Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

**Trabajo Fin de Grado**

# “Desarrollo de una interfaz común para la configuración y adquisición de datos de cámaras de tiempo de vuelo”



**Autora:** María Higuera Pinillos

**Tutora:** Cristina Losada Gutiérrez

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

# Trabajo Fin de Grado

"Desarrollo de una interfaz común para la configuración y adquisición de datos de cámaras de tiempo de vuelo."

Autor: María Higuera Pinillos

Tutora: Cristina Losada Gutiérrez

TRIBUNAL:

Presidente: Carlos Andrés Luna Vázquez

Vocal 1º: Jesús Ureña Ureña

Vocal 2º: Cristina Losada Gutiérrez

FECHA: 4 de julio de 2018

# Contenido

Contenido ..... 5
Lista de figuras ..... 7
Resumen ..... 9
Summary ..... 11
Palabras clave ..... 13
Resumen extendido ..... 15
1 Introducción ..... 17
1.1 Objetivos ..... 18
1.2 Organización de la memoria ..... 19
2 Fundamentos teóricos ..... 21
2.1 Principio de funcionamiento de las Cámaras de Tiempo de Vuelo ..... 21
2.2 Fuentes de error ..... 24
2.2.1 Errores sistemáticos ..... 24
2.2.2 Errores aleatorios ..... 25
2.3 Rango de no ambigüedad ..... 27
2.4 Precisión y repetitividad ..... 28
2.5 Información de salida de las cámaras de tiempo de vuelo ..... 28
2.5.1 Características de la cámara utilizada ..... 31
2.6 Funcionamiento ..... 34
2.7 Aspectos adicionales sobre el manejo de la cámara ToF ..... 40
3 Desarrollo ..... 43
3.1 Introducción ..... 43
3.2 Lenguaje de programación y bibliotecas ..... 45
3.3 Interfaz de programación de aplicaciones desarrollada ..... 46
4 Conclusiones y trabajos futuros ..... 53
4.1 Conclusiones ..... 53
4.2 Trabajos futuros ..... 54
5 Pliego de condiciones ..... 55
5.1 Requisitos de Hardware ..... 55
5.1.1 Requisitos mínimos ..... 55
5.1.2 Hardware de referencia ..... 55
5.1.3 Cámara de Tiempo en Vuelo ..... 56
5.2 Requisitos de Software ..... 56

5.2.1 Versión de referencia de software ..... 56
6 Presupuesto ..... 57
6.1.1 Costes de equipamiento hardware ..... 57
6.1.2 Costes de equipamiento software ..... 57
6.1.3 Costes de tiempo empleado ..... 58
6.1.4 Coste total del presupuesto de ejecución material ..... 58
7 Manual de usuario ..... 59
7.1 Requisitos previos al uso de la aplicación ..... 59
7.2 Conexión de la cámara ..... 60
7.3 Puesta en marcha de la aplicación ..... 61
7.4 Funcionamiento de la aplicación ..... 66
7.5 Almacenamiento de imágenes ..... 69
Bibliografía ..... 71

# Lista de figuras

Figura 1. Principio de funcionamiento tiempo de vuelo ..... 15
Figura 2. Esquema del principio de funcionamiento de una cámara ToF ..... 21
Figura 3. Distribución de la carga en capacidades idénticas ..... 22
Figura 4. Medida de la distancia de tiempo de vuelo mediante modulación pulsada [10] ..... 23
Figura 5. Error “multicamino”. Se observa medida distorsionada [6] ..... 26
Figura 6. Reflexiones multiples (multicamino) ..... 26
Figura 7. Rango de no ambigüedad ..... 27
Figura 8. "Traslado" de objetos hacia el rango de no ambigüedad ..... 28
Figura 9. Representación de los resultados obtenidos a través de una cámara de tiempo de vuelo ..... 29
Figura 10. Imágenes de profundidad (Izq. Monocromática, Dcha. Coloreada en RGB) ..... 30
Figura 11. Representación del eje Z ..... 30
Figura 12. Imagen de intensidad ..... 31
Figura 13. Imagen de confianza ..... 31
Figura 14. Cámara Basler ToF ..... 32
Figura 15. Dimensiones y puntos de montaje de la cámara Basler ToF ..... 33
Figura 16. Descripción frontal ..... 33
Figura 17. Alimentación y conexiones ..... 34
Figura 18. Aplicación proporcionada por el fabricante, PylonViewerApp ..... 35
Figura 19. Origen de coordenadas ROI ..... 36
Figura 20. Origen de coordenadas ..... 37
Figura 21. Formatos posibles de imagen ..... 38
Figura 22. Diagrama de flujo del código ..... 48
Figura 23. Diagrama de flujo de la función MainParameters( ) ..... 49
Figura 24. Diagrama de flujo de la función OpenCamera( ) ..... 50
Figura 25. Diagrama de flujo de la función InitiConfig( ) ..... 50
Figura 26. Diagrama de flujo de la función GrabAcquisition( ) ..... 51
Figura 27. Diagrama de flujo de la función OnImageGrabbed( ) ..... 51
Figura 28. Diagrama de flujo de la función Acquisition( ) ..... 51
Figura 29. Diagrama de flujo de la función Close( ) ..... 52
Figura 30. Conexionado de la cámara ..... 60
Figura 31. Pasos para la ejecución de la aplicación ..... 61
Figura 32. Ejemplo de uso de los parámetros de desplazamiento ..... 63
Figura 33. Izq. Rango elegido entre 0 y 13.320 m. Dcha. Rango elegido entre 0 y 3 m ..... 64
Figura 34. Esquema de funcionamiento ..... 65

# Resumen

El objetivo de este trabajo es el desarrollo de una interfaz de programación de aplicaciones (API) común, que permita la configuración y adquisición de información proveniente de diferentes sensores de profundidad, sin necesidad de conocer las funciones propias de la cámara.

Las cámaras de tiempo de vuelo (ToF) permiten obtener información de profundidad del entorno con una precisión elevada, analizando los cambios que sufre una señal luminosa emitida cuando esta se refleja en los diferentes objetos. Sin embargo, cada fabricante proporciona sus propias funciones y aplicaciones. La API desarrollada en este trabajo permite el acceso a la configuración de la cámara y adquisición de la información sin necesidad de conocer las librerías del fabricante, permitiendo, además, incorporar nuevas cámaras de forma sencilla.

# Summary

The aim of this final degree thesis is the development of a common programming interface (API), which allows the image acquisition from different depth cameras, without the need to know the functions of the camera.

Time of flight cameras (ToF) allow to obtain depth information of the environment with a high precision, analyzing the changes suffered by an infrared signal emitted when the light is reflected from different objects. However, each manufacturer provides its own functions and applications. The API developed in this assignment allows access to the configuration of the camera and acquisition of information without the need to know the manufacturer's libraries functions, also allowing the incorporation of new cameras in a simple way.

# Palabras clave

Cámaras de Tiempo de Vuelo (ToF), Interfaz de programación de aplicaciones, C/C++, Ubuntu

# Resumen extendido

Las cámaras ToF emiten una señal LED infrarroja modulada que rebota en los objetos de una determinada escena. La señal reflejada por los objetos es captada por el sensor. Mediante correlación de la señal emitida y la luz reflectada se obtiene la diferencia de fase que nos cuantifica la profundidad existente entre el sensor y los objetos que se encuentran en su campo de visión.

Recopilando toda la información obtenida sobre la distancia a los objetos, se obtiene una imagen de profundidad. En esta imagen cada uno de los píxeles tiene un valor añadido de distancia, junto con información extra sobre la intensidad, fiabilidad de la medida y la posición según unos ejes de coordenadas determinados.

Estos sensores también son conocidos como sensores 2.5D [1]. Estas cámaras crean un mapa de información dónde únicamente queda reflejada la información de la superficie visible de los objetos situados enfrente de la cámara.

El objetivo de este trabajo es englobar en una única interfaz las funciones necesarias para trabajar con la cámara ToF, es decir, configurar esta y posteriormente mostrar y guardar las imágenes.

El trabajo concluye con la realización de las pertinentes pruebas sobre la cámara Basler ToF.

Conocido el principio de funcionamiento de las cámaras ToF, Figura 1, se implementan funciones para el acceso a la cámara. Estas funciones permiten la configuración y la adquisición de datos de forma sencilla y transparente para el usuario. De esta manera, se reducen las limitaciones existentes en el uso de estas cámaras. Además, se facilita el uso de varias cámaras Basler y resulta posible extender los algoritmos de adquisición de datos obtenidos a través de ellas.

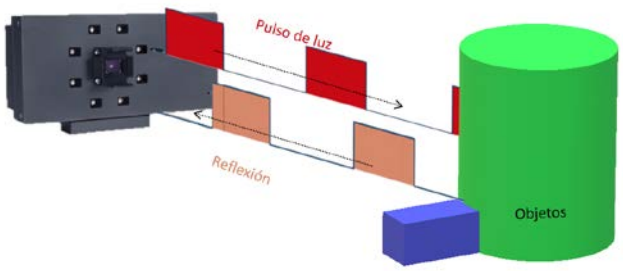



*Figura 1. Principio de funcionamiento tiempo de vuelo*

Este tipo de sensores complementan la visión por computador que tanta importancia ha adquirido en el ámbito de la electrónica. Dentro de esta área, estas cámaras nos aportan alta precisión y velocidad, condiciones necesarias en cualquier dispositivo electrónico.

La robótica móvil, la detección de personas, así como el escaneado de objetos 3D son las principales aplicaciones en las cuales estas cámaras resultan de gran utilidad [3].

Entre las ventajas de estos sensores se encuentra la posibilidad de recuperar información de distancias en la escena utilizando un único sensor [2], así como la capacidad de obtener datos de profundidad sin invadir la privacidad de las personas que aparecen en ella, debido a que no es posible reconocer a dichas personas.

Por otro lado, estas cámaras presentan dificultades a la hora de adquirir las imágenes y gestionar la configuración. Cada fabricante posee su propia interfaz de acceso a las cámaras. Esto hace que se requiera un estudio exhaustivo de la documentación proporcionada por los fabricantes, así como conocimientos de programación. Es por lo que el presente trabajo fin de grado (TFG) crea una interfaz de programación de aplicaciones (API) común para la cámara Basler ToF [10]. Mediante esta aplicación, el uso de la cámara resulta ser una tarea sencilla para cualquier usuario.

Existen otras desventajas en el uso de estas cámaras como pueden ser:

- La luz ambiental adicional a la de los propios LEDs de la cámara interfiere en la medida. La luz que es reflejada una única vez es la que proporciona una medida más exacta de la profundidad.
- La temperatura afecta cuando esta no es estable.
- Se debe de tener cuenta el color de los objetos. Los objetos más claros proporcionan una medida de la distancia más precisa.

Mediante numerosas pruebas, se corrobora que el trabajo funciona correctamente, pudiéndose configurar la cámara para la obtención de los tres formatos de imágenes posibles, (Distancia, intensidad y confidencialidad), siendo posible la grabación continua o la toma de una única imagen. También se ha comprobado que las imágenes son mostradas según prefiere el usuario y que el almacenamiento es correcto.

# 1 Introducción

La visión artificial incluye los métodos existentes para adquirir, analizar y procesar imágenes del entorno con el objetivo de obtener información numérica que puede ser procesada por un ordenador.

De la misma manera que nuestros ojos y cerebro comprenden el mundo que nos rodea, el computador debe de ser capaz de percibir las imágenes y actuar en función de lo que se desee. Esto es posible gracias a diferentes disciplinas científicas como la geometría, la estadística o bien la física.

Mediante la adquisición de secuencias de imágenes, video o datos procedentes de un escáner se pueden detectar y reconocer objetos, reconstruir escenas o restaurar imágenes [3].

La visión digital más sencilla es la captación de imágenes cuya información son los diversos tonos cromáticos. En el presente trabajo la cámara empleada es dotada de otra característica muy importante en visión artificial. Se obtiene en tiempo real información de profundidad. Estas cámaras son denominadas cámaras de Tiempo de Vuelo, aunque comúnmente se utilizan las siglas ToF, (*Time of Flight*). Mediante una única cámara es posible obtener la distancia entre el objetivo y la superficie visible de los objetos situados en la escena [2]. Estas cámaras proporcionan un mapa de distancias mediante la cuantificación del desfase producido sobre la señal emitida una vez que esta ha rebotado en los objetos de una escena. Resulta una aplicación ventajosa respecto al uso de múltiples cámaras convencionales pues la información se obtiene en tiempo real puesto que no hace falta realizar complejos algoritmos. Apenas depende de la iluminación ambiental y de la textura de los objetos. Además, los sensores ToF están dotados de gran precisión y velocidad lo que los hace versátiles y cómodos en infinidad de situaciones. A nivel económico estas cámaras tienen una buena relación calidad precio, encontrándose en el mercado cámaras ToF con buenas prestaciones a un precio asequible. Un claro ejemplo de esto es la cámara Kinect v2 [8] [9] cuyo precio de venta se sitúa en torno a los 200€.

El principio de tiempo de vuelo permite a estas cámaras medir distancias mediante el envío de pulsos de luz emitidos por uno, o varios diodos LED y midiendo de manera indirecta el tiempo que el pulso de luz tarda en reflejarse y ser recibido por el sensor óptico. Por lo que la distancia será mayor cuanto más tarde en reflejarse la luz y alcanzar el sensor y viceversa.

Aunque el uso de estos sensores resulta ser una tecnología cómoda, fidedigna y versátil, existen algunas limitaciones en su uso.

En primer lugar, cada fabricante proporciona una aplicación y unas librerías propias para su cámara, incrementando la dificultad de uso e instalación. Es por ello que se necesita tiempo de formación y conocimientos de programación previos para poder usar cada dispositivo.

Tal y como se verá en puntos posteriores, los resultados de las medidas están expuestos a errores sistemáticos que pueden ser corregidos mediante calibración y a errores aleatorios que producen un mayor problema a la hora de ser compensados.

Este trabajo surge con el objetivo de reducir el primero de los problemas expuestos, realizando una librería que permite configurar la cámara, adquirir, mostrar y almacenar las imágenes solicitadas por el usuario de forma sencilla, y sin necesidad de conocer las librerías del fabricante. Este trabajo se resume en una aplicación apta para personas ajenas a la programación de la cámara, pero con la necesidad de utilizar este tipo de dispositivos, es decir, mediante la introducción de una línea de parámetros, la cámara habrá quedado completamente lista para recibir la información que el usuario desea. Únicamente será necesaria una formación básica sobre la cámara y la información que se obtiene a través de esta.

## 1.1 Objetivos

El objetivo de este trabajo de fin de grado (TFG) es el desarrollo de un interfaz de programación de aplicaciones (API) común, que permita la configuración y adquisición de información de diferentes sensores de profundidad, sin necesidad de conocer las funciones propias de cada cámara.

El objetivo general se divide a su vez en los siguientes objetivos parciales que han sido llevados a cabo durante la realización del TFG:

- Adquisición de conocimientos relacionados con el principio de funcionamiento de las cámaras ToF.
- Manejo de las librerías y demostradores de adquisición y procesamiento de información con cámaras ToF.
- Implementación de funciones para el acceso a la cámara ToF, que permiten su configuración y la adquisición de datos de forma sencilla y transparente para el usuario. Esto reduce las limitaciones existentes en el uso de estas cámaras.
- Evaluación del sistema implementado para la cámara Basler ToF 21886650 [10].

Para alcanzar los objetivos descritos anteriormente, ha sido necesario llevar a cabo diferentes tareas, con una dedicación de 20 horas semanales, durante 4 meses, cuya descripción y temporización se incluyen a continuación:

Tarea 1. Revisión bibliográfica sobre funcionamiento y prestaciones de cámaras de tiempo de vuelo (2 semanas)

Tarea 2. Revisión de funciones y librerías existentes para las diferentes cámaras ToF disponibles (3 semanas)

Tarea 3. Definición de funciones a desarrollar en la interfaz común (2 semanas)

Tarea 4. Implementación de la API común para las cámaras disponibles (6 semanas).

Tarea 5. Evaluación del sistema implementado (2 semanas)

Tarea 6. Documentación del trabajo realizado (2 semanas)

## 1.2 Organización de la memoria

La memoria de este trabajo de fin de grado está compuesta por cuatro apartados.

- La primera parte incluye la introducción. En ella se expone el campo de trabajo en el cual se ha desarrollado esta propuesta, además, se indican los objetivos propuestos, así como las tareas realizadas para lograrlos.
- El segundo capítulo recoge lo principios de funcionamiento de las cámaras de tiempo de vuelo. También se explican las principales fuentes de error de la medida. De este modo, se cubren los conceptos teóricos en los que se ha basado este trabajo.
  Además, se incluye un capítulo con la descripción de la cámara utilizada y sus principales características.
- En el tercer apartado se realiza una descripción de la estructura de la interfaz desarrollada, así como del proceso de creación. Además, se introduce el lenguaje de programación y las librerías utilizadas para la realización de la aplicación.
- En cuarto y último lugar, se cierra la memoria con las conclusiones sobre la solución desarrollada y las posibles futuras líneas de trabajo.

# 2 Fundamentos teóricos

A lo largo de este capítulo se presentan los fundamentos teóricos necesarios para la realización de este TFG. En primer lugar, se describe el principio de funcionamiento de las cámaras de tiempo de vuelo, así como las principales fuentes de errores de la medida, tanto de errores sistemáticos, como no sistemáticos. Posteriormente se explica el rango de no ambigüedad de las cámaras ToF y la precisión que estas pueden alcanzar. También se introduce la información que es posible adquirir desde un sensor de tiempo de vuelo. Por último, se detallan las características propias de la cámara empleada, Basler ToF 21886650.

## 2.1 Principio de funcionamiento de las Cámaras de Tiempo de Vuelo

La Figura 2 muestra el principio de funcionamiento de estas cámaras. La onda infrarroja modulada que producen los LEDs se encuentra dibujada en rojo, una vez que es reflejada en el objeto, obtenemos la onda azul. El sensor detecta esta segunda onda y mediante la medida de la diferencia de fase entre ambas se puede calcular la distancia al objeto conociendo la velocidad de la luz en el vacío [6].

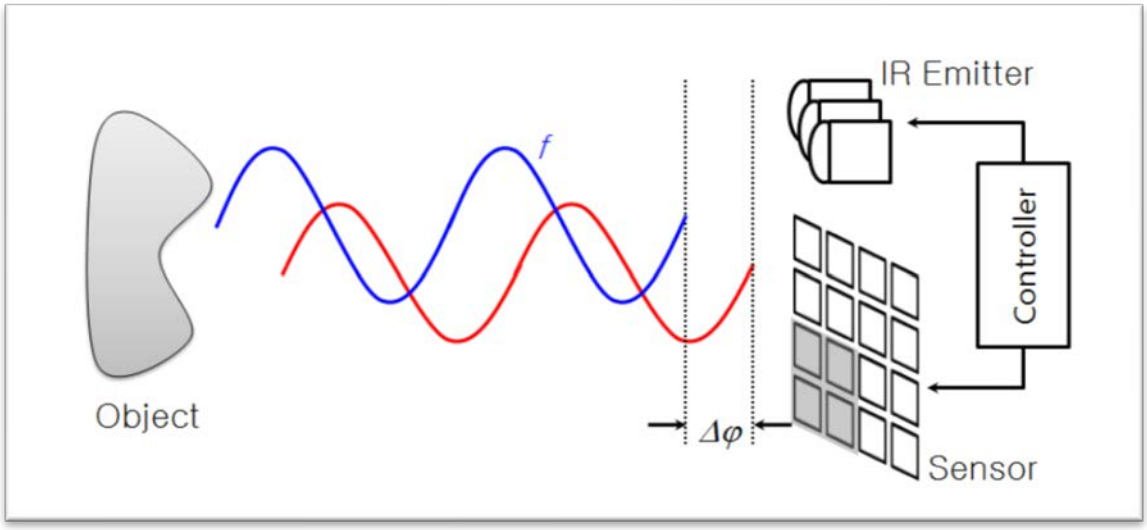



*Figura 2. Esquema del principio de funcionamiento de una cámara ToF*

Como puede verse en la Figura 3 , el conmutador asociado a cada pixel distribuye la carga foto-generada en 2 capacidades de almacenamiento idénticas. La diferencia entre estos dos paquetes de carga se lee y procesa.

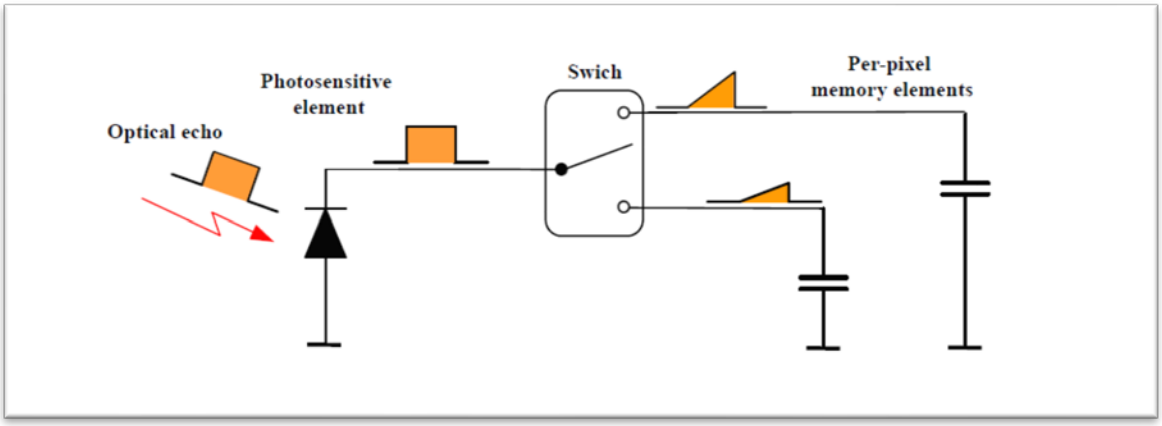



*Figura 3. Distribución de la carga en capacidades idénticas*

En este trabajo se utiliza una cámara de modulación pulsada, por lo que la diferencia de fase es calculada entre dos cargas eléctricas diferentes. Estas determinan la colección de electrones aceptados por el sensor.

La distancia correspondiente puede ser calculada usando la velocidad de la luz *c* en el vacío (aproximadamente $3x10^8$ m/s), y el tiempo que el pulso de luz está activo $t_P$ utilizando la ecuación 1:

$$d = \frac{c}{2} t_p \left(\frac{S1}{S1 + S0}\right)$$

*Ecuación 1. Cálculo de la distancia*

Donde:

S0 es la carga eléctrica acumulada durante la primera ventana de obturación S0

S1 es la carga eléctrica acumulada durante la segunda ventana de obturación S1

En función del tiempo que el pulso de luz esté activo la ambigüedad de la medida será más o menos favorable. Por lo que puede producirse un fenómeno denominado plegado de profundidad que es explicado con más detalle en la siguiente sección.

Los LEDs que posee la cámara emiten pulsos de luz modulada según la Figura 4. Una vez que estos pulsos han sido reflejados en un objeto son captados por una matriz de sensores que determinan la resolución de la futura imagen. Es decir, cada elemento de la matriz es un pixel del mapa de profundidad creado [10].

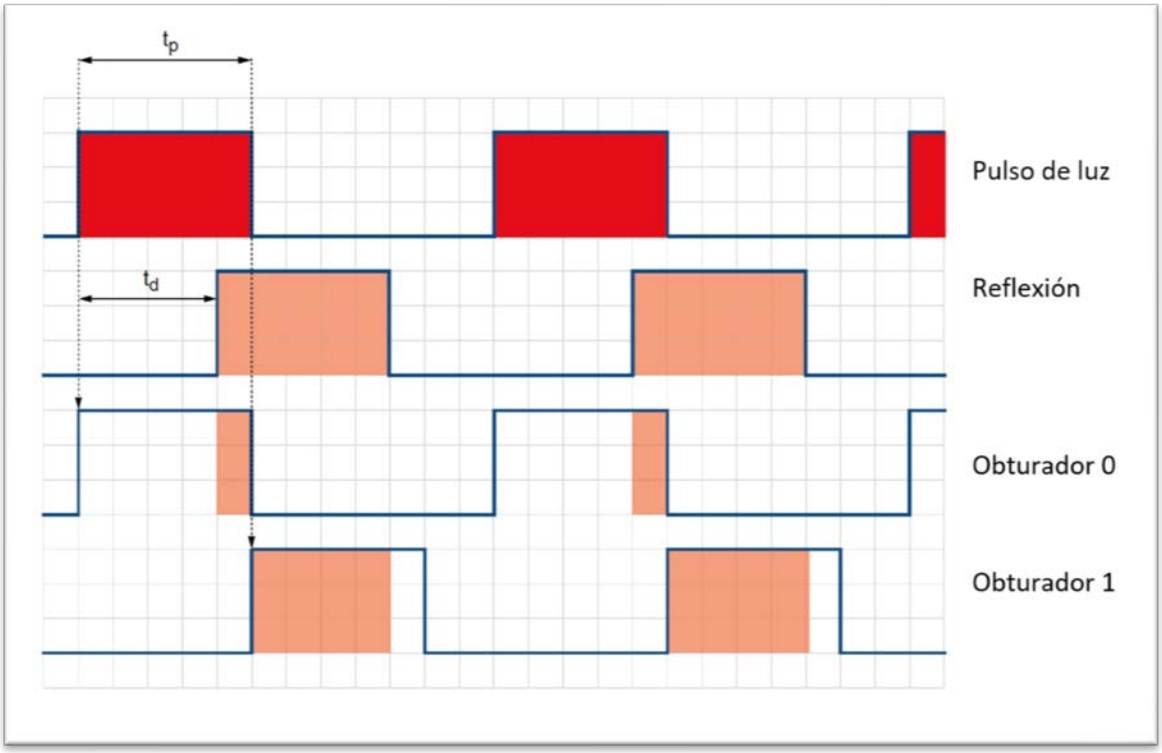



*Figura 4. Medida de la distancia de tiempo de vuelo mediante modulación pulsada [10]*

Para un posterior uso de los valores captados por la matriz es necesario hacer una conversión analógico-digital. Este dato digital recoge la distancia esférica medida entre el sensor y objeto situado en el campo de visión de la cámara.

El uso de la modulación pulsada tiene la ventaja de poder transmitir una gran cantidad de energía en un tiempo muy corto. Esto es beneficioso pues se puede reducir la influencia de la iluminación de fondo y alcanzar una alta relación de señal-ruido manteniendo un valor bajo de potencia óptica, esto es un factor importante para la seguridad ocular.

Este tipo de modulación reduce la demanda de una alta sensibilidad y relación señal/ruido del detector respecto a los sensores de modulación pulsada, lo que permite realizar medidas a larga distancia.

Por otro lado, se encuentra el problema de detectar el instante de llegada del haz de luz reflejado. Hay dos razones que lo justifican:

- El umbral óptico no es un valor fijo, este cambia con el fondo y la distancia del objeto. Es por ello que algunas de estas cámaras no funcionan correctamente en el exterior, por lo que conviene utilizarlas en espacios cerrados cuyo fondo no sature la imagen.

• Es complicado producir pulsos de luz muy cortos con tiempos rápidos de subida y bajada, esto es necesario para detectar de una manera precisa el impulso de luz recibido. Para solucionar este problema, es conveniente realizar varias medidas y obtener un valor medio de todas ellas.

## 2.2 Fuentes de error

Al igual que en cualquier medida, hay que tener en cuenta los posibles errores. Por lo que a la distancia real siempre se le va a sumar una componente de error en el cálculo de la profundidad.

Debido al principio de funcionamiento de estos sensores y a su arquitectura, se producen **errores sistemáticos** en la obtención de la profundidad. Estos errores pueden ser eliminados mediante técnicas de calibración o mejora de las condiciones.

Por otro lado, existen **errores aleatorios** causados por factores externos a la cámara. Minimizar estos errores es una tarea complicada y difícil de llevar a cabo.

### 2.2.1 Errores sistemáticos

- **Temperatura**: al tratarse de un dispositivo electrónico, la temperatura de funcionamiento supone una gran influencia en la adquisición de medidas de confianza. Se debe de considerar lo siguiente:
  - La cámara requiere una temperatura estable, dentro del rango de trabajo permitido, para conseguir los mejores resultados.
  - La cámara necesita al menos 20 minutos para alcanzar una temperatura estable.
- **Iluminación ambiente**: dado que la medida de la distancia depende del reflejo de la luz emitida por la cámara, cualquier luz adicional tanto artificial como natural, puede influenciar en el resultado de la medida.

  Los sensores están preparados para compensar esta luz ambiente, pero existe un límite pues cada pixel está preparado para recibir cierta cantidad de carga eléctrica. Si está cantidad está ocupada por la luz existente en la escena, el pixel recibirá menos cantidad de luz emitida por la cámara.

  Para solucionar esto, existe un filtro óptico que únicamente permite recibir luz del mismo espectro que la que producen los LEDs [12].

La luz solar es capaz de cubrir todo el espectro, por lo que un día soleado puede crear una energía lumínica significante, es por esto que las medidas deben estar protegidas de sobreexposición para conseguir resultados más reales.

- **Reflectividad de los objetos objetivo**: esta propiedad de los materiales tiene influencia en la precisión de la medida. Debemos considerar dos aspectos:
  - Tipo de reflexión determinada por la calidad de la superficie del objeto. Pueden ocurrir dos reflexiones:
    - Reflexión difusa: se produce en objetos mate, (Ej.: Madera o papel). Los pulsos de luz son reflejados de la misma manera en todos los ángulos. Este tipo de reflexión es preferible porque la intensidad de la luz que llega al sensor no está influenciada por el ángulo.
    - Reflexión especular: ocurre en objetos brillantes, (Ej.: Espejos, metales pulidos) y en materiales transparentes. La reflexión especular se produce cuando un rayo de luz incide sobre una superficie pulida y cambia su dirección sin cambiar el medio por donde se propaga. Esto puede producir que el pulso de luz reflejado siga un camino diferente al pulso de luz enviado por la cámara por lo que puede conducir a reflexiones "multicamino" [7].

      Esto puede producir que los píxeles se saturen con la energía de un solo pulso de luz.

      Por las razones anteriores, la medición de profundidad cuando se trata de materiales brillantes o transparentes es difícil y poco precisa.
  - Color del objeto: los objetos claros proporcionan mejores resultados de la medida pues son capaces de reflejar mayor cantidad de luz. En cambio, lo objetos oscuros una gran cantidad de luz es absorbida por lo que la cantidad que le llega al sensor es menor y la calidad de la distancia es peor.

  La proporción de la luz emitida comparada con la luz reflejada; es el coeficiente de absorción de los materiales.

- **Conversión analógica-digital**: debido al necesario redondeo en la conversión se produce una pérdida de información y la precisión es modificada de infinita a finita.

### 2.2.2 Errores aleatorios

A continuación, se indican los principales errores aleatorios que pueden presentarse en las cámaras ToF:

- **Dispersión de la luz**: debido a la baja sensibilidad del dispositivo, la dispersión de la luz puede dar lugar a artefactos en la imagen de profundidad. Esta dispersión de la luz puede generarse mediante superficies brillantes situadas cerca de los LEDs emisores. Simplemente colocando la cámara en medio de una mesa, la luz de la cámara puede ser reflejada por la mesa directamente en la lente y por lo tanto puede falsificar la medición de la distancia [10].
- **Reflexiones "multicamino"**: para una medida precisa de profundidad, la luz reflejada una única vez, proporciona información de confianza. En cambio, cuando se realiza un cálculo de profundidad en un píxel sensor que recibe una luz reflejada varias veces, se produce un error "multicamino" y la medida resulta ser un dato erróneo [7].

  Formas cóncavas, como esquinas de una habitación o el interior de una taza de café, son particularmente problemáticas ya que el pulso de luz puede rebotar hacia adelante y hacia atrás entre las diferentes superficies, aumentando así el tiempo hasta que la luz es recibida por el sensor. Esto puede derivar en la representación errónea de esquinas apareciendo curvas en lugar de líneas rectas como se aprecia en la Figura 5

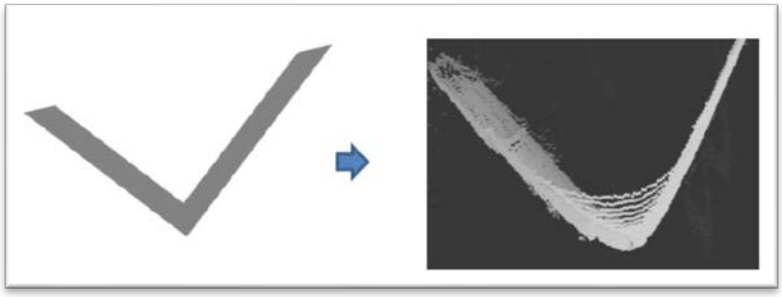


*Figura 5. Error "multicamino". Se observa medida distorsionada [6]*

Espejos y superficies altamente reflectantes también pueden conducir a múltiples reflexiones o incluso pueden desviar el pulso de luz por completo.

La Figura 6. Reflexiones multiples (multicamino) ilustra estos efectos.

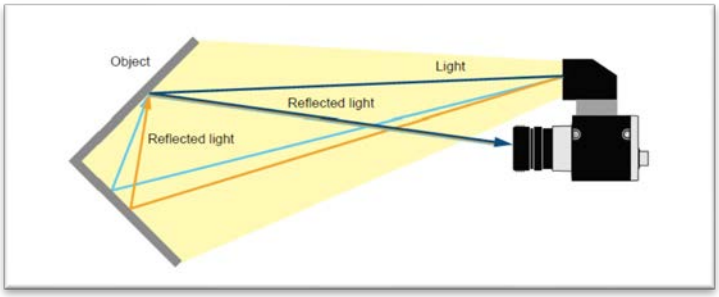



*Figura 6. Reflexiones multiples (multicamino)*

- **Error de ambigüedad en los bordes de algunos objetos:** en caso de querer reconstruir una escena 3D, los pixeles de los bordes producen cierta distorsión pues quedan situados entre un primer plano y el fondo. A esto se le denomina *"flying pixels"* [6].
- **Artefactos de movimiento**: si durante una captura alguno de los objetos en escena se mueve, se producen errores en las mediciones. Esto produce un ruido aleatorio en los bordes del objeto en cuestión.

## 2.3 Rango de no ambigüedad

Debido a que se mide la diferencia de fase entre la luz emitida y la que recibe de nuevo el sensor, y no el tiempo que tarda la luz en emitirse, rebotar en el objeto y volver al sensor, debemos tener en cuenta que en caso de querer medir la distancia a un objeto que se encuentra fuera del rango de medidas valido por la cámara, esta distancia será ambigua [6]. La Figura 7. Rango de no ambigüedadmuestra el momento en el que el pulso de luz ha alcanzado la distancia máxima, a partir de aquí, la distancia recibida es ambigua.

El máximo de este rango ($D_{max}$) corresponde a un periodo completo de modulación ($2\pi$). Por lo que la distancia a cualquier objeto fuera del rango podrá determinarse como $D_{max}$+d, lo que produce una ambigüedad importante, al poder estar situado en *d*, $D_{max}$+*d*, o quizás en $2D_{max}$+*d*. En la cámara que hemos usado en este trabajo, el rango sin ambigüedad sería 0-13.320 m, siendo $D_{max}$ 13.320 m. Esto significa que las distancias a objetos dentro de este rango reflejan la distancia verdadera al objeto.

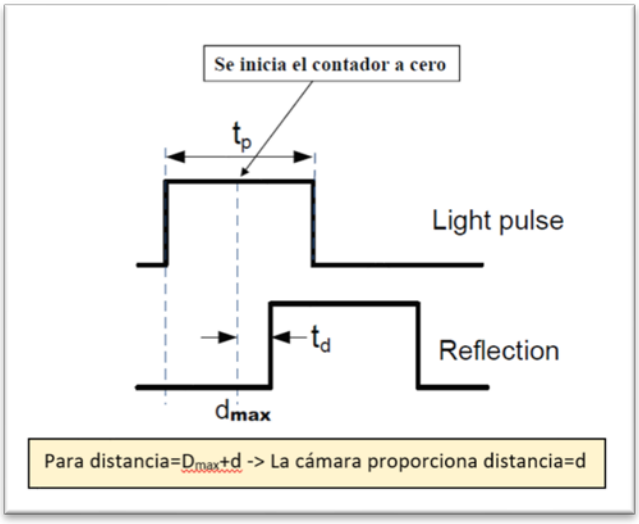



*Figura 7. Rango de no ambigüedad*

Si existe algún objeto a una distancia mayor de 13.320 m con la suficiente intensidad lumínica para ser detectado por la cámara, se situará dentro del rango efectivo y aunque su profundidad real sea 13.320 + d, a efectos prácticos de la cámara, la profundidad será únicamente d. Es decir, si la cámara detecta un objeto a 17 m, el dato que obtendremos será de 3.68 m. Esto queda reflejado en la Figura 8.

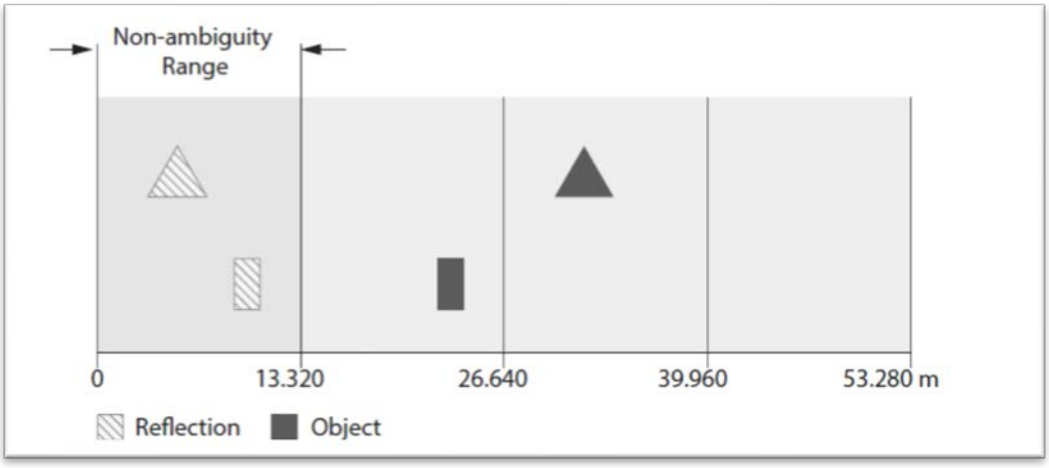



*Figura 8. "Traslado" de objetos hacia el rango de no ambigüedad*

## 2.4 Precisión y repetitividad

La cámara usada en este trabajo está calibrada para una medida entre 0.5 y 5 m. Dentro de este rango, la cámara puede ser caracterizada según el siguiente criterio:

La precisión absoluta es la media de la diferencia entre la distancia medida y la distancia real. Para lograr un resultado lo más preciso posible, las cámaras Basler ToF están calibradas en la fábrica y probadas para garantizar una precisión de confianza. La precisión alcanzada por estas cámaras es de ±1 cm.

La repetibilidad indica la variación de los valores medidos alrededor del valor medio. Con este valor se refleja la cantidad de ruido existente en una medición. Depende de la reflectividad de los objetos existentes en la escena.

## 2.5 Información de salida de las cámaras de tiempo de vuelo

En la mayoría de las cámaras ToF, la información recibida puede clasificarse en tres clases tal y como se muestra en la Figura 9.

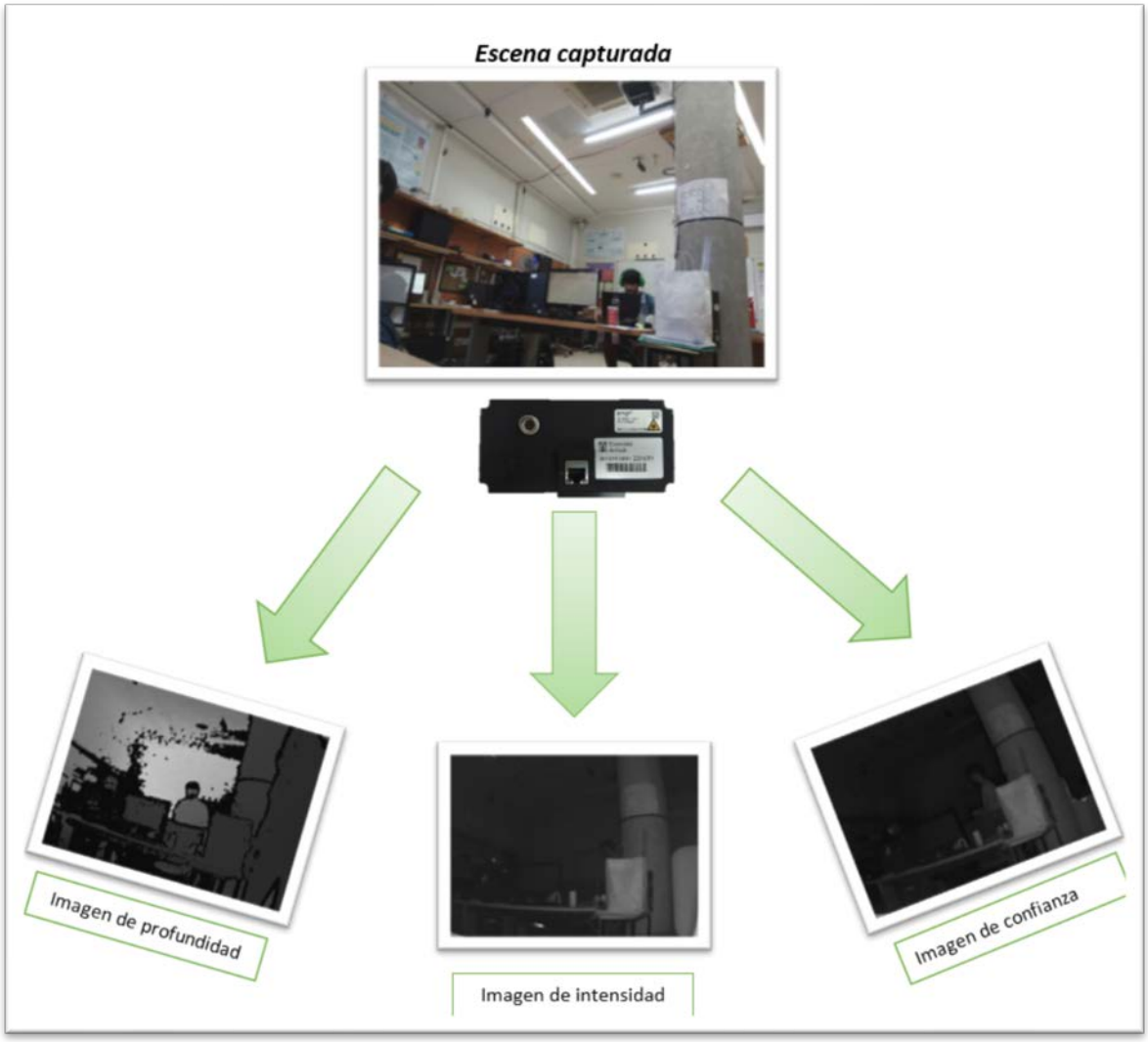



*Figura 9. Representación de los resultados obtenidos a través de una cámara de tiempo de vuelo*

En función de la cámara, los datos e imágenes son recogidos según el formato que ha decidido su fabricante, es por ello que antes de trabajar con cada sensor es imprescindible informarse sobre el tipo de datos que se van a adquirir para posteriormente poder procesarlos y trabajar con ellos.

Los siguientes párrafos explican los datos que puede recoger de manera general cualquier cámara de este tipo. Cada fabricante facilita algunos o todos los datos que se van a estudiar a continuación.

- **Profundidad**: esta información es obtenida según el principio de funcionamiento ToF tal y como se ha explicado anteriormente. El desfase entre la luz emitida y la reflejada, es directamente proporcional a la distancia que recorre la luz hasta alcanzar el objeto y volver al sensor. Los datos que nos proporciona esta medida reflejan la distancia radial

entre cada uno de los pixeles de la cámara y los objetos situados en la escena. Se obtienen imágenes como la Figura 10.

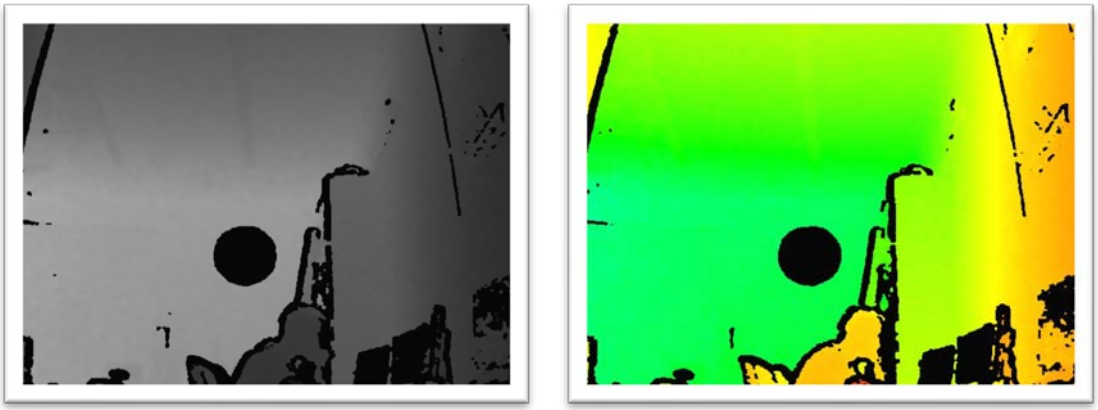


*Figura 10. Imágenes de profundidad (Izq. Monocromática, Dcha. Coloreada en RGB)*

- **Coordenadas XYZ**: las cámaras proporcionan la misma información que en el caso de profundidad, pero en este caso la información es expresada en coordenadas cartesianas. En las características de la cámara Basler se indica cual es el origen de coordenadas. Para obtener dicha información son necesarias las librerías que proporcionan los fabricantes, además es necesaria una separación de los 3 ejes para poder representar una imagen dónde poder apreciar la distancia, eso es lo que se h realizado para obtener la Figura 11.

*Figura 11. Representación del eje Z*

- **Intensidad**: se trata de una información útil para examinar los pixeles sobre saturados. A la hora de crear una imagen de intensidad se muestra el brillo de los pulsos de luz reflejados, esto puede apreciarse en la Figura 12.

Debido a la longitud de onda de la luz emitida por la cámara, la imagen de intensidad puede diferir de la percepción humana sobre la escena.

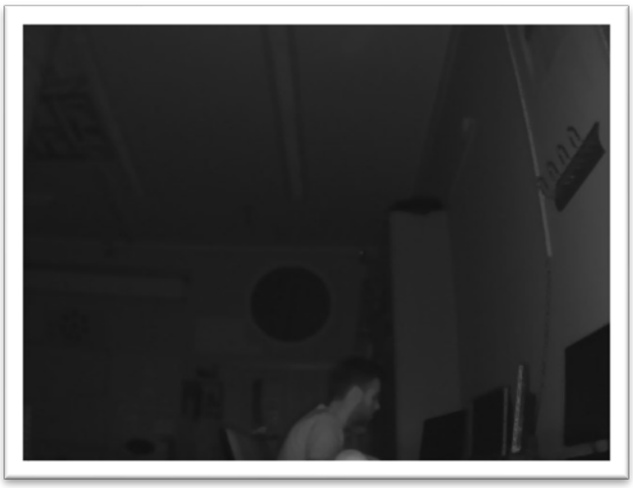


*Figura 12. Imagen de intensidad*

- **Confianza de la medida**: indica como de fiable es la medida, ver Figura 13. Es importante fijarse en estos datos para obtener la mayor precisión posible, pues gracias a ellos se pueden seleccionar regiones con gran calidad de la medida, descartar aquellas regiones que no proporcionan una información veraz, etc.

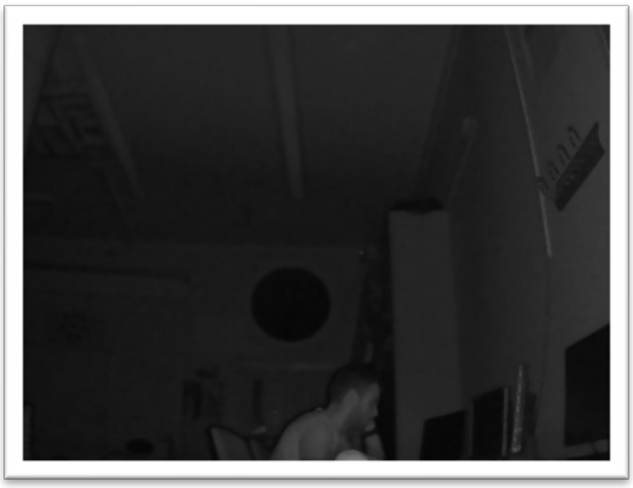


*Figura 13. Imagen de confianza*

### 2.5.1 Características de la cámara utilizada

En este apartado se desarrollan las características principales de la cámara utilizada en este trabajo. La mayor parte de la información reflejada en este capítulo se ha extraído de los

manuales y páginas webs del fabricante de la cámara bajo estudio [10]. A continuación, se detallan las principales características de dicha cámara. La cámara que se ha usado es, por fabricante y modelo: Basler ToF 21886650 (Figura 14)

*Figura 14. Cámara Basler ToF*

La siguiente tabla recoge de manera resumida las características generales.

| **Característica** | **Basler ToF Camera** |
|---|---|
| **Dimensiones (L x W x H)** | 141.9 mm x 61.5mm x 76.4 mm |
| **Fuente de alimentación** | 24 VDC (±10 %), suministrados mediante el conector de 12 pines |
| **Nº de LEDs emisores** | 8 |
| **Longitud de onda emitida** | 850 nm, ±30 nm |
| **Resolución (H x V pixels)** | 640 x 480 |
| **Ángulo del campo de visión** | 57° x 43° |
| **Precisión absoluta** | ±1 cm |
| **Repetividad típica** | 8 mm |
| **Interfaz de comunicación** | Gigabit Ethernet (1000 Mbit/s) |
| **Temperaturas de operación** | 0-50 °C |
| **Peso** | 450 g |
| **Max. Frecuencia de fotogramas** | 20 fps |
| **Mono/Color** | Mono |
| **Tiempo de exposición** | Programable mediante la API de la cámara |

*Tabla 1. Características generales de la cámara Basler*

**Características mecánicas**

A continuación, en la Figura 15 se muestran las dimensiones y puntos de montaje de la cámara Basler ToF. Aunque esta cámara no es voluminosa, existen otras cámaras ToF aún más pequeñas.

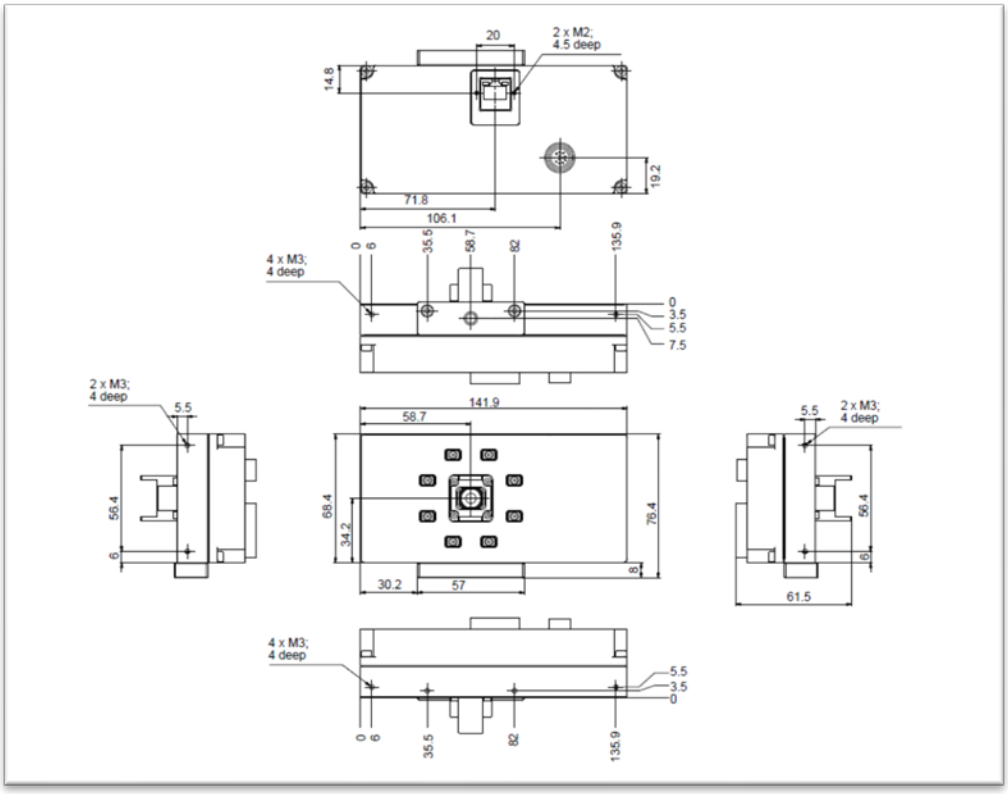



*Figura 15. Dimensiones y puntos de montaje de la cámara Basler ToF*

La Figura 16 muestra la morfología delantera de la cámara. El objetivo de la cámara cuenta con un pequeño parasol. Gracias a este objeto, el problema causado por la dispersión de la luz puede ser solucionado.

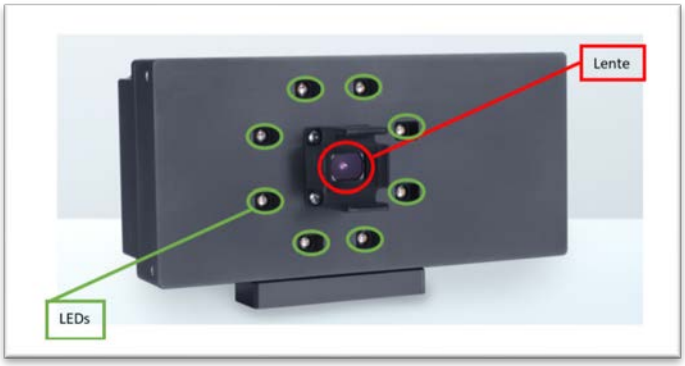



*Figura 16. Descripción frontal*


*Figura 17. Alimentación y conexiones*

Finalmente, la Figura 17 muestra las conexiones de la cámara. La clavija de alimentación, además, es la encargada de transferir la información desde la cámara al ordenador y viceversa, es decir, es un cable entrada/salida. La cámara Basler ToF se conecta a través del puerto Ethernet.

## 2.6 Funcionamiento

Basándose en el principio de tiempo de vuelo, la cámara Basler puede medir distancias mediante el envío de un pulso de luz. La distancia se obtiene como el tiempo que tarda el pulso en volver al sensor una vez sido reflejado en un objeto.

El pulso de luz es emitido desde unos LEDs infrarrojos. Una vez alcanzado el objetivo y registrado el pulso por el sensor, el sensor convierte la carga eléctrica creada por la energía de la luz en información sobre la distancia.

Cuando se produce un disparo, los LEDs lucen todos simultáneamente durante un periodo de tiempo ($t_p$) especificado. Tras reflejarse la luz, esta es recogida por cada uno de los *pixels* del sensor usando dos obturadores con el mismo $t_p$. El mapa de profundidad adquirido cuenta con una resolución de 640x480 *pixels*. El primer obturador abre cuando comienza el pulso de luz, el segundo cuando el pulso ha acabado.

Este proceso ocurre repetidamente hasta que se completa el tiempo de exposición.

En función del fabricante se proporciona una aplicación diferente capaz de mostrar y capturar imágenes, así como adquirir los datos necesarios. Estos softwares no facilitan el tratamiento en tiempo real de los datos.

La manera más sencilla de obtener imágenes y configurar la cámara es mediante la aplicación de la cámara, *PylonViewerApp* proporcionada por el fabricante (Figura 18).

Una vez dentro de esta aplicación se selecciona el dispositivo y ya sería posible la adquisición de imágenes. Pulsando sobre *Continuous Shot* se obtienen *frames* según la frecuencia indicada. En *One Shot* se obtiene una única imagen.

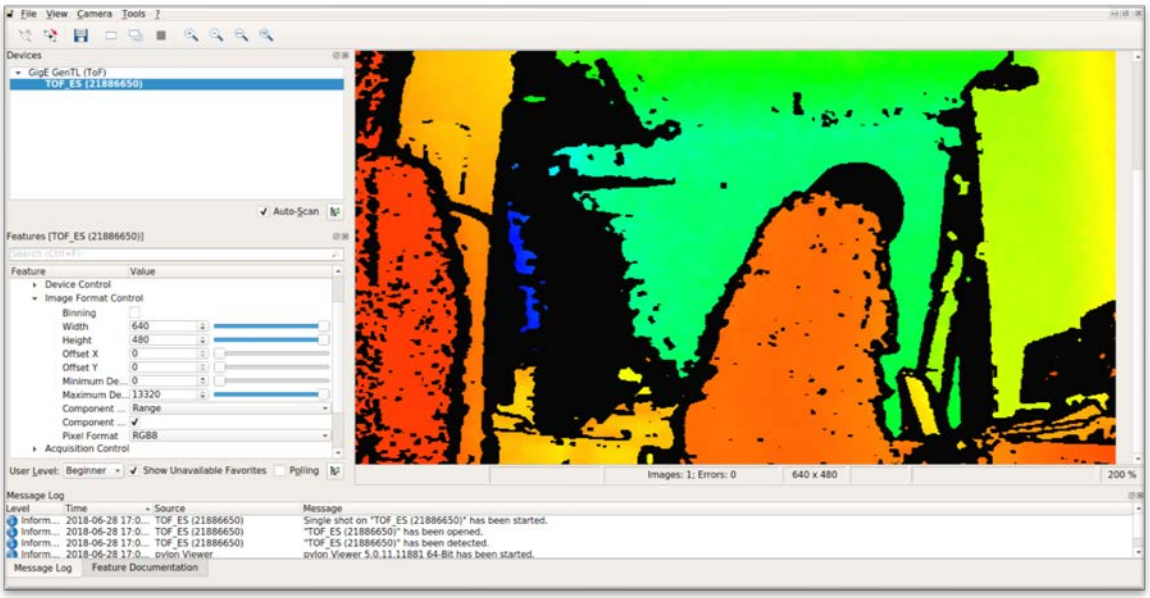



*Figura 18. Aplicación proporcionada por el fabricante, PylonViewerApp*

Hasta este momento las características de la imagen son las impuestas por defecto. A continuación, se describen los diferentes ajustes que permite la cámara.

**Región de Interés (ROI)**

La región de interés permite al usuario centrarse en una determinada área de la escena. Los parámetros que se pueden modificar para concretar la zona deseada son los siguientes:

- Anchura y altura/ ***Width*** **and** ***Height***: mediante estos parámetros se puede cambiar el tamaño de la región de interés. (Valores máximos: **640x480** ***pixels***).
- Desplazamiento eje X y eje Y/ ***OffsetX*** **and** ***OffsetY***: Tras especificar el tamaño de la región de interés, es posible centrarse en una determinada parte de la escena. Los valores posibles de estos dos parámetros dependen del tamaño elegido. Si el tamaño escogido de la región de interés es 430x120 *pixels* podremos desplazarnos por el eje x un máximo de 640 -430 =210 *pixels*. Por el eje y, será posible desplazarse 480 -120 = 360 *pixels*.

El origen de coordenadas de la región de interés se encuentra desplazado respecto al eje y. Si el tamaño escogido es 100x100 *pixels* y ambos *offsets* están configurados a 0 *pixels*, la sección elegida se situará en la esquina superior izquierda de la escena, esto puede verse marcado en rojo en la Figura 19.

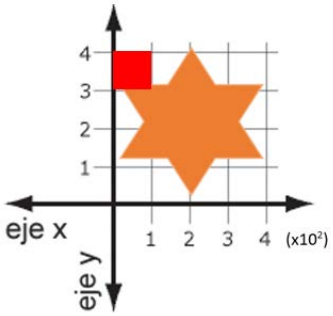



*Figura 19. Origen de coordenadas ROI*

- Profundidad máxima y mínima/ ***Minimum* and *Maximum Depth***: especifican entre que valores es posible adquirir datos de distancia. La información fuera de este rango es descartada. (El máximo rango posible de acuerdo al fabricante es: **0-13.320 m**) Con estos valores es posible reducir la ambigüedad de los resultados. Por lo que, si el objetivo está situado entre los 4 y 8 metros, es conveniente ajustar el parámetro de mínima profundidad a 4000 y el de máxima a 8000. De esta manera todo lo que esté fuera del rango, será obviado.

**Control de formato de imagen**

En esta sección es donde se selecciona el tipo de datos que se obtienen de la cámara. También es posible ajustar el formato de los pixeles, la Figura 21 contiene un ejemplo de cada uno de ellos.

- Datos de distancia/***Range Data***: aquí se obtiene la imagen de profundidad. Esta imagen representa la distancia radian entre el objetivo y la cámara. Los formatos de pixeles posibles son:
    - ***Coord3D_C16*** (Valor por defecto): imagen monocromática de 16 bits enteros sin signo (*Little-endian*). Las zonas más oscuras representan los objetos cercanos a la cámara, mientras que las áreas más claras nos indican lejanía. Objetos fuera de la región de interés aparecen en negro.
    - ***RGB8***: mediante un coloreado falso de la imagen se consigue una mejora de la visualización. Se usan los colores rojo, verde y azul en una imagen de 8 bits

enteros. El color rojo se sitúa sobre los objetos cercanos a la cámara, mientras las zonas azules, indican que los objetos están más alejados. Al igual que en el formato anterior, los objetos que no están dentro de la región de interés aparecen en negro.

- ***Coord3D_ABC32f***: representación en nube de puntos usando 32 bits en coma flotante. Cada punto de la nube representa las coordenadas de la superficie donde ha sido reflejada la luz. El origen de coordenadas se refleja en la Figura 20:

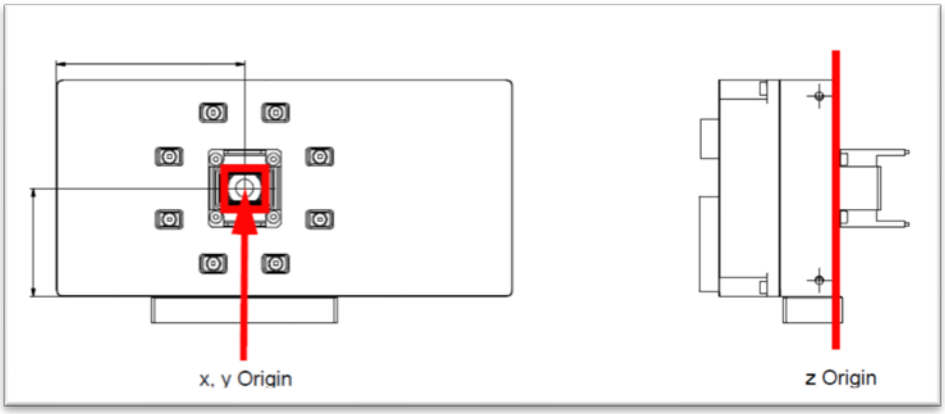



*Figura 20. Origen de coordenadas*

- Imagen de intensidad/ ***Intensity image***: esta imagen está compuesta de 16 bits enteros por *pixels*. En ella se muestra el brillo de la luz reflejada. Está imagen resulta útil para comprobar si existen pixeles sobre saturados o no. En este caso solamente hay un formato de pixeles:
  - ***Mono16***: con él se consigue una imagen monocromática (B&W) usando 16 bits enteros sin signo.

- Mapa de confidencia/***Confidence map***: la imagen obtenida usando este formato representa como de fiable es la medida de la distancia. Se genera una imagen de 16 bits enteros por pixel. Cuanto más brillante es el pixel, más real es la medida. El formato de pixeles existente es:
  - ***Confidence16***: imagen monocromática usando 16 bits enteros sin signo.

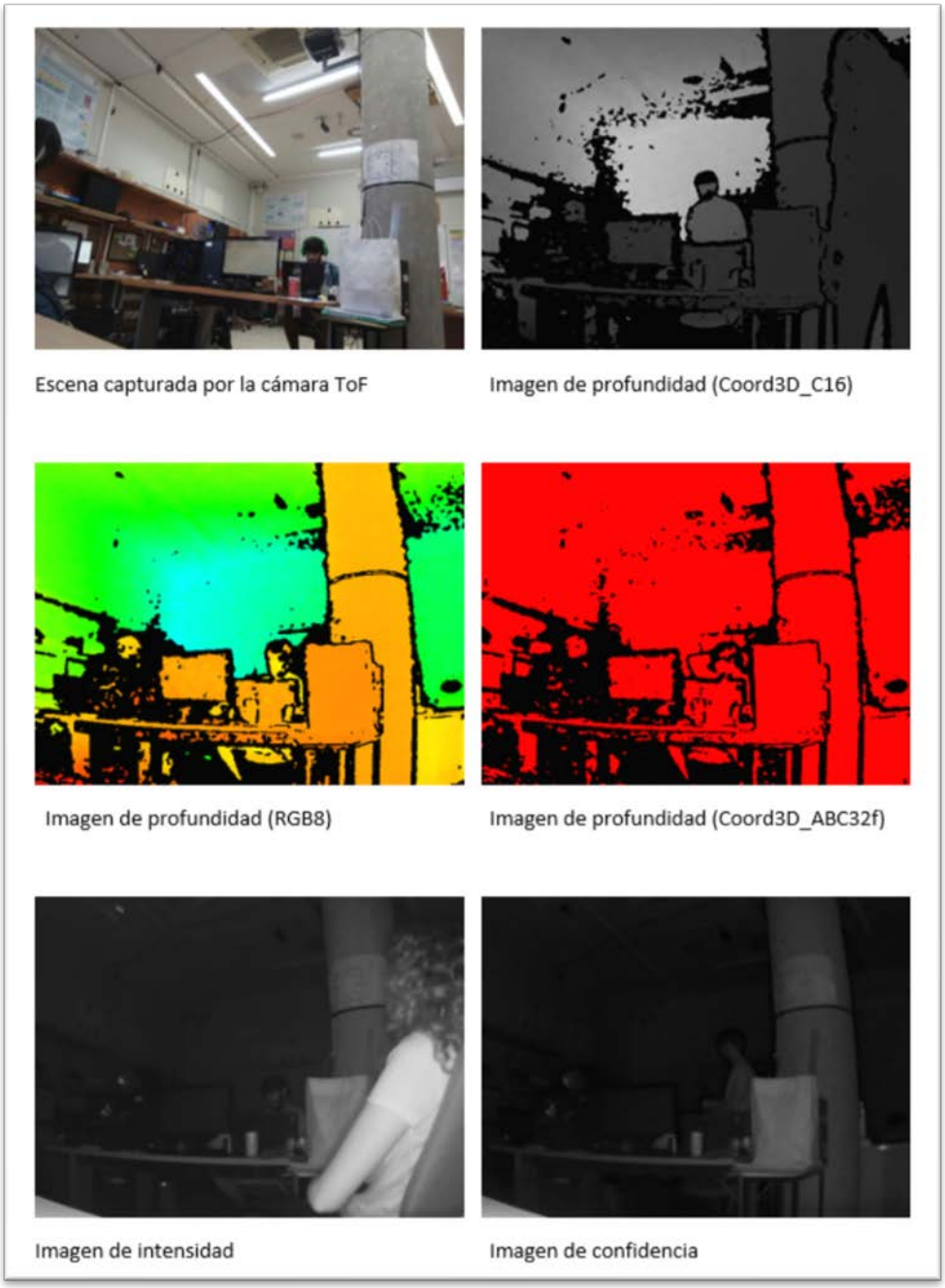



*Figura 21. Formatos posibles de imagen*

**Control de adquisición**

Los siguientes parámetros controlan los tiempos en la adquisición de imágenes. Son importantes para aumentar la velocidad o mejorar los resultados.

- Deshabilitación de los LEDs/***LED Disable***: si los LEDs se apagan resulta imposible adquirir imágenes, por lo tanto, este parámetro no se puede modificar en la aplicación desarrollada.

- Modo de procesamiento/***Procesing Mode***: si se cuenta con una única cámara, existe solo un tiempo de exposición, por lo que el modo elegido es ***Standard***. El modo **HDR** permite usar dos tiempos de exposición, es decir, dos cámaras. Puesto que solo disponemos de una cámara Basler, el modo elegido en este trabajo es Standard.
- Frecuencia de adquisición de imágenes/***Acquisition Frame Rate***: de este parámetro depende la frecuencia a la que se toman las imágenes. Su valor máximo es **20 Hz**.
- Tiempo de exposición automático/***Exposure auto***: con este parámetro se decide si se usa o no el tiempo de exposición automático. En caso de usar ***Continuous***, el tiempo de exposición se adapta continuamente por la cámara. Si queremos modificarlo nosotros, este parámetro debe ser ***Off***. Se recomienda el uso automático por lo que este parámetro ha sido ajustado como Continuous.
- Selector tiempo de exposición/***Exposure Time Selector***: con este parámetro se decide si queremos un único tiempo de exposición (**0**) o dos (**1**). En el modo HDR son necesarios dos tiempos.
- Tiempo de exposición/***Exposure Time***: en caso de estar el modo de procesamiento en *standar*, el parámetro *Exposure Auto* en *Off*, el tiempo de exposición puede ser controlado. El tiempo de exposición indica que, en la adquisición de una imagen, la cámara envía pulsos de luz continuos durante el tiempo seleccionado.

**Calidad de la imagen**

Gracias a estos filtros que la cámara tiene incorporados, la calidad de la imagen es mejorada notablemente.

- Umbral de confianza/ ***Confidence Threshold***: para tener en cuenta los pixeles a la hora de medir la distancia, deben haber superado el nivel de confianza.
- Filtro espacial/***Spatial Filter***: este filtro usa los valores de los píxeles vecinos para filtrar el ruido espacial en una imagen. Esto ayuda aplanar los baches en los planos y crear superficies más lisas. Los bordes del objeto se dejan intactos.
- Filtro temporal/***Temporarl Filter***: este filtro usa los valores que toma el mismo píxel en diferentes momentos para filtrar el ruido temporal en una imagen.
- Fuerza/***Strength***: esto define el valor de la memoria del filtro temporal.
- Tolerancia atípica/***Outlier Tolerance***: esto define cuanto puede diferir la profundidad de un pixel respecto de la de sus píxeles vecinos.

## 2.7 Aspectos adicionales sobre el manejo de la cámara ToF

Además de todos los parámetros necesarios para tomar y adquirir las imágenes, son necesarias tener en cuenta algunas consideraciones que influye en la calidad de las medidas.

**Condiciones hardware y software**

Este tipo de cámaras deben montarse sobre un soporte que impida movimientos no deseados de la cámara y que permita su manejo.

Para evitar sobrecalentamientos de la cámara es importante que no se tape y que la temperatura y humedad de la zona de trabajo se encuentren dentro de unos límites dados por el fabricante.

Deben tenerse en cuenta las posibles interferencias electromagnéticas, así como la cantidad y la naturaleza de la luz incidente en el objetivo.

Dentro del requerimiento hardware para el uso de la cámara debemos destacar lo siguiente:

- Cámara Basler ToF []
- Fuente de alimentación
- Cable GigE
- Soporte de la cámara
- Ordenador con las siguientes especificaciones:
    - Procesador: Intel i5 2.4 GHz o mejor
    - RAM: 4GB mínimo
    - Adaptador de red: Intel Pro-1000 (Recomendado)

Para el uso de la cámara es necesario poseer el software indicado a continuación:

- Sistema operativo
    - 32-bit Windows 7
    - 64-bit Windows 7 (Recomendado)
    - Linux (x86, x64)
- Basler ToF Driver: Disponible en Windows y Linux. Incluye lo siguiente [13]:
    - Pylon Viewer App (Linux only)
    - ToF IP Configurator
    - GenICam GenTL producer
    - Código de muestra

Mediante la guía "*Basler ToF Driver Package 1.2.0 C++ Programmer's Guide*" [10] proporcionada por el fabricante, es posible configurar la cámara y acceder a la información que esta puede proporcionarnos. La principal librería usada en este trabajo es `<ConsumerImplHelper/ ToFCamera.h>` Con ella ha sido posible implementar en C++ la interfaz del presente trabajo.

**Requerimiento eléctrico**

La potencia debe de ser suministrada a la cámara mediante el conector de 12 pines.

La tensión requerida es de 24 VDC (±10 %) 15 W avg y el consumo máximo de energía es de 29 W pico. La corriente de salida de la fuente de alimentación no debe de superar los 2 A. Para evitar sobretensiones de encendido que dañen la cámara, enchufe o desenchufe el cable de alimentación en la cámara si la fuente de alimentación está apagada.

**Temperatura**

Los valores de temperatura y humedad idóneos que recomienda el fabricante son los siguientes:

- Temperatura de la carcasa durante el funcionamiento: 0-50 ° C
- Humedad durante el funcionamiento: 20-80%, relativa, sin condensación.
- Temperatura de almacenamiento: -20-80 ° C
- Humedad de almacenamiento: 20-80%, relativa, sin condensación

Es importante proporcionar la suficiente disipación de calor a la cámara. Los pasos para conseguir esto dependen de la instalación, es por ello que el fabricante propone la siguiente guía general:

- En cualquier caso, usted debe monitorizar la temperatura de la carcasa y asegurarse de que no excede los 50 °C. Esto es posible con el control automático de temperatura que nos facilita la cámara. La cámara Basler ToF está equipada con dos sensores de temperatura, uno de ellos sobre los LEDs y el otro sobre el sensor. Si se alcanza la temperatura máxima de funcionamiento, la adquisición de la imagen será apagada, si esto ocurre debe de reducirse la temperatura bajando el tiempo de exposición o la frecuencia de disparo. Cuando la temperatura logra reducirse un 10% es posible volver a adquirir imágenes.
- Proporcionar la suficiente disipación considerando alguna de las siguientes opciones:
    - Usando un ventilador que proporcione un flujo de aire sobre la cámara. De esta manera se fuerza la circulación del aire.

- Montando la cámara sobre un componente térmicamente conductivo que puede ayudar como disipador.

**Limpieza**

Resulta conveniente mantener limpia la cámara. Con esto dotaremos a la cámara de una mayor vida y, además, su funcionamiento será el adecuado. La emisión y captación de la luz infrarroja no se verá afectada por el polvo.

# 3 Desarrollo

## 3.1 Introducción

La creación de este TFG ha sido posible gracias a la dedicación e implicación en las diferentes tareas necesarias para su desarrollo. A continuación, se especifican los pasos seguidos y las dificultades encontradas en cada momento.

Antes de comenzar con la programación, es importante realizar una revisión bibliográfica sobre el principio de funcionamiento y las prestaciones de los sensores de tiempo de vuelo. Este estudio ha sido una tarea sencilla pues existen muchas ilustraciones donde se detalla de manera clara el funcionamiento de estas cámaras.

Además, tras realizar esta tarea, se han revisado las funciones y librerías de la cámara utilizada en el presente trabajo. Los manuales de usuario proporcionados por el fabricante han sido claves para entender las características que la cámara Basler ToF posee. Cabe destacar que, gracias a estos manuales, se ha podido conocer el tipo de señal moduladora con la que cuenta este sensor. Una vez conocido esto, se ha trabajado en este interfaz sabiendo que la señal que emite la cámara es de modulación pulsada.

Tras el estudio del funcionamiento de estos sensores y las características propias de la cámara Basler, se ha definido el software con el que se va a realizar el interfaz. En este paso se encontraron las siguientes dificultades:

- El entorno de desarrollo elegido al principio fue NetBeans. Este entorno no reconocía correctamente las librerías de OpenCV. Además, este programa complicaba demasiado la creación de proyectos en C++ y su posterior compilación.
- Finalmente, se ha desarrollado la aplicación en Eclipse neon.2. Aunque al principio no compilaba el código desarrollado, rápidamente se dio con la solución. El *makefile* generado automáticamente no contaba con las librerías propias de la cámara. La solución fue crear otro *makefile* que tuviese en cuenta las librerías necesarias para la ejecución de este interfaz.

El siguiente paso ha sido la definición de la estructura del programa. Este paso estaba claro desde el principio, la cámara debía abrirse, configurarse, adquirir imágenes y cerrarse. Inicialmente se contaba con cuatro funciones más la función principal. Al tener que tratar los

datos recibidos vía la terminal, la función *main* se extendía demasiado, siendo esto un problema cuando se quería usar el resto de las funciones creadas como librería. Añadiendo una nueva función que únicamente se encarga del tratamiento de los datos, la función main se ha reducido tal y como se quería.

La adquisición de los datos para configurar y adquirir las imágenes ha dado un giro importante desde la idea inicial hasta la ejecución final del trabajo. Al principio, importaba el orden en el que estos parámetros eran introducidos por la terminal, cada uno tenía su sitio dentro del array de argumentos. Esto alargaba mucho la cadena de parámetros necesarios y dificultaba el uso de la aplicación. La solución encontrada fue la introducción de los parámetros que se quisiesen modificar indicando primero el nombre del parámetro y a continuación el nuevo valor que se quiere otorgar. En caso de querer dejar alguno de estos parámetros en su valor por defecto tan solo hay que añadir *std* al final. De esta manera, ya no importa el orden de introducción ni es necesario introducir todos los argumentos configuradores.

Otro paso importante en el desarrollo de este trabajo ha sido la elección de los posibles parámetros configurables. La cámara no permite hacer grandes modificaciones en su configuración, por lo que se han elegido las más importantes. Algunas de estas modificaciones son el formato de la imagen y la resolución.

Para concluir la implementación de la API se han diferenciado dos modos de adquisición. La adquisición de una única imagen ha sido fácilmente programable. En cambio, la adquisición de secuencias ha dado problemas de almacenamiento, no todos los *frames* son guardados, almacenar es un proceso lento. Además, paralizar la adquisición de imágenes ha sido una de las dificultades encontradas más difíciles de mejorar. Finalmente se ha optado por usar una función de OpenCV. Situándose encima de la ventana que muestra la imagen y pulsando sobre cualquier tecla, la adquisición de imágenes se detiene.

Tras concluir todas las funciones necesarias, estas han sido almacenadas como una biblioteca. Esto facilita su uso y reduce el código a una única función principal desde la cual se llama a las funciones necesarias.

Una vez desarrollado el interfaz ha sido evaluado para demostrar su correcto funcionamiento, los errores que han ido apareciendo durante la creación del programa ha sido subsanados poco a poco por lo que, en la evaluación del trabajo, apenas han aparecido fallos.

Concluida la aplicación, se ha realizado esta memoria y manual de usuario sobre la misma. La dificultad de este paso ha sido que la información encontrada sobre cámaras ToF en su mayoría se refiere a sensores de modulación continua como la cámara Mesa SR4000 [11]. En cambio, la modulación de la cámara utilizada en este trabajo, Basler ToF 21886650, es pulsada.

## 3.2 Lenguaje de programación y bibliotecas

En este apartado se exponen los programas y entornos de desarrollo utilizados para la creación de la aplicación llevada a cabo en este trabajo. Las librerías del fabricante ya han sido definidas anteriormente.

**Lenguaje de programación C++.**

El lenguaje de programación del presente trabajo es C++. Esta decisión ha sido tomada una vez vistas las librerías y los códigos de muestra que ofrece el fabricante. Dado que estas están escritas en C, es necesario que el código de la aplicación haya sido también escrito en C para así permitir la interacción entre el programa y las librerías a usar. Exactamente C++ es un lenguaje compatible con C pero más completo y orientado a objetos

Este lenguaje es fácil de usar en todos los sistemas operativos pues se trata de un lenguaje libre y versátil. Además, los trabajos hechos anteriormente en relación con Cámaras ToF han sido programados en este lenguaje lo que aporta esperanza a la continuidad del trabajo.

**Linux**

Durante el grado, el sistema operativo utilizado en varios laboratorios de programación ha sido Linux. Puesto que este trabajo necesita capacidad suficiente para crearse, ser compilado y su posterior adquisición y almacenamiento de imágenes, se ha decidido que Linux tiene el potencial necesario para su desarrollo. Además, el ordenador donde ha sido llevado a cabo el trabajo contaba con Ubuntu 16.04 LTS como Sistema Operativo y las librerías que han sido necesarias ya descargadas [13].

**Eclipse**

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de programas en C++ o aplicaciones Java entre otros.

No es más que un entorno de desarrollo integrado (IDE) en el que todas las herramientas, funciones y librerías necesarias para tu trabajo están recogidas en una atractiva interfaz que lo hace fácil y agradable de usar [14].

**OpenCV**

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Sus librerías se han utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos [15].

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Sus más de 500 funciones abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos y personas, calibración de cámaras, visión estéreo y visión robótica.

Gracias a estas librerías se puede programar en C y C++ con gran eficiencia.

En este trabajo se utiliza Open para trabajar con las imágenes adquiridas por la cámara en forma de matrices, siendo posible adquirir datos de la profundidad, intensidad y confianza. Además, Open permite generar y mostrar las imágenes en pantalla.

En función de los parámetros pasados a la cámara, las imágenes recibidas tendrán diversas características, que mediante OpenCV podemos representar. Algunas de estas características son el tamaño, color, número de canales, etc.

## 3.3 Interfaz de programación de aplicaciones desarrollada

En este trabajo se ha desarrollado un interfaz de programación de aplicaciones (API) como se había previsto en los objetivos. Ha sido programada en C++ desde el Sistema Operativo Linux.

El código escrito permite al usuario utilizar de manera sencilla la cámara para adquirir, mostrar y guardar las imágenes solicitadas.

A continuación, se detallas las características principales de la aplicación desarrollada.

Esta interfaz permite trabajar con diferentes configuraciones de las características básicas de la cámara. Las modificaciones de los parámetros son internamente gestionadas por la aplicación. La interfaz creada es agradable y transparente, lo que evita una compleja formación por parte del usuario.

La interfaz creada tiene la capacidad de poder solicitar la visualización y almacenamiento de las imágenes. La extensión de almacenamiento es PNG. Además, es posible configurar las características de la imagen, destacando la modificación de su tamaño, formato y rango de profundidad.

El software desarrollado funciona de manera fluida durante los dos modos posibles de adquisición y almacenamiento de datos. Estos son la toma de imágenes mediante un único disparo y la grabación continua.

Cámara soportada en el presente trabajo:

- Basler ToF 21886650 [10]. La cámara se conecta a través de Ethernet, por lo que se debe indicar la dirección IP al abrir la aplicación. para que esta reconozca la cámara.

Las imágenes o secuencias de imágenes que se quiera almacenar son guardadas en carpetas con la fecha y hora del momento que se ha comenzado la adquisición.

Extensibilidad y portabilidad:

- Es posible usar parte del código para la implementación de un software similar para otra cámara ToF. Tan solo cambiarán las funciones propias de la cámara relacionadas con la configuración y adquisición de imágenes.
- En caso de usar cámaras del mismo fabricante, este programa sería portable siempre que las cámaras Basler proporcionen los mismos tipos de imágenes y tengan los mismos parámetros configurables. En este caso las librerías utilizadas por la cámara son las mismas.
- El código está preparado para añadir algoritmos de mejora de la información o de adquisición de datos.

Tamaño:

- Se compone de casi 700 líneas de código.
- El trabajo compilado tiene un tamaño de 509.4 kB.

El software desarrollado permite adquirir varios formatos de imágenes a la vez, de manera fluida y decidiendo si se quiere una grabación continua o un único disparo. Esto es posible mediante una breve línea en la terminal de comandos de Ubuntu.

El formato de almacenamiento de las imágenes adquiridas es .png. Estás se almacenan de forma independiente en carpetas previamente creadas en el mismo programa. La visión de estas

imágenes puede realizarse mediante cualquier visor convencional de imágenes. Estas imágenes pueden ser tratadas, estudiadas o editadas posteriormente.

El desarrollo del programa en diferentes funciones permite añadir nuevas cámaras y algoritmos sin realizar grandes modificaciones en el programa. Además, es posible añadir nuevas funciones para la adquisición de datos, o cambios en las características de la cámara.

Mediante la librería OpenCV ha sido posible la representación y almacenamiento de las imágenes. Facilitándose así la creación de matrices para mostrar y guardar los diferentes formatos de imagen posibles.

El programa se ejecuta de manera secuencial, siendo esta una de las características más importantes. Cada proceso queda separado del siguiente, así se asegura que se acceda a todas las funciones y los resultados sean óptimos. El orden de ejecución es el siguiente:

1. Lectura de los datos recibidos por la terminal
2. Selección de la cámara que se desea abrir
3. Configuración de la cámara
4. Adquisición de la imagen o imágenes
5. Cierre de la cámara

La Figura 22 representa mediante un diagrama de flujo la estructura del código.

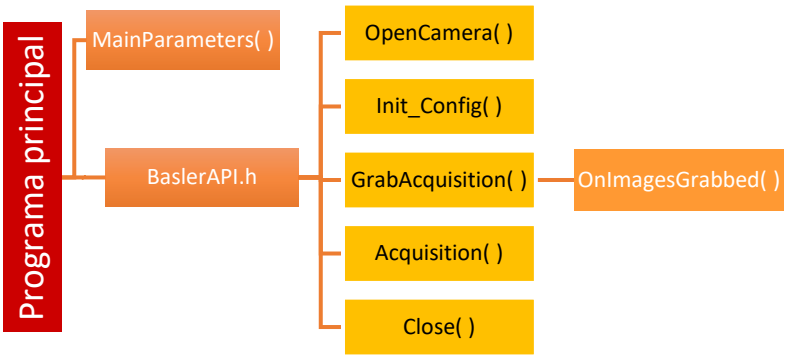



*Figura 22. Diagrama de flujo del código*

Este programa depende de varias librerías que para su correcto funcionamiento es obligatorio tener perfectamente instaladas en el ordenador en cuestión. De esta manera el usuario podrá usar el software desarrollado. La instalación de estas librerías es un requisito indispensable para el objetivo de este trabajo. Con esto evitamos el inconveniente que poseen

estas cámaras ToF pues gracias al uso de estas librerías en el código, el uso de la cámara resulta ser mucho más sencillo que anteriormente.

**Descripción de la solución desarrollada**

En esta sección se explica brevemente el funcionamiento de cada uno de los procesos que componen la aplicación. En el apartado "Manual de usuario" se hará un estudio más detallado de la funcionalidad y uso de esta aplicación.

- Lectura y procesado de los datos introducidos por el usuario:
  `MainParameters(int numargc, char* ImgPar[]);`

  En esta función se reciben los datos que el usuario escribe por la terminal. Una vez que están tratados se llama a las siguientes funciones. Su diagrama de flujo está representado en la Figura 23.

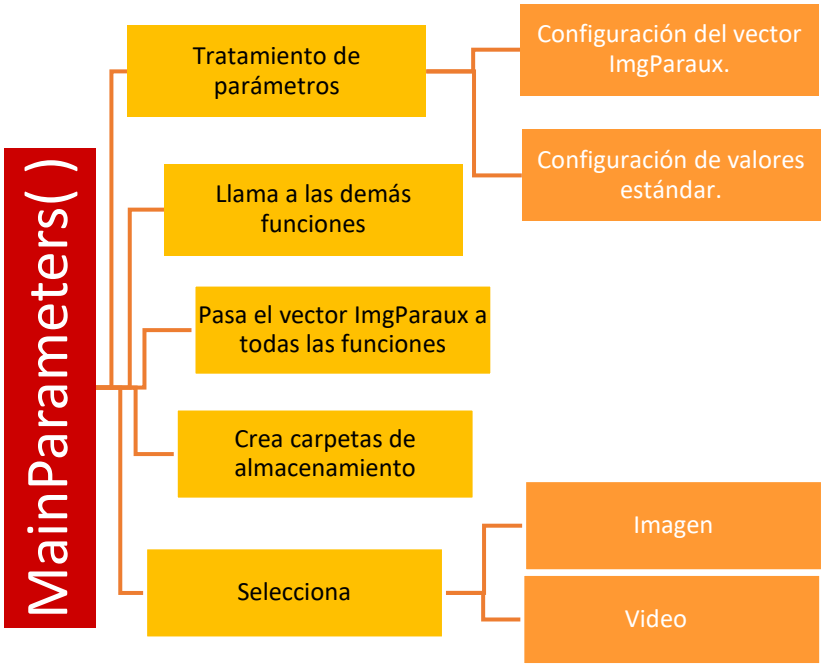



*Figura 23. Diagrama de flujo de la función MainParameters( )*

- Apertura de la cámara seleccionada mediante IP:
  `OpenCamera(char* CameraIP[]);`

  El primer parámetro pasado por la terminal es obligatoriamente la IP de la cámara. Esto queda reflejado en su diagrama de flujo, Figura 24. Cuando la recibe esta función, la cámara se "abre" y se queda preparada para ser usada.

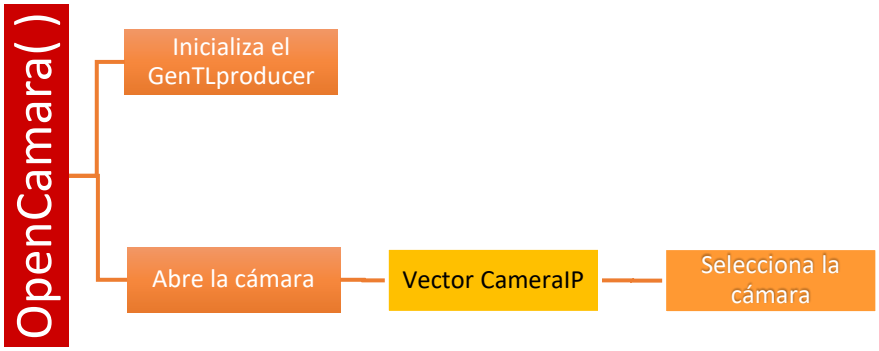



*Figura 24. Diagrama de flujo de la función OpenCamera( )*

- Configuración de las características de la cámara, decidimos formato de imagen, etc:
  `Init_Config(char* ImgFormat[]);`

  Aquí se sitúan todos los parámetros que el usuario ha querido modificar respecto a los valores predeterminados de la cámara, y se configuran las características modificables de la cámara. Queda representado en la Figura 25.

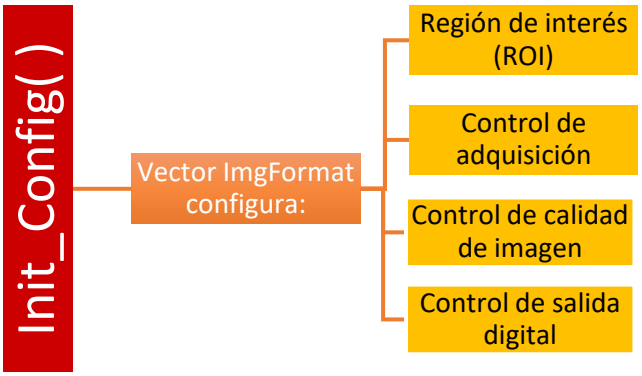



*Figura 25. Diagrama de flujo de la función InitiConfig( )*

- Opción grabación continua, visualización y almacenamiento:
  `GrabAcquisition(char* ImgGrab[]);`
  `onImageGrabbed(GrabResult grabResult, BufferParts parts);`

  En caso de solicitar grabación continua, la primera función se dispone a adquirir imágenes según la frecuencia solicitada, Figura 26, y la segunda se encarga del

almacenamiento y visualización de las imágenes, Figura 27.

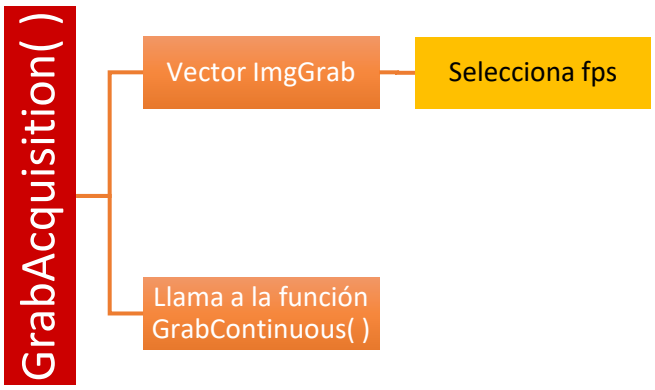



*Figura 26. Diagrama de flujo de la función GrabAcquisition( )*

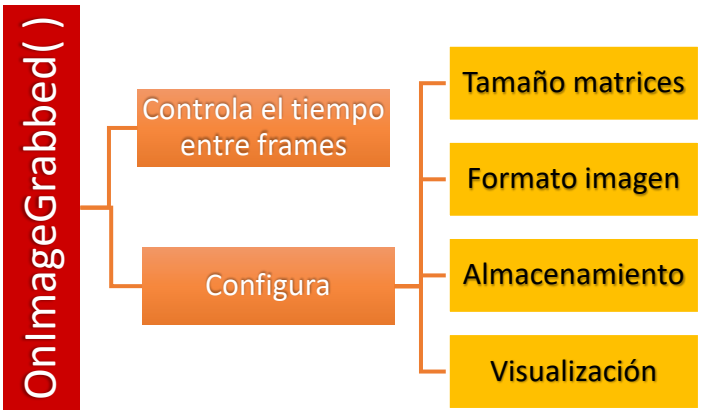



*Figura 27. Diagrama de flujo de la función OnImageGrabbed( )*

- Opción disparo único, visualización y almacenamiento

  `Acquisition(char* ImgAcqui[]);`

  Si únicamente se desea una imagen, esta función se encarga de adquirir la imagen de la cámara, visualizarla y guardarla en el ordenador. Su diagrama de flujo está contenido en la Figura 28.

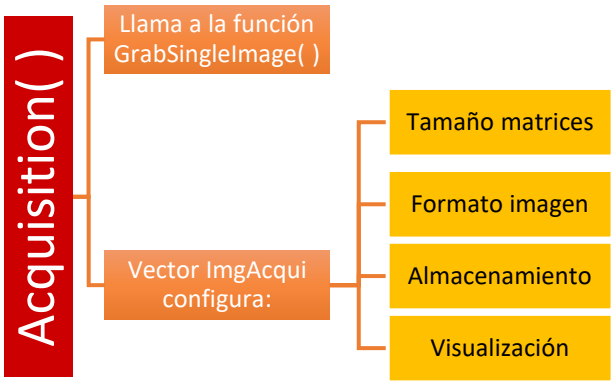



*Figura 28. Diagrama de flujo de la función Acquisition( )*

- Cierre de la cámara

  `Close();`

  Una vez concluidas todas las funciones anteriores, la cámara se cierra, Figura 29.

  Esta función es necesaria para poder solicitar de nuevo la adquisición de imágenes.


*Figura 29. Diagrama de flujo de la función Close( )*

# 4 Conclusiones y trabajos futuros

## 4.1 Conclusiones

En el presente trabajo se ha programado una aplicación capaz de configurar una cámara de tiempo de vuelo, capturar imágenes de diferentes tipos, mostrarlas y almacenarlas en el disco duro. Con ello, los objetivos propuestos para el desarrollo de este TFG se han cumplido.

Mediante formación y trabajo en esta área, se han adquirido los conocimientos necesarios para entender el principio de funcionamiento de estos sensores. Además, el manejo de las librerías de la cámara ha sido una tarea sencilla tras la lectura de los manuales que el propio fabricante aporta.

El principal requisito del trabajo ha sido llevado a cabo mediante la escritura del código. El programa es capaz de implementar mediante funciones el acceso a la cámara, configuración y adquisición de datos de forma sencilla y transparente para el usuario.

Finalmente se puede concluir que el sistema funciona correctamente y es posible trabajar con la cámara ToF elegida, obteniendo de ésta la información de profundidad, intensidad y confianza, que puede ser mostrada y almacenada. La aplicación desarrollada permite configurar los parámetros de la cámara, así como dejar algunos (o todos ellos) en sus valores por defecto para usuarios menos avanzados. Cada vez que se pone en funcionamiento, es posible modificar la configuración. El almacenamiento es opcional, por lo que simplemente mostrando la imagen podemos observar si la configuración elegida es la que deseamos.

La información adquirida es almacenada en carpetas lo que facilita el uso de estas imágenes en cualquier instante. También existe la posibilidad de adquirir secuencias de imágenes. En cualquiera de los casos, dichas imágenes se almacenan en una carpeta con la fecha y hora del primer *frame*. El formato de almacenamiento elegido facilita su uso en cualquier Sistema Operativo, pudiéndose visualizar mediante cualquier programa convencional visor de imágenes.

Las funciones que se encargan de visualizar y guardar las imágenes pueden ser usadas en otros trabajos con la única condición de emplear la librería OpenCV.

En este trabajo las librerías utilizadas han sido las proporcionadas por el fabricante de la cámara, las librerías básicas del lenguaje de programación utilizado (C++), y para el tratamiento de imágenes se ha usado OpenCV. Esta última librería especializada en la visión artificial e

imágenes permite crear las matrices necesarias para presentar las capturas realizadas por la cámara. El software se ha llevado a cabo en el entorno de desarrollo *Eclipse neon*.

El código ha sido programado de manera que sea posible añadir algoritmos de procesamiento de datos, nuevas funciones y diferentes cámaras. Se ha escogido la librería OpenCV para conseguir la máxima portabilidad en otros Sistemas Operativos.

Concluimos este trabajo destacando que las pruebas realizadas para comprobar el funcionamiento del sistema han sido satisfactorias. En ellas se ha configurado la cámara, se han grabado secuencias, tomado imágenes, se han mostrado las fotografías y se han almacenado. Esto ha demostrado el cumplimento de los objetivos propuestos.

## 4.2 Trabajos futuros

Una vez concluida esta primera interfaz, quedan pendientes líneas de trabajo como pueden ser la adquisición de datos en archivos de texto, la incorporación de algoritmos de mejora de la información obtenida, así como el uso de funciones extras. Otro trabajo futuro posible, sería la incorporación de otras cámaras a la aplicación, de esta manera, se podría acceder a todas ellas desde una única interfaz.

La adquisición de datos resulta interesante pues es la manera más precisa de obtener los resultados de profundidad que nos proporciona la cámara. Además, una vez que se conocen esos datos es posible aplicar algoritmos de mejora, algunos ejemplos interesantes son la detección de bordes o la eliminación de "*Flying pixels*". Además, sería interesante la calibración de los errores sistemáticos detallados en esta memoria, y la reducción de los errores aleatorios.

Otro trabajo interesante para el futuro sería la creación de una interfaz gráfica basada en el programa aquí desarrollado. De esta manera de mejoraría la interacción del usuario con la cámara.

Tras facilitar la adquisición de imágenes, sería posible trabajar con esta cámara en la detección de personas o incluso en alguna aplicación de robótica móvil [4].

# 5 Pliego de condiciones

Para obtener un buen funcionamiento de la aplicación creada es necesario un software y un hardware que reúna los requisitos básicos. Estos requisitos están determinados por las características de la cámara usada.

## 5.1 Requisitos de Hardware

### 5.1.1 Requisitos mínimos

El hardware básico necesario debe de ser capaz de leer las librerías utilizadas en el programa y tener memoria suficiente para la adquisición de imágenes.

Los requisitos mínimos de hardware establecidos son los siguientes, siendo el sistema operativo usado durante todo el trabajo Linux Ubuntu 16.04 LTS [13].

- Procesador Intel Core 2 Duo.
- 2 GB de memoria RAM.
- Tarjeta gráfica que soporte las funciones de la librería OpenCV.
- Pantalla con resolución grafica mínima de 1024x768 pixeles. El tamaño máximo de imágenes recogidas por la Basler ToF es de 640x480 Pixels.
- Puerto Ethernet para la conexión y trabajo con la cámara.
- Un mínimo de 10 GB de disco duro, desglosado como se indica a continuación:
    - 870 kB de disco duro para la carpeta que contiene el código fuente.
    - 509,4 kB para el ejecutable.
    - 6 GB para las librerías propias de C++, las de la cámara y la instalación de O*penCV.*
    - El resto, constituye el espacio mínimo para poder almacenar las imágenes adquiridas.

### 5.1.2 Hardware de referencia

El ordenador donde se ha realizado el trabajo está ubicado en el laboratorio GEINTRA, Oeste-201 de la Escuela Politécnica Superior.

Sus características son las siguientes:

- Ordenador de sobremesa modelo Intel NUC BOXNUC5i3RYH.
- 16 GB de memoria RAM DDR3L
- Procesador 64 bits Intel Core i7-5557U con CPU a 3.10GHz x 4.

- Tarjeta gráfica Intel Graphics 5500
- Pantalla LCD 23'6" con resolución 1920x1080 pixeles.
- 2 puertos USB 2.0
- Interfaz de red Ethernet estándar.

### 5.1.3 Cámara de Tiempo en Vuelo

Para comprobar el funcionamiento y poder utilizar la aplicación creada es necesario utilizar la cámara BaslerToF. A continuación, se indica el modelo exacto y el fabricante de esta.

- BaslerToF: modelo 21886650. Fabricante: Basler [10].

La interfaz desarrollada está preparada para poder utilizar este dispositivo a través de la conexión Ethernet.

## 5.2 Requisitos de Software

Antes de ejecutar la aplicación es necesario cubrir todas las exigencias de software.

- Sistema operativo Linux 64 bits. Se recomienda Linux Ubuntu 16.04 LTS [13] o Superior.
- Eclipse (Entorno de desarrollo) versión neon o superior. Eclipse es un IDE multiplataforma de código abierto. Útil en lenguajes de programación como C o Java [14].
- Librería OpenCV 2.4.13 o superior. Es una biblioteca libre multiplataforma de tratamiento de imágenes comúnmente utilizada en visión artificial, basada en C/C++ [15].
- Driver de la cámara BaslerToF, versión 1.4.0 o superior.
- Procesador de textos.

### 5.2.1 Versión de referencia de software

El software de referencia es el establecido para el desarrollo y posterior prueba de la aplicación.

Esta versión consiste en lo siguiente:

- Ubuntu 16.04 LTS [13].
- Eclipse, neon.2 [14].
- OpenCV 2.4.13 [15].
- Driver de la cámara BaslerToF, versión 1.4.0 [10]
- Procesador de textos Microsoft Office 365 (64 bits).

# 6 Presupuesto

En este apartado se realiza el presupuesto general de los costes que conlleva este trabajo.

## 6.1.1 Costes de equipamiento hardware

| CONCEPTO | PRECIO UNITARIO | CANTIDAD | SUBTOTAL |
|---|---|---|---|
| **Ordenador de sobremesa (Intel NUC BOXNUC5i3RYH )** | 240 € | 1 | 240 € |
| **Pantalla LCD** | 174€ | 1 | 174€ |
| **Cámara ToF Basler** | 2340 € | 1 | 2340 € |
| **TOTAL** | | | **2754 €** |

*Tabla 2. Costes de equipamiento hardware*

## 6.1.2 Costes de equipamiento software

| CONCEPTO | PRECIO UNITARIO | CANTIDAD | SUBTOTAL |
|---|---|---|---|
| **Sistema Operativo Ubuntu 16.04** | 0 € | 1 | 0 € |
| **Eclipse** | 0 € | 1 | 0 € |
| **Librería OpenCV** | 0 € | 1 | 0 € |
| **Driver y Librerías de la cámara** | 0 € | 1 | 0 € |
| **Software de edición de documentos Microsoft Word (Licencia gratuita, Alumno UAH)** | 0 € | 1 | 0 € |
| **TOTAL** | | | **0 €** |

*Tabla 3. Costes de equipamiento software*

### 6.1.3 Costes de tiempo empleado

| CONCEPTO | PRECIO UNITARIO | CANTIDAD | SUBTOTAL |
|---|---|---|---|
| **Desarrollo del software/interfaz** | 60 €/hora | 320horas | 19.200 € |
| **Creación del documento-memoria** | 12 €/hora | 120 horas | 1.440 € |
| **TOTAL** | | | **20.640 €** |

*Tabla 4. Costes de tiempo empleado*

El número de horas que han sido necesarias para llevar a cabo este trabajo se resume en 4 meses a media jornada de 4 horas diarias. La creación de esta memoria ha sido posible gracias al trabajo durante 15 días a jornada completa de 8 horas.

### 6.1.4 Coste total del presupuesto de ejecución material

**Concepto Subtotal**

| CONCEPTO | PRECIO UNITARIO | CANTIDAD | SUBTOTAL |
|---|---|---|---|
| **Costes de equipamiento hardware** | 2754 € | 1 | 2754 € |
| **Costes de software** | 0 € | 1 | 0 € |
| **Costes de tiempo empleado** | 20.640 € | 1 | 20.640 € |
| **TOTAL** | | | **23.394 €** |

*Tabla 5. Coste total*

**El importe total del trabajo asciende a la cantidad de:**

**"Veintitrés mil trescientos noventa y cuatro euros, IVA incluido"**

Alcalá de Henares, 11 de julio de 2018

Fdo: María Higuera Pinillos

Graduada en Ingeniería en Electrónica y Automática Industrial

# 7 Manual de usuario

En esta sección se detallan los requisitos básicos para el uso de esta aplicación, así como los pasos necesarios para manejar de manera óptima la interfaz desarrollada.

Mediante sencillas tareas, se puede acceder a la configuración de la cámara, adquirir imágenes y decidir si almacenarlas o no.

## 7.1 Requisitos previos al uso de la aplicación

Los requerimientos *hardware* y la elección del Sistema Operativo deben ser cubiertos antes de comenzar a usar esta aplicación. Además, es importante tener instaladas y actualizadas las siguientes librerías:

- *OpenCV* (Versión 2.4.3 o superior)
  Comprobar que las funciones usadas en este trabajo no cambien en las versiones más actuales [15].
- Driver de la cámara Basler ToF [10]
  Disponible en la web del fabricante al igual que la guía de programación, el manual de usuario de la cámara y la guía de instalación y configuración.

Una vez instaladas las librerías necesarias para la correcta ejecución de la aplicación es necesario crear el ejecutable que permite comunicar la cámara con el usuario vía la terminal de comandos.

Para ello es necesario compilar el proyecto, esto es posible gracias al *Makefile* creado en el entorno de desarrollo Eclipse [14]. Es recomendable no manipular este fichero.

Tras copiar la carpeta ***BaslerAPI*** que contiene el código fuente en la carpeta *workspace* de Eclipse, es posible abrir la aplicación como un proyecto en este entorno de desarrollo. Para su compilación basta con abrir el archivo ***BaslerAPIMain.cpp*** dentro de Eclipse y pulsar ***Ctrl+B***. Esta es la manera más directa y rápida de generar el ejecutable. Otra opción es pulsar sobre el icono con forma de *martillo (build)* que aparece en la parte superior. Una vez realizada esta tarea, se obtiene el ejecutable. Este va a guardarse en la siguiente ruta:

`…/workspace/BaslerAPI/src`.

En caso de recibir errores tras la compilación es posible que las causas sean las siguientes:

Las propiedades del proyecto son erróneas y es necesario hacer modificaciones sobre ellas para eliminar los errores. Un error común es la dirección donde se crea el ejecutable.

Las librerías necesarias no son encontradas. Esto puede deberse a que la instalación no ha sido la correcta o que las direcciones señaladas en las propiedades del proyecto no son reales.

## 7.2 Conexión de la cámara

Antes de ejecutar la aplicación es necesario conectar y alimentar la cámara, para ello, conectaremos el cable *Ethernet* en el conector GigE de la cámara.

El ordenador donde vamos a poner en marcha el ejecutable creado anteriormente debe de estar conectado a la misma red que la cámara. Una vez realizada esta conexión, enchufamos el conector de 12 pines a la cámara y el otro extremo a la red eléctrica. Estas conexiones son mostradas en la Figura 30. En caso de alimentarlo mediante una fuente de tensión debemos asegurarnos de que esta entrega la tensión necesaria para el correcto funcionamiento de la cámara.


*Figura 30. Conexionado de la cámara*

## 7.3 Puesta en marcha de la aplicación

Tras conectar la cámara, el equipo ya está listo para ejecutar la aplicación. Deben de seguirse los siguientes pasos:

1. Abrir la terminal de comandos.
2. Introducir la ruta donde se encuentra la carpeta que contiene el ejecutable con el comando *"cd"*.

   `…/workspace/BaslerAPI/src`

3. Una vez dentro de esta carpeta, se ejecuta el ejecutable creado anteriormente mediante compilación del programa.

   `"./BaslerAPIMain argumento_1 argumento_2 . . . argumento_n"`

   Es importante añadir un espacio después de cada argumento. En la *Figura* 31 se pueden ver los pasos seguidos.

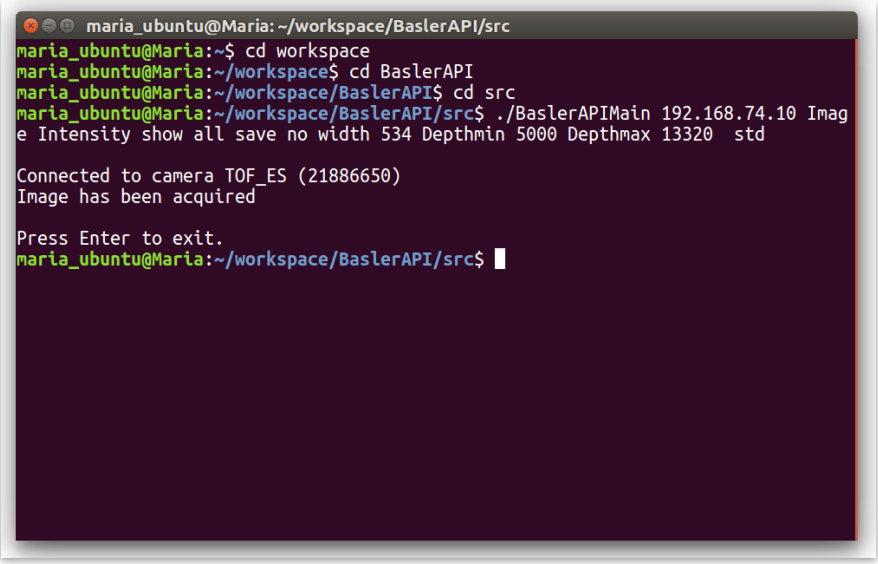



*Figura 31. Pasos para la ejecución de la aplicación*

4. El primer argumento es la dirección IP de la cámara. Este es el único argumento **obligatorio**. Indica la cámara con la cual se va a trabajar.

   Ej.: `192.168.74.10`

5. El resto de los argumentos se encargan de configurar la cámara, seleccionar el formato, visualizar y almacenar las imágenes. Estos parámetros no son obligatorios, no importa el orden en el que son introducidos y en caso de no indicar algunos de ellos, estos

tomaran su valor por defecto. Pueden ser escritos tanto en mayúsculas como en minúsculas.

A continuación, se indican cuáles son, sus valores por defecto y algunos ejemplos de cómo se utilizan.

En primer lugar, hay que decidir si se quiere tomar una única imagen o grabar una secuencia.

a. `Image` o `video`. En caso de seleccionar video, la captura de imágenes parará cuando se pulse alguna tecla sobre la ventana en la cual se está mostrando la secuencia.

El formato de la imagen que se quiere adquirir desde la cámara se decide con los siguientes argumentos:

b. `Range`, en caso de querer una imagen de profundidad. Este formato admite tres tipos de representación. Puede ser seleccionado a la vez que `Intensity`. Una vez escrito `Range`, a continuación, debemos seleccionar uno de estos tres tipos:

   i. `Coord3D_C16`: Se adquiere una imagen en blanco y negro cuyas zonas oscuras representan lejanía y las claras proximidad.

   ii. `RGB8`: Se usan los colores rojo, verde y azul para mejorar la visualización. Los objetos más próximos se representan en rojo, los mas distantes en azul.

   iii. `Coord3D_ABC32f`: Se representa la coordenada Z, una vez separada de x e y. El resultado es el mismo que Coord3D_C16 pero en rojo y negro.

`Intensity`, si preferimos una imagen de intensidad. Mediante esta imagen podemos comprobar si existen pixeles sobresaturados. Puede ser seleccionado a la vez que `Range`.

`Confidence`, si se quiere una imagen de confidencia. La cámara únicamente adquiere este formato de fiabilidad de la medida cuando no se quiere adquirir ninguno de los anteriores.

A continuación, se configuran ciertos aspectos importantes de la cámara. Comenzamos con la región de interés.

a. El tamaño de las imágenes adquiridas puedes ser modificado mediante el argumento `Width`. Tras este parámetro debe indicarse la anchura que interesa, teniendo en cuenta que el máximo son 640 *pixels*. La altura será modificada de la misma manera mediante el parámetro `Height`.

b. Una vez decidido el tamaño de la región de interés, es posible seleccionar la parte de la escena en la que se quiere encuadrar la región de interés. Para ello, hay que escribir `OffsetX` y a continuación el número de pixeles que es necesario desplazarse para alcanzar la zona de interés. Del mismo modo con el eje Y, primero se pone `OffsetY` y seguidamente el número de pixeles. La Figura 32 muestra el funcionamiento de este parámetro.

   Hay que tener en cuenta que, en función del tamaño seleccionado, la región de interés podrá desplazarse mas o menos, es por ello, que si el tamaño es 640x480 *pixels*, el valor de `OffsetX` y `OffsetY` debe ser 0.

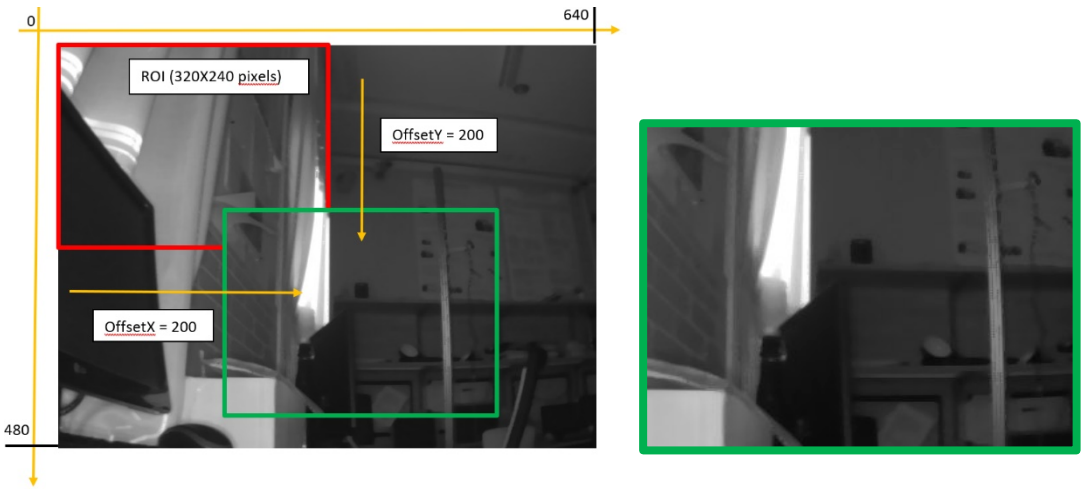



Figura 32. *Ejemplo de uso de los parámetros de desplazamiento*

c. Mediante los siguientes argumentos se permite medir la profundidad de una escena dentro de un rango determinado. Para ello escribiremos en la terminal `DepthMin` y la distancia a la que se comienza a medir la profundidad, con `DepthMax` y un valor en milímetros a continuación, se configura la distancia donde se deja de tener en cuenta la profundidad. Este rango está entre 0 y 13320 mm. La Figura 33 muestra la función de este parámetro.

*Figura 33. Izq. Rango elegido entre 0 y 13.320 m. Dcha. Rango elegido entre 0 y 3 m*

El siguiente argumento es importante cuando se quiere adquirir una secuencia de imágenes. En función de la frecuencia de adquisición de las imágenes, será el tiempo existente entre *frames*.

d. Para configurar esta frecuencia es necesario escribir `fps` y justo después la frecuencia que nos interesa. Hay que tener cuidado con este valor pues en caso de ser un valor muy alto, algunas imágenes no tendrán tiempo de ser almacenadas. Esta cámara no soporta más de 20 Hz.

El código escrito nos da la opción de almacenar las imágenes adquiridas. Habría que indicarlo de la siguiente manera:

e. El argumento que nos da la opción de guardar o no las imágenes es `save`. En caso de querer almacenar todas las imágenes pedidas, se pondrá `save yes`. Si por el contrario no nos interesa guardarlas habrá que escribir `save no`.

La última opción que nos da este ejecutable es la de visualizar o no las imágenes. Para ello hay que realizar lo siguiente:

f. Escribiendo `show` se permite o no visualizar las imágenes según los formatos elegidos. Seguidamente podrán escribirse los siguientes argumentos:
    i. `No`. No se mostrará ninguna imagen.
    ii. `Range`. Se visualizarán las imágenes de profundidad.
    iii. `Intensity`. Serán mostradas las de intensidad.

iv. `Confidence`. Únicamente aparecerán las de confidencia.

v. `All`. Se mostrarán todas las imágenes.

La Figura 34 resume todos los posibles modos, formatos de imagen y configuraciones de la cámara.

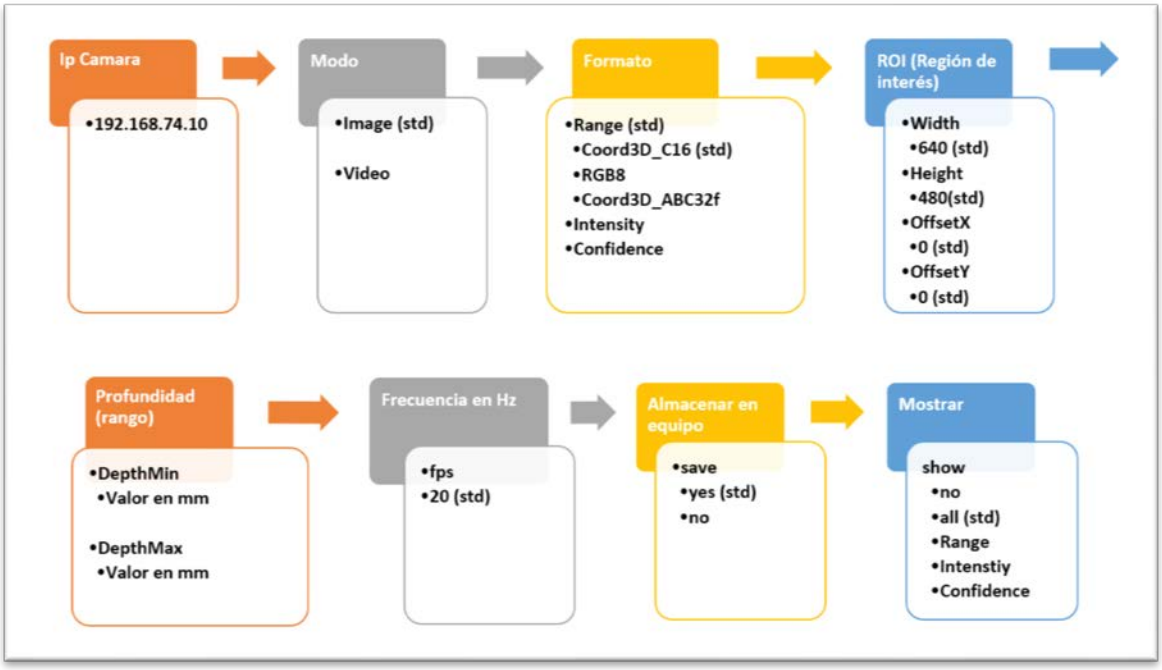



*Figura 34. Esquema de funcionamiento*

En caso de no querer modificar alguno de estos parámetros, será necesario escribir al final el argumento `std`. Con esto, se indica que el resto de los valores que no se han configurado tomarán su valor por defecto.

Estos valores estándar son los siguientes:

- En caso de no indicar si queremos tomar una imagen o grabar una secuencia, se seleccionará por defecto una única imagen.
- El formato estándar es *Range Coord3D_C16*.
- Respecto al tamaño, el valor por defecto es 640x480 *pixels*.
- El rango de profundidad abarca entre 0 y 13320 como profundidad máxima.
- 20 Hz es el valor estándar del parámetro *fps*.
- Salvo cambios, se almacenarán todas las imágenes solicitadas a la cámara y también serán mostradas por pantalla.

## 7.4 Funcionamiento de la aplicación

Las siguientes líneas muestran cómo debe usarse esta aplicación. Además, se indicarán ejemplos no válidos para evitar posibles confusiones.

```
./BaslerAPIMain 192.168.74.10 image Range RGB8 Width 534 show range std
```

*Ejemplo 1. Imagen de profundidad RGB*

Tras introducir la sentencia anterior en la terminal de comandos, obtenemos una imagen, Ejemplo 1, de profundidad en tonos rojos, azules y verdes. Cuya anchura son 543 pixels. Esta imagen será mostrada en pantalla y por defecto, almacenada en el equipo.

```
./BaslerAPIMain 192.168.74.10 video Range RGB8 Width 534 show range std
```

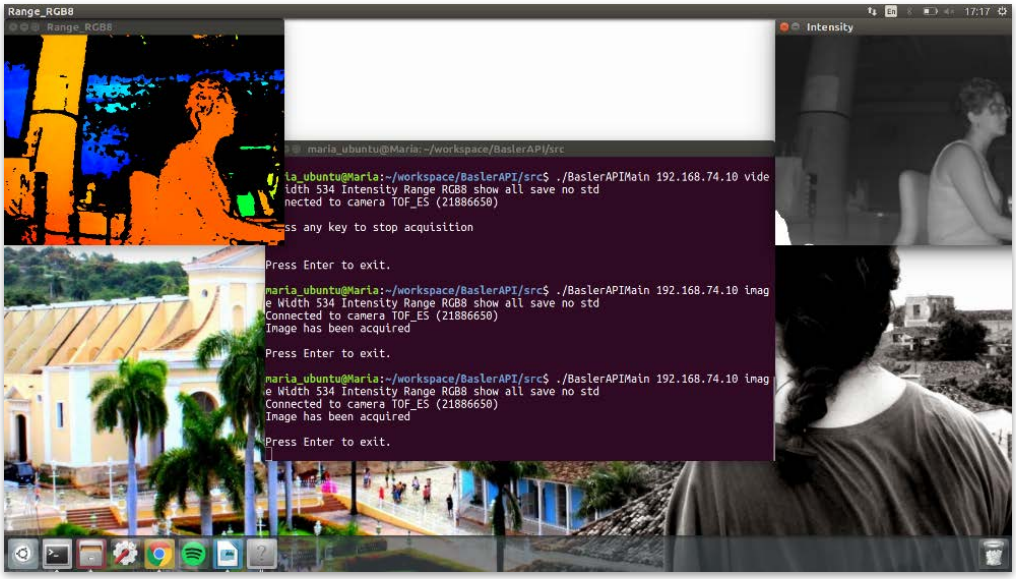
*Ejemplo 2. Secuencia de profundidad RGB*

Con esta línea obtendremos una secuencia de imágenes, Ejemplo 2, con las mismas características que la anterior. Cada 50 ms la cámara capturará un nuevo *frame* pues la frecuencia (fps) no ha sido modificada, toma su valor estándar, 20 Hz.

```
./BaslerAPIMain 192.168.74.10 video Range RGB8 Width 534 show range fps 14 std
```

Mismo caso que el anterior pero la frecuencia ha sido reducida a 14 Hz.

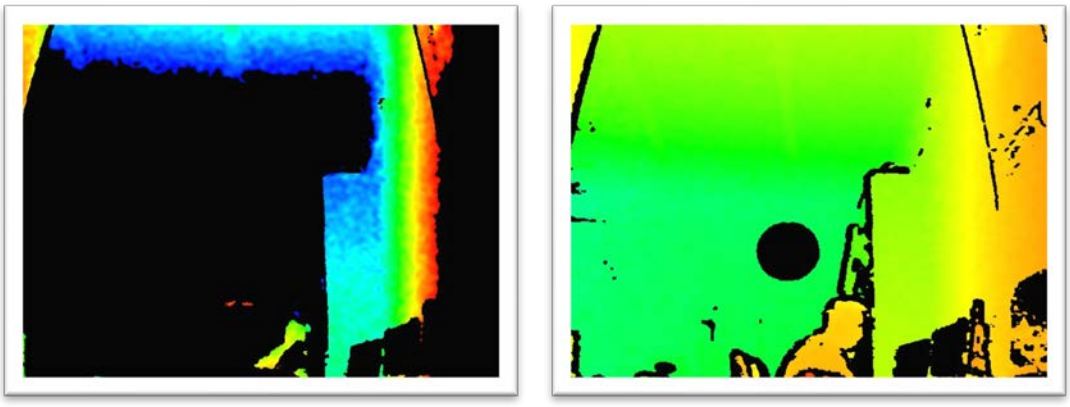
```
./BaslerAPIMain 192.168.74.10 image Width 534 Intensity Range RGB8 show all save no std
```




*Ejemplo 3. Muestra de imágenes de profundidad e intensidad*

En este caso se solicitan dos imágenes, una de intensidad y otra de profundidad de tipo RGB, ambas imágenes son mostradas, pero no almacenadas tal y como se puede ver en el Ejemplo 3.

```
./BaslerAPIMain 192.168.74.10 image RGB8 depthmin 3000 depthmax 5000 show no save yes std
```

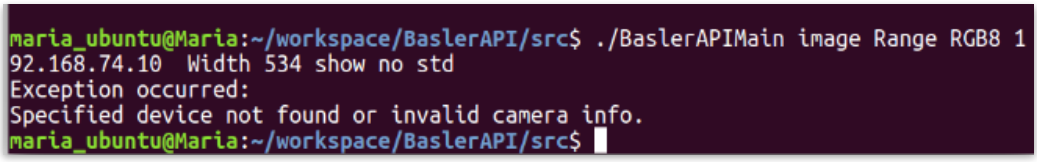


*Ejemplo 4. Izq. Rango de profundidad entre 3 y 5 m. Dcha. Rango estándar.*

Se solicita una imagen de intensidad, tamaño estándar, cuya región de interés queda limitada entre los 3 y 5 m de profundidad, esto queda reflejado en el Ejemplo 4. No se muestra la imagen, pero si queda almacenada.

**Ejemplos erróneos**

```
./BaslerAPIMain image Range RGB8 192.168.74.10  Width 534 show no std
```



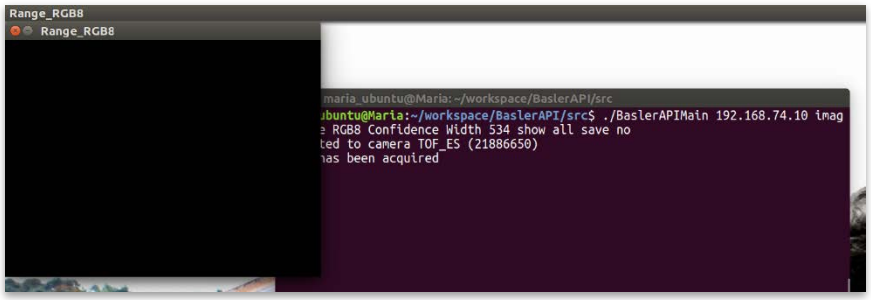

*Ejemplo 5. Error, dirección IP mal colocada*

Con esta sentencia la aplicación devolverá un error, Ejemplo 5. La dirección IP debe de ser el primer argumento pues es necesaria para reconocer la cámara con la cual se va a trabajar. A continuación, se detalla la sentencia correcta:

```
./BaslerAPIMain 192.168.74.10 image Range RGB8 Width 534 show no std
```

Respecto al formato Confidence hay que tener en cuenta:

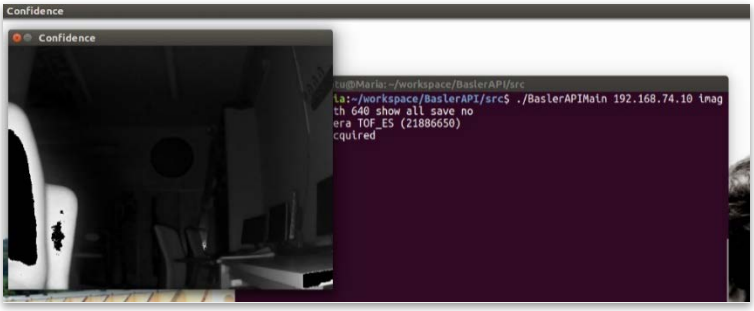
```
./BaslerAPIMain 192.168.74.10 image Range RGB8 Confidence Width 534 show all save no std
```




*Ejemplo 6.Error, imagen de confidencia no es compatible con los otros dos formatos*

En caso de solicitar una imagen de confianza la cámara no permite obtener conjuntamente cualquier de los otros dos formatos, Ejemplo 6. La manera correcta es la siguiente (Ejemplo 7):

```
./BaslerAPIMain 192.168.74.10 image Confidence Width 534 show all save no std
```




*Ejemplo 7. Error solucionado, obtenemos la imagen de confidencia*

Una vez que la cámara ha adquirido la imagen solicitada o ha sido paralizada la adquisición de imágenes, el propio programa cierra la cámara y espera a que el usuario pulse intro para salir.

Esta aplicación es bastante flexible. Únicamente debe cumplirse:

- La dirección IP tiene que ser el primer argumento.
- En caso de no modificarse todos los parámetros debe incluirse al final std.
- Todos los datos deben ir en la misma línea separados por un espacio.

En cambio, debe destacarse lo siguiente:

- Los nombres de datos son insensibles a mayúsculas.
- Pueden ir en cualquier orden a excepción de los citados anteriormente.
- En caso de aparición múltiple del mismo parámetro, solo el último valor será tenido en cuenta.

## 7.5 Almacenamiento de imágenes

La información de profundidad, intensidad y confianza es almacenada en ficheros de imagen reconocidos por *OpenCV*. Las imágenes son guardadas en formato PNG *(Portable Network Graphics)*, con la extensión ".png".

Las imágenes son almacenadas en carpetas creadas cada vez que se ejecuta el programa. El nombre de estas carpetas indica si es imagen o video, y además indican la fecha y la hora en la que han sido tomadas las imágenes. Estas carpetas están situadas en la misma carpeta donde se encuentra el ejecutable.

Una vez dentro de las carpetas, las imágenes se almacenan mediante un nombre descriptivo y una secuencia numérica. Es decir, las imágenes de intensidad en modo video quedan guardadas como *intensity_img1, intensity_img2, …, intensity_imgn*.

# Bibliografía

[1] R. Lange and P. Seitz, “Solid-state time-of-flight range camera”, IEEE J. Quantum Electron., vol. 37, no. 3, pp. 390–397, 2001.

[2] J. Sell and P. O'Connor, "The Xbox One System on a Chip and Kinect Sensor," in IEEE Micro, vol. 34, no. 2, pp. 44-53, Mar.-Apr. 2014.

[3] S. Lee, “Depth camera image processing and applications,” in 19th IEEE International Conference on Image Processing, 2012, pp. 545–548.

[4] R. A. El-iaithy, J. Huang, and M. Yeh, “Study on the Use of Microsoft Kinect for Robotics Applications,” in 2012 IEEE/ION Position Location and Navigation Symposium, 2012, pp. 1280–1288.

[5] J. R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez, “Improving Human Face Detection through TOF Cameras for Ambient Intelligence Applications,” Appl. Adv. Intell. Soft Comput. Ambient Intell. - Softw. Appl., vol. 92, pp. 125–132, 2011.

[6] M. Hansard, S. Lee, O. Choi and R. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*, Springer (noviembre 2012)

[7] D. Jimenez, D. Pizarro, M. Mazo and S. Palazuelos, *Modelling and correction of multipath interference in time of flight cameras*, Image and Vision Computing (2013)

[8] J. Sell and P. O'Connor, "The Xbox One System on a Chip and Kinect Sensor," in *IEEE Micro*, vol. 34, no. 2, pp. 44-53, Mar.-Apr. 2014. (doi: 10.1109/MM.2014.9).

[9] C. S. Bamji *et al*., "A 0.13 µm CMOS System-on-Chip for a 512 × 424 Time-of-Flight Image Sensor With Multi-Frequency Photo-Demodulation up to 130 MHz and 2 GS/s ADC," in *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 303-319, Jan. 2015.

[10] BASLER, the power of sight, “Basler ToF Camera, User´s Manual”, AW001338, v7 english, 7 November 2016. https://www.baslerweb.com/en/products/software/tof-software/ (Junio 2018)

[11] MESA Imaging AG, *SR4000 User Manual, 2012.*

[12] StarForm, “3D Time-of-Flight Camera, User Guide”, v1.3.0, September 21, 2015.

[13] Sistema Operativo Linux Ubuntu. https://www.ubuntu.com/ (Julio 2018)

[14] Entorno de desarrollo Eclipse neon: https://www.eclipse.org/neon/ (Julio 2018)

[15] Librería OpenCV: https://opencv.org/ (Junio 2018)

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR