Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Incorporación de un Módulo para el Manejo de Imágenes RGB a la Interfaz QOP

Autor: Gonzalo Fresno Schmolk

Tutores: Cristina Losada Gutiérrez y David Jiménez Cabello

2015

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Incorporación de un Módulo para el Manejo de Imágenes RGB a la Interfaz QOP

Autor: Gonzalo Fresno Schmolk

Directores: Cristina Losada Gutiérrez y David Jiménez Cabello

Tribunal:

Presidente: María Soledad Escudero Hernanz

Vocal 1º: Daniel Pizarro Pérez

Vocal 2º: Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

A mi abuela Ángeles cuyo consejo siempre escuché, a mis padres Susanne y Javier por el apoyo constante y a mi novia Susi por lo que es y será...

"Existe al menos un rincón del universo que con toda seguridad puedes mejorar, y eres tú mismo."
Aldous Huxley

Agradecimientos

Este trabajo de final de grado es el resultado de mucha dedicación y esfuerzo, para mi representa el cierre de una gratificante etapa como estudiante, y digo esto porque en primer lugar quiero dar las gracias a todo el equipo humano que componen la facultad, hacen más fácil el día a día de todos los estudiantes.

Este esfuerzo no hubiera sido posible sin la ayuda y paciencia de mis tutores Cristina y David, que me han guiado en este trabajo y que siempre han estado disponibles para ayudarme ante mis infinitas dudas.

Mención especial merece mi compañero y amigo Rubén Izquierdo, genio entre los genios, siempre dispuesto a ayudar a los amigos. También quiero agradecerle a mi amigo Pablo Tofé la constancia y capacidad de trabajo que me contagia y el tiempo pasado en todos estos años.

Agradecimientos para otros compañeros docentes como Lourdes y Francisco con los que guardo una excelente relación, y al equipo de cafetería que tantos desayunos me han dado en estos años.

Finalmente, no puedo acabar con esta sección sin agradecer este trabajo a las personas a las que se lo dedico mi familia, novia y amigos, sin ellos no sería lo que soy.

Resumen

El objetivo del presente trabajo consiste en ampliar la funcionalidad de una interfaz de captura de información proveniente de cámaras de tiempo de vuelo, para dotarle así de la capacidad de manejar cámaras *RGB* de disponibilidad comercial.

Se ha llevado a cabo el desarrollo e integración de un software que permite incorporar nuevas funcionalidades a *QOP*, empleando una cámara *RGB*. El módulo desarrollado permite configurar la cámara *RGB*, leer párametros de su configuración actual, capturar y visualizar imágenes a través de la interfaz. Para mantener el programa en el que se creó la interfaz, se usará el entorno de desarrollo de aplicaciones *Qt* y el lenguaje de programación orientado a objetos basado en *C++*. De esta manera se garantiza plena compatibilidad con el resto de la aplicación.

La aplicación *QOP* diseñada como proyecto final de carrera en 2014 es el punto de partida del presente TFG. Dicha interfaz permite la adquisición y tratado de imágenes proporcionadas por dos modelos de cámaras Time of Flight.

Palabras clave: QOP, C++, RGB, API, Qt

Abstract

The objective of the present degree project is to develop and integrate new software in order to increase the possibilities of an interface that can capture images from Time of Flight cameras. This allows to manage commercial *RGB* cameras in order to increase the kinds of cameras supported.

With this new module the functionality of the interface has been increased. The user will be able to configure or read internal *RGB* camera parameters through the *API*. Also the user can capture and visualize images through the interface. To keep the development program where *QOP* was created, the development environment that will be used is *Qt* and the programming language will be *C++* in order to ensure full compatibility with the application.

The application named *QOP* designed and developed in 2014 like dissertation is the start point of this TFG. The interface allows acquisition and post-acquisition image treatment using two Time of Flight camera models.

Keywords: QOP, C++, RGB, API, Qt

Resumen extendido

El objetivo del presente trabajo es el desarrollo, integración y evaluación de un software modular que permite otorgar mayor polivalencia y funcionalidad a la interfaz de manejo de cámaras de tiempo de vuelo *QOP* [1], añadiendo la capacidad de controlar y configurar una cámara *RGB* de escaneo de área proporcionada por el fabricante Basler Scout [2].

El punto de partida de este trabajo de final de grado es la interfaz *QOP* diseñada como trabajo final de carrera en 2014 para la configuración, adquisición y tratamiento de imágenes proporcionadas por dos modelos de cámaras de tiempo de vuelo (*ToF* de sus siglas en inglés, Time of Flight). Las cámaras *ToF* emiten una señal modulada infrarroja que es reflejada por los objetos presentes en su campo de visión y captada por los sensores de las mismas. El valor de salida es la distancia a los objetos captados.

El principio de funcionamiento de las cámaras *RGB* es análogo al ojo humano. La luz atraviesa una lente que la dirige a un filtro encargado de separarla en los colores primarios, los cuales son proyectados sobre un *sensor fotosensible*. Estos sensores están formados en su esencia por materiales semiconductores de metal-óxido *(MOS)* que están distribuidos en forma de matriz. El funcionamiento físico está sustentado por el *efecto fotoeléctrico*, por ello el sensor genera una señal eléctrica en función de la intensidad de la señal que incide sobre él. Esta señal será posteriormente convertida a valores discretos de intensidad de cada uno de los píxeles y que conforman la imagen que se puede visualizar.

En el presente trabajo se añade una nueva funcionalidad a la interfaz *QOP* para el manejo de cámaras *RGB*. Esta solución de carácter modular permite mejorar y dotar de mayor visión de futuro al sistema mediante la integración de la cámara Basler Scout. Con este nuevo módulo es posible acceder y configurar los parámetros internos de la cámara a través de la *API*. También es posible adquirir y visualizar imágenes con los parámetros deseados. Esta implementanción no es invasiva respecto a la aplicación original, ya que se ha creado cada archivo necesario para evitar conflictos con el software ya existente. Se ha realizado la implementación de nuevas clases y funciones en lenguaje *C++*, así como la inclusión de nuevas ventanas para la configuración y manejo de la cámara *RGB* incorporada. De esta manera se asegura la plena funcionalidad de las dos cámaras *ToF*.

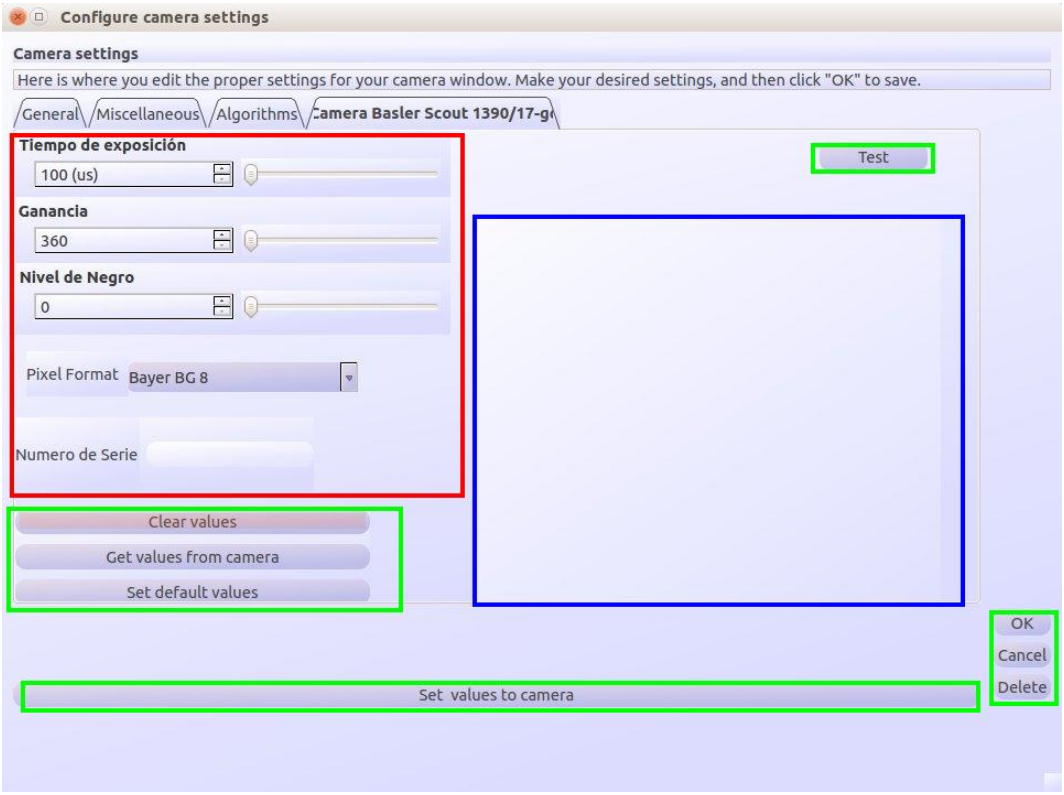

Figura 2: Presentación módulo RGB de la interfaz QOP

La Figura 2 muestra la solución modular propuesta. Se ha recuadrado en diferentes colores cada una de las partes que conforman este módulo. El color `rojo` marca los parámetros configurables de la interfaz. El color `verde` los botones disponibles y el color `azul` se identifica con el visor de imágenes.

Este módulo integra la posibilidad de modificar y obtener de la cámara los parámetros de configuración más relevantes como son:

- Tiempo de exposición: es posible leer el valor actual de trabajo o escribir de forma numérica el valor entero deseado entre un mínimo 100 μs de y un máximo de 4095 μs.
- Ganancia: permite obtener el valor de ganancia de la cámara o configurar la cámara con el valor entero deseado entre 360 y 1023.
- Nivel de negro: el valor leido o escrito está contenido entre 0 y 255.
- Formato de Píxel: el formato de captura de imagen es seleccionable mediante un desplegable que brinda los dos formatos necesarios: Mono8 y BayerBG8. El primero permite la captura en escala de grises y con el segundo se pueden capturar imágenes en color.
- Número de serie: en esta casilla se puede leer el número de serie de la cámara empleada cuando se pulsa el botón Get Default Values.

Encontramos 8 botones en esta ventana remarcados en color verde cuya funcionalidad se detalla a continuación:

- Botón Test: cuando se ejecuta este botón se llama a la función asociada que se encarga de crear el objeto de cámara, de transmitir la información, y de usar las librerías de *OpenCV* para adaptar la imagen a un formato compatible con Qt y así poder visualizar la imagen en el visor marcado en color azul en la Figura 2.

- Botón Clear Values: se encarga de poner al mínimo todos los valores disponibles, tiempo de exposición, ganancia, nivel de negro y formato de píxel, así como de borrar el número de serie.
- Botón Get Values From Camera: permite obtener del dispositivo los valores actuales de los parámetros de interés de la cámara.
- Botón Set Values to camera: se encarga de configurar completamente la cámara. Su función es mandar al objeto de cámara los valores ajustados en la configuración.
- Botones OK, Cancel y Delete: estos botones genéricos a todos los modelos de cámara ya estaban implementados en *QOP*. El botón OK permite crear la nueva configuración de cámara. Si se pulsa el botón Cancel se desecha esta configuración. El botón Delete permite borrar la configuración de cámara creada.

A continuación se enumeran las clases credas que han sido necesarias para llevar a cabo la implementación desarrollada, en esta enumeración se incluyen únicamente las clases creadas:

- *Scout*
- *QScout*
- *QThreadCam_Scout*
- *CamWindow_Scout*
- *CamWindowConfiguration_Scout*

La Figura 3 muestra la dependencia de clases exitentes en el proyecto.

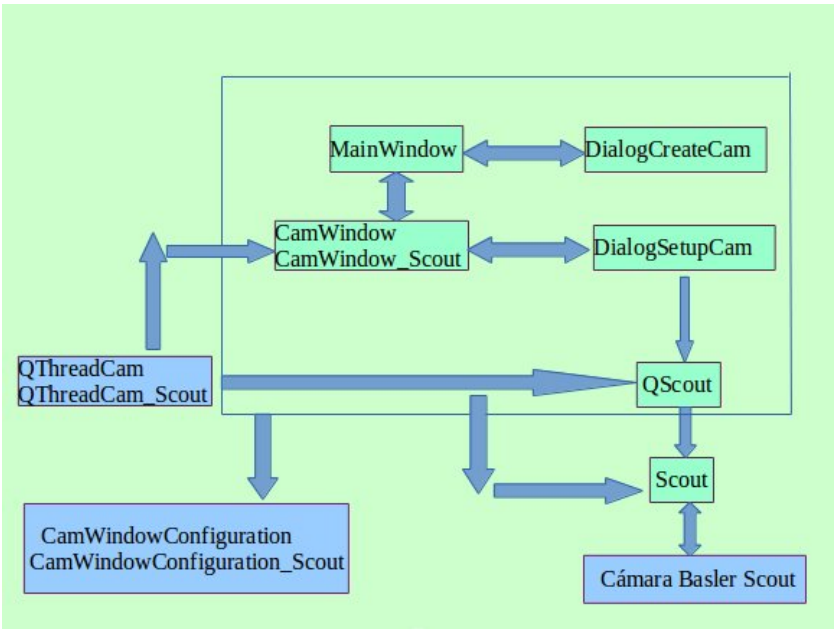

Figura 3: Jerarquía de clases

Se pretende que el usuario final sea capaz en poco tiempo de conectar la cámara, ejecutar la aplicación y empezar a capturar imágenes. Sus prestaciones hacen que la solución propuesta sea válida para múltiples propósitos o ámbitos. Sobre este desarrollo pueden aplicarse algoritmos de reconocimiento o de procesado digital para la función requerida, ya que está diseñada y programada para ser fácilmente extendida.

Para plasmar la utilidad y versatilidad práctica de la solución desarollada se ha llevado a cabo una evaluación que abarca todas las posibilidades de configuración en la parte final de la memoria. Así se verifica el funcionamiento del módulo desarrollado, como su correcta inserción y adaptación en la aplicación sobre la que se ha implementado.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Memoria

Es mejor saber después de haber pensado y discutido
que aceptar los saberes que nadie discute para no tener
que pensar.

Fernando Savater

1.1 Introducción

Este trabajo de final de Grado que se enmarca dentro del área de visión computacional se centra en la incorporación de un módulo para la configuración, adquisición y manejo de imágenes *RGB*. Este módulo se incorpora a la interfaz *QOP* [1] para la adquisición, calibración y procesado de imágenes provenientes de cámaras de tiempo de vuelo. Se ha ampliado la funcionalidad de la interfaz con esta nueva incorporación puesto que se han incrementado los diferentes tipos de imagen que pueden ser tratados por la interfaz, dotándola de mayor polivalencia.

1.1.1 Objetivos generales del trabajo

El objetivo fundamental de este proyecto es aumentar la polivalencia de la interfaz *QOP*, mediante el diseño, implementación y evaluación de un nuevo módulo asociado al análisis e interpretación de las imágenes capturadas por la cámara *RGB*, por lo que se podrá gestionar imágenes tanto de las cámaras de tiempo de vuelo *ToF* como de la cámara *RGB*. El manejo de esta interfaz permitirá al usuario final con menor conocimiento técnico, la manipulación y gestión de ambos tipos de imagen.

Los objetivos específicos de este proyecto son los siguientes:

- La adquisición de conocimiento y búsqueda de información sobre la interfaz ya disponible. Para ello se llevará a cabo un análisis y estudio del estado del arte actual. Esta información permite conocer y entender la aplicación para el desarrollo del módulo *RGB*.
- Aprendizaje de la programación de interfaces gráficas utilizando *Qt* y *C++*.
- Análisis de la información proveniente de una cámara *RGB* de disponibilidad comercial.
- Diseño, implementación y evaluación un nuevo módulo de adquisición y tratamiento de imágenes *RGB* que se integra en la interfaz existente y que tiene las siguientes características:

1. Incorporación de las librerías de procesamiento y manipulación proporcionadas por el fabricante Basler *Pylon*.
2. Adecuación del código para continuar con el paradigma de *QOP*.
3. Gestión controlada de los datos y valores obtenidos.
4. Facilidad de uso para un usuario final con menor conocimiento técnico.

- Evaluación de la funcionalidad del nuevo módulo asociado.
- Documentación de todo el proceso llevado a cabo para una mayor facilidad de comprensión de otros usuarios o desarrolladores.

1.1.2 Descripción de las tareas desarrolladas

Para alcanzar los objetivos del proyecto he llevado a cabo las siguientes tareas:

1. Estudio de los requerimientos funcionales del software de partida.
2. Análisis de las especificaciones y lectura de la documentación técnica proporionada por el fabricante de la cámara, siendo muy necesario la lectura del manual de usuario y el manual del programador.
3. Aprendizaje del lenguaje de programación *C++* y del entorno de desarrollo de aplicaciones *Qt*.
4. Interpretación y análisis de la estructuración, información y datos proporcionados por la cámara. También se probará la interfaz software que proporciona el fabricante para conectividad del dispositivo, adquisición y tratado.
5. Programación e implementación de un software de captura, tratado, y configuración para este dispositivo, y así dotar de un nuevo módulo funcional a la interfaz.
6. Revisiones y pruebas continuas del funcionamiento y la calidad del código fuente.
7. Preparación y elaboración de la documentación necesaria para la presentación de este proyecto y para los futuros usuarios de la aplicación logren una rápida adaptación.

1.1.3 Interés y alcance del trabajo

Actualmente no se dispone de ninguna aplicación que integre de manera conjunta la adquisición y tratado de imágenes digitales provenientes de cámaras *ToF*, como de imágenes obtenidas de cámaras *RGB* en un sistema operativo abierto como *Linux*.

Las diferencias entre fabricantes y modelos son notables por ello es necesario aprender y manejar el lenguaje de programación propio de cada cámara proporcionado por el fabricante. Esto es una condición necesaria para integrar la cámara en la aplicación proporcionada conocida como *QOP*.

La ampliación de polivalencia de esta interfaz permite a futuros usuarios tener una guía más amplia de los distintos modelos y de los parámetros proporcionados por las cámaras para facilitar posteriores integraciones de otros dispositivos en la aplicación *QOP*. Esta ampliación deberá ser modular y extensible para seguir en sintonía de la aplicación original.

No obstante, para el usuario final es necesaria una mínima formación técnica básica sobre las características de estas cámaras y la información obtenida de las mismas. Para el usuario avanzado que desee desarrollar y ampliar la interfaz estos conocimientos deben ser superiores.

1.2 Estudio Teórico

En este apartado se documentará el fundamento teórico que sustenta la adquisición y procesado digital de imágenes.

1.2.1 Punto de partida

El punto de inicio de este trabajo de final de grado es la interfaz *QOP* [1] para la adquisición y tratamiento de imágenes proporcionadas por cámaras de tiempo de vuelo. Existen algunas librerías comerciales para cámaras *ToF* como Metrilus MetriCam, MVTec HALCON y otras de código abierto como OpenNI o ROS. La necesidad de esta aplicación surge debido a que no hay interfaces ni aplicaciones que permitan realizar esta tarea para diferentes modelos de cámaras *ToF* y *RGB*.

Esta es una aplicación modular y extensible que permite su desarrollo sin excluir ningún tipo de cámara para aumentar la funcionalidad de la misma. El nuevo módulo *RGB* se desarrolla usando las funciones propias de la *API* de la cámara, así como funciones de visión computacional (*OpenCV*) ampliamente utilizadas por la comunidad científica.

Estas librerías y funciones requieren de conocimientos técnicos de programación para poder ser implementadas. La motivación en el comienzo de este trabajo fue iniciarse en la programación de un nuevo lenguaje y entorno de desarrollo, para así adquirir los conocimientos técnicos necesarios y poder brindar al usuario final con menor conocimiento técnico una aplicación intuitiva y amigable.

Por ello, en este trabajo se incorporarán los módulos necesarios a la aplicación para que integre la adquisición y configuración, así como el tratamiento de imágenes digitales provenientes de cámaras de *ToF*, como imágenes provenientes de cámaras *RGB* en un entorno de código abierto (open source) como es un sistema operativo Linux.

1.2.2 Contexto y principio de funcionamiento de una cámara RGB

En este apartado se pretende explicar el funcionamiento técnico de las cámaras digitales, asi como contextualizar historicamente la aparición de estos dispositivos.

El desarrollo de esta era digital esta ligado al descubrimiento de los sensores *CCD* por `Fairchild Semiconductor` en el año 1973. Dos años mas tarde `Kodak` incorporó estos sensores a una cámara, obteniendo una foto con una resolucion de 0.01Mpix en blanco y negro, y con un peso de 4Kg.

En el año 1988 la compañía japonesa `Fuji` desarrolló la primera cámara completamente digital bajo el nombre *DS-1P* que almacenaba los datos capturados en una memoria interna de 16MB y que necesitaba de alimentación para mantener la información.

Figura 1.1: FujiDS-1p

Posteriormente en el año 1991 Dycam comercializó el primer modelo en emplear un sensor *CCD* Charge-Coupled Device, también llamado dispositivo de carga acoplada.

Se puede decir que la aparación de los sensores digitales y su posible integración en las cámaras abrió las puertas al mundo de la fotografía y video digital.

A continuación se explica el principio de funcionamiento de la cámara digital empleada en el presente proyecto y de sus distintas partes. El funcionamiento de la cámara *RGB* proporcionada por el fabricante alemán Basler es como el de una cámara digital estándar.

Para comprender este proceso es necesario tener el conocimiento de como funciona el ojo humano, pues su funcionamiento se usó de base para desarrollar las cámaras fotograficas. En la Figura 1.2 se detalla el flujo de la luz en ambos sistemas.

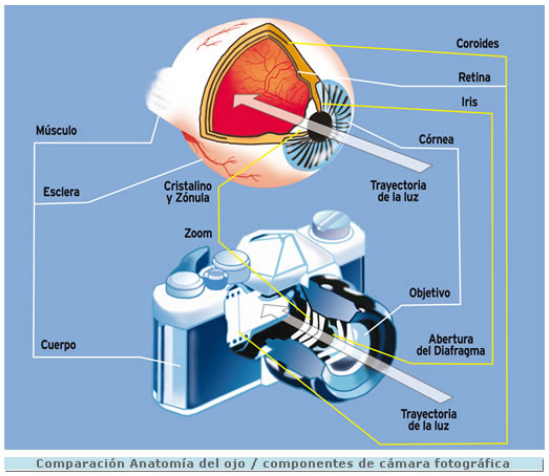

Figura 1.2: Flujo de luz a través de ojo humano y cámara digital

De la figura se extrae que la luz atraviesa una lente que la dirige a un filtro encargado de separarla en los colores primarios, los cuales son proyectados sobre un sensor fotosensible.

La semejanza con el ojo reside en que esa lente se asemeja a la córnea y el sensor que capta la intensidad de luz es la retina.

El nervio óptico actua como el cable transmisor de información y el iris como diafragma de apertura regulando la cantidad de luz que se deja pasar al interior del dispositivo, en nuestro caso al sensor. A continuación se enumera y explica el funcionamiento del elemento principal de una camara digital, el sensor fotosensible.

Los 2 tipos de sensores mas distribuidos y conocidos son el *CCD* Charge-Coupled Device y el *CMOS* Complementary Metal Oxide Semiconductor. Existen más tipos como el Super CCD y el Foveon X3. En este apartado se explicarán los dos modelos más comunes.

El funcionamiento básico de ambos es bastante similar, están formados en su esencia por semiconductores de metal-óxido (*MOS*) que están distribuidos en forma de matriz.

El funcionamiento físico está sustentado por el efecto fotoeléctrico, que consiste en la conversión espontánea de luz recibida en corriente eléctrica que ocurre en algunos materiales, en definitiva, el sensor genera una señal eléctrica en función de la intensidad de la señal que incide sobre él.

Su función es la de acumular una carga eléctrica en cada una de las celdas de esta matriz. Estas celdas son los llamados píxeles. El número de píxeles determina la resolución del sensor. La carga eléctrica almacenada en cada píxel, dependerá en todo momento de la cantidad de luz que incida sobre el mismo. Cuanta más luz incida sobre el píxel, mayor será la carga que este adquiera.

La sensibilidad del sensor depende de la `eficiencia cuántica` del chip, es decir, de la cantidad de fotones que deben incidir sobre cada detector para producir una corriente eléctrica.

Indenpendientemente del sensor empleado, el incoveniente del sensor es que no distingue los colores, sino variaciones de intensidad, por tanto, para obtener una imagen en color es necesario descomponer la imagen en los colores primarios (rojo, verde y azul). Para ello empleamos los filtros de color que proyectan los colores rojo, verde y azul *RGB* sobre distintas zonas del sensor, el cual reconoce la cantidad de intensidad de cada uno de ellos por separado, generando la señal eléctrica asociada.

Toda esta información es común a ambos tipos de sensores, la diferencia reside en la forma de transferir la carga eléctrica generada en los píxeles. En la Figura 1.3 se ilustra la diferencia entre los sensores *CCD* y *CMOS*.

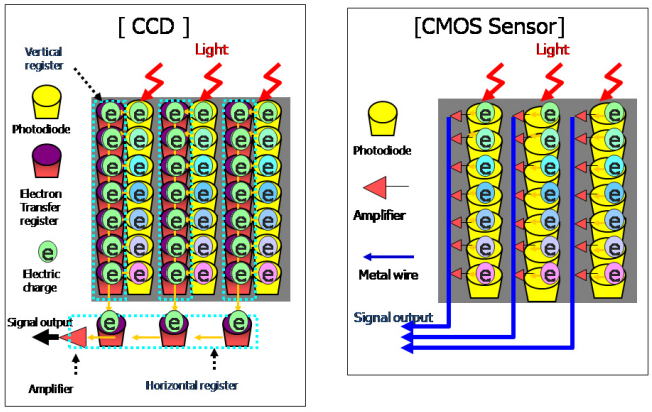

Figura 1.3: Diferencia de transmisión de la carga eléctrica entre sensor CCD y CMOS

- ***CCD***: en este tipo de sensores la carga eléctrica de cada píxel es transferida a través de un número limitado de nodos de salida (generalmente 1) siendo posteriormente convertida a voltaje para entregar a la salida una señal analógica. Esta señal es enviada a un *ADC* (Convertidor analógico-digital) que discretiza los valores obtenidos. La cámara `Basler Scout` empleada en este proyecto posee un sensor de este tipo.

  - Ventajas:

    No hay una determinación específica acerca de cuando se emplea un tipo de sensor u otro. Cada uno tiene ventajas e inconvenientes. Esta elección depende de muchas circunstancias siendo la más importante la aplicación a desarrollar, o el uso que se pretenda dar.

    Los sensores *CCD* presentan un mejor `rango dinámico` que es el coeficiente entre la saturación de los píxeles y el umbral por debajo del cual no captan señal, tolerando mejor los extremos de intensidad lumínica. Dynamic Range (dB) = 20 x Log(Nsat/Nnoise)

    En cuanto al ruido, los sensores *CCD* presentan menor distorsión por ruido debido a que el procesado de la señal se realiza en un *ADC* externo. Por ello puede reducirse su magnitud actuando sobre este chip y/o con algunos configuraciones de diseño.

    En estos sensores al ser toda la matriz de píxeles uniforme, se tiene un mejor comportamiento en cuanto a uniformidad de respuesta se refiere, debido a que todos los pixeles están sometidos al mismo nivel de excitación. Una respuesta uniforme implica un menor ruido.

    Para finalizar éstos son un producto más maduro, y tienden a tener una mayor calidad y más pixeles.

- ***CMOS***: en un sensor de este tipo cada píxel es independiente del resto y cada uno de los píxeles dispone de un propio convertidor carga-voltaje y *ADC* correspondiente, por lo que la digitalización de la intensidad de luz recibida se hace de manera independiente.

  Todo el proceso se lleva a cabo dentro del sensor y no se hace necesario disponer de un *ADC* externo encargado de esta función. Los sensores *CMOS* aparecieron después de los *CCD*, y también presentan una serie de ventajas a tener en cuenta.

  – Ventajas:

    El procesado se realiza dentro del sensor no siendo necesario ningun chip externo que realice esta función, por ello se consigue un mayor nivel de integración pudiendo fabricar dispositivos más compactos y pequeños. Así mismo son más baratos.

    Consumen mucha menos energia a igualdad de alimentación, siendo más interesantes para la aplicación a dispositivos portátiles y haciendo estos sensores muy atractivos.

    Se define la `responsividad` como el nivel de señal que es capaz de ofrecer el sensor por cada unidad de energía óptica incidente. Dado que interesa que el sensor tenga una responsividad elevada, se necesita que con poca cantidad de luz proporcione una señal aceptable. Con esto en mente, se puede ver que los sensores *CMOS* son superiores a los de tecnología *CCD*. Por tanto en condiciones pobres de iluminación se comportan mucho mejor.

    Otra de las grandes ventajas es su velocidad, son mucho más rapidos que los *CCD* debido a la ausencia del *ADC* externo. Evitamos tener que esperar los tiempos propios de adquisición, procesado y conversión de un *ADC*.

    Por último otra virtud de estos sensores es que presentan un mejor comportamiento frente al efecto conocido como `blooming` (casi inexistente). Éste fenómeno se produce cuando un píxel se satura por la luz que incide sobre él y a continuación empieza a saturar a los píxeles que están a su alrededor.

1.2.3 Calidad de la medición

En las cámaras digitales existen muchos parámetros que afectan a la calidad de la medición. Estas variables que influyen a la hora de adquirir una imagen con un nivel de calidad bueno, pueden ser intrínsecas del dispositivo usado o extrínsecas y por tanto dependendientes del medio donde se esta trabajando.

- ***Parámetros internos***

  Los factores internos que determinan la calidad de la medición están normalmente determinados por el modelo de lente y sensor utilizado. Éste último determina la sensibilidad disponible a la hora de capturar energía lumínica, en función de su efecto cuántico. Cuando se aumenta la sensibilidad se pierde definición y aumenta el nivel de ruido, se tiene menor nitidez y por lo tanto menor calidad.

  Así mismo su geometría, dimensiones y distribución de píxeles también importan, pues habrá más píxeles en un sensor de mayor tamaño y por ende mayor resolución. Otra característica a mencionar es la densidad de píxeles. El aumento de la densidad de pixeles disminuye la sensibilidad del sensor. Pues cada pixel es tan pequeño que recoge muy pocos fotones, y así para conservar la relación señal-ruido se debe iluminar más el sensor. Esta disminución de la sensibilidad conduce a cuadros ruidosos, calidad pobre en sombras y generalmente a imágenes de pobre calidad si están escasamente iluminadas.

  Se puede definir la resolución como la capacidad de reproducir fielmente los detalles de una imagen. Utilizaremos esta palabra, lógicamente, al referirnos a la resolución de una imagen digital, pero también, al referirnos a una cámara, un monitor o un escáner. Así pues, se trata de un concepto fundamental, que posee distintas acepciones, según el contexto en el que se utilice.

Cuanto mayor es la resolución de una cámara más calidad tiene su visualización, con transiciones de color más sutiles. Además la resolución está determinada por la transformación opto electrónica en la cámara, que es un proceso analógico con filtros y limitaciones. Es recomendable que la resolución de la cámara sea el doble que la resolución óptica del objetivo. En el contexto de las cámaras digitales se habla de *MegaPíxeles*. La resolución de una imagen digital está limitada por la cantidad de píxeles que la componen. Por ejemplo una imagen de 640 píxeles de ancho por 480 píxeles de alto tendrá 307,200 píxels. La densidad de pixeles de suele medirse en píxeles por pulgada (ppi) o píxeles por centímetro (pcm).

A través de la *API* (Apliccattion Programmable Interface) o de algunas herramientas proporcionadas por el fabricante es posible leer o modificar algunos de los parámetros más importantes en la adquisición como es la ganancia, el nivel de negro, el balance de blancos, la corrección gamma o el tiempo de exposición siendo factores clave en la calidad de la imagen tomada.

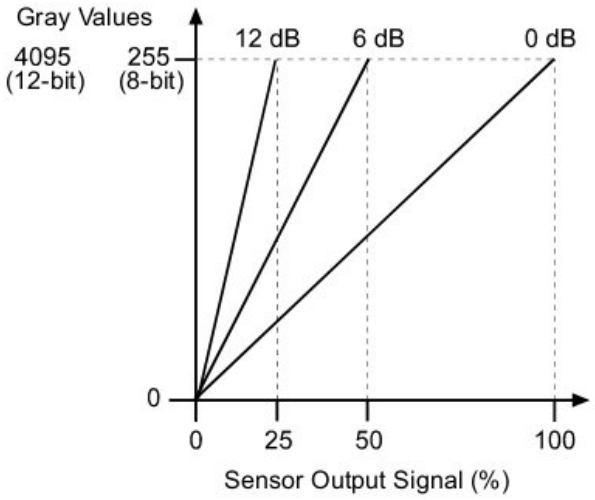

Figura 1.4: Ganancia en dB

La ganancia `Gain` está definida por el parámetro de ganancia en bruto `Raw Gain`. Como se observa en la Figura 1.4 cuando se incrementa la ganancia se incrementa la pendiente de la curva de respuesta de la cámara. Esto implica un valor de gris más elevado a a salida de la cámara, para una una determinada cantidad de carga a la salidad del sensor. Si se disminuye la ganancia se consigue el efecto inverso. Modificando la ganancia se está modificando la sensibilidad del sensor. En el presente trabajo, la cámara empleada permite un autoajuste de la ganancia para obtener resultados óptimos. El rango de ganancia permitida en nuestro modelo (para 8bits) oscila entre 360 y 1023.

Las ecuaciones que se exponen a continuación permiten calcular la ganancia en dB en función del margen de valores del parámetro GainRaw.

Si el valor está comprendido entre 110 y 511:

$$Gain_{dB} = 20 * log_{10}((658 + GainRaw)/(658 - GainRaw)) - Gc \tag{1.1}$$

Si el valor está comprendido entre 512 y 1023:

$$Gain_{dB} = (0{,}0354 * GainRaw) - Gc \tag{1.2}$$

Ecuación para calcular el valor de la constante Gc :

$$Gain_{dB} = 20 * log_{10}((658 + GainRawMin)/(658 - GainRawMin)) \tag{1.3}$$

Incrementando el `nivel de negro` de la cámara se consigue obtener un offset positivo en los valores digitales de salida de los píxeles. Si se decrementa este valor resulta un offset negativo a la salida. Este valor de negro puede ajustarse en un rango de 0 a 255 en el modelo de cámara usado.

Respecto al `balance de blancos`, este parámetro permite ajustar el balance de la intensidad de color rojo, verde y azul. Posibilita obtener una imagen con el color deseado.

La `corrección gamma` permite modificar el valor de salida de brillo de los píxeles a través del sensor para corregir la no linealidad en la percepción humana de brillo. El valor máximo de brillo de los píxeles (para 8bit) es de 255. La corrección gamma puede ajustarse en un valor de 0 a 3,999. Un valor de este factor de corrección entre 0 y 1 incrementará el brillo, si es igual a 1 el brillo no será alterado, y por encima de este valor el brillo se ve atenuado .

El `tiempo de exposición` es un factor clave en la fotografía. Es el tiempo en el que el sensor capta la luz que incide sobre él. Se profundiza en la sección 1.3.2.

La `velocidad de disparo` está estrechamente relacionada con el tiempo de exposición, pues es su función inversa. Si se aumenta el tiempo de exposición disminuye la velocidad de disparo y vicerversa. Permite manipular los efectos visuales de la imagen final más allá de su luminosidad.

Los tiempos de exposición de una cámara fotográfica pueden ajustarse en valores discretos, para este caso el fabricante informa de un tiempo mínimo de exposición $34\mu s$ y un máximo $10000000\mu s$.

A nivel práctico un tiempo de exposición corto deja pasar menos luz al sensor fotosensible y por ello se consigue congelar o reducir notablemente el movimiento, mientras que un tiempo de exposición largos permite obtener imágenes movidas o desplazadas, otorgando mayor sensación de desplazamiento.

Así mismo, también existen parámetros propios de las cámaras que trabajan en red, que no son relevantes en este apartado, aunque si necesarios para una correcta obtención de imágenes, y se detallan en la sección 1.3.1. Algunos de estos pueden ser el tamaño de los paquetes, o el tiempo que transcurre entre estos envíos de paquetes al sistema de proceso de datos.

Otro parámetro a tener en cuenta para cualificar una imagen es la `profundidad del color`. Este parámetro se refiere al número de bits utilizados para describir el tono de cada píxel. Determina el número de colores. Por ejemplo una profundidad de 1 bit implica 2 colores (blanco y negro).

La `saturación` de una imagen es la propiedad que describe la viveza del color. Un color muy saturado es un color con una tonalidad intensa y pura. Por el contrario, un color poco saturado es el que tiene una tonalidad apagada.

La saturación de un color se expresa en porcentaje y oscila entre el 100, que corresponde a los colores puros, saturados al máximo y el 0, que corresponde a los colores apagados en los que ya no se distingue la tonalidad.

La Figura 1.5 corresponde a la rueda de color HSB, la saturación se representa a lo largo del radio de la circunferencia. Los colores muy saturados se encuentran cerca del borde y los colores poco saturados son los que están cerca del centro del círculo.

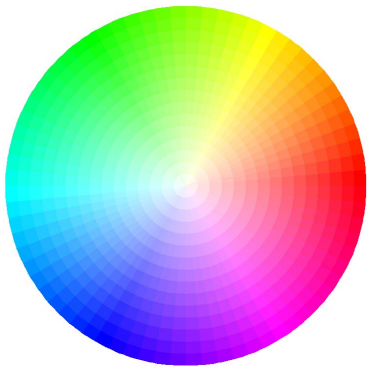

Figura 1.5: Rueda de color HSB donde la mayor saturación se da en el perímetro

- ***Parámetros externos***

  Después del largo listado de factores internos que influyen la calidad de imagen, existen factores ambientales o posteriores a la captura de imagen que pueden afectar a la calidad de ésta.

  La calidad obtenida está ligada al mantenimiento de la cámara en una zona de trabajo adecuada, bien iluminada y considerando factores ambientales como humedad y temperatura para los cuales los fabricantes recomiendan unos valores adecuados.

  En el presente modelo proporcionado por el fabricante `Basler` éste recomienda una humedad relativa contenida entre el 20 y 80 por ciento para almacenaje y uso de la cámara (sin condensación). En cuanto a la temperatura, la cámara debe estar almacenada entre -20°C y +80°C y debe operar a una temperatura de funcionamiento comprendida entre 0°C y +50°C, lo que permite múltiples ambientes de trabajo.

  La calidad de la imagen puede verse afectada por las *EMI* (interferencia electromagnética) generadas por otros dispositivos que estén en el mismo entorno, y que son propensos a causar descargas electrostáticas (*ESD*). Excesivas *EMI* y *ESD* pueden causar varios problemas y han de seguirse unas pautas para evitarlos. Entre ellas el uso de cables apantallados de alta calidad y de longitud adecuada, para evitar enredos entre cables. También se debe evitar la cercanía a aquellos cables por los que circula elevado amperaje. Así mismo se debe evitar el acoplo de interferencias que se produce a través de los nodos de la toma de tierra.

  Para poder visualizar una imagen en un dispositivo es necesario tener en cuenta la resolución del sistema que muestra la imagen, pues ha de disponer de la resolución suficiente para apreciar la imagen con toda la calidad con la que se capturó.

  Si se tiene que reconstruir la señal proporcionada por el sensor, se ha de hacer a una frecuencia de al menos el doble de la frecuencia del detalle más pequeño que se quiera reproducir de la imagen.

  Así, de no cumplirse la regla de sobre muestreo de `Nyquist`, el principal defecto de imagen es el `Aliasing`, que se manifiesta como una distorsión en patrones, bordes y texturas. Una distorsión en bordes y texturas con contenido de color indeseado que le da un aspecto muy característico y lo hace indeseable. Como se ha dicho, el aliasing es un patrón del rango de espurios del que no se puede desprenderse. No se puede borrar cuando ya forma parte de la imagen.

  Cabe mencionar que en futuros tratamientos de imagen ya sea de compresión o cambio de formato se puede perder calidad. La compresión es un método que emplea algoritmos y métodos matemáticos para reducir su tamaño facilitando su transferencia. Encontramos dos tipos de compresión, aquella

que presenta pérdidas como los formatos `.jpeg, .png, .gif` y la compresión sin pérdidas se da en los formatos `.bmp, .tiff, .pict` entre otros. Existen más formatos como los vectoriales (compuestos por figuras planas, lineas y textos) que almacenan información vectorial como `.pdf, .ai, .eps`. Los cambios de tamaño de estas imágenes vectoriales no afectan a la calidad de la imagen.

1.2.4 Fuentes de error

En todos los procesos la calidad de la medida puede verse afectada por un margen de error.

Una forma analítica de representar esta situación quedaría de la siguiente manera: $X = X_{real} \pm e$ donde X es la variable medida, X real es el valor real de la variable, y e es el error en la medida.

Debido a las múltiples causan que pueden originar errores, el error total que afecta a la medición es el resultado de la suma de todos los errores. Existen dos tipos de errores que pueden afectar al buen funcionamiento de la cámara. Se clasifican en errores sistemáticos y errores aleatorios, es decir, los primeros son errores que se dan siempre que se repite el experimento o medición, y los segundos son aquellos que se dan de forma aleatoria.

Debido a las dos componentes de error la fórmula anterior puede reescribirse de la siguiente manera:

$$X = X_{real} \pm (e_{sistematico} + e_{aleatorio})$$

en donde $e_{sistematico}$ es la desviación de la medida debido a errores sistemáticos y $e_{aleatorio}$ debido a errores aleatorios. A continuación se explican ambos tipos:

- ***Errores Sistemáticos***

  Los errores de naturaleza sistemática son aquellos causados por los elementos y factores propios de la cámara o del sistema de medición. Son errores que aparecen en la medición de forma constante debido a que puede estar originado en un defecto del instrumento, en una particularidad del operador. Estos errores no pueden evitarse, lo que no implica que se minimizar o anular su magnitud e influencia. Para ello se lleva a cabo un proceso de calibracion del dispositivo.

  Un error de este tipo puede ser la distorsión de la lente, o la deriva temporal de los componentes, haciendo que todas las medidas se vean afectadas.

- ***Errores Aleatorios***

  Estos errores estan causados por factores externos al sistema de medida en este caso la cámara, independientemente de que sean predecibles o no. Se puede decir que es aquel error inevitable que se produce por eventos únicos e imposibles de controlar durante el proceso de medición.

  Las fuentes de los errores aleatorios son difíciles de identificar y sus efectos no pueden corregirse del todo. Los factores que provocan errores aleatorios suelen seguir una distribución gausiana [1] pudiendo establecerse un intervalo de confianza y las medidas obtenidas fluctúan alrededor de una media.

  Los objetos en movimiento o la luz solar pueden ser fuente de errores aleatorios.

[1] Distribución gaussiana: Distribución de probabilidad de la variable continua que con más frecuencia aparece aproximada en fenómenos reales

- ***Fuentes de errores sistemáticos***

  – Conversiones analógico-digitales:

  La conversión de valores analógicos a valores digitales, realizada por el *ADC* implica un error de cuantificación, esto es una discretización que normalmente conlleva una cierta pérdida de información. Se debe a los redondeos necesarios aplicados a los valores analógicos cuya resolución es por naturaleza infinita, para convertirlos a un medio digital con precisión finita.

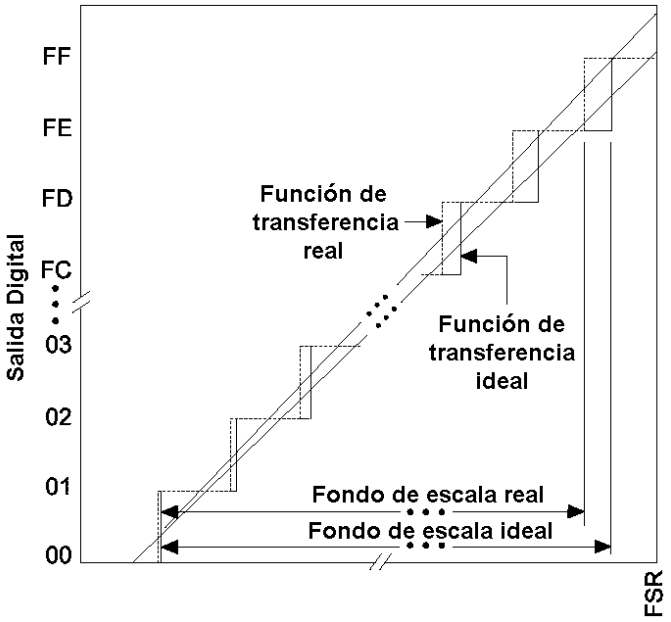

Figura 1.6: Función de transferencia real de un ADC

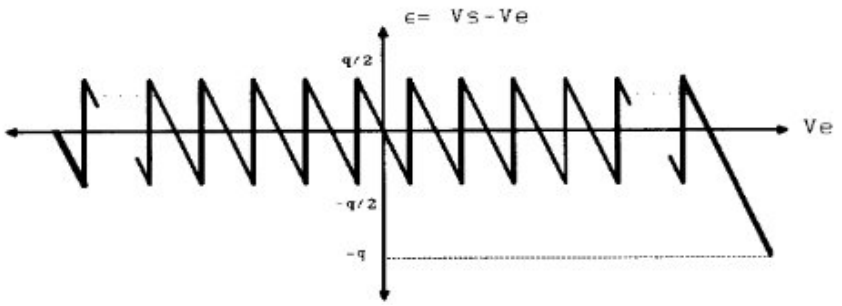

Figura 1.7: Error cometido en la cuantificación de un ADC en función de la señal de entrada. Varia entre $\pm q/2$

  – Iluminación:

  La condiciones de iluminación pueden provocar algunos efectos físicos sobre los sensores semiconductores y la electrónica de la cámara, redundando en un error de la medida.

  – Temperatura:

  Los componentes semiconductores y electrónicos de la cámara son sensibles en mayor o menor medida a la temperatura. En función de la temperatura a la que opera la cámara, la deriva térmica de los componenentes puede introducir un error sistemático en las medición.

– Reflectividad de los materiales:

La reflectividad de los objetos tiene gran influencia en la repetitividad de la medición. Por esta razón, es importante comprender cómo se define la reflectividad en un objeto, y cómo se mide la misma.

La reflectividad puede definirse como la fracción de radiación incidente reflejada por una superficie y viene determinada por dos factores:

El primero es el coeficiente de absorción de los materiales que relaciona la luz emitida con la luz recibida.

El segundo es la distribución angular de la luz emitida y reflejada. Esta propiedad es conocida como `foreshortening`.

Material	Reflectividad	Material	Reflectividad
Tarjeta Kodak gris brillante	107%	Panel de madera rugosa	25%
Papel blanco	80 – 100%	Hormigón liso	25%
Mampostería blanca	85%	Tarjeta Kodak gris oscuro	33%
Periódico	70%	Neumático de caucho negro	2%
PVC gris	40%		

Tabla 1.1: Tabla de reflectividad de distintos materiales

En la Tabla 1.1 se indican las reflectividades de diversas superficies. La referencia del 100 % viene dada por el denominado reflector Lambertiano perfecto, donde el 100 % de la luz es difundida de vuelta con una distribución de intensidad independiente del ángulo de observación.

Un método sencillo de determinación de reflectividad de una superficie un objeto consiste en situar al lado un elemento de referencia con reflectividad conocida, y comparar ambas reflectividades, siempre con idéntica luz y bajo el mismo ángulo de incidencia.

- ***Fuentes de errores aleatorios***

  – Objetos en movimiento:

  Los objetos en movimiento pueden ser una fuente importante de errores. Si durante el tiempo de exposición, tiempo durante el cual el sensor capta la luz que incide sobre él, se mueve un objeto dentro de la escena no se logra una calidad de imagen adecuada. Este movimiento se traduce en un ruido aleatorio en la imagen.

  – Luz solar:

  La luz solar afecta a la calidad de la imagen obtenida de estas cámaras. La mayoría de las cámaras digitales poseen prestaciones que permiten ser utilizadas bajo muchas condiciones de iluminación. Por lo tanto, se debe procurar evitar su exposición bajo luz solar directa, pues no se lograría obtener la imagen correcta. No obstante, la luz solar indirecta proveniente de ventanas, puertas, y otras aberturas en interiores no suele causar ningún problema.

La calidad de la medición y la existencia de errores en el proceso o instrumento puede determinarse por la precisión absoluta y por la repetividad de la misma.

– La precisión absoluta puede definirse como la diferencia entre el valor real de una variable y el valor promedio. Representa el mayor error sistemático posible en la medición.

– La repetividad se caracteriza como la propagación de la medición alrededor del valor promedio, y es relevante en la precisión de una medida única. Este valor da una indicación del ruido implicado en una medición dada.

Figura 1.8: Diferencia entre precisión y exactitud de la medida

1.2.5 Características de la cámara empleada

En este proyecto se empleará una cámara del fabricante alemán Basler (compañía fundada en 1988) [3], dentro de las múltiples gamas disponibles pertenece a la familia Scout de cámaras de escaneo de área (Area scan camera) [2]. Son cámaras extremadamente versátiles y de utilidad conocida en variadas aplicaciones. Entre otras destaca su aplicación a sistemas de control e inspección industrial, al ámbito médico, etc.

Figura 1.9: Basler Scout

La familia scout ofrece una amplia selección de resoluciones y velocidades de captura debido a que lleva instalados los mejores sensores CCD de la marca Sony. La comunicación con el dispositivo puede

realizarse a través de dos interfaces: Gigabit Ethernet (GigE) o FireWire-bTM (IEEE 1394b). En el presente proyecto el modelo empleado dispone de un protocolo de comunicación GigE, que más adelante se explicará. En la Figura 1.10 se muestra el diagrama de bloques común a todos los modelos scout.

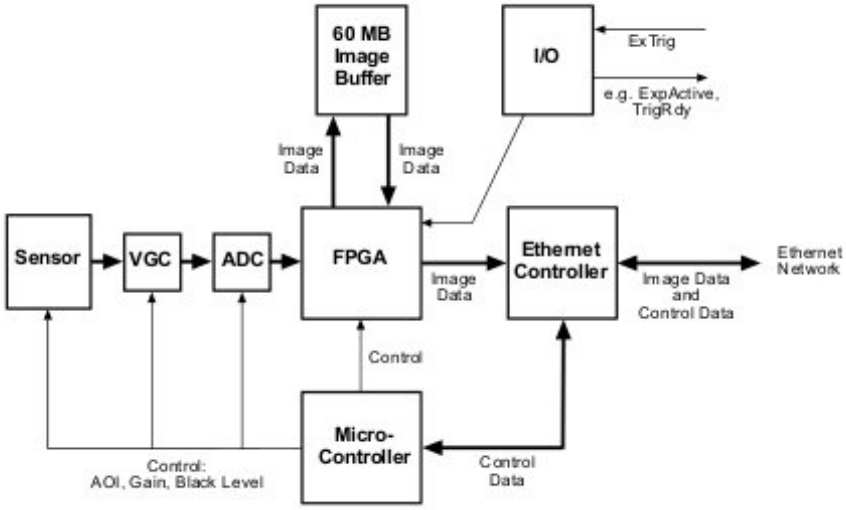

Figura 1.10: Diagrama de bloques interno Basler Scout

Para lograr una imagen de calidad el fabricante asegura una calibración y alineación precisa del sensor realizada con instrumentos de ultra-alta precisión que aseguran la calidad del producto, así mismo, todas las cámaras se ejecutan en modo de exploración progresiva para asegurar un buen resultado. Basler lidera el esfuerzo para estandarizar mediciones empleando EMVA 1288 [4] que describe un método unificado para medir y computerizar imágenes y parámetros. Basler asegura una garantía de 3 años.

En la Tabla 1.2 se muestran las especificaciones generales de la cámara, proporcionadas por el fabricante:

Mono / Color	Mono / Color
Interface	Gigabit Ethernet (screw lock possible)
Video Output Format	Mono 8: 8 bits/pixel | Mono 16: 16 bits/pixel | YUV 4:2:2: 16 bits/pixel average YUV 4:2:2: (YUYV):16 bits/pixel average | Raw 8: 8 bits/pixel (R,G, or B) | Raw 16: 16 bits/pixel (R,G, or B) | scA750-60gm/gc (8 bits/pixel only) | RGB 8 packed
Synchronization	Via external trigger or free run
Exposure Control	Programmable via GigE Vision (camera API)
Housing Size (L x W x H)	73.7 mm x 44 mm x 29 mm (without lens adapter)
Housing Temperature	Up to 50 °C
Lens Mount	C-mount
Digital I/O	2 opto-isolated input ports, 4 opto-isolated output ports
Power Requirements	12-24 VDC; via Hirose 12-pin connector (max. 10 meter cable length)
Conformity	CE, FCC, RoHS, IP 30, GigE Vision, GenICam
Driver	Basler pylon Camera Software Suite or 3rd party GigE Vision Software

Tabla 1.2: Especificaciones generales Basler Scout

De la tabla superior se extrae información común relevante para todos los modelos de la familia Scout.

- Formato de salida de video: los formatos de salida disponible son Mono8 (8 bits por píxel), Mono16 (16 bits por píxel), YUV (16bits), RAW8 (8 bits por píxel) y RAW16 (16 bits por píxel).
- Sincronización: la sincronización puede realizarse por disparo externo.

- Control de exposición: es programable a través de la API de la cámara, en este caso la interfaz GigE. Esta interfaz se caracteriza por una fácil integración en los programas de procesamiento de imágenes mediante el uso de librerías de software, ofreciendo una alta compatibilidad con otros dispositivos.
- Digital I/O: dispone de 2 puertos de entrada opto-aislados y 4 puertos de salida opto-aislados. Los opto-aisladores previenen que altos voltajes afecten al sistema que recibe la señal.
- Alimentación requerida: 12-24 VDC

Basler scout	Resolution (HxV pixels)	Sensor	Sensor Technology	Sensor Size (optical)	Pixel Size (µm)	Frame Rate	Power Consumption (typical)	Weight (typical)
scA1390-17gm/gc	1392 x 1040	Sony ICX267	Progressive Scan CCD	1/2"	4.65 x 4.65	17	3.5 W	160 g

Tabla 1.3: Especificaciones técnicas Basler Scout scA1390-17gm/gc

De la Tabla 1.3 se extrae información importante y específica del modelo como:

- Resolución: 1392 x 1040 píxeles
- Modelo del sensor: Sony ICX267: este sensor de tipo *CCD* posee una matriz cuadrada de 1,45 millones de píxeles efectivos y una longitud diagonal de imagen de 8mm. El escaneo progresivo permite la lectura de la señal de los píxeles independientemente. Puede llegar a soportar hasta 30 imágenes por segundo (fps).

  Entre sus características destacan: alta resolución horizontal y vertical, alta reproductividad del color, alta sensibilidad, dispone disparador contínuo de velocidad variable, además presenta un comportamiento excelente antiblooming. Por último se hace referencia a una baja dark current, que en castellano se entiende como una pequeña corriente eléctrica que es generada en los dispositivos fotosensibles en aunsencia de fotones incidentes. Esta corriente es debido a los defectos cristalográficos del sensor.

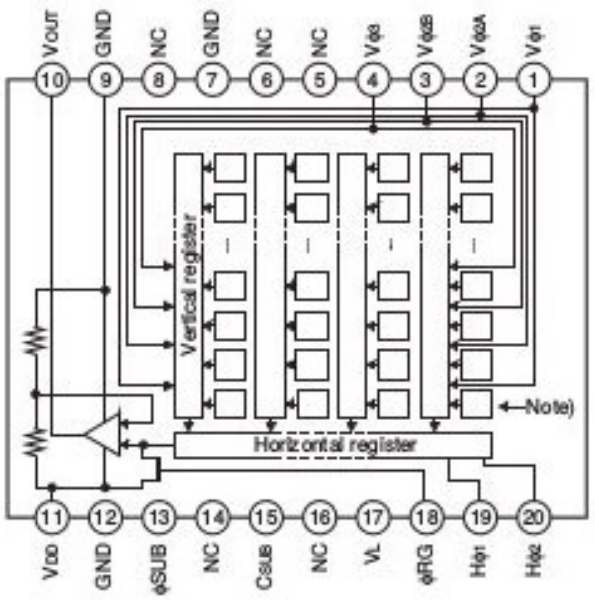

Figura 1.11: Sensor Sony ICX267: Diagrama de bloques

- Tamaño del sensor: (1/2)”
- Tamaño del píxel: (4.65 x 4.65) μm^2
- Frame rate: este término hace referencia a las imágenes por segundo, es decir, a la medida de la frecuencia a la cual un sistema de imágenes genera o procesa distintos fotogramas.
- Consumo energético: 3.5 Watios
- Peso: 160 gramos

La Figura 1.12 representa la respuesta espectral del sensor en función de la longitud de onda incidente.

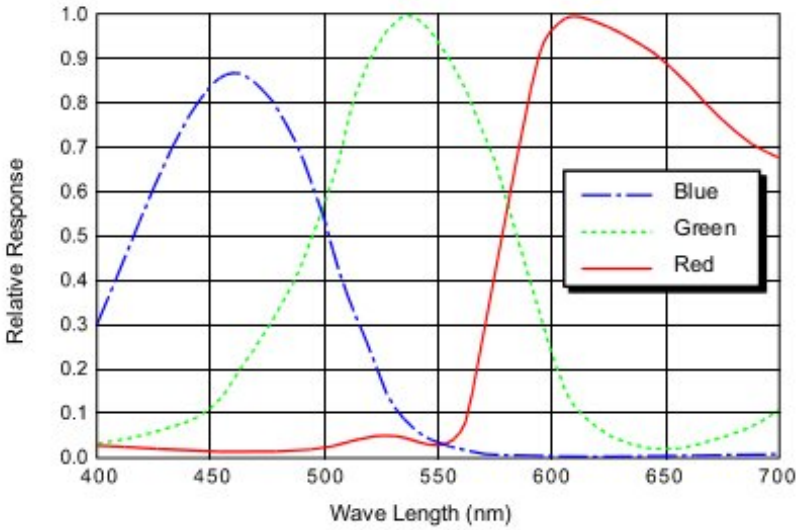

Figura 1.12: Respuesta Espectral Sensor Sony ICX267

A continuación se muestra la Figura 1.13 que de manera esquemática reproduce el funcionamiento del software Basler *Pylon* empleado para trabajar con la cámara.

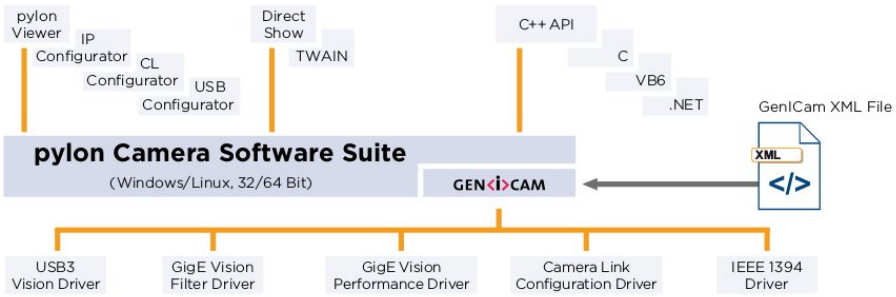

Figura 1.13: Pylon Camera Software

Este software opera con todas las cámaras de línea y área de Basler. La arquitectura de este software está basada en la tecnología GenICam que ofrece acceso a los últimos modelos de cámara y propiedades. Incorpora algunas aplicaciones como PylonViewer para ajustar los parámetros de la cámara, capturar y mostrar imágenes. La aplicación IpConfigurator permite conectar y configurar la conectividad y el ajuste de la dirección Ip de la cámara. Este software incorpora un potente SDK para el desarrollo de aplicaciones [5].

1.3 Estudio Experimental y Solución Desarrollada

En este apartado se documentará el desarrollo llevado a cabo y la técnica que sustenta la adquisición y procesado digital.

1.3.1 Datos de salida proporcionados por la cámara

Las librerías de la cámara con la que se realiza el presente proyecto incluyen multitud de funciones que permiten modificar los parámetros de trabajo de ésta, ya sean paramétros propios de conectividad o parámetros que afecten a la calidad de la imagen.

Esta parte requiere de un estudio previo de la información disponible de la cámara como el manual de usuario y el manual del programador, pues es necesario conocer los modos de configuración de la cámara así como los valores de trabajo disponibles para poder desarrollar el nuevo software y asegurar su funcionamiento.

En la solución desarrollada se permite modificar los parámetros más relevantes que afectan a la calidad en la adquisición de la imagen. También se puede modificar el modo de adquisición de la cámara, para adquirir imágenes en escala de grises o en color. Debido a que la aplicación está orientada a un usuario final con menor conocimiento técnico se simplifica en gran medida los parámetros disponibles para la captura, reduciendo el manejo de la cámara a unas pocas configuraciones.

A nivel de programación, la información proporcionada por cada cámara puede ser accedida mediante la *API* (Application Programmable Interface) proporcionada por cada fabricante. En el presente trabajo, las librerías pueden ser utilizadas mediante *C/C++*, tanto en Linux como Windows.

En este apartado se explican y se describen los parámetros necesarios para el buen funcionamiento de la cámara, así como aquellos parámetros que garantizan una captura óptima. Los primeros se refieren a aquellos parámetros que se configuran una vez dependiendo de las carácteristicas del equipo en el que se ejecute la aplicación, cuya configuración se debe realizar externamente a *QOP*. Los segundos serán configurables en su mayoría en cada adquisición en *QOP*. Ambos serán abordados con mayor profundidad en el manual de usuario que figura como anexo.

1. **Parámetros de funcionamiento**

   Esta nomenclatura hace referencia a aquellos parámetros que determinan como se transmiten los datos al PC a través de la conexión a internet, y que serán propios de cada equipo. Estos datos pueden ser de salida (lectura) o de entrada/salida (permite lectura y escritura). Procedemos a describir los más relevantes para asegurar una correcta conexión y funcionamiento:

   - *Payload Size*:(Salida). Indica el tamaño total de la información que transmite la cámara, es decir, el tamaño de los datos que conforman la imagen en bytes. En su lectura devuelve un valor de tipo entero de 64bit (int64_t).
   - *Packet Size*:(Entrada/Salida). Mediante este valor se ajusta el tamaño de los paquetes de datos que envía la cámara en bytes. Este valor es independiente de cada equipo y será necesaria su configuración a su valor máximo dependiendo del modelo de adaptador de red usado. En su lectura devuelve un valor de tipo entero de 64bit (int64_t).
   - *Inter-packet Delay*:(Entrada/Salida). Permite ajustar el retardo entre el envío de paquetes de la cámara mediante ticks de reloj. Cada tick equivale a $8ns$. Cuando se aumenta este valor decrementa la efectividad de la cámara en la transmisión, así como el ancho de banda usado por la cámara. En su lectura devuelve un valor de tipo entero de 64bit (int64_t).

- *Frame Transmission Delay*:(Entrada/Salida). Permite configurar el retardo existente entre el inicio teórico de la transmisión de datos y el inicio real. Normalmente cero, es configurable con ticks de reloj como los del apartado anterior. En su lectura devuelve un valor de tipo entero de 64bit (int64_t).
- *Bandwidth Assigned*:(Salida). Indica el ancho de banda en bytes por segundo que emplea la cámara para transmitir información. Está relacionado con Packet Size y con Inter-packet Delay, según la siguiente ecuación:

$$BandwidthAssigned = \frac{X * Y}{[(X * Y * 8ns) * ((X - 1) * IPD * 8ns)]} \tag{1.4}$$

  Siendo X el número de paquetes por imagen, Y el número de bytes por paquete, IPD el parámetro *Inter-packet Delay*.
- *Resulting Frame-Rate*:(Salida). Este parámetro indica el máximo ratio de adquisición en imágenes por segundo. Está relacionado con el área de interés, tiempo de exposición y ancho de banda. En su lectura devuelve un valor de tipo double.

2. **Parámetros de captura**

   En este apartado se explican los parámetros que influyen en la captura de imágenes. Los parámetros de este tipo más relevantes son configurables a través del módulo implantado en *QOP*. Estos parámetros son de lectura/escritura.

   - *Acquisition Mode*. Indica el modo de adquisición de la cámara asociada. Se puede configurar en modo SingleFrame o en modo ContinousFrame, para captura una sola imagen en el primer caso y más de una en el segundo. En su lectura devuelve un valor booleano, ya que solo hay dos modos de configuración.
   - *Trigger*. Acerca del trigger o disparador existen 4 funciones que permiten su configuración. Mediante la función TriggerMode se activa o desactiva. Usando TriggerActivation se puede configurar el tipo de activación del disparador, por flanco de subida por ejemplo.
   - *Gain*. Mediante la función GainRaw se puede ajustar la ganancia al valor deseado. Es un parámetro fundamental en la captura y del que se ha hablado anteriormente en la sección 1.2.3. Devuelve un valor de tipo entero.
   - *Black Level*. Usando la función BlackLevelRaw puede obtenerse el valor de negro de una imagen. El incremento de este parámetro provoca un offset positivo en los valores digitales de salida de los píxeles. Devuelve un valor de tipo entero.
   - *White Balance*. El balance de blanco puede ser ajustado usando la función BalanceWhiteAuto, que ajustará el valor medio de los píxeles de color azul y color rojo con el valor medio de los de color verde.
   - *Exposure Time*. También conocido como tiempo de integración es uno de los parámetros más determinantes para adquirir una imagen de calidad. Es el tiempo durante el que la luz incide sobre el sensor. Mediante la función ExposureTime puede leerse o ajustarse su valor de tipo entero. Se dedica un apartado completo a explicar este parámetro 1.3.2.
   - *Pixel Format*. Empleando la función PixelFormat podemos ajustar el formato de color de la cámara, para capturar imágenes en escala de grises o en color. En nuestro modelo disponemos de cinco formatos diferentes, en contra de lo que dice el manual que permite mayor número de tipos. En el desarrollo del módulo se han incorporado los dos formatos más importantes.

     El primero llamado `PixelFormat Mono8` permite capturar imágenes en blanco y negro. Cuando la cámara se ajusta con este formato la salida son 8bits de intensidad de brillo por píxel. Pudiendo tomar cada píxel un valor entre 0 y 255.

El segundo formato está denominado como `PixelFormat BayerBG8` y tiene utilidad para capturar imágenes en color. La salida está formada por 8bits de datos por píxel. Estos datos no están procesados ni interpolados. La lectura de cualquier color depende de la disposición del filtro bayer. La intensidad de color de cada píxel depende del color de la lente que tenga asociada, y oscila en un valor comprendido entre 0 y 255.

Las siglas *BG* en el nombre Bayer determinan la alineación de los colores en el filtro. Para las filas pares el píxel uno será azul, el segundo verde, el tercero azul, etc. Para las filas impares el píxel uno será verde, el siguiente rojo, el tercero verde y así sucesivamente.

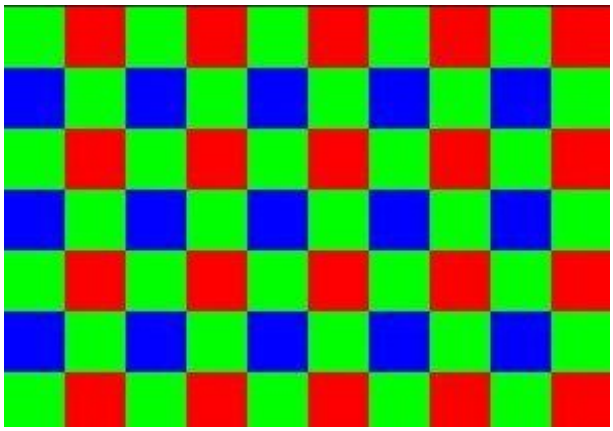

Figura 1.14: Filtro Bayer

- *Area of Interest.* El área de interés hace referencia al tamaño y posición de la escena de relevancia en la captura. A través de la función Width y Height se configura con un valor entero el tamaño de la escena a capturar. Por defecto la resolución está al máximo en nuestro modelo 1278 x 958 pixeles siendo el primer valor correspondiente al ancho(eje X) y el segundo al alto(eje Y). Se puede ajustar un offset en el area de interés. Este offset esta referenciado respecto de la esquina superior izquierda del sensor, que corresponde con la columna y fila 0 de la matriz de píxeles que conforman el sensor.

1.3.2 Tiempo de integración

El *tiempo de exposición* o *tiempo de integración* es un factor clave en la fotografía. Es el tiempo en el que el sensor capta la luz que incide sobre él. Este valor de trabajo de la cámara es especialmente importante pues determina la velocidad a la que se capturan las imágenes, así como la calidad final de la imagen.

La *velocidad de disparo* está estrechamente relacionada con el tiempo de exposición, pues es su función inversa. Si aumenta el tiempo de exposición disminuye la velocidad de disparo y vicerversa. Permite manipular los efectos visuales de la imagen final más allá de su luminosidad.

- A nivel técnico el tiempo de integración es el tiempo durante el que la luz incide sobre las células del sensor, estas células producen corriente eléctrica en función de la cantidad de luz incidente, debido al efecto fotoeléctrico. Posteriormente esta corriente carga unos condensadores que proporcionan una tensión que el *ADC* discretiza para formar la imagen.

- A nivel práctico un tiempo de exposición corto deja pasar menos luz al sensor fotosensible y por ello se consigue congelar o reducir notablemente el movimiento, mientras que un tiempo de exposición largo permite obtener imágenes movidas o desplazadas, otorgando mayor sensación de desplazamiento.

Los tiempos de exposición de una cámara fotográfica pueden ajustarse en valores discretos, para nuestro caso el fabricante informa de un tiempo mínimo de exposición $34\mu s$ y un máximo $10000000\mu s$.

Escoger un tiempo de exposición óptimo para cada escena es una tarea necesaria para obtener una medición con la mejor calidad posible. Este valor está determinado por tres aspectos como se observa en la figura inferior.

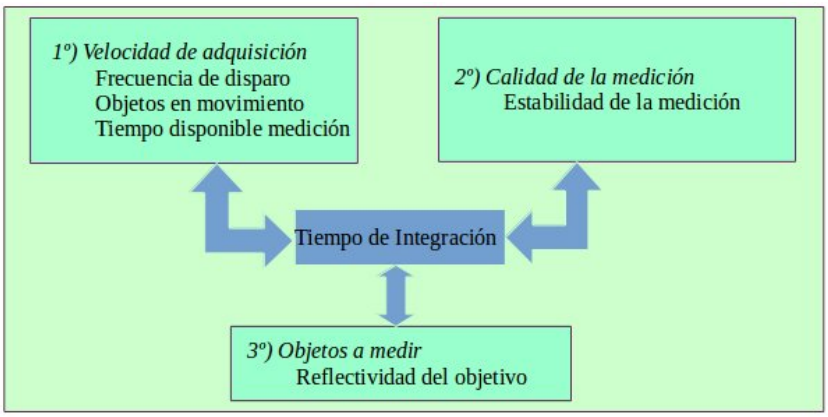

Figura 1.15: Factores a considerar en el ajuste del tiempo de integración

Se explica la figura con mayor detalle:

- Velocidad de adquisición

  La frecuencia a la que se captan las imágenes es inversamente proporcional al tiempo de integración. Por tanto cuando se reduce el tiempo de integración se aumenta la frecuencia de disparo y se disminuyen los artefactos de movimiento que provocan los objetos móviles, pudiendo aumentar la velocidad de estos objetos. Se consume menos tiempo para la medición total.

  La *trepidación* es el efecto que se produce cuando una foto sale movida por un tiempo de exposición demasiado largo o por no controlar el movimiento de la cámara.

Figura 1.16: Efecto debido a artefactos de movimiento

- Calidad de la medición

  Para obtener mayor calidad en la imagen debemos emplear un mayor tiempo de integración, lo que nos permite capturar mayor cantidad de luz reflejada, lo que se traduce en un menor nivel ruido.

  Un tiempo excesivo provocará que la imagen se sature.

  De forma inversa un menor tiempo de integración implica mayor nivel de ruido.

- Escena a medir

  Para alcanzar una medida de calidad (repetitividad) los objetos o materiales que presentan bajos coeficientes de reflectividad necesitan mayores tiempos de integración que los que no. Esto es debido a que menos cantidad de luz incidente requiere mayor tiempo de captación del sensor.

  Los objetos lejanos requieren de mayor tiempo de integración que los cercanos debido a la atenuación que sufre la intensidad lumínica con la distancia.

1.3.3 Consideraciones adicionales

Además de los factores que hemos mencionado en otros apartado 1.2.3 existen algunos aspectos que hay que tener en cuenta para trabajar con cámaras digitales:

- Condiciones de montaje: la cámara deberá ser montada y asegurada sobre un soporte firme, recto y sin vibraciones y que permita la circulación del aire alrededor de la misma para evitar sobrecalentamientos.

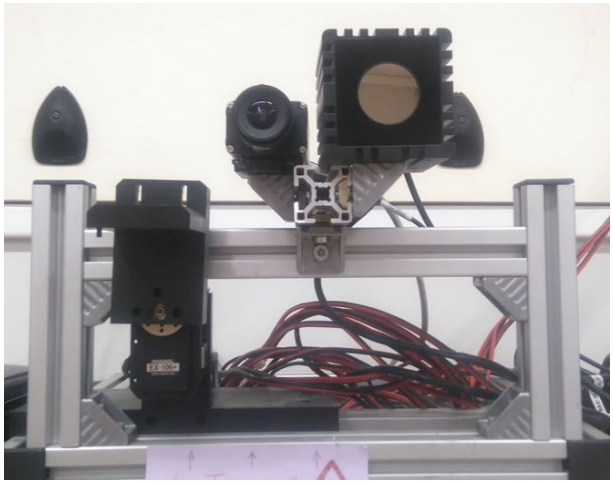

Figura 1.17: Montaje Basler Scout junto a ToF

- Condiciones ambientales: el ambiente de operación de la cámara es un factor clave si se quiere asegurar un correcto funcionamiento. La humedad y temperatura de la sala debe estar dentro del rango de operación proporcionado por el fabricante. Así mismo un entorno libre de fuentes emisoras de radiación electromagnética o lumínica es recomendable para asegurar la máxima calidad posible de medición.

- Condiciones de temperatura: durante el funcionamiento de la cámara digital hay que evitar el sobrecalentamiento. Para ello se recomienda: forzar la circulación del aire mediante ventilación externa, montar la cámara junto a una masa térmica mayor para lograr disipación de calor por conducción termal a través de su soporte o contactos, reducir el tiempo de exposición y velocidad de disparo al mínimo. Por último es recomendable montar disipadores sólidos o masillas disipadoras.

- Condiciones de limpieza: como todo dispositivo electrónico conviene limpiar la cámara periódicamente y mantenerla libres de polvo, esto permitirá que la captación de fotones por el sensor sea correcta y no se vea afectada por motas de polvo. Así mismo la limpieza influye en la disipación de calor generado.

1.3.4 Lenguaje de programación y librerías

Todas las herramientas necesarias para el desarrollo de la aplicación son gratuitas, es decir de código libre (open source), pues no tienen fines lucrativos ni comerciales.

La libertad de código se aplica también a las tareas complementarias como la compilación, enlazado, edición y análisis de datos e imágenes del proyecto, para estas tareas secundarias se ha empleado las herramientas más difundidas.

En este proyecto se continúa con la filosofía de código libre existente en el proyecto QOP [1]. A continuación se enumeran los elementos necesarios para la ejecución del proyecto:

1. Lenguaje de programación *C++*

   En el presente trabajo se continúa con la buena praxis de programar el nuevo módulo en *C++* preferiblemente sobre *C* ya que es más completo y polivalente, sin perder rendimiento. A su vez esta elección está tomada ya que la *API* proporcionada por el fabricante `Basler` está implementada en *C* natural. Además es soportado por casi todas las arquitecturas y sistemas operativos, permitiendo una mayor capacidad de funcionamiento en múltiples plataformas. Todas las herramientas utilizadas soportan *C++* ya que es un lenguaje totalmente vigente en las soluciones software actuales.

   Esta elección facilita la comprensión de la aplicación a futuros desarrolladores.

2. Sistema Operativo Linux

   Se escogió este sistema operativo por ser código libre, así como el punto de partida para el desarrollador nobel y por contar con un gran apoyo de la comunidad. En este proyecto se emplea una distribución de Linux conocida como Ubuntu por su fiabilidad, coste cero y carácter multiplataforma [6].

3. Entorno de desarrollo Qt

   Qt Creator junto con el editor integrado Qt Designer conforman un entorno de desarrollo gráfico muy completo. Este software propiedad de la compañia `Digia` se distribuye gratuitamente para fines no comerciales [7].

   El nuevo módulo *RGB* se integrará a la aplicación *QOP* a través de este entorno de desarrollo cuyas características son:

   - Entorno de desarrollo muy intuitivo: constituye una aplicación amigable, con autocompletado y coloreado de los diferentes elementos (lo que permite ahorrar tiempo en la programación) y muy versátil a la hora compilar diferentes versiones del mismo programa (versiones de desarrollo o definitivas), con diseñador visual de ventanas y formularios, e integración total con analizadores de código (que permite la verificación de memoria), depuradores y otras herramientas de desarrollo avanzadas.
   - Posibilidad de aprendizaje rápido: permite una iniciación sencilla con múltitud de ejemplos para adentrarse en la programación de aplicaciones y una comunidad de usuarios muy activa. También presenta un asistente de proyecto, y un sistema avanzado de ayuda contextual.

- Multiplataforma: su polivalencia para funcionar en diferentes sistemas lo convierte en un entorno de desarrollo de gran interés. Con una buena implantación y metodología la misma aplicación en este caso *QOP*, puede funcionar en Linux y Windows de 32 y 64bit. También es posible su implantación en sistemas como Android y Mac.
- Compatibilidad: ofrece diversas características que pueden ser utilizadas independientemente del sistema operativo: manejo de eventos, capacidad de procesos en paralelo, manejo y presentación de imágenes, administración de variables permanentes de una sesión a otra, etc.
- Fácil inserción de nuevas funcionalidades: añadir librerías que permitan aumentar la funcionalidad del código es una tarea sencilla, por ello esta característica es recomendable para poder implementar las librerías propias de la *API* de la cámara, así como otras que resulten de interés.
- Multi-idiomas: las aplicaciones realizadas en Qt tienen como idioma base el inglés. Es posible su traducción a otros idiomas, sin embargo, el castellano aún no está implementado.

4. Librerías de tratamiento de imagen Open CV

   Es una herramienta de manejo de visión artificial e imágenes, sus siglas significan *Open Computer Vision* y también son código libre. Están muy extendidas en el tratado de imágenes digitales. En este trabajo se emplean para llevar a cabo la conversión de formato de los datos de salida digitales de los píxeles de la cámara a un formato disponible en Qt, mediante la aplicación de mapas de color.

5. Librerías proporcionadas por el fabricante Basler

   La *API* proporcionada por el fabricante `Basler` está formada por un grupo de librerías que son implementadas en *C* natural. Es necesario estudiar la documentación disponible para programar correctamente las funciones de la cámara. Estas librerías permiten la configuración de la cámara y captura de imágenes, siendo implementadas en el presente proyecto según nuestras necesidades.

1.3.5 Solución desarrollada

El punto de inicio de este trabajo de final de grado es la interfaz *QOP* [1] para la adquisición y tratamiento de imágenes proporcionadas por cámaras de tiempo de vuelo. Trabaja con diversas configuraciones de cámaras *ToF* y valores de trabajo, permitiendo editarlas y guardarlas en cualquier momento, así como realizar testeos y procesado de imágenes mediante algoritmos entre otras funciones.

Para poder llevar a cabo este proyecto es necesario un minucioso estudio intensivo de la aplicación *QOP* que permita comprender la multitud de funciones y el enorme flujo de datos presente en la aplicación. Sin esta labor el diseño, implementación y evaluación del nuevo módulo no hubiera sido posible. Para lograr otro de los objetivos de gestión y control de los valores obtenidos es necesaria una lectura detallada de los documentos y de la *API* proporcionada por el fabricante que permite implementar correctamente las funciones.

La dificultad es debida a la dependencia del software de diferentes librerías, las cuales deben estar perfectamente instaladas y acopladas entre sí antes de poder ser utilizado por el usuario.

La aplicación *QOP* se caracteriza por su modularidad y extensibilidad. A continuación se expondrán los diagramas y módulos que componen la interfaz *QOP* a nivel interno para justificar la estructura de desarrollo.

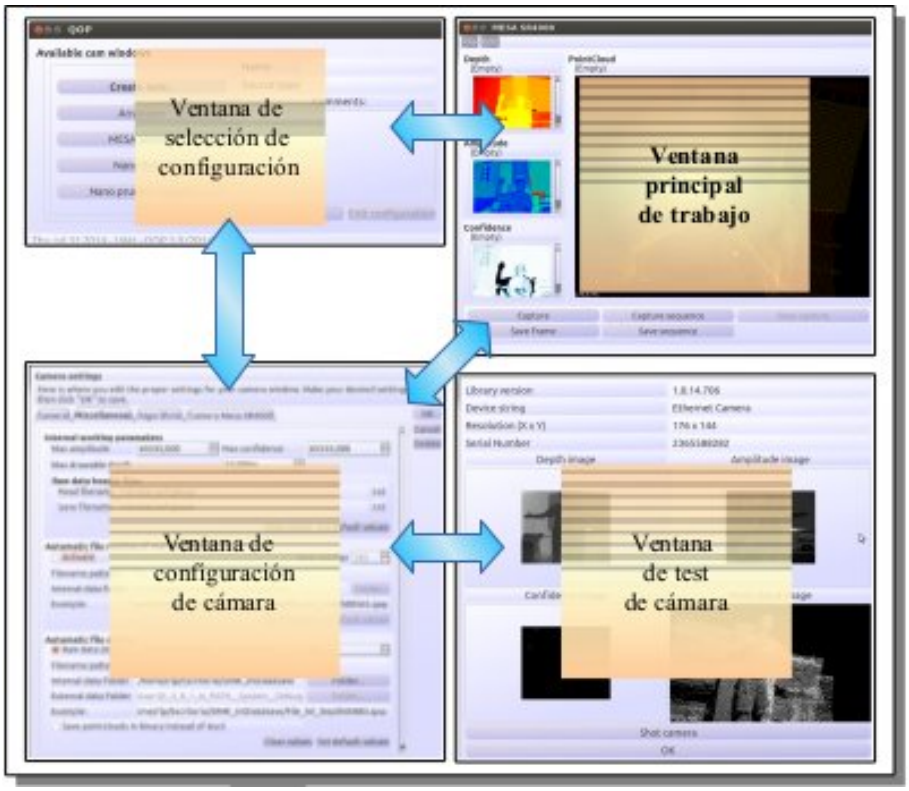

Figura 1.18: Diagrama funcional de QOP

La Figura 1.18 tomada de [1] pertenece al diagrama general de funcionamiento de *QOP*, como se observa existen cuatro ventanas que se explican a continuación:

- Ventana de selección de configuración

  Cuando se arranca la aplicación aparece la ventana de selección de configuración que permite crear y editar o abrir una configuración de cámara entre los tres modelos disponibles.

- Ventana de configuración

  Permite ajustar todos aquellos parámetros generales y característicos de cada cámara. Está asociado al sistema gestor de configuraciones que es el responsable de almacenar, leer y eliminar del disco los valores de todas las configuraciones creadas por el usuario, y de servir estos datos al resto de componentes del software cuando lo necesiten. Esta es la ventana en la que se ha incorporado la posibilidad de configurar y probar el funcionamiento de la nueva cámara *RGB*.

- Ventana de Test

  Desde la ventana anterior de configuración puede accederse a esta permite comprobar el correcto funcionamiento de las diferentes cámaras *ToF* y *RGB*.

- Ventana principal de trabajo

  La ventana principal de trabajo se encarga de capturar y mostrar imágenes y llevar a cabo las acciones de guardado. La visualización de la imagen en esta ventana para la cámara *RGB* añade un visor superpuesto para no modificar los existentes, debido a que los datos de salida de la cámara *ToF* son diferentes a los de la cámara usada. Es la ventana principal de la interfaz que está manejada por el hilo principal. También el hilo secundario es manejado por el principal y se encarga de la capa de entrada de datos (via *API*), de la capa de normalización y de la salida de datos.

1.3.5.1 Diseño

Una vez abordado el estudio de *QOP* se puede comenzar el diseño y programación del nuevo módulo. La línea de desarrollo de este proyecto mantiene la forma en la que se programó *QOP* para que el futuro desarrollador pueda comprender de manera sencilla como funciona la aplicación, y sea capaz de implementar nuevas funcionalidades teniendo tres modelos de cámara de referencia, pues en este proyecto se ha tenido como referencia las cámaras de tiempo de vuelo cuyo funcionamiento es distinto al de una cámara *RGB* convencional.

Se pretende ser minimamente invasivo con el código original, que en pocas ocasiones ha tenido que ser modificado, aunque si se han tenido que añadir funcionalidades y código propio.

Como ya se ha comentado, el objetivo a cumplir en este trabajo es la implantación un nuevo módulo *RGB* asociado a la interfaz gráfica *QOP* que permite aumentar la funcionalidad de la aplicación sin restringir ninguna característica previa. Con esta incorporación es posible realizar la configuración y trabajar con las cámaras *ToF* como con la cámara *RGB*.

En la solución desarrollada Figura 1.19 se permite configurar la cámara *RGB* proporcionada por el fabricante Basler con los valores más relevantes, testear su funcionamiento y adquirir imágenes en escala de grises y en color. Este nuevo módulo está perfectamente integrado en el flujo de la aplicación. Esta incorporación permite al usuario final con menor conocimiento técnico, la manipulación y gestión de ambos tipos de cámara. Por cuestiones temporales alguna funcionalidad prevista que requiere de mayor conocimiento técnico no ha podido ser implementada. Sin embargo el sistema presenta un módulo fiable y de fácil manejo.

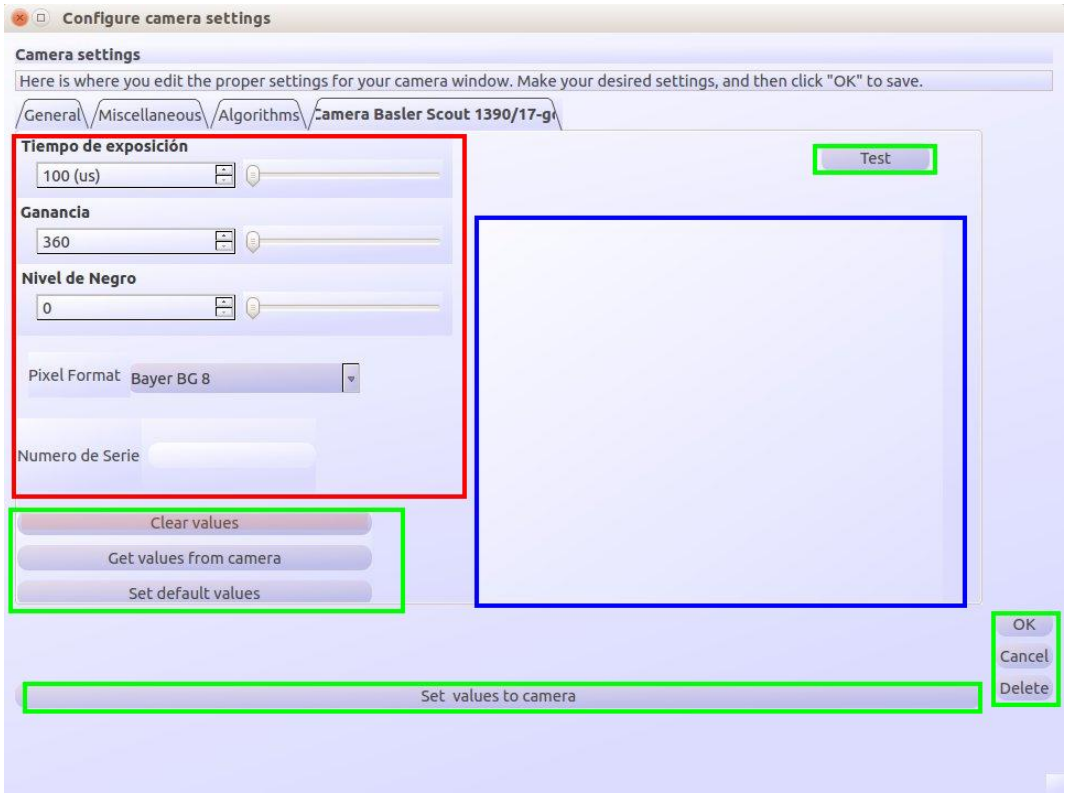

Figura 1.19: Presentación módulo RGB de la interfaz QOP

Se ha recuadrado en diferentes colores cada una de las partes que conforman este módulo. El color `rojo` marca los parámetros configurables de la interfaz. El color `verde` los botones disponibles y el color `azul` se identifica con el visor de imágenes.

Este módulo integrado en la ventana de configuración de la aplicación permite escribir de manera numérica o mediante un cursor los parámetros más importantes como son:

- **Tiempo de exposición**: es posible leer el valor actual de trabajo o escribir de forma numérica el valor entero deseado entre un mínimo 100 μs de y un máximo de 4095 μs.
- **Ganancia**: permite obtener el valor de ganancia de la cámara o configurar la cámara con el valor entero deseado entre 360 y 1023.
- **Nivel de negro**: el valor leido o escrito está contenido entre 0 y 255.
- **Formato de Píxel**: el formato de captura de imagen es seleccionable mediante un desplegable que brinda los dos formatos necesarios: Mono8 y BayerBG8. El primero permite la captura en escala de grises y con el segundo se pueden capturar imágenes en color.
- **Número de serie**: en esta casilla se puede leer el número de serie de la cámara empleada cuando se pulsa el botón Get Default Values.

Estos parámetros pueden ser obtenidos de la cámara y serán representados individualmente en sus casillas.

Se pueden aplicar los valores deseados a la cámara, los valores por defecto o los valores mínimos. Los valores por defecto aseguran una buena captura.

La imagen se mostrará con los parámetros deseados en el la esquina inferior derecha en el espacio habilitado para ello cuando se pulse el botón Test.

Encontramos 8 botones en esta ventana remarcados en color verde cuya funcionalidad se detalla a continuación:

- **Botón Test**: cuando se ejecuta este botón se llama a la función asociada que se encarga de crear el objeto de cámara, de el canal para transmitir la información, de inicializar el búffer y por último de usar las librerías de *Opencv* para adaptar la imagen a un formato compatible con Qt y así poder visualizar la imagen en el visor marcado en color azul en la Figura 1.19.
- **Botón Clear Values**: cuando se hace click sobre este botón se ejecuta la función asociada que se encarga de poner al mínimo todos los valores disponibles, tiempo de exposición, ganancia, nivel de negro y formato de píxel, así como de borrar el número de serie.
- **Botón Get Values From Camera**: permite obtener del dispositivo los valores actuales de los parámetros de interés de la cámara, como el tiempo de exposición, ganancia, nivel de negro y formato de píxel.
- **Botón Set Values to camera**: se encarga de configurar completamente la cámara. Su función es mandar al objeto de cámara los valores enteros y enumerados el (tiempo de exposición, ganancia, nivel de negro y formato de píxel) que se han ajustado en cada uno de los campos de Qt propios de cada variable.
- **Botones OK, Cancel y Delete**: estos botones genéricos a todos los modelos de cámara ya estaban implementados en *QOP*. El botón OK permite crear la nueva configuración de cámara. Si se pulsa el botón Cancel se desecha esta configuración. El botón Delete permite borrar la configuración de cámara creada.

1.3.5.2 Implementación software

Esta sección de implementación requiere de una identificación de las clases existentes en *QOP* cuando abrimos el proyecto con Qt Creator. Esta identificación de clases permite relacionar las diferentes ventanas e interfaces de usuario con los código fuente existentes y por tanto saber que se tiene que dejar intacto, que hay que modificar y que hay que añadir. En la Figura 1.20 se remarcan las clases exitentes en color amarillo en el proyecto *QOP* una vez añadido el módulo.

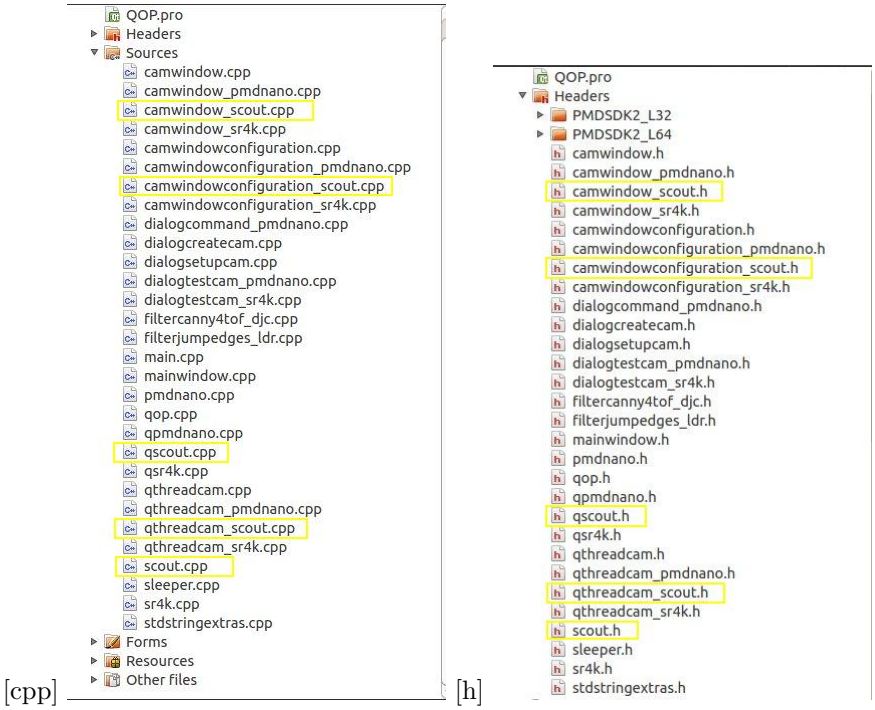

Figura 1.20: Clases existentes en QOP con el módulo RGB

Se observa el gran volumen de datos que tiene la aplicación. Cada clase tiene un archivo de codigo fuente y una cabecera asociado, y en ocasiones una interfaz de usuario que corresponde con las ventanas de la aplicación. En la Figura 1.21 se muestran las interfaces de usuario, archivos con extensión .ui que se identifican con las diferentes ventanas que conforman la aplicación.

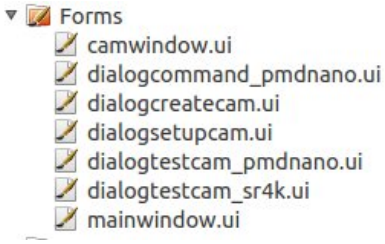

Figura 1.21: Interfaces de usuario en QOP

Siguiendo el flujo de la aplicación, se procede a explicar estos archivos y las modificaciones o adiciones necesarias. El nombre asociado a cada archivo es su nombre de clase, por ejemplo de forma genérica nombreclase.cpp, nombreclase.h y nombreclase.ui corresponden a la clase NombreClase. Aquellos cuya nomenclatura nombreclase_nombrecamara.cpp, nombreclase_nombrecamara.h son subclases de la anterior y pertenecen a la clase NombreClase_NombreCamara. Aquellas subclases de la forma NombreClase_Scout han sido creadas y son necesarias para que el programa elija el hilo correcto.

A continuación se detallan las clases descritas y la implementación desarrollada. La Figura 1.22 muestra de forma esquemática la jerarquía de clases exitente en el proyecto.

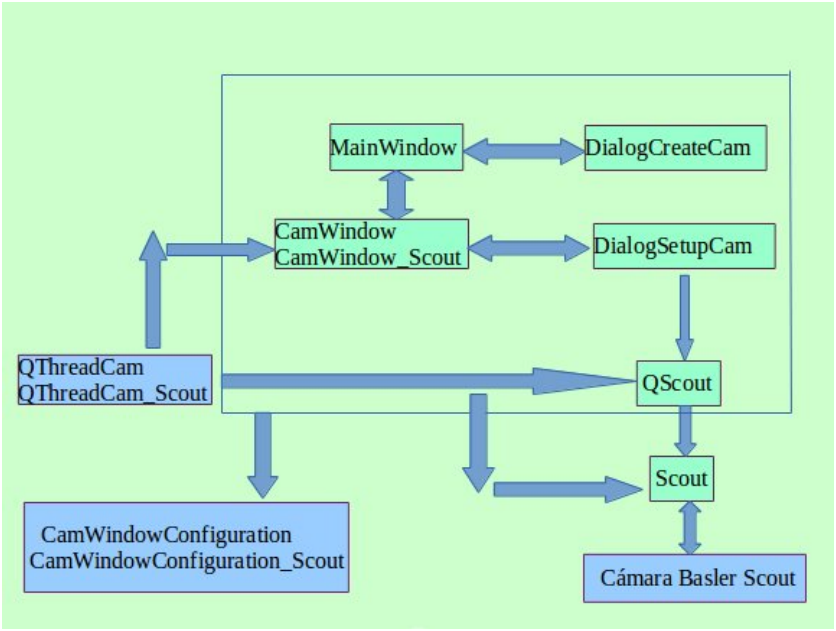

Figura 1.22: Jerarquía de clases

El primer paso en la implementación es poder crear una nueva configuración de cámara Basler Scout cuando se pulsa el botón Create New, para ello se describe el proceso a continuación. Una vez implementada esta posibilidad se procede a crear las clases de la *API* de la cámara.

- *QOP:*

  Clase genérica formada por qop.h y qop.cpp cuya función principal es proveer algunas definiciones empleadas en todo el proyecto, como los enumerados que establecen los modelos de cámara admitidos. Por ello es necesario definir estos nuevos enumerados y variables correspondientes a la nueva cámara.

  - qop.h: es necesario definir dos nuevas variables estáticas STR_CAMWINTYPE_BASLERSCOUT que serán rellenadas con cadenas de caracteres. También hay que añadir un nuevo enumerado de tipo CamWinType que ha sido definido como CAMWINTYPE_BASLERSCOUT.
  - qop.cpp: en este archivo el enumerado definido como CAMWINTYPE_BASLERSCOUT es añadido al switch de la función que devuelve la cadena de caracteres que forma el nombre de cámara asociada.

- *Mainwindow:*

  Esta clase maneja la ventana inicial de la aplicación y se corresponde con la ventana de selección de configuración una vez desaparecen las imágenes iniciales. Está formada por mainwindow.h, mainwindow.cpp y mainwindow.ui. En el presente proyecto esta ventana permanece inalterada pues es común a todas las cámaras. Mediante esta ventana se puede crear una nueva configuración de cámara o escoger una ya existente para abrirla o editar sus parámetros. Cuando se pulsa en el botón Create New se puede crear una nueva configuración de cámara en función de las existentes en la clase DialogCreateCam que se explica a continuación. La Figura 1.23 muestra la interfaz asociada a la clase MainWindow.

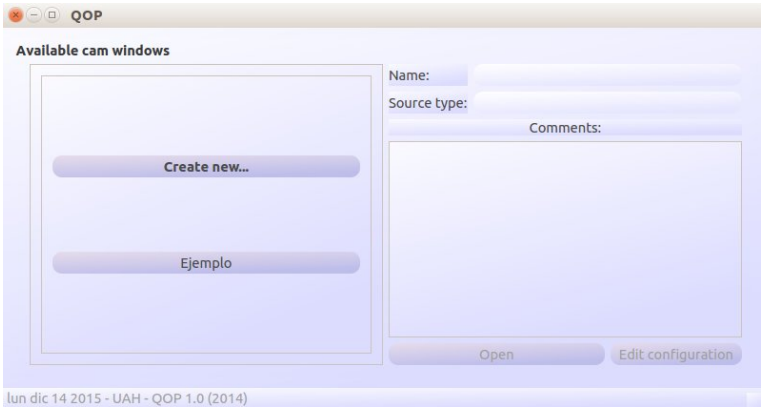

Figura 1.23: Interfaz asociada a la clase MainWindow

- *DialogCreateCam:*

  Cuando se hace click en el botón de creación de cámara se abre un desplegable que permite elegir entre los tres modelos disponibles. Se debe dar un nombre al modelo escogido o de lo contrario se producirá un error. Los archivos incluidos son dialogcreatecam.h, dialogcreatecam.cpp y dialogcreatecam.ui. Para llevar a cabo esta integración es necesario inicializar este nuevo objeto cuyo nombre será el contenido en CAMWINTYPE_BASLERSCOUT en la función de inicialización del menú desplegable. Una vez realizados estos pasos es posible seleccionar el modelo de cámara Basler Scout, si bien no tendrá ninguna funcionalidad asociada hasta que se implante todo el módulo. Como se observa en la Figura 1.24 se puede seleccionar el modelo implantado.

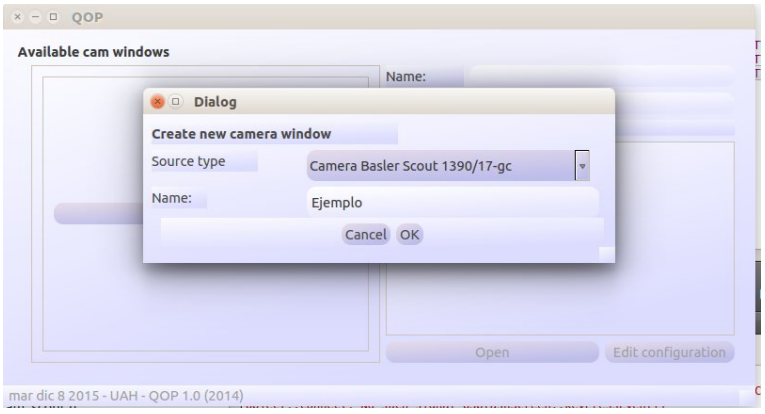

Figura 1.24: Selección del modelo Basler Scout

- *Scout:*

  Actúa de intermediario entre el programador de *C++* y la *API* del fabricante implantada en *C* para la Basler Scout. Permite manejar la cámara desde *C++*, de forma limpia y con la filosofía *POO* (Programación Orientada a Objetos). Esta clase se ha creado íntegramente desde cero con las variables y funciones de interés de la cámara empleada, incluye también los valores por defecto en los ficheros scout.cpp y scout.h

```
unsigned int            intTime;
unsigned int            gain;
unsigned int            blacklevel;
enum PixelFormatEnums   pixelformat;
std::string             serialNumber;
```

Figura 1.25: Variables relevantes en la clase Scout

- *QScout:*

  Clase creada cuyo único objetivo es proveer algunas funciones generales útiles para el manejo de la cámara Basler Scout dentro del framework Qt. Algunas funciones se han programado siguiendo la filosofía de otras clases similiares. Incluye los archivos qscout.cpp y qsocut.h. Mención especial merece la función de normalización del formato de píxel que se muestra en la figura inferior.

```
QString QSCOUT::qStr_PixelFormat(Basler::Scout::PixelFormatEnums _pixelformat)
{
    QString ret;
    switch (_pixelformat)
    {
        case Basler::Scout::PixelFormat_None:
            ret = tr("(None)"); break;
        case Basler::Scout::PixelFormat_Mono8:
            ret = tr("Mono 8"); break;
        case Basler::Scout::PixelFormat_BayerBG8:
            ret = tr("Bayer BG 8"); break;
    }
    return ret;
}
```

Figura 1.26: Función de normalización del formato de píxel

- *CamWindowConfiguration:*

  Clase encargada de guardar y gestionar las configuraciones de cámaras y valores de trabajo del usuario. Contiene todo aquello que afecte al trabajo y no sea específico de un modelo. Debe emplearse la subclase del modelo de cámara a utilizar, por ello en el constructor es necesario añadir nuestro enumerado CAMWINTYPE_BASLERSCOUT. Está formada por camwindowconfiguration.h y camwindowconfiguration.cpp.

- *CamWindowConfiguration_Scout:*

  Subclase de la anterior creada para añadir las configuraciones específicas. Está formada por camwindowconfiguration_scout.h y camwindowconfiguration_scout.cpp. Incluye las funciones setDefaultValues y setClearValues que serán usadas en la configuración de la cámara posible en el módulo implementado.

- *CamWindow:*

  Es la encargada de la ventana principal de trabajo del usuario. Siempre va asociada a una CamWindowConfiguration. Además de gestionar la interfaz, se encarga de la supervisión de los hilos encargados del trabajo con las cámaras QThreadCam. Siempre debe utilizarse la subclase apropiada al modelo de cámara a utilizar, en el constructor es necesario añadir nuestro enumerado CAMWINTYPE_BASLERSCOUT. Está constituida por camwindow.h, camwindow.cpp y camwindow.ui. Cuando se seleccione o cree la configuración de la Basler Scout y se pulse el botón Capture la imagen se muestra en un visor externo para no modificar la estructura de la ventana ya que el visor principal de pcl es tridimensional. En la Figura 1.27 se muestra la ventana Camwindow genérica a los tres modelos de cámara soportados.

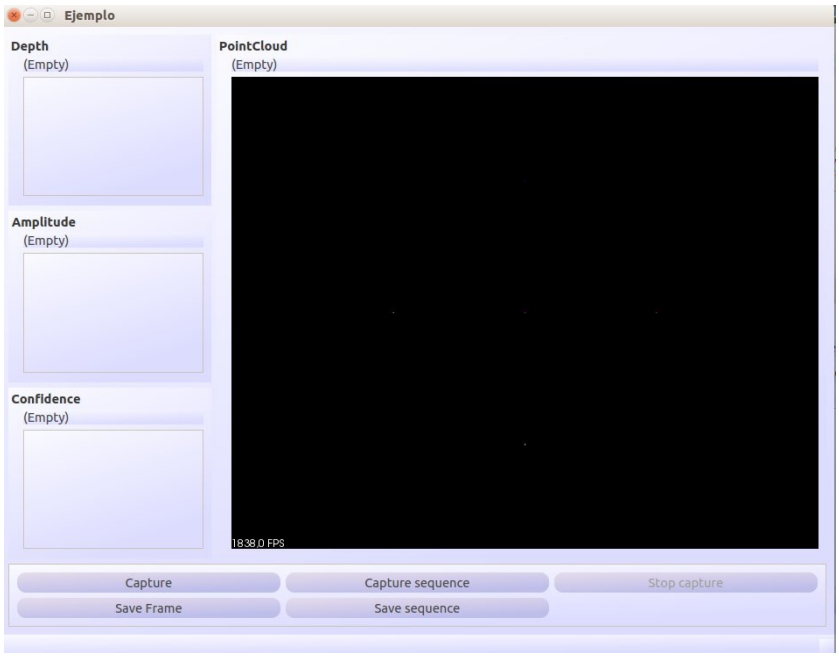

Figura 1.27: Ventana principal donde se mostrará la imagen al pulsar Capture

- *CamWindow_Scout:*

  Es la extensión de la clase anterior para trabajar con la cámara Basler Scout. Asegura que se utilice el hilo de trabajo adecuado a dicha cámara y se configure adecuadamente con la configuración correspondiente a la misma. Se crean los ficheros camwindow_scout.h y camwindow_scout.cpp.

- *QThreadCam:*

  Es la clase principal encargada de trabajar y gestionar las cámaras. Está preparada para funcionar como un hilo secundario de modo que los procesos de computación de datos de cámaras y sus imágenes obtenidos no interfieran con la interfaz de usuario. Esta clase emite una señal al final de cada ciclo. Formada por qthreadcam.h y qthreadcam.cpp. Es una clase específicamente diseñada para *QOP* y que no sufre modificación alguna durante esta integración.

- *QThreadCam_Scout:*

  La variante de la clase anterior que se encarga del modelo Basler Scout. Incluye qthreadcam_scout.h y qthreadcam_scout.cpp. Cuando se pulsa el botón Capture de la ventana principal CamWindow son las funciones de esta clase las que se ejecutan si estamos en el hilo de trabajo adecuado.

- *DialogSetupCam:*

  Es la cara visible al usuario de la clase CamWindowConfiguration, y se encarga de la interacción entre el usuario y las configuraciones de la cámara. Se utiliza la misma clase para gestionar todos los modelos de cámara, adaptándose la interfaz de acuerdo a dicho modelo. Está formada por dialogsetupcam.h, dialogsetupcam.cpp y dialogsetupcam.ui. En el presente trabajo se implementan las funciones de inicialización, configuración y adquisición del módulo *RGB* dentro de la ventana Basler presente en esta interfaz. Es la ventana más importante de este trabajo ya que corresponde con la solución desarrollada. Esta interfaz mantiene las pestañas originales de configuración general, así como la que se asocia a la cámara empleada.

  – dialogsetupcam.h: en este archivo será necesario mantener todas las funciones iniciales e incluir correctamente las rutas donde se encuentran instaladas las librerías proporcionadas por el fabricante de la cámara. También se incluirán las funciones propias de *Opencv* que permiten

adaptar los mapas de color de la imagen a un formato válido en Qt. Es necesario incluir las funciones de inicialización de los objetos que forman la interfaz como los cursores y aquellas funciones que permiten el buen flujo del programa. La Figura 1.28 muestra las funciones creadas para inicializar los objetos implementados.

```
///Basler section

void scout_InitWidgets();
void scout_InitFromCamWindowConfiguration(CamWindowConfiguration_Scout &_camWinConf_Scout);
void scout_InitCamWindowConfigurationFromDialog(CamWindowConfiguration_Scout *_camWinConf_Scout);
Basler::Scout::PixelFormatEnums        scout_ComboPixelFormat_Data(int _idx);
Basler::Scout::PixelFormatEnums        scout_GetDialogPixelFormat();
```

Figura 1.28: Funciones de inicialización

Se deben definir las funciones que se ejecutan cuando se produce la pulsación de los botones existentes como se muestra en la Figura 1.29. Por ejemplo cuando se hace click el botón Clear Values se activará la función on_scout_ButtonClearValues_clicked() y así con el resto de botones propios.

```
//OnBasler Section
void on_scout_ButtonClearValues_clicked();
void on_scout_ButtonDefaultValues_clicked();
void on_scout_ButtonGetValuesFromCamera_clicked();
void on_scout_ButtonSetValuesToCamera_clicked();

void on_BaslerScout_ButtonTest_clicked();
```

Figura 1.29: Funciones que se ejecutan al pulsar los botones

– dialogsetupcam.cpp: en este archivo se ejecutan las funciones anteriores y otras necesarias para lanzar y gestionar esta ventana que permite trabajar con la cámara.

1.3.5.3 Funcionamiento

Una vez comentado como realizar la implementación se procede a explicar la funcionalidad del módulo implementado, la Figura 1.30 muestra la solución desarrollada en funcionamiento, que se integra en la clase DialogSetupCam.

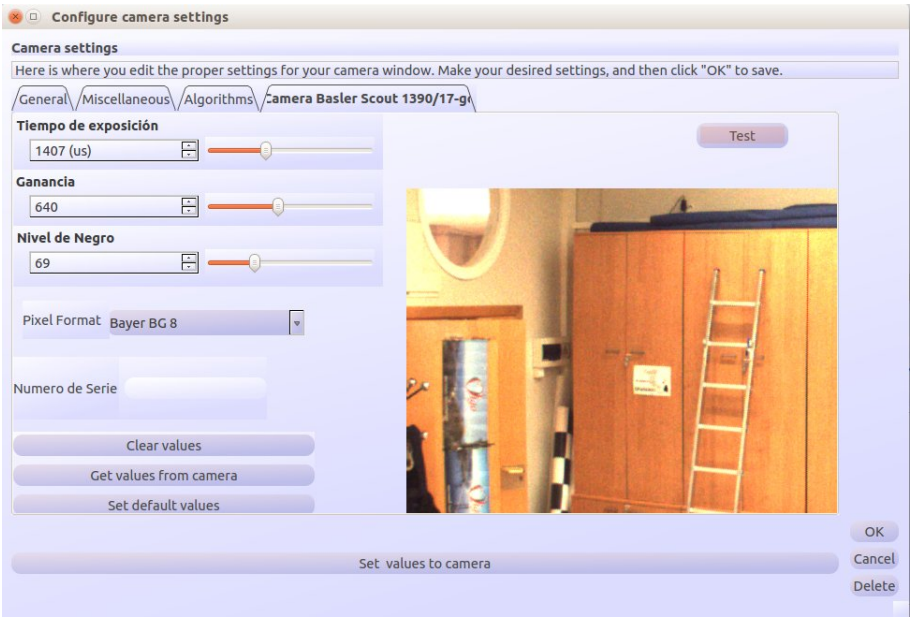

Figura 1.30: Solución Desarrollada. Módulo RGB en funcionamiento

Para abrir esta ventana es necesario clicar en CreateNew en la ventana de selección de configuración (MainWindow). Después hay que seleccionar el modelo de cámara Basler Scout sc1390/17gc y darle el nombre deseado. Una vez que se crea la configuración de cámara se tienen dos opciones en función del botón pulsado. Si se selecciona Open se abre la ventana principial de trabajo (CamWindow) para capturar imágenes. Si se pulsa el botón Edit se despliega el módulo implementado en la ventana ventana de configuración (DialogSetupCam). Se mantienen las pestañas genéricas a todos los modelos de cámara. Situandose en la pestaña de la cámara Basler Scout se muestra la interfaz de usuario desarrollada, y que tiene los siguientes elementos integrados.

- **Tiempo de exposición**: está formado por un grupo de objetos de Qt Creator llamados SliderIntTime (cursor) y SpinIntTime (casilla de texto editable) que permite leer el valor actual de trabajo o escribir de forma numérica el valor entero deseado entre un mínimo de 100 μs y un máximo de 4095 μs. Está relacionado con la función de la *API* de la cámara ExposureTime.SetValue() que permite escribir o leer el valor de este parámetro.

- **Ganancia**: igual que el tiempo de exposición los objetos que lo forman tienen de nombre SliderGain (cursor) y SpinGain (casilla de texto editable). Permite obtener el valor de ganancia de la cámara o configurar la cámara con el valor entero deseado. Llama a la función GainRaw.SetValue() de la *API* de la cámara. Ajustable entre un valor absoluto de 360 y 1023.

- **Nivel de Negro**: ídem que los anteriores nombrados como SliderBlackLevel y SpinBlackLevel. Se basa en el uso de la función BlackLevelRaw.SetValue() disponible en la *API*. El valor leido o escrito está contenido entre 0 y 255.

  Para que la sincronización sea correcta entre cada cursor y cada casilla de estas variables es necesario emplear señales. Estas señales permiten que un cambio en la posición del cursor, provoque un cambio

numérico en la casilla editable, y viceversa. Así se logra que el valor concuerde en los dos objetos para la configuración de la cámara o lectura de estas variables.

- **Numero de Serie**: en esta casilla se puede leer el número de serie de la cámara empleada cuando se pulsa el botón Get Default Values. Es un parámetro meramente informativo que se incluye para comprobar la funcionalidad de otro tipo de funciones disponibles en la *API* no relacionadas con la captura.
- **Pixel Format**: permite ajustar o leer el formato de captura de la cámara entre dos posibles enumerados que constituyen dos modos de funcionamiento de la cámara, nombrados como Mono8 que permite la captura en escala de grises y BayerBG8 con el que se pueden capturar imágenes en color. Para esta variable se emplea un desplegable denominado en Qt como Basler_ComboPixelFormat que brinda las dos opciones disponibles.

  Estos elementos conforman las posibilidades de configuración y se ilustran en la Figura 1.31

Figura 1.31: Parámetros configurables. Módulo RGB

- **Visor de imagen**: cuando se pulsa el botón Test se da la orden de visualizar la imagen en este objeto de de Qt Creator con los parámetros actuales de la cámara.
- **Botón Test**: este botón fue el primero que se implementó en la interfaz para poder comprobar la buena conexión de la cámara y visualizar la imagen configurada con los parámetros deseados en el visor situado debajo de él. Cuando se ejecuta este botón se llama a la función asociada que se encarga de crear el objeto de cámara. También se encarga de crear el canal entre la cámara y el adaptador de red del PC para transmitir la información. Lleva a cabo la inicialización del búffer y el manejo de las librerías de *Opencv* para adaptar la imagen a un formato compatible con Qt y así poder visualizar la imagen. Es necesario parar la grabación y adquisición, borrar el contenido de los búffers y demás variables después de la captura.

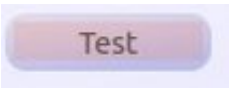

Figura 1.32: Botón Test. Módulo RGB

En la Figura 1.33 se muestran las variables especificas necesarias para adquirir una imagen, nótese que algunas son punteros.

```
//Variables a usar en captura basler button test

QImage m_image;
QImage m_qimgOriginal;
Camera_t *m_camera;
Camera_t::StreamGrabber_t *m_streamGrabber;
size_t m_imageSize;
GrabResult m_result;
uint8_t *m_pBuffer;
StreamBufferHandle m_hBuffer;
```

Figura 1.33: Variables necesarias para una correcta ejecución del botón Test

- **Botón Clear Values**: cuando se hace click sobre este botón se ejecuta la función asociada que se encarga de poner al mínimo todos los valores disponibles, tiempo de exposición, ganancia, nivel de negro y formato de píxel, así como de borrar el número de serie. Si se intenta capturar con estos valores puede no obtenerse una buena calidad de imagen.

- **Botón Get Values From Camera**: esta opción permite mostrar en los objetos de Qt asociados a cada variable los valores actuales de los parámetros de interés de la cámara, como el tiempo de exposición, ganancia, nivel de negro y formato de píxel.

- **Botón Set Default Values**: permite situar en los cursores y casillas editables los valores por defecto de la cámara que aseguran una buena captura.

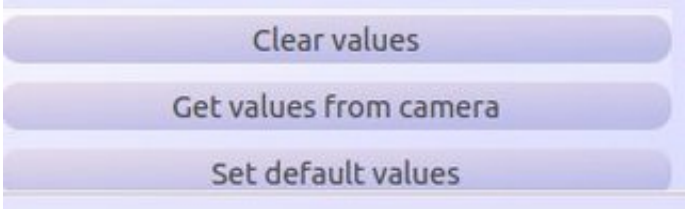

Figura 1.34: Botones implementados. Módulo RGB

- **Botón Set Values to camera**: este botón cada vez que es pulsado se encarga de configurar completamente la cámara con los parámetros actuales. Su función es mandar al objeto de cámara los valores enteros y enumerados el (tiempo de exposición, ganancia, nivel de negro y formato de píxel) que se han ajustado en cada uno de los campos de Qt propios de cada variable. La creación e implementación de este botón permite configurar cada parámetro con el valor deseado dentro de los límites. Debe de ser pulsado antes de realizar la captura (botón Test) para que la cámara adquiera con los valores seleccionados. Este botón se visualiza en la Figura 1.35

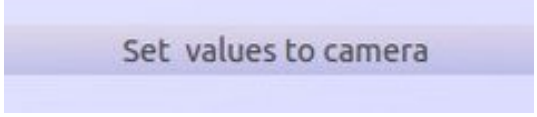

Figura 1.35: Botón Set Values To Camera. Módulo RGB

- **Botones OK, Cancel y Delete**: estos botones genéricos a todos los modelos de cámara ya estaban implementados en QOP. Se ha comprobado su correcto funcionamiento el módulo desarrollado con este modelo de cámara. El botón OK permite crear la nueva configuración de cámara, visualmente se muestra esta nueva configuración en la ventana de inicio del programa (MainWindow) debajo de la opción CreateNew. Si se pulsa el botón Cancel se desecha esta configuración volviendo a la ventana de inicio. El botón Delete permite borrar la configuración de cámara creada.

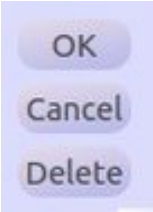

Figura 1.36: Botones nativos. Módulo RGB

1.3.5.4 Evaluación

La evaluación del software creado se realiza a continuación en la ventana de configuración. Mediante esta técnica se confirma su completa incorporación al sistema. En este apartado se avala experimentalmente la validez de la solución propuesta, con demostraciones de cada parámetro de manera individual. Para estas pruebas se procede a configurar la cámara con diferentes valores, siendo necesario hacer click en el botón Set Values to Camera y en el botón Test para obtener las imágenes expuestas. El modo de configuración del formato de píxel de la cámara es Bayer BG8.

- **Tiempo de exposición**: se muestran tres figuras obtenidas al configurar la cámara con diferentes tiempos de exposición

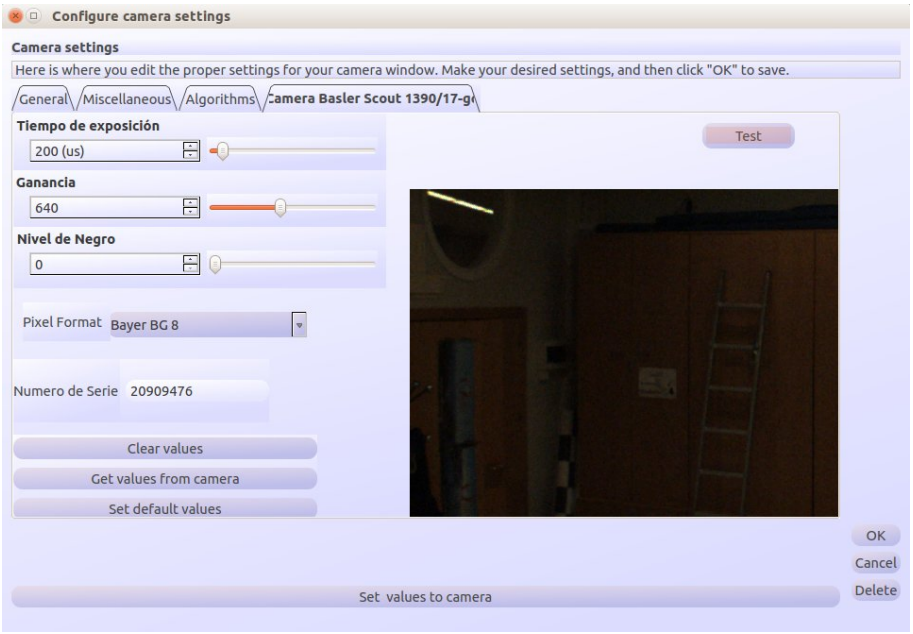

Figura 1.37: Imagen con tiempo de exposición bajo

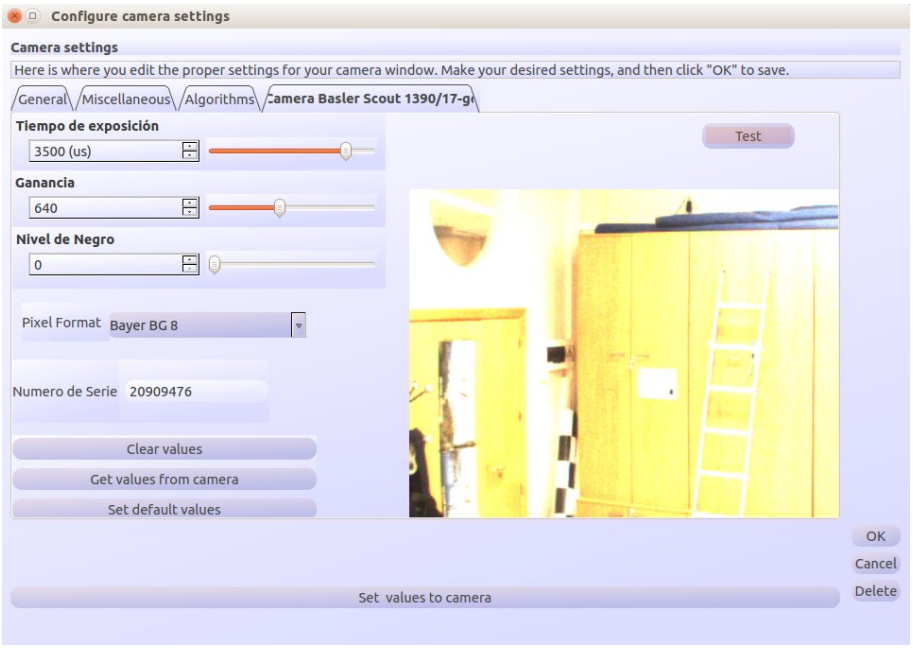

Figura 1.38: Imagen con tiempo de exposición alto

Figura 1.39: Imagen con tiempo de exposición óptimo

De las imágenes se extrae la influencia del tiempo de exposición en la luminosidad de la imagen con el resto de parámetros configurados a los valores por defecto. Si el tiempo de integración de la imagen es bajo la luz incide durante muy poco tiempo en el sensor, obteniendo una imagen con poca luz. Si este tiempo es demasiado alto la imagen refleja la mayor cantidad de luz que ha incidido en el sensor, pudiendo llegar a saturarse.

- **Ganancia**: de forma similar al apartado anterior se muestran tres imágenes asociadas a la configuración de la ganancia. El tiempo de integración y el nivel de negro están ajustados a su valor por defecto para apreciar únicamente la influencia de la ganancia.

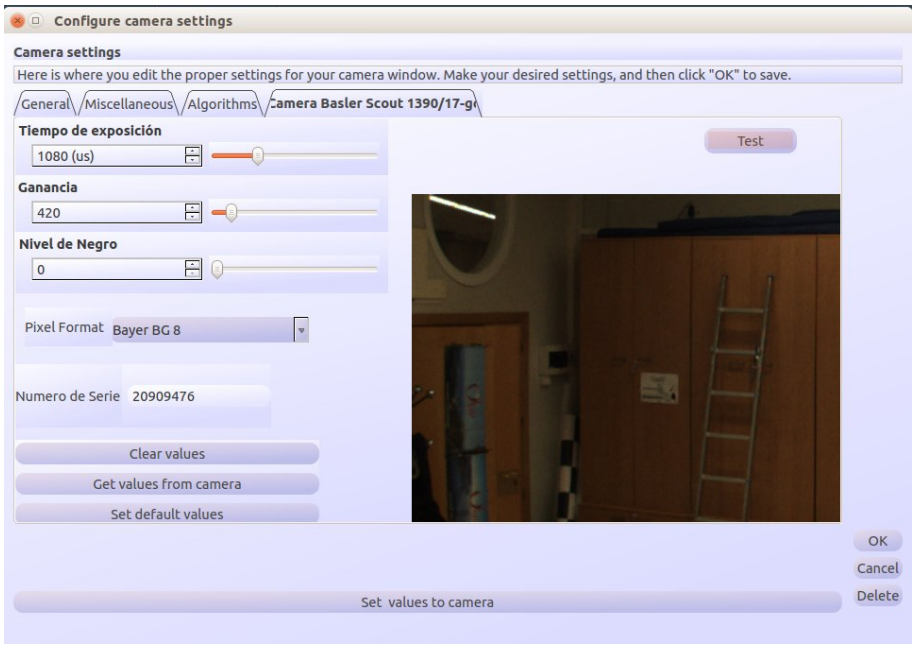

Figura 1.40: Imagen con ganacia baja

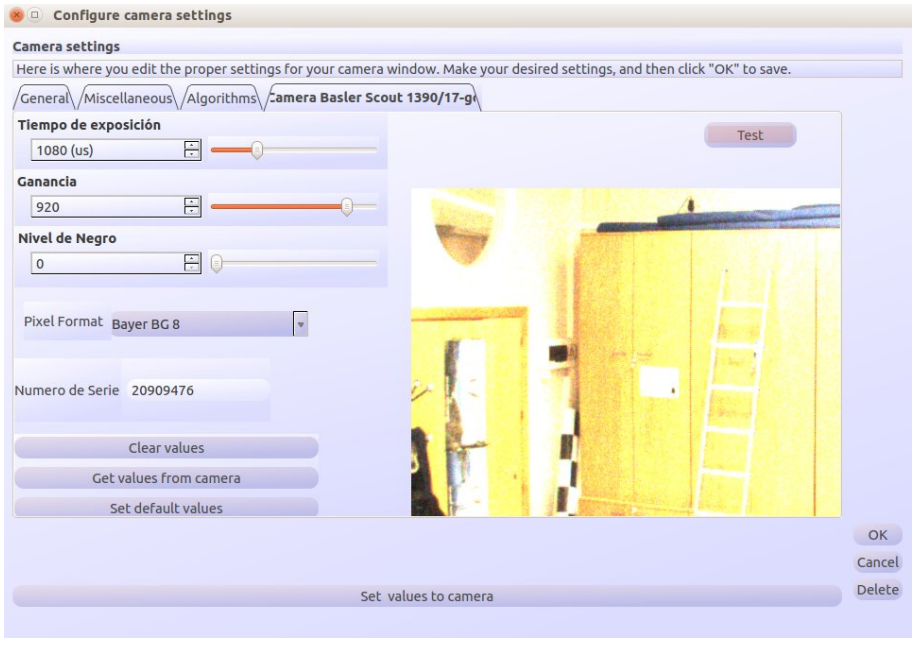

Figura 1.41: Imagen con ganacia elevada

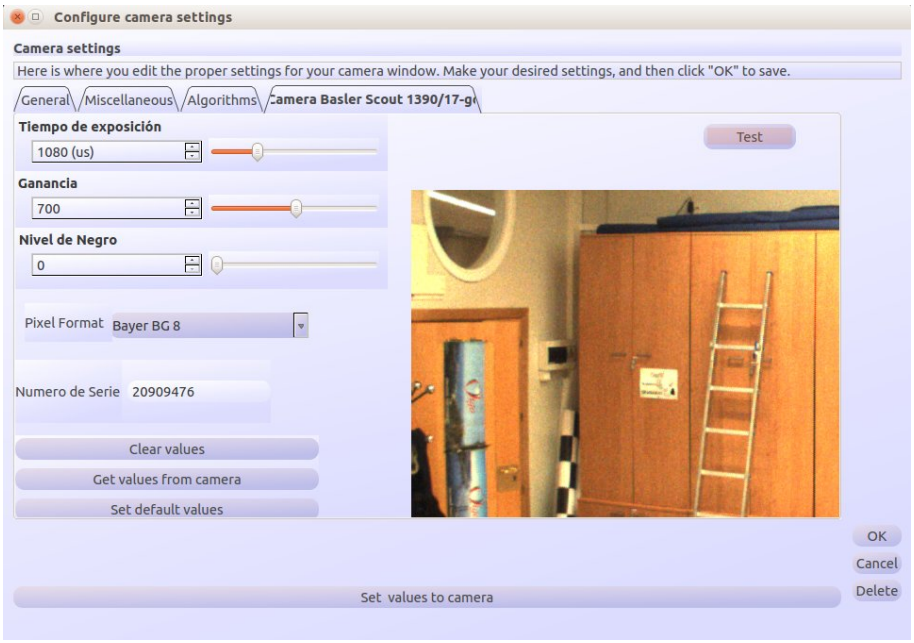

Figura 1.42: Imagen con ganacia óptima

Una baja ganancia provoca una imagen con escaso nivel de color, en la imagen con elevada ganancia se observa la saturación existente. La tercera imagen representa un valor óptimo. El valor mínimo disponible es igual a 360, y el máximo 1023. Estos valores límites son indicados en la documentación suministrada por el fabricante. El máximo es 1023 para un formato de píxel de 8bit como son los empleados en este proyecto. Si el formato de píxel fuese de 16bit el valor máximo de la ganancia se vería reducido a la mitad, es decir 511. El valor mínimo no se ve afectado.

- **Nivel de Negro**: en este apartado se expondrán dos imágenes debido a que la influencia de este parámetro oscila en un margen estrecho de valores. El resto de parámetros permanecen configurados a sus valores por defecto.

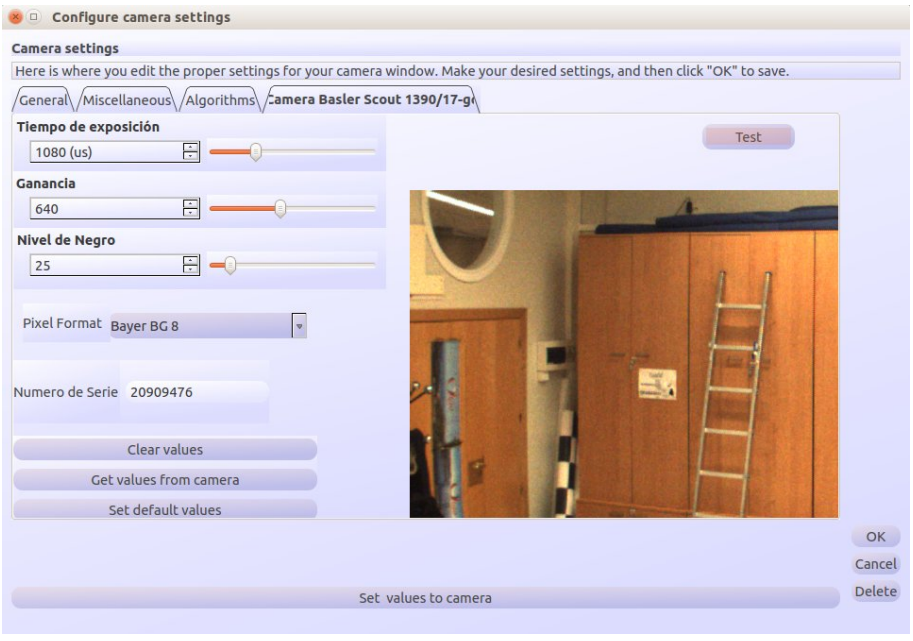

Figura 1.43: Imagen con nivel de negro bajo

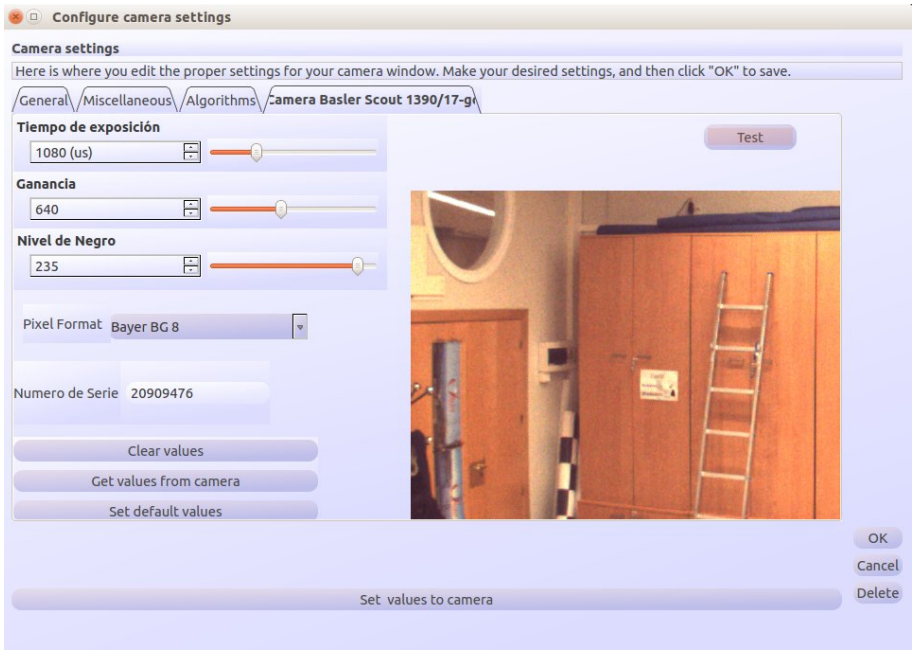

Figura 1.44: Imagen con nivel de negro bajo elevado

Incrementando el nivel de negro de la cámara se consigue un offset positivo en los valores digitales de salida de los píxeles. Si se decrementa este valor resulta un offset negativo a la salida. Este valor de negro puede ajustarse en un rango de 0 a 255 en el modelo de cámara usado.

- **Formato de Píxel**: respecto a este formato la validez del software hace referencia a la posibilidad de adquirir de la cámara imágenes en escala de grises e imágenes en color. Se muestran dos imágenes capturadas con los parámetros por defecto en las que se varía el formato de adquisición.

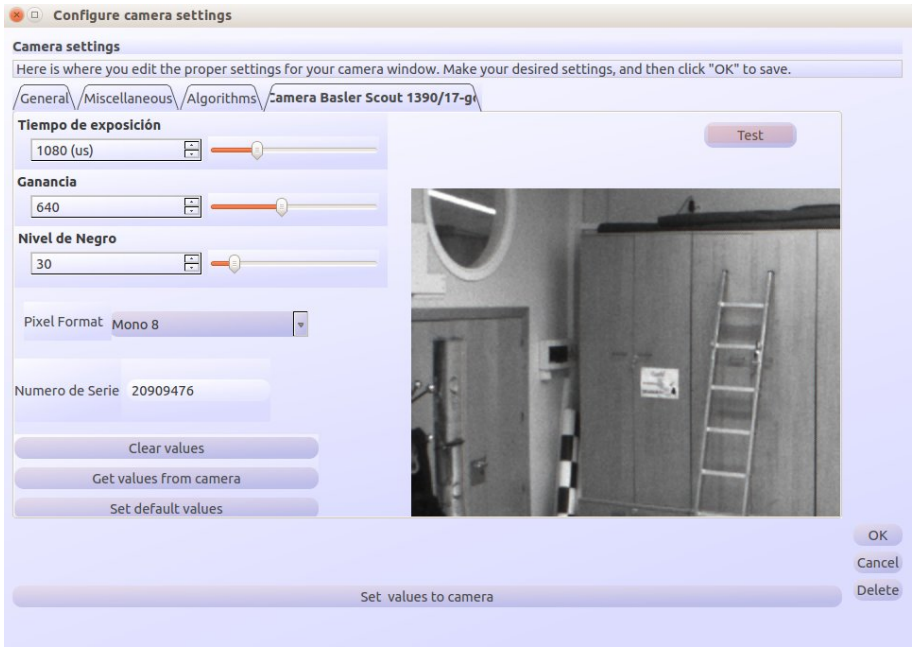

Figura 1.45: Imagen con formato Mono8

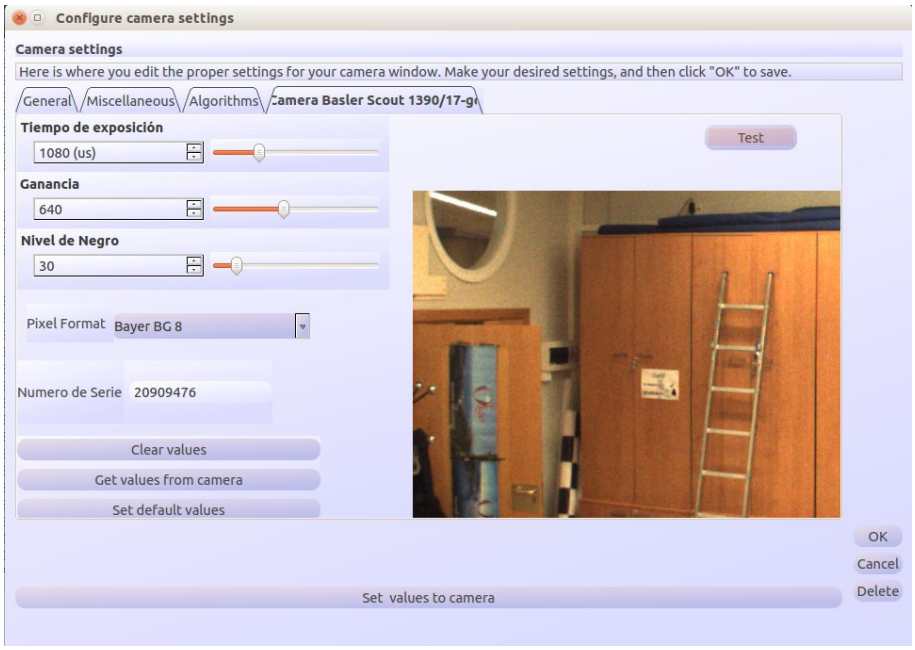

Figura 1.46: Imagen con formato BayerBG8

- **Ventana principal del trabajo**: una vez ajustados los parámetros de trabajo deseados en la configuración de la cámara se evalúa la implementación del módulo en la ventana principal del trabajo.

Figura 1.47: Imagen tomada al pulsar el botón Capture

La Figura 1.47 evalúa la funcionalidad del nuevo módulo en la ventana principal de trabajo (CamWindow). Cuando desde la ventana de configuración de la cámara (DialogSetupCam) se configura y pulsa el botón OK se guardan los ajustes de la cámara y se vuelve a la ventana de selección de configuración (MainWindow). Desde esta ventana se puede acceder a la interfaz principal (CamWindow) clicando el botón Open que abre la configuración seleccionada.

La demostración se materializa cuando se pulsa el botón Capture, el hilo que maneja la cámara Basler Scout se encarga de realizar la adquisición y visualización de la imagen con la configuración ya establecida. Se muestra la imagen adquirida en un visor no integrado originalmente. Esta decisión es debida a que los visores existentes en esta ventana no se adecuaban a las necesidades, bien por ser de carácter tridimensional o por tener un tamaño inadecuado. La funcionalidad del módulo es parcial en esta interfaz ya que algunas funciones como la carga y guardado de archivos no han sido implementadas.

1.4 Conclusiones

En esta sección se comentan las conclusiones asociadas a la realización del proyecto y los posibles trabajos futuros.

1.4.1 Conclusiones

Se ha diseñado un software modular que permite la configuración, captura y visualización de las imágenes proporcionadas por la cámara Basler Scout. Se han cumplido los objetivos generales y específicos del proyecto. El objetivo general era elevar la polivalencia de la interfaz *QOP* para que pudiese trabajar con una cámara *RGB*, objetivo alcanzado con la nueva implementación. Los objetivos específicos han sido cumplidos. Se ha realizado un minucioso estudio de la interfaz disponible, así como un estudio y lectura de la documentación proporcionada por el fabricante. Para lograr estos objetivos se ha dedicado mucho tiempo a aprender un nuevo lenguaje de programación como *C++*, competencia no adquirida en mi formación pero necesaria para entender el funcionamiento de la aplicación y de la cámara empleada. Estas tareas son imprescindibles para el desarollo del módulo *RGB*.

La solución desarrollada permite una gestión ordenada y limpia de los valores de trabajo más significativos de la cámara *RGB* usada, brindando al usuario final con menor conocimiento técnico la posibilidad de conectar la cámara y capturar una imagen clicando un botón.

El manejo del módulo y por tanto la programación asociada ha sido realizada de manera limpia e intuitiva para que el futuro desarrollador sea capaz de entender el funcionamiento del módulo en poco tiempo siempre que disponga de unos conocimientos mínimos de programación. Se ha diseñado, implementado y evaluado un nuevo módulo que incluye las librerias y variables de entorno necesarias para utilizar la *API* del fabricante. También se ha utilizado las librerías más extendidas en el procesamiento de imágenes (*OpenCV*), así como las librerías asociadas al diseño de interfaces multiplataforma (Qt). La librería de manejo de visión artificial e imágenes *OpenCV* se ha empleado principalmente para adaptar mediante mapas de color la imagen proporcionada por la cámara a un formato disponible en Qt, lo que permite visualizar correctamente la imagen. Este módulo es portable a los sistemas operativos como son Windows y Linux.

Todo el software creado en el presente trabajo ha sido evaluado mediante pruebas experimentales de adquisición de datos, configuración de cámara y visualización de imagen. Con la inserción de este módulo no se ven afectadas las funcionalidades de las cámaras de tiempo de vuelo, pues se ha creado cada elemento necesario para el correcto funcionamiento de la cámara Basler Scout. Se puede decir que convive en simbiosis con la aplicación proporcionándola mayor funcionalidad al incluir una cámara *RGB* que convive con las dos cámaras de tiempo de vuelo disponibles originalmente. Se puede crear y editar cualquier configuración de cámara entre los tres tipos disponibles. Durante la edición de las configuraciones, existe la posibilidad de verificar el funcionamiento de las cámara sin necesidad de guardarla ni proceder desde la ventana principal de trabajo. El idioma escogido es para la interfaz creada es el castellano.

Se ha documentado todo el proceso llevado a cabo desde el estudio teórico y experimental al diseño e implementación de esta interfaz.

Como conclusión personal puedo decir que este trabajo ha sido una tarea complicada. A lo largo de todo el proceso han surgido multitud de errores algunos sencillos de resolver y otros todo lo contrario, que han retrasado la ejecución del proyecto. Se puede decir que los errores me han acompañado en la instalación del sistema operativo, en el proceso de compilación y ejecución de *QOP*, en las primeras pruebas de funcionamiento de la cámara Basler y en el diseño e implementación de la solución desarrollada. Sin embargo, cada error solucionado ha sido útil para aprender nuevas técnicas. Estos serán abordados en el manual de usuario que figura como anexo.

Para finalizar a lo largo de todo el proyecto he adquirido bastantes conocimientos que serán de utilidad en el futuro. Entre ellos se puede mencionar el conocer como funciona física y digitalmente una cámara de estas características, así como ser capaz de desarrollar un software para trabajar con ella. Al haber empleado un compilador de texto como Latex [8] se descubren las múltiples ventajas de usar este tipo de programas. Por último la mayor satisfacción personal ha sido otorgada al cumplir el objetivo personal más importante, haber podido mejorar mi nivel de programación en *C/C++* que no era especialmente elevado cuando me embarqué en este proyecto.

1.4.2 Trabajos futuros

La realización de este trabajo ha supuesto una implementación de un módulo encargado de configurar, capturar y visualizar imágenes empleando una cámara *RGB* a la aplicación *QOP*. La línea de futuro más evidente es incorporar nuevos modelos de cámaras *RGB* ya que se tiene como referencia la cámara Basler empleada en este proyecto. Anteriormente sólo se tenían como referencia las cámaras *ToF* que tienen pocos elementos en común con la adquisición de imágenes en color.

Si se desea continuar con el desarrollo realizado de la cámara Basler y la implementación realizada existe la posibilidad de mejorar la interfaz de usuario creada, donde nuevos diseños, opciones o incluso nuevas formas de interacción usuario-máquina pueden ser aplicados. Para mejorar la solución desarrollada se abre un abanico de posibilidades, como sería implantar más funcionalidades al módulo *RGB* que en el presente proyecto no han sido desarrolladas, como ejecutar las funciones de guardado y cargado, funciones de calibración de la cámara y así como la incorporación de nuevos algoritmos de procesamiento de imágenes.

Capítulo 2

Pliego de Condiciones

En este aparatado se indica el software y hardware mínimo del que debe disponerse para una correcta ejecución de la aplicación.

2.1 Requisitos Hardware

En esta sección se expondrán los requisitos hardware mínimos así como los de referencia en los que se ha desarrollado este proyecto.

2.1.1 Requisitos mínimos

No existe un nivel de hardware mínimo, con equipos estándares lograremos ejecutar la aplicación, si bien es necesario que sea capaz de ejecutar las librerías con fluidez.

- Procesador: Intel de 32bits o AMD de 64 bits. La frecuencia de la CPU debe ser superior a 1GHz. Necesidad de doble núcleo. En Intel algunos procesadores Celeron cumplen estos requisitos y generalmente los modelos posteriores también.
- Memoria RAM: 2GB de memoria RAM, equipos con 1GB podrían no soportar la aplicación.
- Tarjeta gráfica: que soporte OpenGL. Debería disponer de un mínimo de 32 MB de memoria RAM dedicada. Las fabricadas a partir de 2007 suelen cumplir estas características.
- Pantalla: con resolución gráfica mínima de 1024x768 píxeles
- Conectividad: tarjeta de red para la conexión con la cámara Basler Scout y Mesa SR4000. Se recomienda que la tarjeta soporte *Jumbo Frames* para asegurar una mejor conexión con la cámara Basler. Puerto USB 2.0 necesario para la conexión de la cámara PMD.

2.1.2 Hardware de referencia

En esta sección se expondrá el hardware con el que se ha ejecutado este proyecto y que garantiza la buena ejecución de la interfaz.

- Portatil modelo Lenovo G50
- 4 GB de memoria RAM DDR3

- Procesador Intel® Core i3-4005U. Cuatro núcleos a 1.70GHz
- Trajeta gráfica AMD Radeon Graphics
- Pantalla LCD 15'6" con resolución 1920x1080 píxeles.
- Disco duro de 512GB
- 2 puertos USB 2.0, 1 puerto USB 3.0
- Tarjeta de red Realtek 8168
- Cámara MESA Imaging AG: modelo SwissRanger 4000.
- Cámara Pmdtechnologies GMBH: modelo CamBoard Nano.

2.1.3 Cámara RGB

Cámara de escaneo de área Basler Scout 1390/17-gc [2]. El módulo *RGB* de *QOP* está preparado para emplear esta cámara a través del interface Ethernet

2.2 Requisitos Software

A continuación se lista el software necesario:

- Sistema operativo Linux 32/64 bits. Se recomienda Linux Ubuntu 12.04 o superior. Enlace de descarga [6]
- Framework de desarrollo Qt 4.8 o superior hasta la versión 5.3. Qt es un Framework de desarrollo multiplataforma de aplicaciones, de código abierto. Su lenguaje nativo es C++. Actualmente es propiedad de Digia. Qt Creator deberá instalarse por separado pues hasta la versión 5.3 no se incluye en el mismo paquete. Enlace de descarga [7]
- Librería OpenCV 2.4.3 o superior. Es una biblioteca libre multiplataforma de visión artificial, basada en C/C++, que permite múltiples procesos sobre imágenes, principalmente 2D. Es propiedad de Intel. Enlace de descarga [9]
- Librería Pylon 2.0 o superior para el manejo de la cámara Basler, actualmente el fabricante distribuy la versión 4.0 [5]
- Librería VTK versión 5.8 o superior [10].
- Librería PCL 1.6 o superior [11].
- Drivers para el funcionamiento de las cámaras ToF provistos por cada fabricante.

  Los tres últimos requisitos de esta enumeración no han sido empleados en la solución *RGB* desarrollada, aunque sí son necesarios para el funcionamiento íntegro de la interfaz *QOP*.

2.2.1 Versiones de referencia

La base en la que se ha desarrollado el módulo es la siguiente:

- Ubuntu 12.04 LTS
- Qt 4.8.6 con Qt Creator 2.5.2
- OpenCV 2.4.9
- Basler Pylon 4.0
- VTK 5.8
- PCL 1.7

2.2.2 Versiones alternativas

Se pueden utilizar versiones alternativas de software que se muestran a continuación:

- Ubuntu 14.04 LTS
- Qt 5.3 que contiene a Qt Creator
- OpenCV 2.4.3
- Basler Pylon 2.0
- VTK 6.1
- PCL 1.7.2

Capítulo 3

Presupuesto

A continuación aparecen los distintos gastos agrupados en función del origen: costes de hardware, costes de software y costes de tiempo, concluyendo con un presupuesto total que engloba los anteriores.

3.1 Costes de Hardware

Engloba los distintos gastos relacionados con el equipo utilizado para el desarrollo del proyecto.

COSTES DE HARDWARE			
Concepto	*Precio Unitario Orientativo*	*Cantidad*	*Subtotal*
Ordenador portátil Lenovo G50	500,00 €	1	500,00 €
Cámara Basler Scout 1390/17gc	1.760,00 €	1	1.760,00 €
TOTAL	***2.260,00 €***		

Tabla 3.1: Tabla de costes de hardware

3.2 Costes de Software

Se incluyen los costes de los programas empleados.

COSTES DE SOFTWARE			
Concepto	*Precio Unitario Orientativo*	*Cantidad*	*Subtotal*
Sistema operativo Ubuntu 12.04	0,00 €	1	0,00 €
Framework Qt (uso no comercial)	0,00 €	1	0,00 €
Librería OpenCV	0,00 €	1	0,00 €
Librería Pylon Basler	0,00 €	1	0,00 €
Librería PCL	0,00 €	1	0,00 €
Librería VTK	0,00 €	1	0,00 €
Software de edición de documentos LibreOffice	0,00 €	1	0,00 €
TOTAL	***0,00 €***		

Tabla 3.2: Tabla de costes de software

3.3 Costes de Tiempo

En esta sección se muestran los costes derivados de las horas empleadas en la ejecución del proyecto.

COSTES DE TIEMPO			
Concepto	*Precio Unitario Orientativo*	*Cantidad*	*Subtotal*
Estudio previo y requisitos	60€/hora	100	6.000,00 €
Diseño, Implementación, Evaluación	60€/hora	150	9.000,00 €
Mecanografiado y maquetado	25€/hora	50	1.250,00 €
TOTAL	***16.250,00 €***		

Tabla 3.3: Tabla de costes de tiempo

3.4 Coste Total

COSTES TOTALES	
Concepto	*Subtotal*
Costes de Hardware	2.260,00 €
Costes de Software	0,00 €
Costes de Tiempo	16.250,00 €
TOTAL	***18.510,00 €***

Tabla 3.4: Tabla de coste total

El importe final del proyecto asciende a la cantidad de **DIECIOCHO MIL QUINIENTOS DIEZ EUROS**.

Alcalá de Henares, 14 de diciembre de 2015

Bibliografía

[1] R. L. Peña, "Diseño de un interfaz común para la captura de datos provenientes de cámaras de tiempo de vuelo," Proyecto Fin de Carrera, Escuela Politécnica Superior, Madrid, 2014.

[2] "Página del sensor rgb," http://www.baslerweb.com/en/products/area-scan-cameras/scout/sca1390-17gc.

[3] "Página del distribuidor basler," http://www.baslerweb.com/.

[4] "Página de la asociación emva," http://www.emva.org/.

[5] "Página de descarga de la librería pylon," http://www.baslerweb.com/en/products/software/.

[6] "Página del sistema operativo ubuntu," http://www.ubuntu.com/.

[7] "Página del entorno de diseño qt," http://qt-project.org/.

[8] "Página del procesador de textos latex," http://www.latex-project.org/.

[9] "Página de la libreria bidimensional," http://opencv.org/.

[10] "Página de la libreria tridimensional," http://www.vtk.org/.

[11] "Página de la librería pcl," http://pointclouds.org/.

Apéndice A

Manual de usuario

En este manual de usuario se explica el conjunto de pasos a seguir para guiar al usuario a una correcta ejecución y funcionamiento de la aplicación.

A.1 Instalación de prerrequisitos

En este apartado se enumeran una serie de requisitos que deben de ser cumplidos una vez cubiertas las necesidades hardware, para el buen funcionamiento de la interfaz. Debe instalarse todas las librerías requeridas por *QOP* y satisfacer las dependencias de cada una. Para comenzar con la instalación debe tenerse instalado el sistema operativo Ubuntu, ya sea la versión 12.04 LTS o la posterior 14.04 LTS. Se expone el listado de prerrequisitos, cuyo orden de instalación es el recomendado y se muestra a continuación:

1. `Framework Qt:` las versiones 4.8 a 5.2.1 garantizan el funcionamiento de la aplicación. Estas versiones requieren la instalación adicional de Qt Creator (versiones válidas 2.5.2 a 3.2.1). La última versión disponible y estable es la 5.3, que ya incluye Qt Creator. Si se instala esta versión la interfaz puede no ejecutarse con normalidad.

   Puede descargarse desde los repositorios como desde el auto-instalador que puede obtenerse aquí[7], o bien compilando e instalando los archivos fuentes. La ruta recomendada donde se deben instalar los paquetes es /usr/local

2. `Librerías Pylon:` estas librerías son proporcionadas por el fabricante de la cámara *RGB* Basler Scout. Pueden obtenerse en su web [5]. Las versiones disponibles abarcan desde la 2.0 a la 4.0, siendo esta última la de referencia en el presente proyecto. Una vez instaladas estas librerías es necesario configurar las variables de entorno para una correcta ejecución del módulo *RGB*. La ruta por defecto de instalación es /opt

3. `Aplicaciones Pylon:` una vez descargado e instalado el software proporcionado por el fabricante Basler, es necesaria una configuración inicial de la cámara *RGB*. Se emplearán dos aplicaciones que están incluidas en estos paquetes. La primera llamada IpConfigurator es una interfaz amigable que permite detectar el estado de conexión de la cámara, y de estar desconectada llevar a cabo su conexionado Ip empleando diferentes métodos. La segunda aplicación denominada PylonViewer requiere para su funcionamiento de una buena conexión de la cámara. Permite configurar todos los parámetros de la cámara, así como adquirir y visualizar imágenes. Es un prerrequisito indispensable configurar el dispositivo para pueda ser utilizado a través del módulo *RGB* implementado. En la siguiente sección se profundiza acerca de estas dos aplicaciones.

Estas interfaces de configuración se encuentran en el directorio de instalación de Pylon. En este caso están alojadas en /opt/pylon4/bin

4. `Librerías OpenCV:` las versiones válidas de esta librería son la 2.4.3 o versiones superiores. Si se instala desde el repositorio verificar la compatibilidad con Qt. Pueden descargarse desde esta web [9]. El directorio donde se han instalado estos paquetes en el presente proyecto es /usr/local/include

5. `Librería VTK:` la versión válida es la 5.8 o superior. No debe instalarse desde repositorios, ya que con dicho origen no trae el componente QVTKWidget, que es un componente fundamental de *QOP*. Una vez instaladas conviene comprobar la existencia de este componente. Estos archivos pueden descargarse aquí [10]. El directorio donde se instaló estos archivos es /usr/local/include

6. `Librería PCL:` versión recomendable 1.6 o superior. Puede descargarse aquí [11] o desde los repositorios. La ruta de instalación empleada es /usr/local/include

7. `Drivers:` de la cámara MESA SR4000 y de la cámara PMD Nano. Pueden descargarse e instalarse desde las páginas web de cada fabricante. Puede buscarse más información sobre los requerimientos aquí [1].

Los tres últimos requisitos de esta enumeración no han sido empleados para llevar a cabo el desarrollo del módulo *RGB*, aunque sí son necesarios para el funcionamiento íntegro de la interfaz *QOP*.

A.2 Preparación de la cámara RGB

En esta sección se ilustra cómo configurar los parámetros relacionados con la conectividad de la cámara *RGB*, así como aquellos parámetros relativos a la adquisición y visualización de imágenes. Esta labor es necesaria para el correcto funcionamiento del módulo *RGB* y se realiza para adaptar la cámara a las características del hardware disponible.

Para esta tarea se emplearán las dos aplicaciones proporcionadas por el fabricante denominadas IpConfigurator y PylonViewer, que se lanzarán desde la terminal de Ubuntu.

- **`IpConfigurator`**: esta aplicación identifica el estado de conexión de la cámara presente en la red. Se recomienda cerrar otros programas que puedan acceder a la cámara durante esta configuración. Si el dispositivo no está conectado, existen tres métodos que pueden ser utilizados por la cámara para obtener y asignar una dirección Ip.

  - Ip estática: el usuario puede asignar una dirección Ip fija.
  - DHCP: es un protocolo de red que devuelve una dirección Ip de un servidor DHCP.
  - Auto Ip (LLA): la cámara puede ajustar automáticamente su dirección con el PC al que está conectado.

  Cuando se ejecuta, IpConfigurator lista todas las cámaras conectadas. Se selecciona la cámara de la lista de dispositivos disponibles para su configuración. Si la columna de estado indica que la conexión no es correcta, el usuario debe elegir el método deseado para llevar a cabo la configuración Ip. Cuando se escoge el método Ip estática se debe introducir una dirección Ip válida y una máscara de subred apropiada. Generalmente la dirección de salida (Gateway) puede dejarse vacia. En el presente proyecto el método de conexión escogido es DHCP.

Independientemente de la configuración escogida se debe pulsar el botón Save que escribirá los cambios en la cámara. El dispositivo se reseteará y los cambios tendrán efecto. Si la conexión es satisfactoria será indicado en la columna de estado. La Figura A.1 muestra el correcto conexionado del dispositivo en esta aplicación.

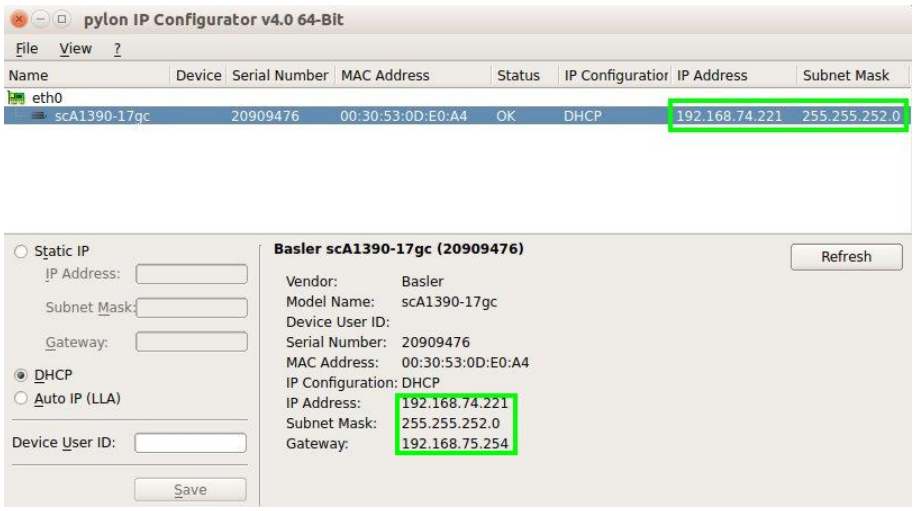

Figura A.1: IpConfigurator conexionado correcto DHCP Basler Scout

En la columna de estado se puede observar OK, lo que indica que la conexión es correcta. Para demostrar la validez de esta conexión se incluye una figura de la conexión de la tarjeta de red usada (eth0) a la que se conecta este dispositivo.

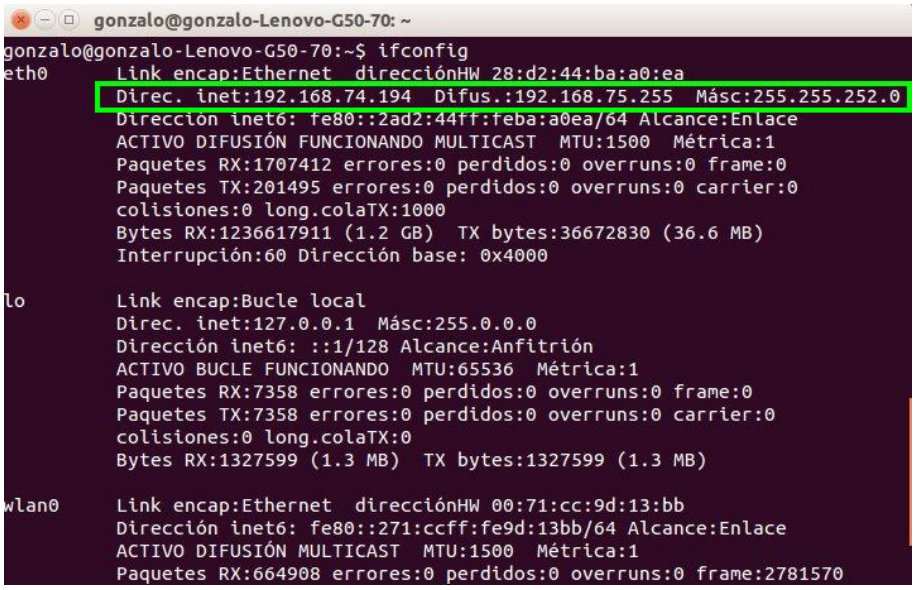

Figura A.2: Terminal conexionado correcto DHCP tarjeta de red

Las figuras ponen de manifiesto la importancia de que los tres primeros dígitos de la dirección Ip sean iguales para una correcta conexión. La tarjeta de red tiene asignado el valor 192.168.74.194 y la cámara la dirección 192.168.74.221. Así mismo otro factor clave, es la dirección de subred o máscara que debe coincidir plenamente, siendo en ambos casos 255.255.252.0

- **`PylonViewer`**: la ejecución de esta aplicación permite configurar todos los parámetros de la cámara, así como adquirir y visualizar imágenes. En esta interfaz se podrán configurar todos aquellos parámetros de conectividad que garanticen el correcto flujo de datos y paquetes entre el dispositivo y el adaptador de red presente en el PC. Para poder ejecutar esta aplicación es necesario haber asignado anteriormente una dirección Ip válida al dispositivo y conocerla, ya que al arrancar PylonViewer será necesario introducir la dirección para establecer la conexión como se muestra en la Figura A.3.

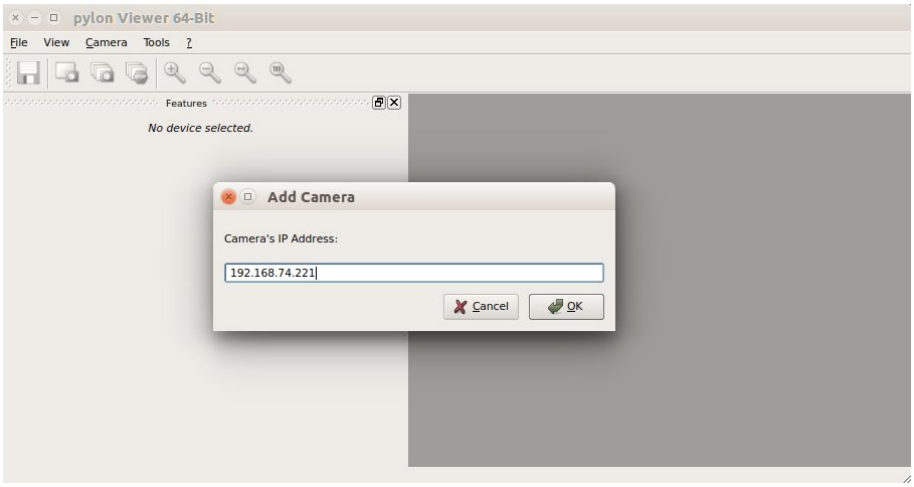

Figura A.3: PylonViewer dirección Ip

Si la dirección introducida es válida se despliega la siguiente ventana que permite configurar multitud de parámetros y opciones, así como capturar y visualizar imágenes.

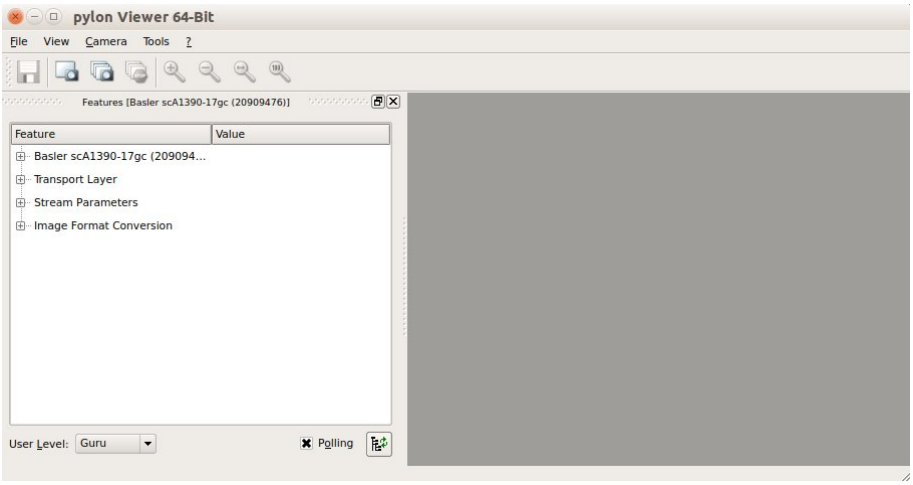

Figura A.4: Parámetros configurables PylonViewer

Como se extrae de la Figura A.4 exiten cuatro desplegables diferentes cuyos parámetros más relevantes se explican en detalle a continuación.

– Basler scA1390-17gc (20909476)

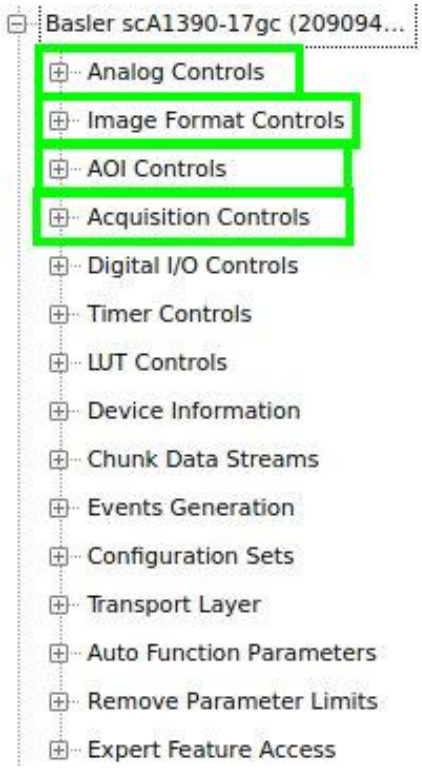

Figura A.5: Parámetros configurables Basler scA1390-17gc (20909476)

Cuando se despliega esta opción aparecen numerosas opciones de configuración, siendo las más importantes para la captura:

* Analog Controls: permite configurar los parámetros analógicos como la ganancia, el nivel de negro o el balance de blancos.

Figura A.6: Analog Controls

* Image Format Controls: se puede configurar el formato de píxel para adquirir imágenes en escala de grises o color. Es importante desactivar la opción Test Image para evitar visualizar una imagen de testeo.

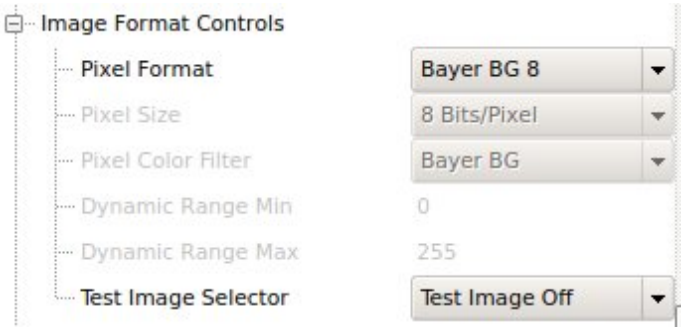

Figura A.7: Image Format Controls

* AIO Controls: se puede configurar la resolución de la imagen a partir de los dos selectores disponibles Width y Height. Los offset asociados permiten desplazar el área de interés de la captura.

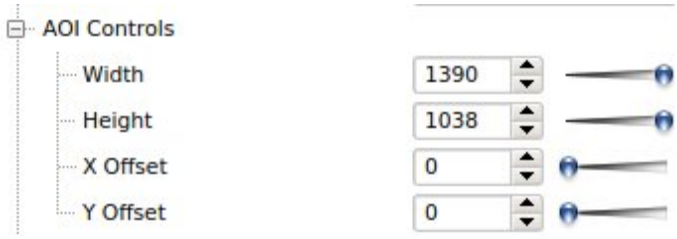

Figura A.8: AIO Controls

* Acquisition Controls: la opción más importante de este desplegable es el ajuste del tiempo de exposición. Se pueden activar y configurar los disparadores (Trigger).

Figura A.9: Acquisition Controls

– Transport Layer

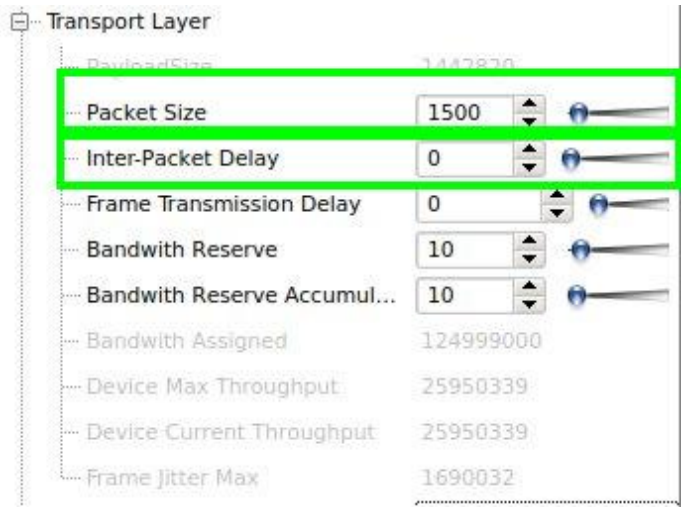

Figura A.10: Transport Layer

Los parámetros configurables en esta sección son aquellos relacionados con la transmisión de datos y conectividad de la cámara. Son de especial relevancia, ya que si el ajuste no es correcto no se podrá visualizar imágenes.

Los valores más dependientes del modelo de adaptador de red empleado son denominados como Packet Size e Inter-Packet Delay.

– Stream Parameters

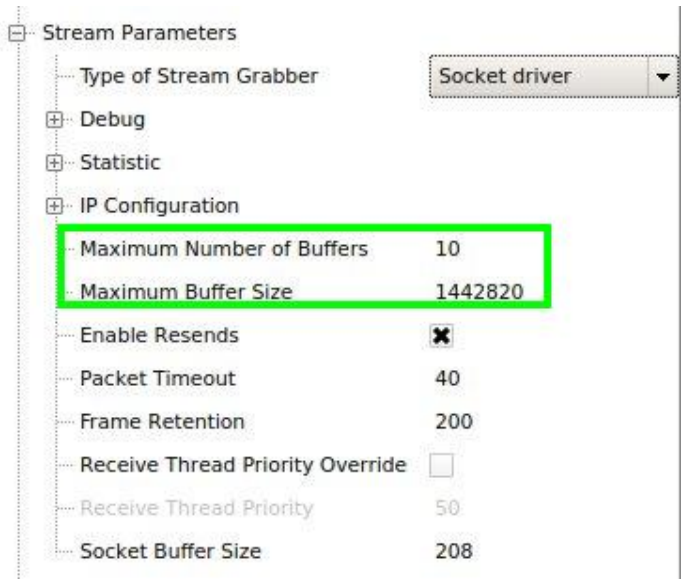

Figura A.11: Stream Parameters

En esta sección es posible configurar los parámetros relacionados con el número de buffers y el tamaño de estos para adecuarlos en función de las necesidades de adquisición. Algunas opciones de la dirección Ip pueden ser configuradas.

– Image Format Conversion: permite algunos ajustes de conversión si fuesen necesarios.

Una vez conectada la cámara y austados estos parámetros es recomendable realizar diferentes pruebas para comprobar el correcto funcionamiento del dispositivo. Si no se obtienen errores a lo largo del proceso puede darse por concluida la configuración inicial y continuar con la preparación de la aplicación.

A.3 Preparación de la aplicación

Con todos los prerrequisitos de la aplicación correctamente instalados se procede a abrir Qt y compilar el archivo `QOP.pro`, que es el proyecto en sí. Para llegar a este punto es necesario seguir las siguientes instrucciones:

1. Mover y copiar la carpeta QOP: esta carpeta contiene el proyecto, todo el código fuente y todos los archivos necesarios a la ubicación que se desee. Debe comprobarse que se dispone de permisos sobre la misma, ya que Qt necesita tener acceso a estos archivos.

2. Apertura del proyecto en Qt: desde Qt Creator, se procede a cargar el fichero `QOP.pro` que se encuentra dentro de la carpeta anteriormente copiada y ubicada. Automáticamente se regenerará el proyecto creando una configuración de compilación, una de debug (versión de depuración) y otra de release (versión liberada). Se recomienda crear nuevas configuraciones en el caso que Qt Creator lo pregunte. Así se minimizarán errores en la adaptación del código al nuevo entorno.

3. Adaptación del proyecto en Qt: para adaptar la aplicación al entorno se abrirá el archivo `QOP.pro`. Debe ser configurado en función de las librerías instaladas siempre que sean versiones válidas. Mención especial a las librerías descargadas e instaladas que deben encontrarse en los directorios que están definidos en Qt, de no ser así se obtendrán fallos y no se logrará una buena compilación y ejecución de la aplicación

4. Compilación del proyecto en Qt: una vez realizadas todas las tareas anteriores debe escogerse una de las configuraciones de versión (debug o release). A continuación se compilará el proyecto que generará el ejecutable asociado a la aplicación. Si todo va bien, Qt informará del éxito de la compilación, y tendremos la aplicación lista para su ejecución.

A.4 Ejecución de la aplicación

Una vez realizadas todas las tareas descritas anteriormente se puede comenzar la ejecución del proyecto. Existen dos formas para lanzar la aplicación:

1. Desde el entorno de desarrollo de Qt: se debe pulsar la flecha verde (botón play) situada en la barra vertical izquierda. Si no hay cambios en el proyecto, Qt no recompilará nada y procederá a ejecutar la aplicación. Si se ha modificado algún fichero Qt procederá a recompilar y regenerar la carpeta con los ejecutables.

2. Desde la terminal de Linux: Se puede ejecutar la aplicación si se conoce el directorio en donde se ha creado el ejecutable. Tecleando [DirectorioConEjecutableCreado]/QOP deberá comenzar la ejecución de la aplicación y se mostrarán las imágenes de la presentación.

A continuación se muestra la arquitectura de funcionamiento de la aplicación una vez han desaparecido las imágenes de presentación. La primera imagen que verá el usuario es la ventana de selección de configuración (MainWindow). Desde esta ventana puede crearse una nueva de configuración de cámara mediante el botón Create New o abrir una configuración de cámara ya creada pulsado el botón Open.

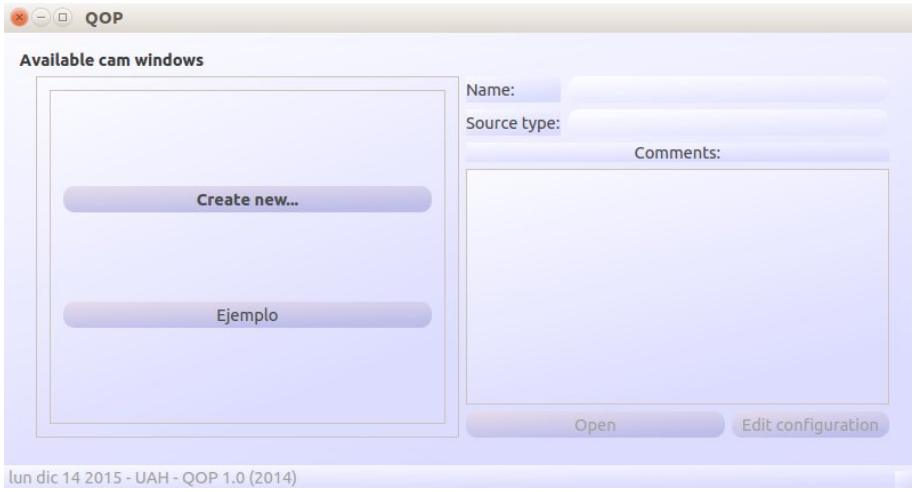

Figura A.12: Interfaz inicial. MainWindow

La primera vez que se pretenda emplear la cámara *RGB*, se debe crear una nueva configuración con el modelo y nombre solicitado en el diálogo que se despliega al pulsar Create New, en este caso Basler Scout sc1390-17/gc.

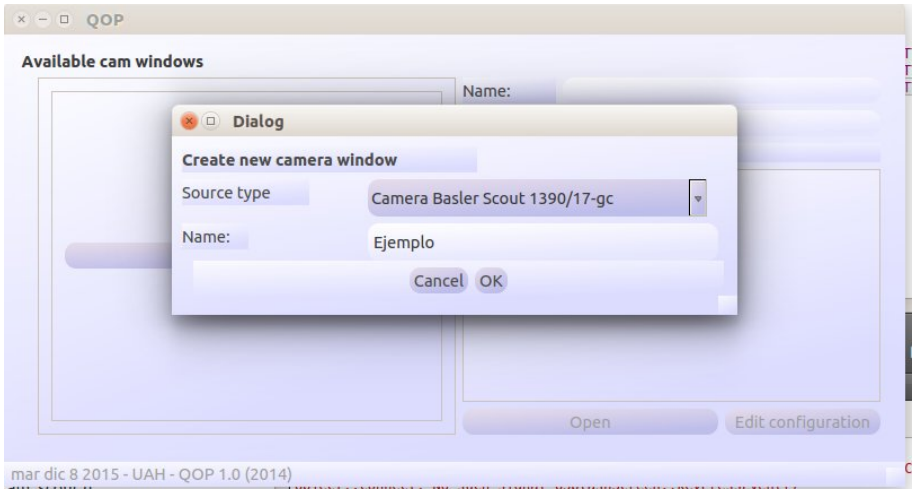

Figura A.13: Selección del modelo Basler Scout

Cuando se pulsa el botón OK se despliega la interfaz implementada de configuración del dispositivo *RGB*.

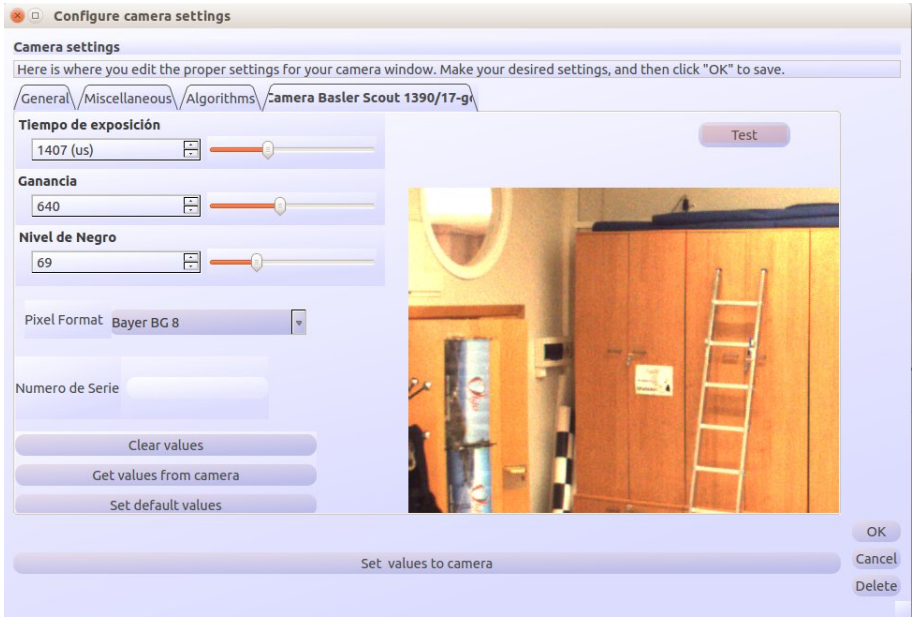

Figura A.14: Módulo RGB en funcionamiento

Cuando desde la ventana de configuración de la cámara (DialogSetupCam) se configura la cámara y se pulsa el botón Test, se guardan los ajustes del dispositivo y se muestra la imagen en el visor correspondiente. En esta interfaz pueden configurarse u obtenerse de la cámara los parámetros más relevantes de funcionamiento.

Si se hace click sobre el botón OK se vuelve a la ventana de selección de configuración (MainWindow). Desde esta ventana se puede acceder a la interfaz principal (CamWindow) pulsando el botón Open que abre la configuración seleccionada.

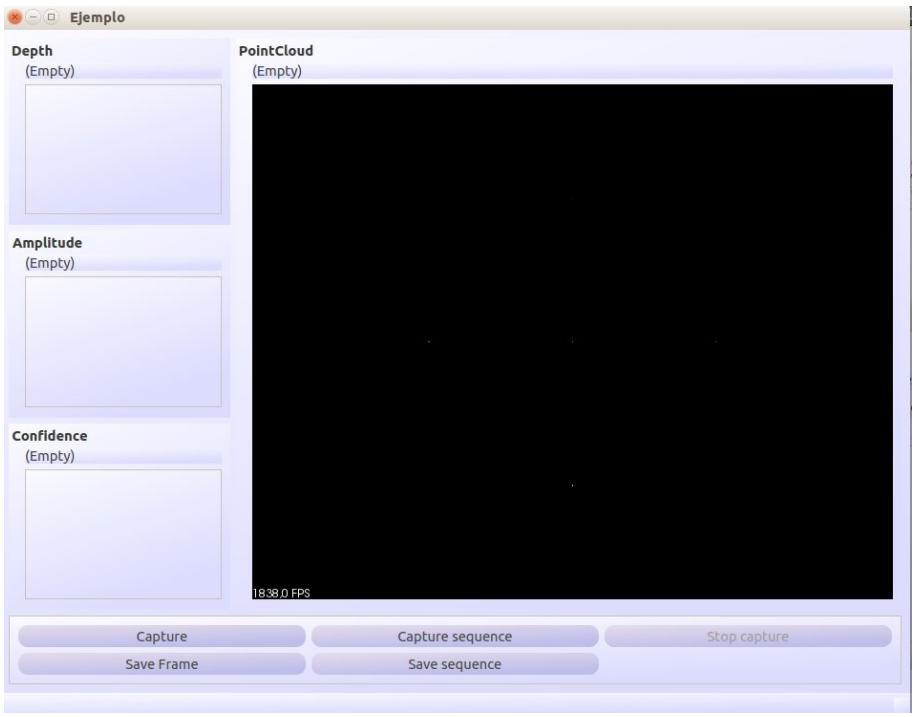

Figura A.15: Interfaz principal de trabajo. CamWindow

Cuando se pulsa el botón Capture, el hilo que maneja la cámara Basler Scout se encarga de realizar la adquisición y visualización de la imagen con la configuración ya establecida. Se muestra la imagen adquirida en un visor no integrado originalmente. Esta decisión es debida a que los visores existentes en esta ventana no se adecuaban a las necesidades, bien por ser de carácter tridimensional o por tener un tamaño inadecuado.

Figura A.16: Imagen tomada al pulsar el botón Capture

A.5 Solución de errores

El objetivo de esta sección es servir de soporte y ayuda al futuro usuario, dotándole de una colección de errores surgidos a lo largo del proceso y la solución llevada a cabo. Se mantiene el orden del manual. Para empezar se detallarán aquellos errores relacionados con la cámara y las aplicaciones asociadas, posteriormente los errores relacionados con la aplicación y que surgen en el proceso de compilación o ejecución. Se plantearán individualemente los posibles errores y su solución.

- Errores en la cámara y aplicaciones asociadas

  – No se lista ninguna cámara cuando se ejecuta IpConfigurator.

    `Solución`: el primer paso que se debe ejecutar es obviamente comprobar que la cámara está conectada físicamente. En ocasiones el firewall del adaptador de red no permite este reconocimiento, se recomienda desactivar el firewall y volver a lanzar la aplicación. La Figura A.17 muestra la interfaz cuando no hay ninguna cámara conectada.

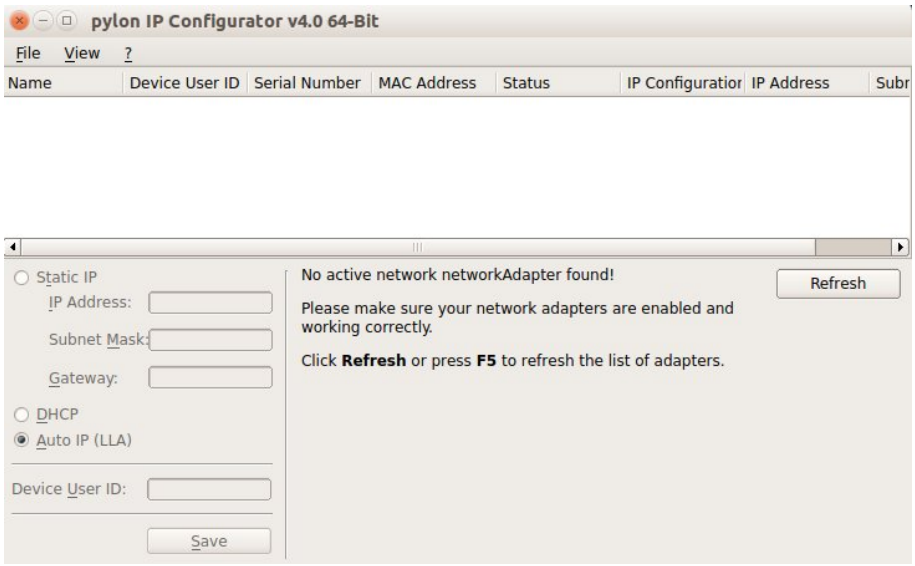

Figura A.17: IpConfigurator cámara sin conectar

– Se lista una cámara no conectada cuando se ejecuta IpConfigurator.

  `Solución`: si la cámara es reconocida por la aplicación y en la columna de estado aparece Not Reached significa que el dispositivo y el adaptador de red presentan direcciones Ip válidas, sin embargo la dirección de subred no coincide pudiendo ser este el origen del problema. Se recomienda igualar las direcciones de subred y refrescar el estado de la conexión.

– No se reconoce el dispositivo cuando ejecuto PylonViewer.

  `Solución`: comprobar el estado de conexión del dispositivo en IpConfigurator, si es correcto introducir la misma dirección Ip cuando PylonViewer lo solicite.

– En la adquisición y visualización de imágenes mediante PylonViewer la carga de la CPU es mayor de la esperada.

  `Solución`: aumentar el tamaño de los paquetes que se transfieren entre la cámara y el adaptador de red mediante el parámetro Packet Size, asegurarse también que el adaptador de red dispone de Jumbo Frames y que esta caraterística está habilitada.

– En la adquisición y visualización de imágenes mediante PylonViewer no se muestra nada y la terminal devuelve el código de error 0x81010014. Este error se produce para indicar que el PC recibió imágenes incompletas.

  `Solución`: comprobar la equidad del valor Packet Size con el MTU (Maxium Transfer Unit) del adaptador de red. Si *Jumbo Frames* está activado, el valor al que debe configurarse Packet Size y por tanto MTU es igual a 8192.

  Si esto no funciona probar a aumentar el tamaño de buffer para asegurar una adquisición estable. Hay que asegurarse que la aplicación tiene los permisos requeridos para ajustar la prioridad del hilo de ejecución a tiempo real.

– En la adquisición y visualización de imágenes mediante PylonViewer no se muestra nada y la terminal devuelve el código de error 0xe1000014. Este error se produce para indicar que el valor ajustado en Packet Size y MTU no es válido porque el adaptador de red no soporta Jumbo Frames. Se produce cuando la cantidad de datos de salida de la cámara exceden el ancho de banda del adaptador de red.

Solución: comprobar la equidad del valor Packet Size con el MTU (Maxium Transfer Unit) del adaptador de red. El parámetro Packet Size no debe ser mayor que MTU. Este valor debe ser igual a 1500 en ambos parámetros, lo que asegura que no se pierden paquetes de datos en la transmisión.

Si esto no soluciona el problema debe probarse a incrementar gradualmente el parámetro Inter Packet Delay que ralentiza la transmisión de imágenes por segundo, lo que permite descongestionar el canal transmisor. La Figura A.18 muestra el error obtenido por la terminal de comandos.

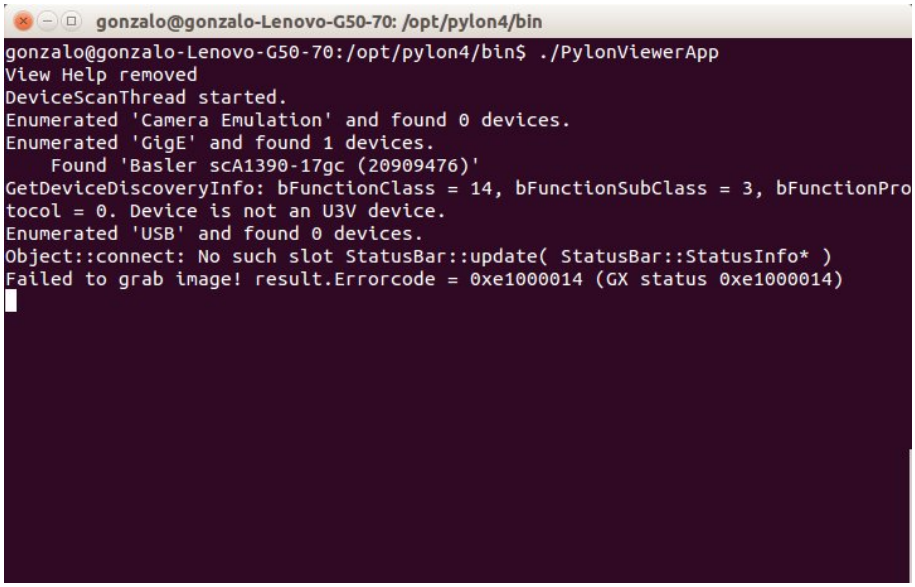

Figura A.18: Error 0xe1000014

– En la adquisición y visualización de imágenes mediante PylonViewer la terminal devuelve el error No such file or directory libpylonbase.so. Este error se debe a que el link dinámico no encuentra la librería libpylonbase.so instalada en /opt/pylon4

  Solución: ejecutar el comando sudo sbin/ldconfig -v en la terminal que permite linkar las librerías.

– En la adquisición y visualización de imágenes mediante PylonViewer la terminal devuelve cualquiera de los errores siguientes No such file or directory libXercesC_gcc40_v2_7_1.so/libXMLLoader_gcc40_v2_3.so o el error No absolute path for /opt/GenICam_v2_3/bin/Linux64_x64/GenApi/Generic/libXMLLoader_gcc40_v2_3.so

  Solución: este error está provocado por un inexistente o mal ajuste de las variables de entorno en el sistema. La Figura A.19 muestra la configuración necesaria de las variables de entorno en el archivo /home/usuario/.bashrc del equipo usado para que el ajuste de estas variables sea exitoso. Puede utilizarse el script provisto para esta labor pylon-setup-env.sh, que en el presente proyecto no ha sido funcional.

```
#source /bin/pylon-setup-env.sh /opt/pylon4

export PYLON_ROOT=/opt/pylon4
export GENICAM_ROOT_V2_3=/opt/pylon4/genicam
export GENICAM_ROOT=/opt/pylon4/genicam
#mkdir -p $HOME/genicam_xml_cache
export GENICAM_CACHE=/home/gonzalo/genicam_xml_cache
export GENICAM_CACHE_V2_3=/home/gonzalo/genicam_xml_cache


export LD_LIBRARY_PATH=${PYLON_ROOT}/lib64:${GENICAM_ROOT_V2_3}/bin/Linux64_x64:$
${GENICAM_ROOT_V2_3}/bin/Linux64_x64/GenApi/Generic:$LD_LIBRARY_PATH
```

Figura A.19: Configuración variables de entorno .bashrc

- Errores en la compilación y ejecución de la aplicación

  – La definición del proyecto contenida en `QOP.pro` no está bien adaptada al sistema.

    Normalmente se detecta cuando ciertas librerías o cabeceras de las mismas no se encuentran o dan errores.

    `Solución`: rectificar los directorios en dicho fichero para ajustar *QOP* al entorno. La otra opción es situar estos archivos en los directorios requeridos por *QOP*.

  – El código fuente no compila.

    Los síntomas suelen encontrarse en variables o definiciones incorrectas o ausentes, o llamadas a métodos inexistentes o con un perfil de parámetros erróneo.

    `Solución`: revisar todo lo anteriormente mencionado. Debe analizarse con qué librería se relaciona el código problemático, y refactorizar el código de *QOP* para adaptarlo a dicha versión de librería (nunca debe intentarse lo contrario: reprogramar la librería para adaptarla *QOP*).

  – En la adquisición y visualización de imágenes mediante el módulo *RGB* implementado, el panel de salida de Qt devuelve cualquiera de los errores siguientes, No such file or directory libXercesC_gcc40_v2_7_1.so/libXMLLoader_gcc40_v2_3.so o el error No absolute path for /opt/GenICam_v2_3/bin/Linux64_x64/GenApi/Generic/libXMLLoader_gcc40_v2_3.so

    `Solución`: Este error está provocado por un inexistente o mal ajuste de las variables de entorno en la aplicación *QOP*. Es necesario configurar el proyecto en Qt. La siguiente figura muestra qué variables de entorno son necesarias y dónde deben definirse en Qt para que la aplicación sea capaz de abrir y utilizar la cámara implementada.

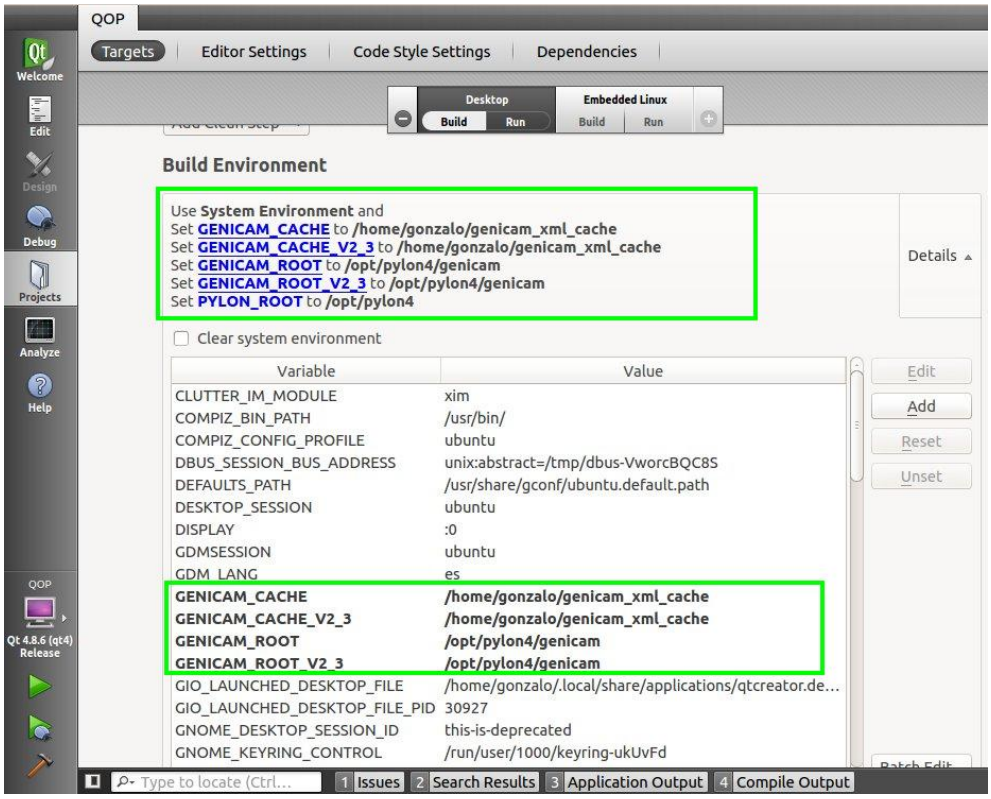

Figura A.20: Configuración variables de entorno en Qt para el proyecto QOP

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá