práctica se utilizará una resolución de 640x480 (VGA) para reducir el tiempo de ejecución de los cálculos.

2.4.3 ESTIMACIÓN DE LA DURACIÓN DE LA BATERÍA

Para estimar la duración de la batería tenemos que tener en cuenta los siguientes factores:

- La batería que está integrada en el sistema tiene una carga de 6000mAh según el fabricante y suministra un voltaje de 3,7V.
- La Raspberry Pi demanda 5V de entrada por el puerto mini-USB.
- La batería se conecta a la Lipo Rider Pro, la cual se encargará de subir el voltaje de los 3,7 V de la batería a los 5V que requiere la Raspberry Pi.

En un caso ideal la intensidad suministrada por la batería llegaría íntegramente a la Raspberry Pi, pero como esto no es cierto sino que se pierde parte al pasar por la Lipo

Rider consideramos que existe un cierto factor λ que juega en nuestra contra decrementando la duración de la batería.

De forma experimental medimos la intensidad demandada por la Raspberry Pi montando el siguiente circuito:

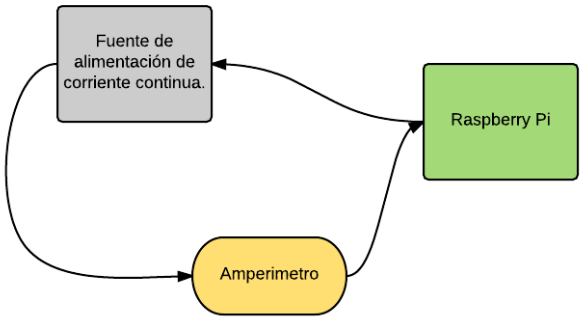


Ilustración 19: Circuito elaborado para medir el consumo de la Raspberry Pi.

Las medidas obtenidas se muestran en la tabla Tabla 12:

Tabla 12: Relación de medidas obtenidas del experimento.

Estado de la Raspberry Pi	**Intensidad máxima leída (mA)**	**Duración aproximada de la batería**
Con la cámara apagada y sin conexión Wifi.	280	21,42 horas
Con la cámara apagada y con una conexión	400	15 horas
Cámara encendida(tomando y guardando imágenes) y con conexión	500	12 horas

Este último caso es con el que nos vamos a encontrar a la hora de tomar el conjunto de imágenes de entrenamiento. Por tanto, tenemos como máximo 12 horas para tomar tantas fotografías como nos sea posible.

3 DESARROLLO DEL SISTEMA DE VISIÓN POR COMPUTADOR

3.1 CAPTURA DE LAS IMÁGENES

Podemos clasificar las imágenes que se pueden tomar en el medio marino en tres categorías:

- Solo la superficie del agua
- La superficie del agua con el horizonte y el cielo
- La superficie del agua con el horizonte, el cielo y un perfil orográfico.

En cada una de estas categorías podremos encontrar objetos flotantes de diferentes dimensiones. En este proyecto circunscribiremos el problema de detección objetos flotantes a la primera categoría y con objetos flotantes de pequeñas dimensiones. Así pues, nos dirigimos a Sardina del Norte a tomar las fotografías con una serie de objetos, compatibles con muchos de los objetos de origen antrópico que se pueden encontrar flotando en el mar como "basura marina". Los objetos poseen distintas características y que se detallan en la tabla:

Tabla 13: Relación de los objetos utilizados y sus principales características

Objeto	Característica principal	Foto.
Bote de refrigerante	Su superficie es de color uniforme.	
Brick de zumo	Su superficie tiene distintos colores y texto.	
Brick de leche	Su superficie tiene distintos tonos del mismo color.	
Botella de refresco	Es transparente	

Del total de imágenes se seleccionaron 203 para utilizarlas como conjunto de entrenamiento y validación del sistema de visión por computador.

3.2 SISTEMA DE VISIÓN POR COMPUTADOR

El sistema de visión por computador tiene por objetivo determinar si hay un objeto en el agua para - eventualmente - evitarlo. La detección de los objetos se realiza bajo la suposición de que todos los píxeles que pertenecen al mar son similares en cuanto a color entre ellos.

La definición de color depende del espacio de color en el que se está trabajando, para este caso utilizaremos el espacio de color YCbCr. Este espacio de color está formado por tres canales: Y correspondiente información de la luminosidad de la imagen, Cb y Cr son las componentes de información de color azul y rojo respectivamente.

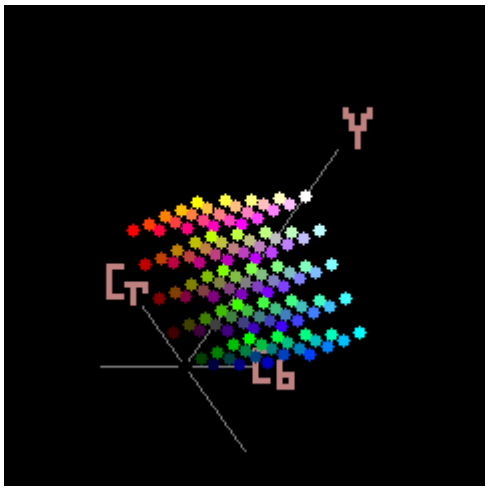


Ilustración 20: Representación del espacio YCbCr

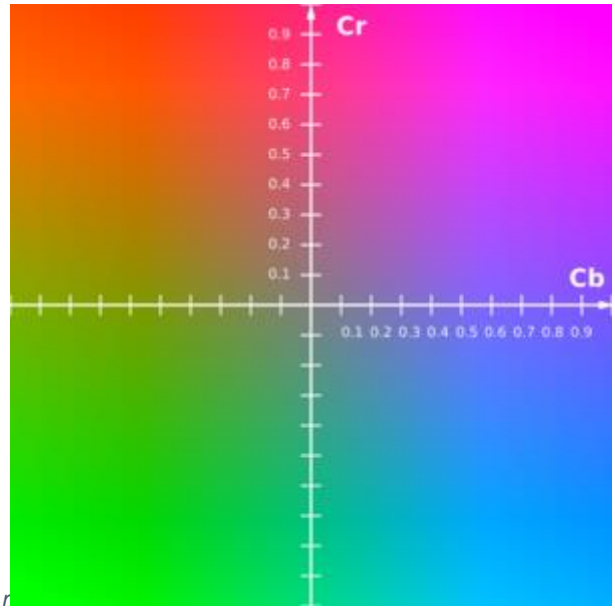


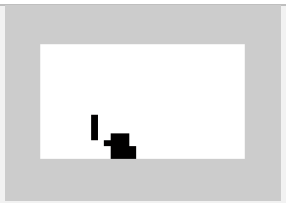
Ilustración 211: Plano de colores en el espacio YCbCr cuanod Y =0.5

En este espacio de color definiremos el color como el vector (Cb Cr) para cada píxel. En Matlab los valores que pueden adoptar cada uno de estos componentes pertenecen al intervalo entero [16, 240]. Al utilizar únicamente las componentes de color se obtiene un sistema que muestra a los cambios de intensidad debidos a las olas superficiales.

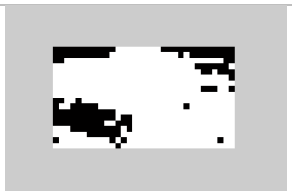
La imagen que captura la cámara se subdivide en ventanas de n x m píxeles que en este caso son 40 x 40 píxeles. Sobre cada una de ellas aplicamos un descriptor que nos ayude a decidir si el contenido de la ventana es objeto o no.

Aprovechando que un canal del espacio de color contiene información sobre el color azul, y las escenas tienen en su mayoría mar, como primera intuición se desarrolla un descriptor basado en la media del componente Cb del espacio YCbCr. Se prueba utilizando un umbral que discrimina entre fondo y objeto puesto a mano para cada uno de los ejemplos:

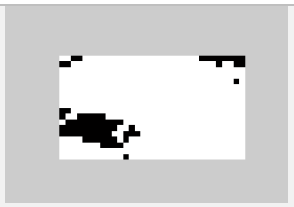
Tabla 14: Resultados del descriptor Cb

Original	**Umbral**	**Resultado**
	5	
	2.5	
	2	
	3	
	4	

4



6

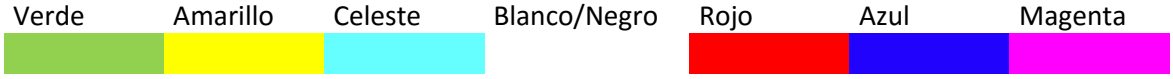


De estos resultados extraemos como conclusión que los objetos de color rojo no son se discriminan bien porque no se tiene información sobre este color. Por ello se desarrolla un descriptor basado en la multiplicación de los valores medios de los dos canales mostrados anteriormente, $D = Cb_{medio} * Cr_{medio}$ de todos los píxeles de la ventana. Esta nueva dimensión nos da un valor para color que nos permite ordenarlos en la escala mostrada en la Tabla 16 según los valores de la Tabla 15.

Tabla 15: Valores de los principales colores en los canales RGB y sus correspondientes valores CbCr del espacio de color YCbCr.

Color	R	G	B	Cb	Cr	Cb*Cr
Rojo	255	0	0	90	240	21600
Azul	0	0	255	240	110	26400
Verde	0	255	0	54	34	1836
Magenta	255	0	255	202	222	44844
Amarillo	255	255	0	16	146	2336
Celeste	0	255	255	166	16	2656
Blanco	255	255	255	128	128	16384
Negro	0	0	0	128	128	16384

*Tabla 16: Escala de colores según su valor Cb*Cr.*



La diferenciación entre mar y objeto está sujeta, al menos en esta primera versión, a la hipótesis de que la mayoría de las ventanas contendrán agua. Esto nos lleva a asumir

que la media del descriptor para todas las ventanas es muy similar al valor del descriptor para una ventana con mar cuyos colores son similares al azul.

Con esto podemos determinar que si para una ventana el descriptor varía mucho es porque en esa ventana hay objeto o parte de él. Esta diferencia la calculamos como el valor absoluto de la distancia euclídea entre el descriptor medio para toda la ventana y el descriptor para esta ventana en concreto. Si el valor absoluto de la distancia euclídea es mayor que cierto umbral de marca dicha ventana como objeto y si queda por debajo esta se marca como mar.

Tabla 17: Escala de colores obtenida al calcular el valor absoluto de la distancia a la media cuando la media es un tono de azul.

Azul	Rojo/Magenta	Blanco/Negro	Celeste	Amarillo	Verde

Aunque a priori nos parezca plausible la hipótesis de que los puntos pertenecientes al mar estén concentrados en una zona del espacio de color CbCr y los del objeto en otra zona de forma que ambos conjuntos sean linealmente separables las pruebas experimentales nos muestran lo contrario tal y como se muestra en la Ilustración 24. En ella se ha tomado una imagen (Ilustración 22) y una máscara como entrada (Ilustración 23). La máscara tiene en negro todos aquellos píxeles que pertenecen al objeto o a su reflejo y en blanco todos aquellos valores que son el mar. A partir de la máscara tendremos marcados en rojo los puntos pertenecientes al mar, en azul los pertenecientes al objeto.

Ilustración 22: Imagen original

Ilustración 23: Máscara

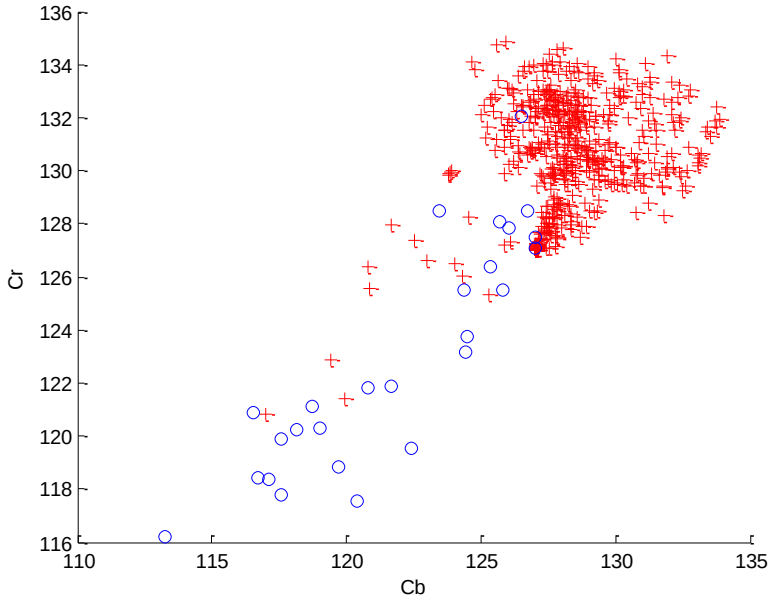


Ilustración 24:Valores de los canales para cada ventana. En rojo los puntos pertenecientes al mar, en azul los pertenecientes al objeto.

A la vista de ambos conjuntos no son linealmente separables se desarrolló un clasificador basado en el cálculo de un umbral que detectase parte de los puntos que pertenecen al objeto.

El cálculo del umbral está basado en cálculos sobre el histograma de la imagen. Damos por hecho de la hipótesis anterior que habrá una clase con mayor número de ocurrencias asociado a una distancia pequeña al valor del descriptor medio; dicha clase se corresponderá con las ventanas que contienen mar. Esta clase aparecerá en el extremo izquierdo del histograma.

A continuación, vendrán un número pequeño de clases con menor número de ocurrencias y una distancia un poco mayor que la primera clase. Estas clases están asociadas con las zonas de reflejos y sombras del mar.

Por último, tendremos una última clase que será la clase del objeto que está será la que tenga menor número de ocurrencias. Sabiendo esto el histograma ideal para distinguir los objetos sería como el que se muestra en la Ilustración 25.

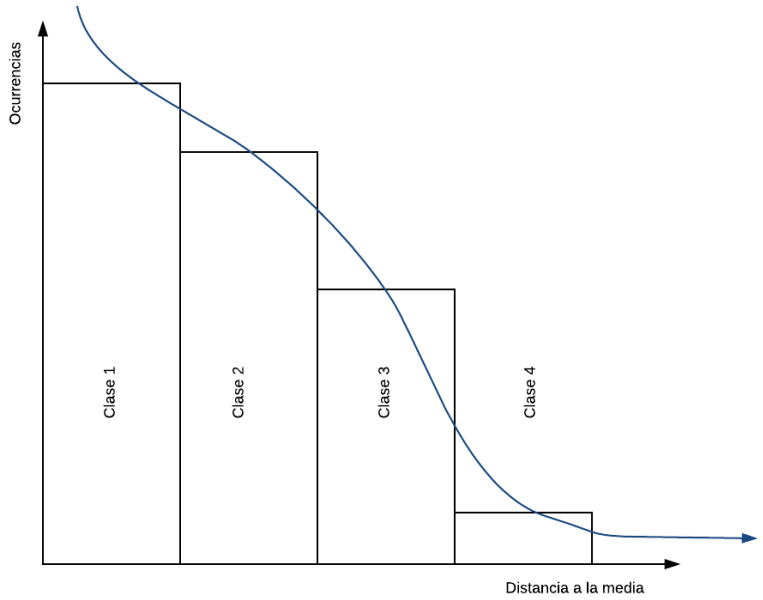


Ilustración 25: Histograma ideal del sistema de visión por computador.

De dicha ilustración podemos determinar que la clase que determina dónde está el objeto se encuentra más allá del punto de inflexión. Para encontrar dicho punto de inflexión calculamos la segunda derivada numérica sobre el historial para encontrar un valor que la aproxima a 0[2], siendo este valor el que tomaremos como umbral.

El resultado del sistema es una imagen de mucha menor dimensión que la original en la que los puntos negros representan las ventanas en las que hay un objeto o parte de él y en blanco todas aquellas zonas donde no hay objeto.

3.3 RESULTADOS DEL SISTEMA DE VISIÓN POR COMPUTADOR

Con el sistema implementado lo ejecutamos con un conjunto de 43 imágenes cuyos resultados podemos clasificar en cuatro tipos de resultado:

1. *Admisible*: el punto más cercano al origen del barco, es decir, el punto más cercano al píxel situado en la última línea inferior de la imagen en la columna del centro, coincide con parte del objeto.
2. *No admisible de tipo 1*: el umbral está mal ajustado porque admite valores de otras clases como parte de la clase objeto.

[2] Se determina que está próximo a 0 cuando el valor absoluto de la segunda derivada cae por debajo de 10^(-3).

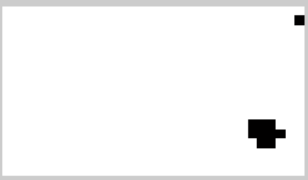
3. *No admisible de tipo 2*: existe un elemento que tiene una distancia mayor que los píxeles que representan al objeto, desplazando así el umbral más hacia la derecha y clasificando los píxeles del objeto como fondo.
4. *No admisible de tipo 3*: no existe objeto en la imagen por lo que se toman los píxeles más lejanos del descriptor medio como objeto.

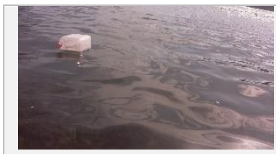
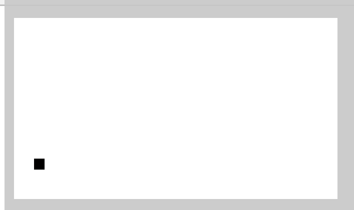
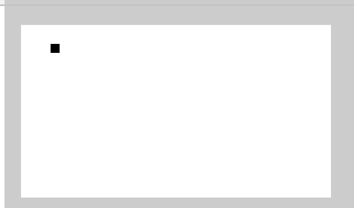
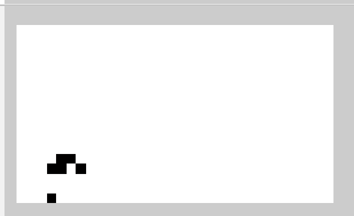
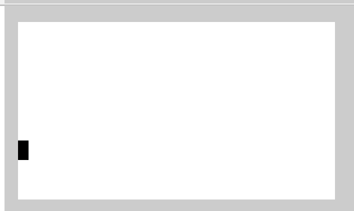
Los casos de los resultados no admisibles de tipos 1 y 3 pueden ser corregidos si el umbral que discrimina entre objeto y mar tuviese en cuenta el umbral obtenido para la imagen anterior.

Tabla 18: Resultados del sistema de visión por computador

Original	Resultado	Valoración
		Admisible
		Admisible
		Admisible
		No admisible Tipo 1
		No admisible Tipo 1
		Admisible

		Admisible
		Admisible
		No admisible Tipo 2
		Admisible
		Admisible
		Admisible
		Admisible
		Admisible

		Admisible
		Admisible
		No admisible de Tipo 2
		No admisible Tipo 2
		No admisible Tipo 2
		No admisible: Tipo 3
		No admisible: Tipo 2
		No admisible: Tipo 2

		Admisible
		Admisible
		Admisible
		Admisible
		Admisible
		Admisible
		Admisible
		Admisible

		Admisible
		Admisible
		Admisible
		Admisible
		No admisible: Tipo 1
		No admisible Tipo 1
		Admisible
		No admisible: Tipo 2

		No admisible: Tipo 1
		Admisible
		No admisible: Tipo 1
		Admisible
		Admisible

3.4 BONDAD DEL SISTEMA DE VISIÓN POR COMPUTADOR

De los resultados mostrados anteriormente extraemos las siguientes estadísticas:

Admisibles	29		67%
No admisibles	14	Clases 1 y 3: 7	16%
		Clase 2: 7	16%

De ellas podemos concluir que el sistema no es muy fiable puesto que falla el 32% de los casos.

Pero en los casos de los errores de tipo 1 y tipo 3 se pueden corregir si en lugar de recalcular el umbral en cada imagen, el umbral obtenido para la imagen anterior tuviera influencia en el umbral actual, en otras palabras, el sistema aprendiera el umbral basado en la imagen anterior.

De esta forma los casos de los errores de tipo 1 y tipo 3 serían corregidos y los podríamos añadir al conjunto de los admisibles consiguiendo así el sistema un 83% de acierto.

3.5 SISTEMA DE APRENDIZAJE DEL UMBRAL

La corrección de los resultados no admisibles de tipos 1 y 3 consiste en el aprendizaje del umbral de forma que el umbral para la nueva imagen se ve influenciado por el umbral de la imagen anterior y el umbral calculado para la imagen actual, siguiendo el modelo matemático siguiente:

$$T_t = (1 - \alpha)T_{t-1} + \alpha T$$

Donde:

- T_t es el umbral que se está calculando para esta imagen.
- T_{t-1} es el umbral calculado para la imagen anterior.
- α es el factor de aprendizaje. Este factor toma valores dentro del intervalo [0,1].
- T es el umbral calculado al calcular el punto de inflexión sobre el historial para la imagen actual.

3.6 COMUNICACIÓN CON EL CONTROL DEL BARCO

El sistema de visión por computador devuelve a su salida una imagen. Esta imagen coincide en dimensiones con el número de ventanas en la horizontal y en la vertical. Los valores que pueden tomar lo píxeles es negro (0) o blanco (1), de forma que los píxeles negros representan objetos y las blancas fondo.

Dado que la cámara es fija, y tiene un ángulo fijo, podemos asociar a cada uno de estos píxeles una distancia aproximada a la que se encontraría el objeto.

El cálculo de esta distancia se basa en la geometría del sistema, tal y como se muestra en la simulación creada con el motor para el desarrollo de videojuegos Unity[3] de la Ilustración 26.

[3] Este motor de videojuegos se caracteriza por tener las unidades del mundo en metros de forma que la simulación se corresponde con una situación real. También permite modificar el ángulo de la cámara para que concuerde con las especificaciones del sensor de la cámara de la Raspberry Pi.

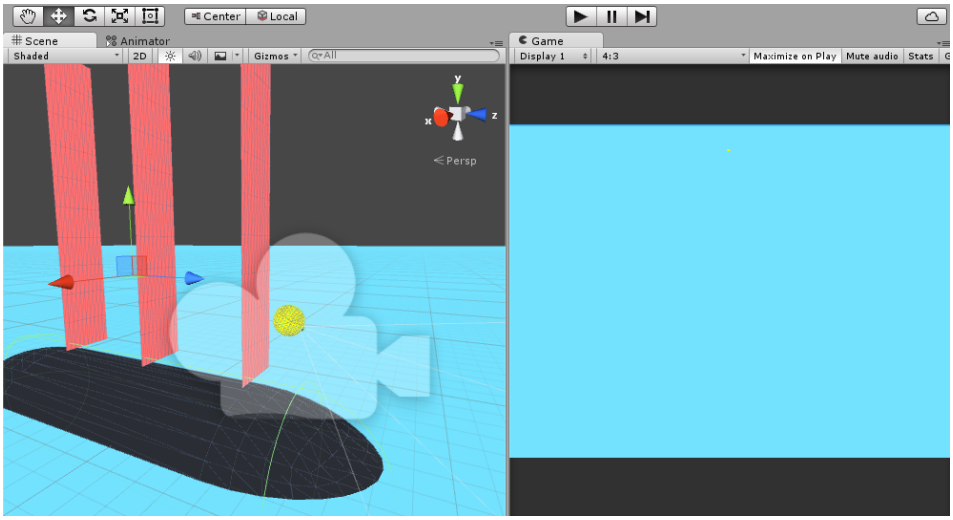


Ilustración 26:Simulación del barco en Unity.

En la Ilustración 26, tenemos a la izquierda un esquema en 3D de lo que sería el barco con la cámara, representada con la esfera amarilla, colocada a 0,5 metros sobre la superficie del mar. La cámara está inclinada -26,6º sobre la horizontal para garantizar que la parte superior de la imagen tomada coincide con la línea del horizonte tal y como se muestra en la parte derecha de la Ilustración 26.

Conociendo el ángulo de cobertura en la vertical del sensor, 41.4º según las especificaciones, podemos calcular la distancia a la que están los puntos de la imagen en la realidad.

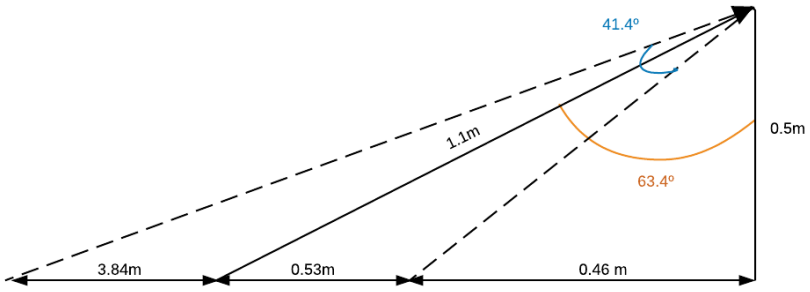


Ilustración 27: Cálculos de la geometría de la imagen.

De la Ilustración 27 extraemos que los píxeles de la parte inferior de la imagen se encuentran a una distancia de 0.46 metros de la cámara, los puntos situados en la mitad de la imagen están a una distancia de 0.99 metros y los puntos de la parte superior de la imagen están a 4.83 metros.

De las especificaciones del sensor extraemos que el ángulo de cobertura en la horizontal es de 53.3º por lo que los límites de la imagen están a -0.44m y a 0.44m a izquierda y derecha respectivamente. Con toda esta información podemos determinar a qué distancia están los cuatro vértices de la imagen en la realidad.

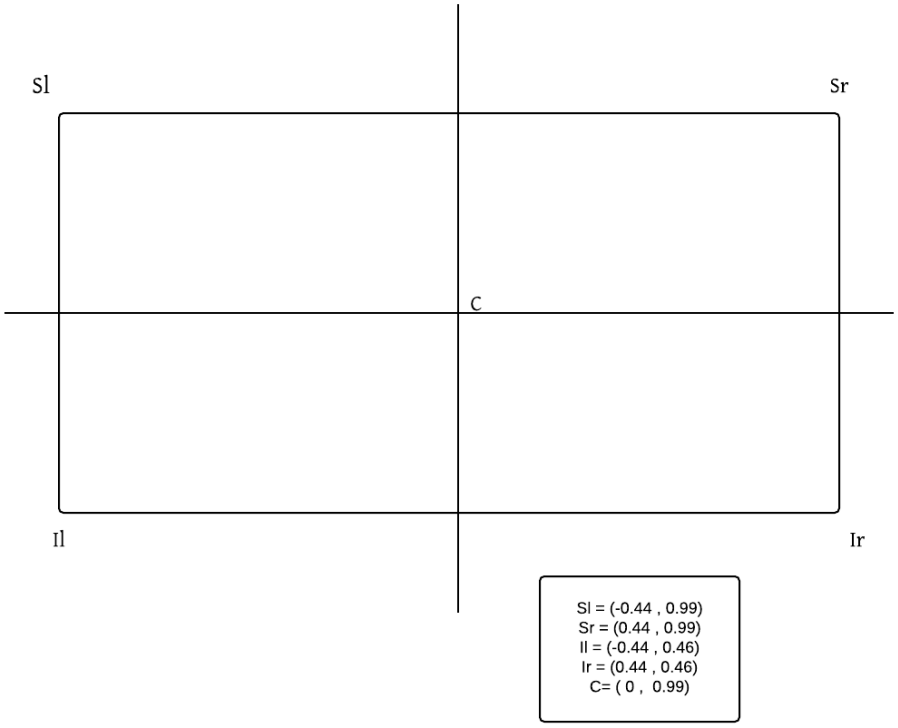


Ilustración 28: Medidas de los vértices de la imagen la realidad, respecto del sistema de coordenadas 2D coincidente con la superficie del mar, con el eje Y alineado con la línea de crujía del barco y cuyo origen se sitúa en la vertical del poste que sostiene a la cámara.

En el sistema definitivo de visión por computador se van a usar una resolución de imagen de 640 x 360 píxeles para acelerar el proceso de detección de obstáculos. Esta nueva resolución al ser 4 veces inferior nos permite mantener el tamaño de ventana que teníamos anteriormente de 40 x 40píxeles.

La salida del algoritmo de visión por computador es una imagen que tiene de ancho el número de ventanas en la horizontal en las que se subdividió la imagen de entrada y de alto el mismo número de ventanas en la vertical que las que se subdividió la imagen de entrada. Es decir, obtendremos una imagen de 16 x 9 píxeles.

Tener una imagen más pequeña no facilita el cálculo del punto más cercano al barco al tener un menor número de puntos.

Conociendo este punto más cercano al velero, tenemos sus índices en la horizontal y en la vertical. Esto junto con las coordenadas de los vértices nos permite dar una estimación de a qué distancia está el objeto siguiendo las Ilustración 29.

```
function [x,y] = getDistance(i,j)

x= i*((2*0.44)/16)-0.44;
y = 4.8384 - (j*((4.8384-0.4614)/9));

end
```

Ilustración 29: Función que devuelve la estimación de la distancia a la que está.

La nueva coordenada en x se calcula como el i-ésimo paso a partir de la posición -0.44, donde el paso se calcula como la división del intervalo en 16 partes iguales correspondiendo cada parte a un píxel en la horizontal de la imagen resultado.

De manera similar se calcula la coordenada en y, pero en este caso el paso se calcula restando al valor 4.8384 puesto que la coordenada 0 para la j está en la esquina superior izquierda y al ir aumentando los valores de j deben ir disminuyendo los valores de la y.

4 SISTEMA DE EVASIÓN DE OBSTÁCULOS

4.1 DESCRIPCIÓN DEL PROBLEMA

La evasión de obstáculos es el fin último de este trabajo. La presente aproximación es a nivel teórico y acota el problema de la siguiente forma:

- Se asume que la densidad de obstáculos es muy baja, de manera que sólo puede haber un obstáculo en cada imagen.
- Las direcciones en las que el barco puede evitar el obstáculo quedan determinadas por la dirección del viento.
- Los puntos marcados como destino están lo suficientemente lejos como para permitir que la evasión de inicie y se recalcule la ruta en vez de describir una trayectoria curva en un periodo breve de tiempo para esquivar el objeto.

4.2 ANÁLISIS DE LOS CASOS A AFRONTAR

Es necesario definir el concepto de ángulo crítico β_0, es aquel ángulo mínimo definido sobre la dirección del viento que permite que el velero pueda seguir navegando.

Con la anterior simplificación del problema nos podemos encontrar con los siguientes casos:

- El nuevo rumbo permite seguir navegando al no entrar dentro del ángulo crítico con el viento. $|\beta| > \beta_0$
- El nuevo rumbo entra dentro del ángulo crítico $|\beta| < \beta_0$, pero el obstáculo está en la misma banda(estribor o babor) desde la que sopla el viento.
- El nuevo rumbo entra dentro del ángulo crítico $|\beta| < \beta_0$, pero el obstáculo está en la banda contraria desde la que sopla el viento.
- El nuevo rumbo coincide con la dirección del viento. $|\beta| = 0$

4.3 ALGORITMO DE CONTROL

El algoritmo de control se describe a continuación:

```
función control_evasión_obtáculos {
        entrada real x,y ; // salida del sistema de visión por computador
        salida  betha;
        constante betha_critico;

        theta := estimación_nuevo_rumbo(x,y);
        awd := lee_sensor_direccion_viento_aparente();
        betha := theta – awd;

        abs_betha := abs(betha); //valor absoluto.
        si abs_betha > betha_critico
                modifica_rumbo(betha);
        si abs_betha > 0
                si están_en_la_misma_banda (x,y,awd)
                        betha += betha – betha_critico;
                sino
                        betha := obtener_cambio_brusco_trayectoria(x,y,awd);
                fin si
        si abs_betha == 0
                betha :=
                obtener_nueva_trayectora_banda_contraria_al_viento(x,y,awd);
        fin si
}
```

A continuación, se adjunta una descripción de apoyo al algoritmo para mejor entendimiento:

El sistema de control recibe la estimación de la posición del obstáculo en el caso en el que se detecte alguno. Con ella se estima un nuevo rumbo θ.
Se lee la dirección del viento aparente, AWD.
La diferencia entre ellos da un ángulo β.
Si el valor absoluto de β es mayor que el ángulo crítico del sistema β_0, la nueva dirección está bien calculada y se procede a modificar el rumbo.
En caso de que el valor absoluto sea menor o igual pero distinto de 0:
Si el obstáculo está en la misma banda que el viento:
Se incrementa el ángulo θ como mínimo en la diferencia entre $\beta - \beta_0$ y se procede a modificar el rumbo.
En caso contrario, es necesario cambiar bruscamente la trayectoria. Este cambio se representa como $\Delta\theta$ depende de lo próximo que esté el objeto.
En caso de que el valor absoluto sea 0 se cambia el rumbo a la banda contraria al viento.

El anterior algoritmo no se ha podido probar porque el velero está siendo modificado y no se ha podido llevar al mar para probarse.

5 CONCLUSIONES Y TRABAJO FUTURO

El desarrollo del Trabajo de Fin de Grado ha permitido tener una visión completa de cómo sería un verdadero proyecto de investigación: desde la idea inicial hasta el primer prototipo pasando por todas las fases de la implementación y testeo del desarrollo.

Durante el desarrollo se ha podido experimentar como se integran las distintas tecnologías estudiadas en las diferentes asignaturas del grado, como son física, redes, métodos numéricos, programación web, sistemas operativos y sistemas inteligentes que han dado como resultado:

- La configuración de la interfaz de red inalámbrica de la Raspberry Pi para su integración dentro de una red WiFi. Con esta estructura se pueden desarrollar aplicaciones en un sistema aparte, como es el caso del MacBook Pro, apoyándose en el compilador cruzado para posteriormente enviar el ejecutable a la Raspberry Pi.
- El desarrollo de una aplicación concurrente de acceso a la cámara que permite la captura y guardado de las imágenes a la vez que escucha por el puerto 3490 a la espera de que lleguen los parámetros de configuración requeridos por el usuario.
- El desarrollo de una página web para la visualización de las imágenes capturadas por la cámara y la modificación de los parámetros de forma cómoda para el usuario desde cualquier sistema y plataforma que esté dentro de la red.
- La construcción de una carcasa protectora para el sistema que ha probado ser eficaz a la hora de proteger el sistema del salitre.
- La implementación de un primer prototipo de sistema de visión por computador. Este prototipo tiene algunos defectos como la detección de partes del objeto, no de la totalidad del volumen junto con la mala discriminación que se puede producir en algunos casos y que es el motivo principal por el que se propone utilizar un sistema de aprendizaje del umbral. Pero también tiene sus puntos fuertes como la insensibilidad a los cambios de intensidades en la imagen causados por las ondas superficiales del mar.
- La propuesta teórica de un sistema de aprendizaje del umbral para el sistema de visión por computador. De haberse implementado y ajustado correctamente podría mejorar el sistema de visión de manera considerable.
- El algoritmo de control del velero para evitar obstáculos teniendo en consideración la dirección del viento con respecto del barco. Evidentemente hay más factores a tener en cuenta como la inclinación del barco debido a su posición en una superficie marítima con oleaje que en

esta primera versión no se han tenido en cuenta y que se deja para desarrollos posteriores.

Además de que ha apreciado la importancia de conocimiento de las especificaciones técnicas de la cámara y la Raspberry Pi así como el contexto físico en el que se sitúa el velero para el correcto desarrollo de los experimentos.

En cuanto a los resultados son satisfactorios para ser una primera versión del sistema puesto que se detecta bien los objetos en las imágenes de muestra, quedando su perfeccionamiento para desarrollos posteriores.

Como trabajo futuro queda la implementación del sistema de aprendizaje del umbral junto con la implementación del sistema de visión por computador una vez probado el nuevo sistema de visión por computador habría que implementarlo en C/C++ para que pueda utilizarse en la Raspberry Pi.

A partir de ahí, se podría elaborar un sistema basado en redes neuronales que permitiese al sistema de control aprender los ángulos que tiene que desviarse cuando detecta un obstáculo, apoyándose en una estructura de radiocontrol.

También se podría modificar la estructura para que la cámara en lugar de estar estática, una vez se detectase un objeto lo mantuviera en el campo de visión hasta haberlo sobrepasado para ello la cámara se rotaría con ayuda de un servo motor.

6 REFERENCIAS

1 Timothy S. Veenstra, James H. churnside, Airborne sensor for detectin large marine debris at sea. Marine Pollution Bulleting 65 (2012).Pie

2 Alexander Borhgraef, Olivier Barnich, Fabian Lapierre, Marc Van Droogenbroeck, Wilfried Phillips and Marc Acheroy, An Evaluation of Pixel-Based Methods for the Detection of Floating Objects on the Sea Surface.

3 Tarek El-Gaaly, christopher Tomaszewki, Abhinav Valada, Prasanna Valagapudi,Balajee Kannan and Paul Scerri, Visual Obstacle Avoidance for Autonomous Watercraft using Smartphones, Rutgers University & Carnegie Mellon University.

4: Raspberry Pi 2 https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

5: Librería OpenCV http://opencv.org/

6: Servidor Apache https://httpd.apache.org/

7: Librería OMXCam https://github.com/gagle/raspberrypi-omxcam

8: Compilador cruzado Linaro http://www.linaro.org/downloads/

9: ARM Architecture Reference Manual
https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&sqi=2&ved=0ahUKEwiljZeDiNzNAhVGvBoKHRoRCx4QFggcMAA&url=https%3A%2F%2Fwww.scss.tcd.ie%2F~waldroj%2F3d1%2Farm_arm.pdf&usg=AFQjCNGskgpM2kWGnaan

APÉNDICE A: MÓDULO DE LA CÁMARA

1. INTRODUCCIÓN

La cámara de la Raspberry Pi tiene un sensor de 5 megapíxeles y se conecta mediante cable a la interfaz CSI de la Raspberry Pi. La calidad del vídeo y de la imagen estática es mejor que si utiliza una cámara USB de precio similar.

La versión NoIR, utilizada en este trabajo no tiene el filtro infrarrojo por lo que se pueden recoger ondas con una longitud de onda próxima a la frecuencia de las ondas de infrarrojos. Esto es interesante para aplicaciones de seguridad. A cambio de esta característica los colores no son tan nítidos como en la versión con filtro infrarrojos. Tanto la versión con el filtro IR como sin él usan el mismo software.

2. PARÁMETROS TÉCNICOS

Parámetro	Valor
Tipo de sensor	OmniVision OV5647 Color CMOS QSXGA de 5Mpx
Tamaño del sensor	3.67 x 2.74 mm (1/4 pulgadas)
Número de píxeles	2592x1944
Tamaño del píxel	1.4 x 1.4 um
Lente	f=3.6 mm, f/2.9
Ángulo de visión	53.3 x 41.4 grados
Campo de visión	2.0 x 1.33 m a los 2 metros
Lente SLR Equivalente	35 mm
Foco fijo	De 1 metro al infinito
Video	1080p a 30fps con el códec H.264 (hasta 90fps con la resolución VGA)
Tamaño de la placa	25 x 24 mm (excluyendo el cable)

APÉNDICE B: CONFIGURACIÓN DE LA RASPBERRY PI.

La Raspberry Pi es el sistema base para todo el desarrollo anterior, para poder llevarlo a cabo hay que realizar la configuración previa que se recoge en éste apéndice saber:

- Conexión de la cámara y su configuración para que sea reconocida por el sistema operativo
- Habilitación de la conexión WiFi haciendo uso de la antena USB.
- Instalación de las librerías necesarias para el desarrollo del TFG: OpenCV y OMXCam, BGSLibrary

1. CONEXIÓN DE LA CÁMARA Y SU CONFIGURACIÓN PARA QUE SEA RECONOCIDA POR EL SISTEMA

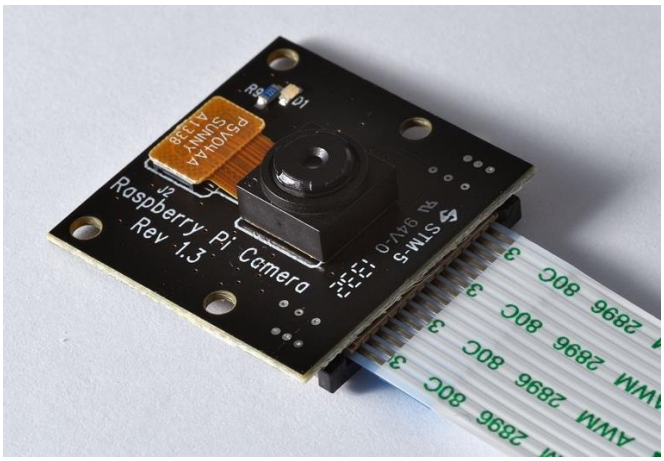


Ilustración 30: Cámara de la Raspberry Pi NoIR

La cámara de la Raspberry Pi es un módulo que se vende aparte de la propia Raspberry Pi y tiene dos variantes: con filtro de corte infrarrojo y sin él. En este caso contamos con la versión sin el filtro infrarrojos.

La conexión con la propia Raspberry Pi se realiza mediante el puerto CSI de ésta. La lectura desde este puerto está por defecto deshabilitada por lo que hay que ir al menú de configuración del sistema operativo mediante la orden de línea de comandos 'sudo raspi-config' que habilitará un entorno gráfico por el que podremos navegar con los cursores hasta la opción de la cámara y la habilitaremos. Tras ello reiniciamos el dispositivo.

Para comprobar que funciona correctamente ejecutamos la orden 'raspistill -v –o test.jpg' que mostrará en pantalla durante cinco segundos lo que está capturando tras los cuales tomará una foto y la guardará como 'test.jpg'.

2. HABILITACIÓN DE LA CONEXIÓN WIFI HACIENDO USO DE LA ANTENA USB.

Ilustración 31: Adaptador WiFi USB NetGear WNA3100M

Hasta el modelo 3, introducido en el mercado hace poco, la Raspberry Pi no contaba con la capacidad de conectarse a redes WiFi por sí misma por tanto se veía necesitada de dispositivos externos para llegar hasta ella o simplemente la tendríamos que conectar por la interfaz Ethernet. Por ello para nuestro proyecto tenemos la antena Wifi NetGear que conectaremos en uno de los 4 puertos USB disponibles.

La configuración de este nuevo dispositivo necesita de la configuración de dos servicios del sistema operativo *wpa_supplicant y dhcpcd.* El servicio *wpa_supplicant* controla el acceso a las redes WiFi seguras cifradas WPA, mientras que el servicio *dhcpcd* nos permite configurar las interfaces de red para que obtengan una configuración estática o dinámica.

Para configurar el servicio *wpa_supplicant* se configura mediante el fichero */etc/dhcp/dhcpcd.conf* donde configuraremos para que se conecte a la red del móvil de la siguiente manera:

```
country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        id_str="pr_laura"
        scan_ssid=1
        ssid="Lala"
        psk="[redacted]"
        priority=5
}
```

Ilustración 32: Configuración del servicio wpa_supplicant

Para crear una red se crea un bloque *network* con el identificador "*pr_laura"* que es una forma de identificar las redes que se configuran dentro de este fichero. El parámetro *scan_ssid=1* indica la técnica de escaneo de redes para que busque aquellas redes que se encubren a ellas mismas por no difundir su SSID. A continuación indicamos que el SSID que estamos buscando en los paquetes es "Lala", al cual accedemos con la contraseña definida en el psk. Lo último que se configura es la prioridad de la red, ante la posibilidad de conectarse a varias redes se selecciona aquella con menor valor en el parámetro *priority*.

Lo interesante para este trabajo es que la Raspberry Pi esté accesible en una dirección fija dentro de la red y puesto que el servidor del protocolo DHCP está en el

móvil protegido por varias capas de software del sistema operativo Android tendremos que configurar la dirección de la interfaz de red inalámbrica en la propia Raspberry Pi.

La configuración de la interfaz la realizaremos sobre el fichero */etc/dhcp/dhcpcd.conf* en el que añadiremos al final las siguientes líneas:

```
#Static confg

interface wlan0
static ip_address=192.168.43.2
static routers=192.168.43.1
static domain_name_servers=192.168.43.1
```

Ilustración 33: Configuración estática de la interfaz

En estas líneas se indica que para la interfaz *wlan0* la dirección, el punto de acceso por defecto y el servidor de nombres de dominio son estáticos y son los que se establecen.

3. INSTALACIÓN DE LAS LIBRERÍAS NECESARIAS PARA EL DESARROLLO DEL TFG: OPENCV Y OMXCAM

El sistema final será un sistema que necesita de una capacidad de reacción rápida, para minimizar los tiempos de ejecución se opta por utilizar los lenguajes C y C++ para el desarrollo del software involucrado ya que está a un nivel más cercano al nivel máquina.

En este nivel utilizaremos las librerías OMXCam para el acceso a la cámara y a sus parámetros de configuración, OpenCV y BGSlibrary para las operaciones de separación del fondo y el frente.

Nótese que la instalación de las librerías sólo se realiza en la Raspberry Pi; añadiéndose en los directorios de librerías correspondientes del sistema operativo, mientras en el Macbook Pro solamente se compilan usando el compilador cruzado en una carpeta destinada a almacenar las librerías compiladas de forma cruzada y se enlazan a la hora de crear los proyectos.

OMXCAM

La librería OMXCam sirve como capa de abstracción sobre las librerías OpenMax IL que controlan las operaciones de la cámara.

Funciona a través de la creación de un hilo secundario conectado a la cámara que realiza sucesivas llamadas a un método llamado *on_data* definido por el usuario pasándole un búffer con una sección de la imagen. Se espera que el método *on_data* del usuario almacene dicha información para transformarla finalmente en una imagen completa.

Esta librería es una librería compartida por lo que para utilizarla tras haberla compilado hay que trasladar una copia a la carpeta de librerías /usr/lib

OPENCV

OpenCV es la librería más conocida relacionada con los problemas de visión por computador. Está continuamente actualizándose, saliendo la versión 3 a comienzos de septiembre del año 2015, pero se prefirió utilizar una versión más estable como es la versión 2.4.11.

La instalación se realiza mediante el uso del programa CMake. Éste programa se usa de modo generalizado para la compilación cruzada pero también tiene la opción de utilizar el propio compilador del sistema para llevar a cabo la compilación.

Al abrir CMake lo primero que tenemos que determinar son las rutas donde se encuentran los ficheros CMake.list y la ruta en la que se quiere guardar el código de instalación producido por CMake (éste último suele ser una nueva carpeta llamada build dentro del directorio anterior). Una vez establecidas ambas rutas hay que realizar dos pasos dentro de CMake: Configurar y Generar.

Configurar permite elegir tanto el compilador que se va a utilizar como las opciones de compilación que más se ajustan a nuestro sistema como pueden ser la integración con otras librerías así como las rutas de instalación si se establece en el fichero CMake.lists que todo se puede modificar. Un ejemplo de los parámetros de configuración de esta fase se muestran en la Ilustración 34.

El paso de Generar genera archivos Make (también conocidos como *Makefiles*) para la instalación correcta según lo que hayamos establecido en los parámetros de la fase de configuración.

ANT_EXECUTABLE	ANT_EXECUTABLE-NOTFOUND
BUILD_DOCS	ON
BUILD_EXAMPLES	ON
BUILD_JASPER	ON
BUILD_JPEG	ON
BUILD_OPENEXR	ON
BUILD_PACKAGE	ON
BUILD_PERF_TESTS	ON
BUILD_PNG	ON
BUILD_SHARED_LIBS	ON
BUILD_TBB	OFF
BUILD_TESTS	ON
BUILD_TIFF	ON
BUILD_WITH_DEBUG_INFO	ON
BUILD_ZLIB	ON
BUILD_opencv_apps	ON
BUILD_opencv_calib3d	ON
BUILD_opencv_contrib	ON
BUILD_opencv_core	ON
BUILD_opencv_features2d	ON
BUILD_opencv_flann	ON
BUILD_opencv_gpu	ON
BUILD_opencv_highgui	ON
BUILD_opencv_imgproc	ON
BUILD_opencv_legacy	ON
BUILD_opencv_ml	ON
BUILD_opencv_nonfree	ON
BUILD_opencv_objdetect	ON
BUILD_opencv_ocl	ON
BUILD_opencv_photo	ON
BUILD_opencv_python	ON
BUILD_opencv_stitching	ON
BUILD_opencv_superres	ON
BUILD_opencv_ts	ON
BUILD_opencv_video	ON
BUILD_opencv_videostab	ON
BUILD_opencv_world	OFF
CLAMDBLAS_INCLUDE_DIR	CLAMDBLAS_INCLUDE_DIR-NOTFOUND
CLAMDBLAS_ROOT_DIR	CLAMDBLAS_ROOT_DIR-NOTFOUND
CLAMDFFT_INCLUDE_DIR	CLAMDFFT_INCLUDE_DIR-NOTFOUND
CLAMDFFT_ROOT_DIR	CLAMDFFT_ROOT_DIR-NOTFOUND
CMAKE_BUILD_TYPE	Release
CMAKE_CONFIGURATION_TYPES	Debug;Release
CMAKE_INSTALL_PREFIX	/usr/local

Ilustración 34: Algunos parámetros de configuración de la librería OpenCV.

Con todo lo anterior hecho solamente queda ejecutar los comandos *make; make install;* para que se comience la instalación de la librería.