UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Adquisición y tratamiento de imágenes mediante sensor estereoscópico en un sistema embebido basado en Linux. Ejemplos de aplicación

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Gomez Gonzalez, Raul

Tutor/a: Blanes Noguera, Juan Francisco

Cotutor/a: Simó Ten, José Enrique

CURSO ACADÉMICO: 2022/2023

ADQUISICIÓN Y TRATAMIENTO DE IMÁGENES MEDIANTE SENSOR ESTEREOSCÓPICO EN UN SISTEMA EMBEBIDO BASADO EN LINUX. EJEMPLOS DE APLICACIÓN

Autor
Raúl Gómez González

Directores: **Blanes Noguera, Juan Francisco**
Simó Ten, José Enrique

TRABAJO FIN DE MÁSTER
MÁSTER DE AUTOMÁTICA E INFORMÁTICA INDUSTRIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Valencia, Curso 2022-2023

“Porque hasta los campeones del mundo necesitan penaltis para alzarse con la copa”

AGRADECIMIENTOS

A Patri, mi hoyu

ÍNDICE

RESUMEN

La visión por computador es una de las ramas de la tecnología y la ciencia que mayor crecimiento ha experimentado durante las últimas décadas. Nuevos descubrimientos en otras áreas como las redes neuronales o la inteligencia artificial han permitido mejorar las técnicas de procesamiento y el desarrollo de los algoritmos utilizados en visión. Estas mejoras han espoleado la inclusión de la visión por computador en numerosos espacios de trabajo, desde industriales a domésticos, estando hoy día presente desde en los robots más modernos como en el simple telefonillo de una vivienda.

Sin embargo, tan solo un nicho muy específico de los sistemas de visión contempla la adquisición de información de la profundidad de la imagen, estando por norma general muy orientados a procesos industriales o dispositivos tecnológicos avanzados como robótica.

Este proyecto tiene como finalidad el diseño y puesta en marcha de un sensor de visión estereoscópico basado en StereoPi que nos permita obtener la profundidad de los distintos elementos que contenga la imagen.

Un sistema estereoscópico es aquél formado por, al menos, dos cámaras el cual es capaz de proporcionar información tridimensional del entorno. Es decir, un sistema estéreo es capaz de indicar la distancia a la que se encuentran los objetos respecto a las cámaras que obtienen la imagen.

En este proyecto se detallarán los procesos de captura y procesamiento de las imágenes suministradas por el sensor, la obtención de la profundidad de los elementos que las componen, la trasmisión de la información desde el sensor hasta la aplicación de usuario, y la gestión de dicha aplicación para suministrar información de utilidad al usuario.

ABSTRACT

Computer vision is one of the branches of technology and science that has experienced the greatest growth in recent decades. New discoveries in other areas such as neural networks or artificial intelligence have allowed us to improve processing techniques and the development of algorithms used in vision. These improvements have spurred the inclusion of computer vision in numerous work spaces, from industrial to domestic, being present today in the most modern robots as well as in the simple telephone in a home.

However, only a very specific niche of vision systems contemplates the acquisition of image depth information, being generally very oriented to industrial processes or advanced technological devices such as robotics.

The purpose of this project is the design and implementation of a stereoscopic vision sensor based on StereoPi that allows us to obtain the depth of the different elements contained in the image.

A stereoscopic system is one formed by at least two cameras which is capable of providing three-dimensional information about the environment. That is, a stereo system is capable of indicating the distance at which objects are located with respect to the cameras that obtain the image.

This project will detail the processes of capturing and processing the images supplied by the sensor, obtaining the depth of the elements that compose them, transmitting the information from the sensor to the user application, and managing said information. application to provide useful information to the user.

OBJETIVOS Y PLIEGO DE CONDICIONES

El objetivo general de este trabajo es el diseño y puesta en marcha de un sensor de visión estereoscópico que proporcione imágenes en tiempo real así como la información tridimensional de los píxeles que la componen. Se requiere además que el sensor sea operado a través de una aplicación de usuario ejecutada desde un sistema externo como un ordenador.

Afrontamos este trabajo desde la perspectiva de obtener un sensor de visión 3D que sirva como punto de partida para diferentes aplicaciones que puedan aprovechar los datos de profundidad de un streaming de video en tiempo real.

Por ellos las imágenes obtenidas y procesadas deben enviarse a la aplicación cliente en varios modos de trabajo diferentes. Cada modo de trabajo tendrá una función específica y mostrará una determinada información en la aplicación cliente.

- Imagen original. El sensor enviará la imagen original rectificada sin ningún otro tipo de procesamiento.
- Imagen original y matriz de posición. El sensor enviará para cada fotograma la imagen original y su matriz correspondiente con los datos de profundidad de cada píxel.
- Mapa de profundidad. El sensor enviará la imagen de un mapa de profundidad. En este mapa el color de cada píxel vendrá determinado por un color en función de la distancia a la que se encuentre el objeto del sensor.
- Imagen original y distancia del punto central. El sensor enviará la imagen original rectificada a la que se añadirá un marcador numérico que indicará la distancia en (cm) del objeto que haya en el punto central de la imagen.

Por último, debemos tener en cuenta que el sensor debe ser operable por una aplicación remota por lo que debe contar con los medios necesarios para conectarse por red con el dispositivo que ejecute la aplicación cliente.

ESTADO DEL ARTE

- ***Visión***

Capacidad de interpretar el entorno gracias a los rayos de luz que alcanzan al ojo. Sentido que le brinda a distintos organismos la posibilidad de detectar la luz y reconocer lugares, personas y objetos.

- ***Visión por computador***

Ciencia que desarrolla las bases teóricas y algorítmicas para obtener información sobre el mundo real a partir de una o varias imágenes.
(Escalera)

Rama de la inteligencia artificial, cuyo objetivo es proveer del sentido de la vista a un computador o robot para que éstos puedan interactuar de forma más eficiente en ambientes complejos.

- ***Visión 3D***

Denominamos visión tridimensional o 3D a la capacidad de percibir el mundo en las tres dimensiones espaciales. La visión tridimensional consiste en la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de ese mundo.

Los seres humanos poseemos la capacidad innata de observar el mundo en tres dimensiones, lo que nos permite calcular mentalmente las distancias y situar los objetos en el espacio. La sincronización de nuestros ojos posibilita que el cerebro fusione en una sola imagen las procedentes de ambos ojos, logrando una única imagen en tres dimensiones.
En el ámbito de la tecnología, nos referimos con visión 3D al conjunto de técnicas que se encargan de proporcionar la capacidad de emular la visión humana a un ordenador o dispositivo. Con dicha capacidad este dispositivo será capaz de generar un modelo tridimensional de un objeto o escena.

Historia e hitos en la visión por computador

La ciencia que hoy conocemos por visión artificial o visión por computador, aun siendo una rama novedosa, lleva en desarrollo más de medio siglo. Las primeras investigaciones en este campo datan de mediados de los años cincuenta, en concordancia con la aparición de los primeros computadores de propósito general como el UNICAV y el ENIAC. Sin embargo la limitada capacidad de cómputo y recursos informáticos de estos sistemas hicieron que se abandonara esta línea de investigación al no lograr resultados relevantes.

Durante la década de los años setenta se produce una gran evolución tecnológica, evolucionan los materiales y los procesos de fabricación de los dispositivos informáticos.

Aparecen en el mercado una nueva gama de ordenadores que ya hacen uso de los primeros circuitos integrados lo que les permite unas características y capacidad de cómputo muy superiores a sus antecesores. En este punto, las investigaciones sobre la visión por computador son retomadas obteniendo resultados, ahora sí, significativos, consiguiendo establecer las primeras técnicas de detección de bordes. Podemos citar como ejemplo el trabajo del doctor Roberts en 1963 que demostró la posibilidad de procesar una imagen digitalizada para obtener una descripción matemática de los objetos de la escena. O el trabajo del doctor Wichman en 1967, que presentó un equipo de Cámara y ordenador que podía identificar objetos y posiciones en tiempo real.

En la siguiente década se desarrollan numerosas investigaciones sobre procesamiento de imágenes y se analizan diferentes metodologías para realizar análisis completos de imágenes a través un ordenador. Si bien cada metodología cuenta con técnicas y algoritmos muy diferentes, todas ellas pueden resumirse en un mismo proceso de funcionamiento que puede ser esquematizado en la siguiente imagen.

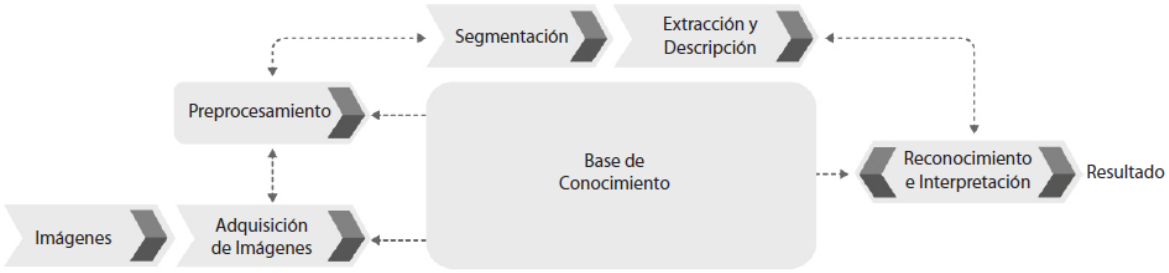


Figura. Esquema de procesos en un sistemas de visión por computador
(día, Número 10)

Las décadas de los ochenta y noventa son en pistoletazo de salida en el desarrollo a gran escala de la visión por computador, nacen las primeras aplicaciones funcionales fuera de laboratorio. Por otro lado estas aplicaciones necesitaban de un escenario "altamente controlado" para funcionar correctamente, por lo su campo de aplicación era muy reducido en escenarios comunes de la vida real. Podemos mencionar como ejemplo de esta dicotomía la proliferación de avances en el reconocimiento de caracteres, el cual tuvo buena aceptación en procesos de digitalización de documentos, pero sin embargo, no obtuvo buenos resultados a la hora de aplicarse al reconocimiento de matrículas de vehículos en vía pública.

Ya entrados en la década de los dos mil nacen dos nuevas metodologías que serán claves en el avance de la disciplina, hablamos del Machine Learning (ML) y del algoritmo de Transformada de Características Invariantes a Escala (SIFT en inglés).

La aplicación del Machine Learning provoca grandes beneficios relacionados directamente con el proceso de reconocimiento e interpretación de características visuales. El uso de Redes Neuronales comienza a postularse como una herramienta de utilidad en el procesamiento de imágenes.

Por otro lado las técnicas de SIFT ayudan a eliminar los problemas que dificultan el reconocimiento de elementos del entorno debido a su localización, como pueden ser cambios de posición, orientación o escala.

Estas dos metodologías permiten la proliferación de aplicaciones que pueden trabajar en entornos “no controlados”, o al menos con mucho menos control con el que trabajábamos en las décadas anteriores. Paralelamente la eficacia de resultados en procesos tales como identificación de elementos aumentan hasta tasas que permiten que la comercialización de los primeros productos y tecnologías al ámbito convencional.

Durante las dos últimas décadas se generaliza la implementación de las Redes Neuronales Convolucionales (RNAC). Este nuevo modelo permite crear “redes especializadas” lo que genera como resultado un aumento significativo en las tasas de eficacia y una disminución del grado de dependencia existente entre la efectividad y las características aleatorias del escenario. En este contexto se destaca que el procesamiento de una imagen a diferencia del procesamiento de diversas regiones de esta permite obtener información global de la imagen, lo que mejora y facilita la clasificación.

Hoy día la visión por computador, de la mano de las redes neuronales, sigue avanzando a pasos agigantados, generando soluciones tan fascinantes como Yout Only Look Once (YOLO) que permite reconocer y posiciones a tiempo real objetos de una imagen, escáneres tridimensionales de alta fidelidad desde aplicaciones móviles, y un sinfín más que están por llegar.

Finalmente, a modo de pincelada, mostramos una cronología con los descubrimientos de algunas técnicas y metodologías que fueron hitos relevantes en la evolución de esta ciencia.

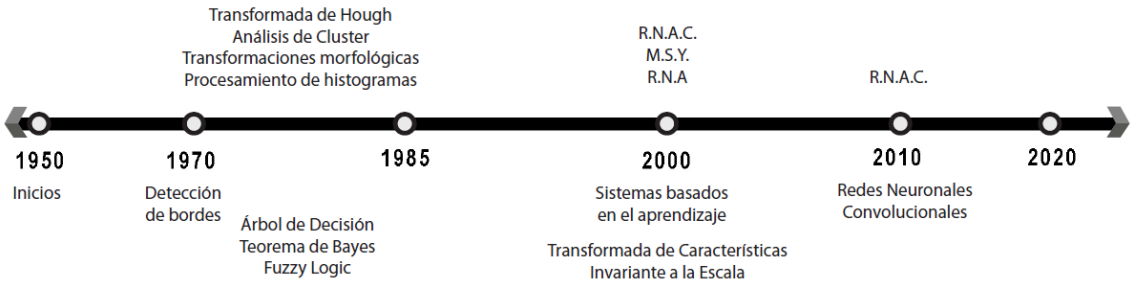


Figura. Cronología de la visión por computador

Métodos de obtención de modelos 3D

A la hora de obtener los datos que necesitamos para generar modelos del entorno que nos proporcione información tridimensional de los elementos que contiene nos encontramos

con una amplia variedad de métodos de adquisición. Los sistemas de adquisición son los encargados de capturar información de las superficies de los objetos que muestrean a partir de la cual somos capaces de generar un modelo tridimensional u obtener información de las tres dimensiones espaciales que lo componen.

Observamos una primera clasificación en dichas técnicas de adquisición en función de la interacción del sistema con la escena, diferenciando en técnicas activas o pasivas según si el sistema realiza o no algún tipo de actividad encaminada a intervenir en el entorno de la imagen.

Dentro de cada uno de estos grupos encontramos técnicas muy variadas y diferentes, cada una de las cuales está más orientada a unos determinados campos de aplicación.

Técnicas activas

Son aquéllas que cuentan con un sensor "activo" que está implicado en alguna parte de la operación de adquisición, es decir, que interfiere en la escena. Por norma general se basan en la emisión de haces controlados de energía, principalmente luz, desde una posición y orientación conocidas.
Los sensores de rango activos pueden utilizar una gran variedad de principios físicos: radares, sonoros, interferometría holográfica, enfoque o triangulación activa. Esta última es con diferencia la más extendida y utilizada en la mayoría de entornos y aplicaciones industriales.

Cabe mencionar otro tipo de técnicas activas que requieren de contacto físico con el entorno. En estos casos se hace uso de marcadores físicos o brazos robóticos mediante los cuales se determina la posición y dimensiones del objeto. Sin embargo, estas técnicas quedan completamente fuera de nuestro estudio al no contemplarse como sistemas de visión propiamente dichos.

Principio de triangulación en técnicas de adquisición activas

Los sensores de rango basados en el principio de triangulación activa utilizan una cámara junto de una fuente de emisión de un haz de luz que se proyecta sobre el objeto. La Cámara detecta en la imagen del objeto la posición donde se proyecta el haz emitido. A partir de este conjunto, y conocidas la dirección del haz de luz y la posición de la cámara respecto de la fuente del haz, se puede calcular una medida de la distancia en el espacio tridimensional desde el sensor al punto del objeto donde se proyecta el haz.

El análisis de la deformación que sufre el patrón al proyectarse sobre la superficie de los objetos de la escena permite extraer información tridimensional de éstos. Los patrones que

se utilizan en esta técnica son usualmente un punto o una línea generados a partir de una fuente de luz láser.

Esta técnica se basa en el cálculo de triángulos semejantes entre los elementos sensor óptico, emisor de luz y objeto de la escena. El proceso de cálculo se realiza estableciendo una relación de semejanza entre dos triángulos que comparten un vértice común: el punto focal de la cámara.

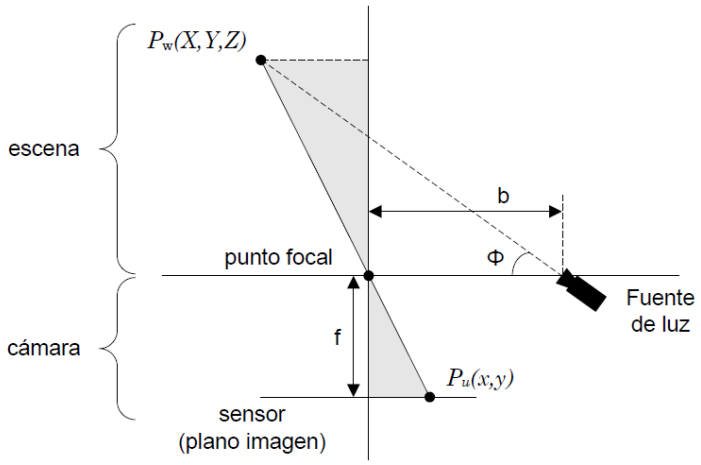


Figura. Relación de triángulos semejantes en triangulación activa

De esta relación entre triángulos semejantes podemos extraer las coordenadas del punto $P(X,Y,Z)$.

$$x * Z = X * f$$

$$y * Z = Y * f$$

Donde f es la distancia focal de la cámara y ϕ es el ángulo que forma la fuente de luz con respecto al plano de la cámara.

$$X = \frac{b}{f * \cot(\phi - x)} * x$$

$$Y = \frac{b}{f * \cot(\phi - x)} * y$$

$$Z = \frac{b}{f * \cot(\phi - x)} * f$$

Los métodos basados en el principio de triangulación activa tienen las ventajas de proporcionar mapas densos y precisos de coordenadas 3D, de no necesitar de una calibración de las cámaras y de no tener que resolver el problema de la correspondencia de puntos entre imágenes, etapa costosa y sujeta a una tasa de error demasiado elevada para muchas aplicaciones.

Es por tanto que las técnicas ópticas activas son una opción a considerar desde el punto de vista de la reconstrucción tridimensional de escenas y en ambientes industriales.

Muchos de los sistemas de visión de luz estructurada utilizan este proceso para el cálculo de coordenadas tridimensionales.

Luz estructurada

Los dispositivos de medición de luz estructurada se postulan como una de las solucionen más comunes y utilizadas tanto comercialmente como para entornos industriales. Son ampliamente usados en todo tipo de procesos como control de calidad, testing, reconstrucción 3D. Su alta precisión los hace especialmente útiles en estos campos. Como contrapartida estos sistemas presentan algunas limitaciones tales como la dependencia a la iluminación, interferencias y ruido, rango de profundidad que puede medir, o elevado coste económico.

Las técnicas ópticas de luz estructurada utilizan patrones de luz proyectados sobre la escena para obtener información tridimensional de ésta. En estas técnicas, la información tridimensional se obtiene analizando las deformaciones de la proyección del patrón sobre la escena con respecto al patrón original proyectado.
(J. Battle, 1998)

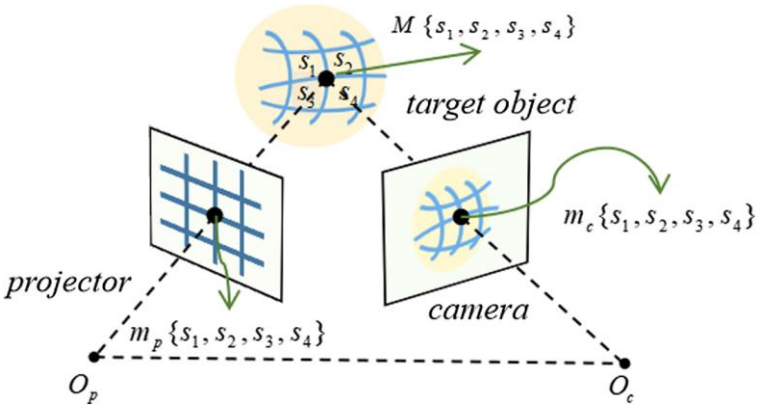


Figura. Esquema de funcionamiento de un sistema de luz estructurada.

La codificación del patrón de luz permite resolver de forma sencilla al problema de correspondencia entre los puntos de la imagen y los puntos del patrón original. Las estrategias a la hora de codificar los patrones son amplias y diversas en función de los

resultados que queramos obtener. Entre los más usuales encontramos patrones tipo punto, línea, barras o rejilla. Los patrones pueden ser proyectados en espectros visibles o no para el ojo humano, siendo común las soluciones de haces de luz infrarroja.

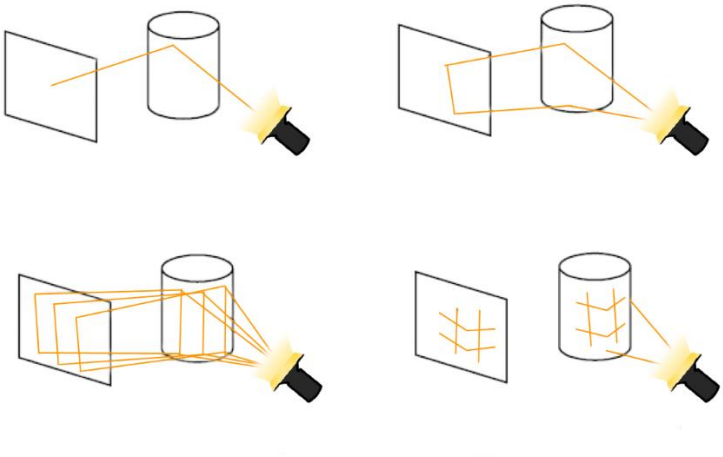


Figura. Distintos tipos de patrones de luz estructurada.

Actualmente existen numerosas soluciones comerciales de sensores de luz estructurada en función del campo de aplicación y de los requisitos de la actividad, uno de los más interesantes y que puede entrar dentro del estudio de este trabajo son los sensores con patrón de luz caótica.

Sensores con patron de luz caótica.

Estos dispositivos proyectan una cantidad muy elevada de puntos, que a simple vista pueden parecer distribuidos aleatoriamente. Dichos puntos pueden diferir en intensidad y estar localizados en sectores o cuadrículas para mejorar la adquisición de datos.

Este tipo de sensores suele ser utilizado para obtener información tridimensional de un entorno amplio. Es por ello que suele utilizarse en aplicaciones de escaneo 3D tales como reconstrucción de escenas, capturas de movimiento, videojuegos, etc.

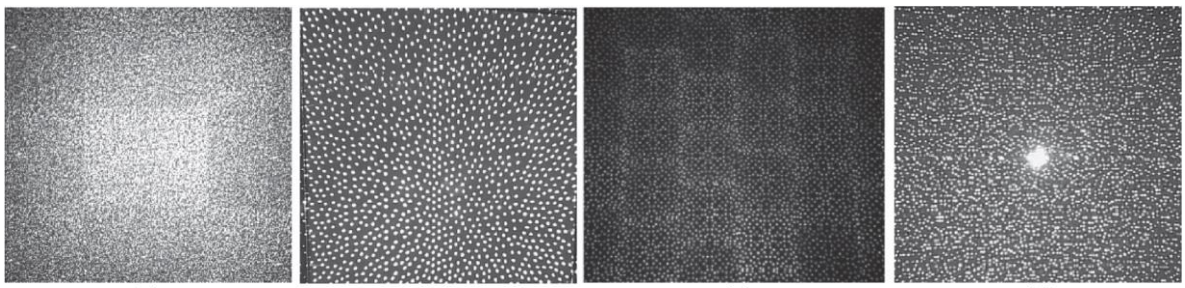


Figura. Patrones de sistemas de luz caótica. (a) Microsoft Kinect v1; (b) Intel RealSense D435; (c) Orbbec Astra Pro

Tiempo de vuelo

Los sensores de tiempo de vuelo (TOF), al contrario de los de luz estructurada, no proyectan un patrón visual que se deforma en función de la superficie del entorno, sino que se basan en el principio en que cada uno de los píxeles determina la distancia de la cámara al objeto mediante la medida muy precisa del tiempo de retardo de un haz de luz modulada emitido por el propio sensor.
El funcionamiento de estos sistemas consiste en el envío una señal óptica modulada, normalmente en espectro infrarrojo (IR), por un transmisor que ilumina la escena sobre la cual pretendemos extraer la información 3D. La luz reflejada se detecta por el sensor el cual determina el tiempo de vuelo (ida y vuelta) para cada uno de los píxeles. La información tridimensional se captura en simultáneo paralelo para cada uno de los píxeles de la imagen, lo que elimina la necesidad de cualquier otro procesado adicional.

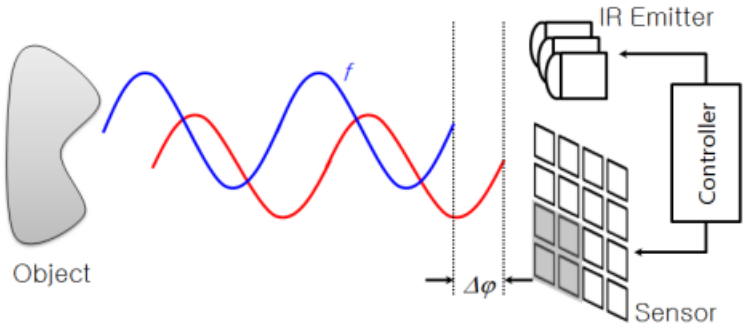


Figura. Diagrama de sensor TOF

Si bien es cierto que el principio físico de estos sensores consiste en la determinación de la distancia en función del tiempo que tarda el haz de luz en viajar y volver al objeto que deseamos medir, el funcionamiento real del sensor es algo diferente.
Los haces de infrarrojos viajan a la velocidad de la luz $(300.000.000\ m/s)$ por lo que el más mínimo error en la obtención del tiempo de vuelo supondría un error enorme en cálculo de la distancia

$$distancia = \frac{c * tiempo}{2}$$

Los sensores TOF solucionan este problema midiendo la diferencia de fases entre el haz de luz emitido y el rebotado. El cálculo de la distancia queda determinado pues de la siguiente manera.

$$distancia = \frac{c}{2} * \frac{\Delta\varphi}{2\pi f}$$

En esta ecuación el valor de la velocidad del haz de luz puede ser contrarrestada por una alta frecuencia del mismo. El sensor calcula la diferencia de fase entre las señales $(\Delta\varphi)$ y a

partir de esta la distancia del píxel en particular. En la siguiente imagen observamos como quedarían superpuestas las señales emitida y rebotada y su diferencia entre fases

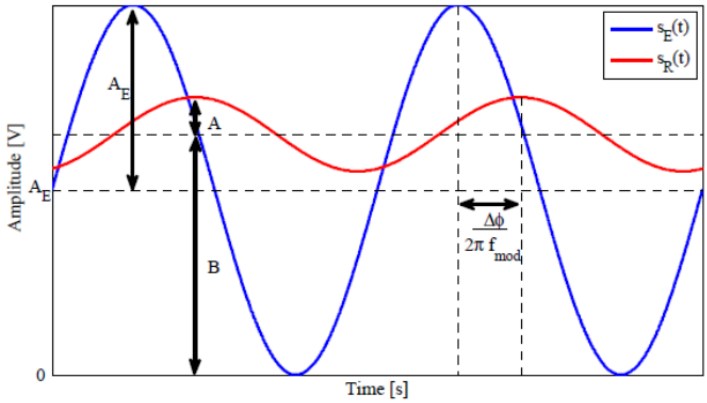


Figura. Señal emitida (azul) y señal rebotada (rojo)

Los sistemas de visión por tiempo de vuelo proporcionan medidas de alta calidad, estables en repetitividad y precisión, incluso en objetos de distintos colores y reflectividad dentro de la misma imagen, además, dado que el proceso es independiente de la luz ambiente, son útiles en ambientes de poca o nula iluminación externa.

Todo esto unido a su relativo bajo coste hace que este tipo de sistemas estén siendo implementados cada vez en más aplicaciones y campos.

Técnicas pasivas

Denominamos por técnicas pasivas aquellas que hacen uso exclusivamente de sensores pasivos, generalmente cámaras de vídeo. Estos sensores capturan imágenes bidimensionales de intensidad o secuencias de vídeo con las que se reconstruye la escena en tres dimensiones. Es decir, en un sensor pasivo no hay ningún sistema ni elemento que intervenga o modifique la escena.

Las técnicas de tipo pasivo tienen un rango más amplio de utilización que las de tipo activo, pues los sensores implicados consisten en cámaras convencionales usualmente de bajo coste y de un tamaño más reducido que los sensores de rango activos. Sin embargo, al no utilizar una fuente de iluminación controlada, el grado de incertidumbre con el que se realiza la correspondencia entre puntos de la escena y puntos de la imagen limitan en muchas ocasiones la exactitud de la medida. Por otro lado, requieren una rigurosa etapa de calibración de las cámaras susceptible a la generación de errores.

Flujo Óptico

Cuando un observador se mueve en dirección lateral con respecto a su campo de visión, los objetos que se encuentran a distancias diferentes proyectan unas imágenes que se mueven en sentido y a velocidades diferentes. A este efecto se le denomina paralaje.

Podemos definir el paralaje de movimiento como el desplazamiento diferencial de las imágenes, proyectadas por distintos objetos, debido a un cambio lateral en la posición del observador y a la distancia relativa de los objetos con respecto al punto de fijación. Los objetos más cercanos parecen desplazarse más lejos y a mayor velocidad mientras que para los más alejados el desplazamiento es menor y más lento. El paralaje de movimiento es una clave de profundidad muy efectiva a grandes distancias incluso cuando no están presentes otras claves de profundidad.

El cálculo del flujo óptico consiste en la estimación del movimiento aparente de los objetos en una secuencia de imágenes. Dado un conjunto de imágenes, el objetivo es calcular el desplazamiento de los pixeles entre las distintas imágenes. El desplazamiento de los pixeles no es más que la proyección en una imagen del movimiento tridimensional por lo que dado un conjunto de imágenes se puede estimar el desplazamiento de los píxeles entre las distintas imágenes.

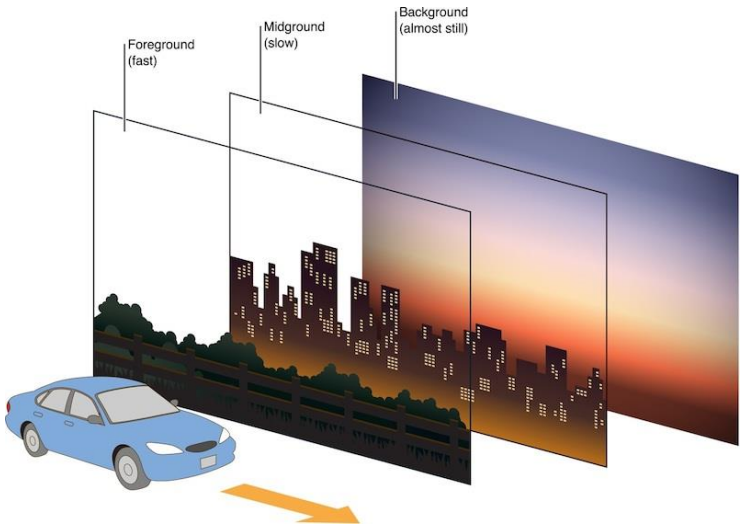


Figura. Ejemplo de paralaje

A la hora de calcular el flujo óptico encontramos diferentes métodos en función de los métodos matemáticos y los elementos de la imagen en los que se aplica.
Los más representativos son los métodos diferenciales. En estos métodos se calcula el desplazamiento de los píxeles a partir de las derivadas espaciales de las intensidades de la imagen. En función de cómo se utilice la información de dichas derivadas podemos establecer algunas subcategorías siendo las más relevantes:

- Métodos locales. Utilizan la información en una vecindad alrededor de un píxel para estimar el movimiento.

- Métodos de contorno. Utilizan la información de los bordes de los objetos para detectar el desplazamiento.

A parte de los métodos diferenciales cabe mencionar los métodos variacionales. La idea subyacente de este tipo es la definición de una energía que penaliza las desviaciones respecto al modelo.

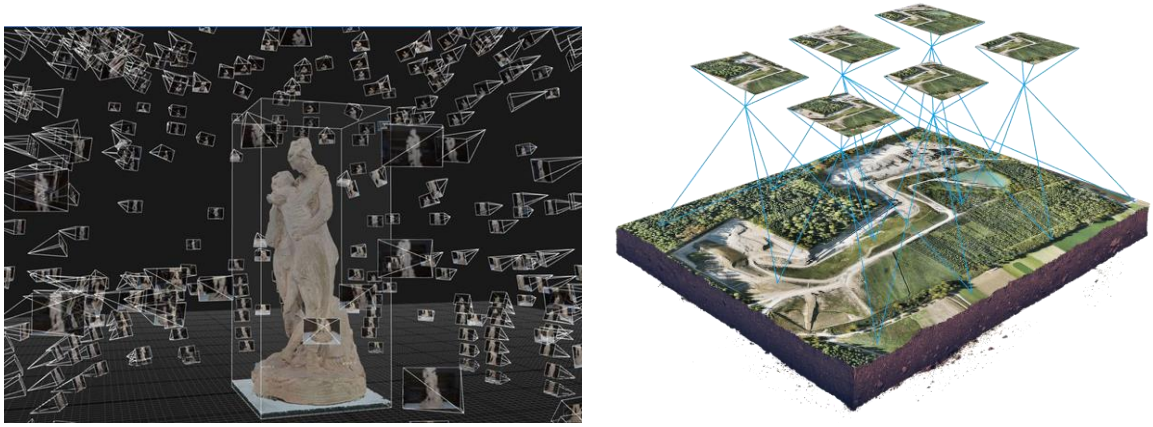
Fotogrametría

Los sistemas de fotogrametría son aquellos que permiten determinar las información tridimensional de un entorno mediante la interpretación de un conjuntos de imágenes realizadas sobre este.

El método utilizado consiste en tomar una amplia muestra de fotografías solapadas de un objeto tomadas desde numerosos ángulos distintos, que se procesan posteriormente para obtener el modelo 3D de dicho objeto. Para que este modelo 3D sea preciso, se deben tomar un número suficiente de imágenes desde varias posiciones y ángulos necesarios con el fin de que todas las partes del objeto queden completamente registradas

Esta técnica presenta algunas ventajas pero numerosas desventajas en tu uso. Su principal ventaja radica en que se puede llevar a cabo con una única cámara de fotos. Para reproducciones 3D que no requieran altas resoluciones o detalles, la cámara de un simple teléfono móvil sería suficiente para tomar las imágenes de muestra.

Por otro lado, la necesidad obtener imágenes de toda la superficie del objeto hace que por norma general se necesite un amplio set de cámaras, así como girar o voltear la pieza. La iluminación del objeto es otro factor a tener en cuenta ya que debe ser suficiente para evitar sombras dispares entre las diferentes perspectivas. Y por último y quizás el más importante, las imágenes requieren de un procesamiento posterior, costoso computacionalmente, por lo que este método queda completamente descartado para cualquier actividad que requiera una interpretación de imágenes a tiempo real. Es por ello que esta técnica es ampliamente utilizada en topografía o en generación de modelos 3D de objetos, pero totalmente inaplicable en otra gran cantidad de áreas, incluida las que se estudian en este trabajo.



(a) (b)

Figura. (a) Fotogrametría aplicada a generación de modelos 3D; (b)Fotogrametría aplicado a topografía

Visión estereoscópica

Un sistema de visión estereoscópica permite determinar la información tridimensional de la escena a partir de la comparación de puntos comunes en imágenes tomadas en un mismo instante de tiempo por dos o más cámaras.

La visión estereoscópica se basa en el modelo estereoscópico biológico, de hecho, podríamos decir que esta técnica trata de emular el sistema de visión humano, en el que se analizan las diferencias de la proyección de la escena en dos imágenes tomadas desde dos posiciones diferentes.
(J. M. López-Vallés, 2006)

El cálculo de profundidad de la imagen se realiza a partir de la diferencia existente entre las proyecciones de puntos comunes de cada una de las cámaras. A estas diferencias se les denomina disparidad que podemos definir como el desplazamiento entre el punto 2D situado en la proyección de una de las cámaras y su punto correspondiente en la proyección de la otra.
Para cada punto del espacio real se tiene una proyección en ambas cámaras. A partir de la disparidad entre ambas imágenes es posible calibrar y ajustar el sistema para obtener información tridimensional del entorno.

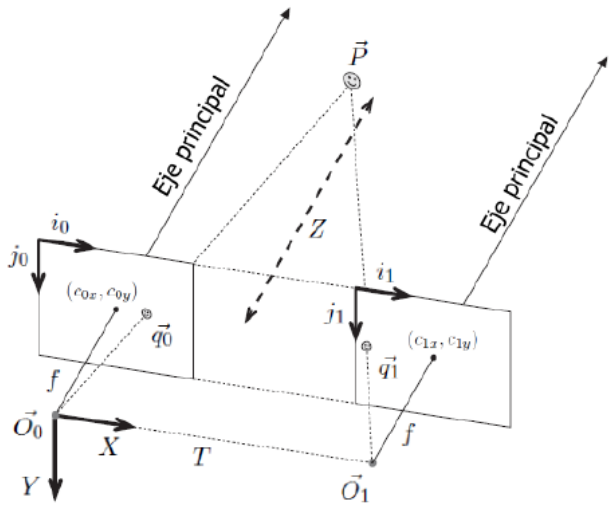


Figura. Sistema de visión estereoscópica con cámaras alineadas.

En la imagen podemos observar como el punto de la escena se proyecta en diferentes posiciones dentro del plano de proyección de cada cámara, la diferencia de estas medidas determina la disparidad, a partir de la cual podemos calcular la distancia del punto por simple trigonometría. Para que este cálculo sea correcto la geometría de las cámaras debe cumplir una serie de condiciones, muchas veces imposibles por propios defectos de fabricación. La correcta calibración de las cámaras es la que permite solventar estos defectos.

Teniendo que las proyecciones del punto $\vec{p}$ para cada una de las son $\overrightarrow{q_0}$ y $\overrightarrow{q_1}$ y que los centros de las mismas están alineados en el plano horizontal.

$$\overrightarrow{q_0} = (i_0, j_0)$$

$$\overrightarrow{q_1} = (i_1, j_1)$$

La disparidad se calcula como la diferencia entre la componentes horizontal de dichas proyecciones.

$$d = i_0 - i_1$$

Un sistema de visión estereoscópica requiere de un proceso de calibración previo a su puesta en funcionamiento. El proceso de calibración se puede dividir a su vez de dos partes. La calibración a partir de los parámetros intrínsecos de la cámaras, que están asociados a las propias características físicas de cada una, y la calibración de parámetros extrínsecos, que relaciona la posición y orientación entre ambas.

El proceso de calibración supone un punto de fundamental importancia a la hora de trabajar con visión estéreo, ya que cualquier error introducido en este punto inducirá a errores permanentes en el funcionamiento posterior del sistema.

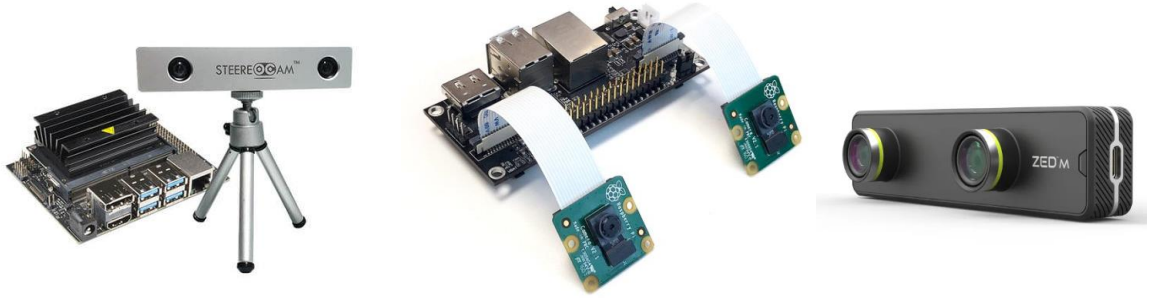
Las características de los sistemas de visión estereoscópica presentan una serie de ventajas y desventajas en su uso, por lo que sus campos de aplicación quedan bastante delimitados por estos.

Al tratarse de un sistema de visión pasivo, los dispositivos de visión estéreo no requieren de ningún otro elementos más que los sensores ópticos, lo que simplifica los componentes y el funcionamiento del mismo. Una vez realizado el proceso de calibrado el sistema puede trabajar en diversos entorno sin necesitar de ningún otro tipo de intervención, esta característica los hace particularmente interesantes, por ejemplo, para su uso en robótica móvil o en aplicaciones en los que el dispositivo modifique su posición o entorno.

Por el contrario, el proceso de calibración es un paso delicado y determinante en el comportamiento futuro del sistema, para el correcto funcionamiento del sensor el proceso de calibración debe ser realizado correctamente y de forma meticulosa.

El campo de aplicación del sistema viene determinado también por una relativa baja precisión a la hora de elaborar mapas de profundidad, así como un rango de medida limitado. Como analizaremos más adelante, la relación entre disparidad y distancia real de los objetos no es lineal, por lo que la precisión del sistema decrece considerablemente con la distancia al sensor, por ello, los rangos de profundidad son limitados con este tipo de técnicas. Por último también debemos destacar que al tratarse de un sistema óptico pasivo, la iluminación de la escena es determinante para unos correctos resultados.

A continuación se muestran algunos ejemplos comerciales de soluciones de visión estereoscópica. Entre ellos se incluye la elegida para este proyecto, el sistema de visión StereoPi. Entre las principales razones que se han tenido en cuenta destaca su bajo precio y su Hardware basado en la plataforma Raspberry.



(a) (b) (c)

Figura. Ejemplos de soluciones estereoscópicas comerciales. (a) STEEReo CAM; (b) StereoPi; (c) Zed Mini

ENTORNO DE DESARROLLO

Para el desarrollo de este trabajo se ha dispuesto un entorno de trabajo con el fin de reproducir diferentes escenarios en los que poder realizar tanto la captura de imágenes como la comprobación de resultados.

Dicho entorno está compuesto los siguientes elementos:

- Sensor de visión. Encargado de capturar y procesar las imágenes de la escena.
- Set de muestras. Compuesto de varios objetos con distintas formas, tamaños.
- Cinta métrica. Que nos facilite comprobar empíricamente la fiabilidad de los resultado de profundidad calculados
- PC. Equipo donde se ejecuta la aplicación de usuario y se visualizarán los resultados.

A continuación se enumeran algunas de las herramientas más destacadas tanto a nivel de Hardware como de software de las que se ha hecho uso en este proyecto.

StereoPi

El sistema de visión estereoscópica *StereoPi* es un dispositivo desarrollado por la empresa *StereoPi Team*, diseñado específicamente para capturar, guardar y transmitir imágenes y video a tiempo real utilizando cámaras estereoscópicas. Está basado en la placa computadora *Raspberry Pi* y agrega la capacidad de trabajar con cámaras estéreo para crear contenido tridimensional.

La elección de este dispositivo viene motivada por su accesibilidad a la hora de montar el setup del hardware y por trabajar en una plataforma ampliamente conocida como es Raspberry y Raspbian. Tal es así que este módulo puede ser replicado sin ningún problema por un módulo y dos cámaras para RaspberryPi.

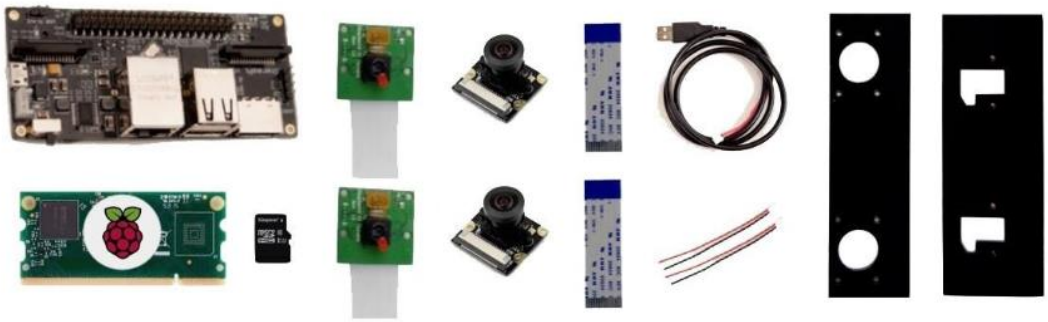


Figura. Componentes del StereoPi Deluxe Kit

Otra de las razones a tener en cuenta es el relativo bajo coste de este dispositivo en comparación con otras soluciones tanto de visión 3D en general como estereoscópicas en particular. A continuación se muestra una comparación de distintas soluciones comerciales.

	StereoPi	RealSense D435+Aaeon	QooCam	ZED Mini	Duo MLX
Open hardware	Yes	No	No	No	No
3D livestream	Yes	Yes	Yes	Yes	No
Depth map	Yes	Yes	No	Yes	Yes
Swappable cameras	Yes	No	No	No	No
Adjustable stereobase	Yes	No	No	No	No
On-board CPU for user code	Yes	Yes	No	No	No
Raspberry Pi inside	Yes	No	No	No	No
Price for complete setup	**$125**	**$259**	**$399**	**$449**	**$695**

Figura. Comparación de características y precio de soluciones de visión estereoscópica

A la hora de trabajar con el sistema estereoscópico debemos tener en cuenta que el montaje del sensor debe permitir una alineación de las cámaras con los menos errores posibles. Para ello utilizaremos las guías que se proporcionan en el propio kit y unos soportes para mantener el setup en una posición fija. Una curiosidad de estas guías es que la distancia entre cámaras coincide con la distancia media entre los ojos humanos.

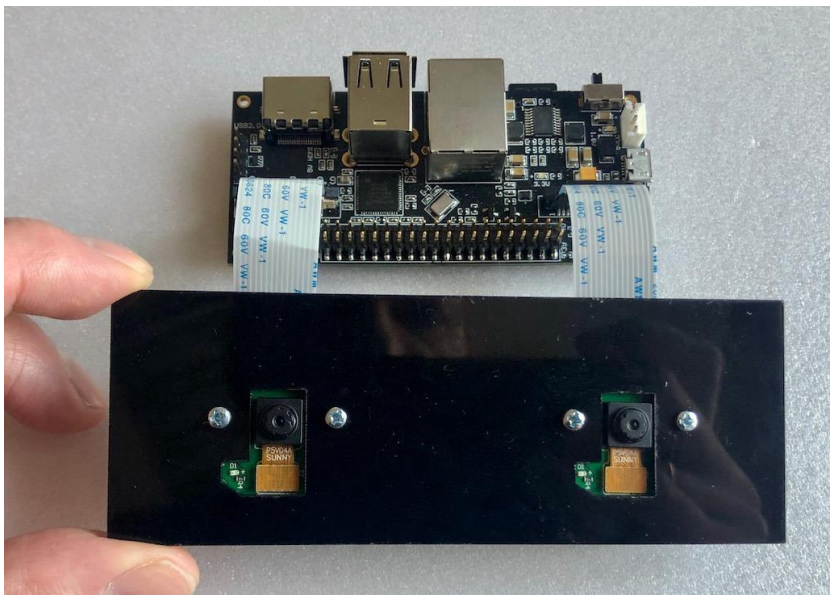


Figura. Montaje de las cámaras en estéreo

En la parte referente al software cabe destacar que todo el código ha sido escrito en lenguaje de programación Python. Como en todo desarrollo se ha hecho uso de varias librerías para realizar ciertos procesos, en nuestro cabo destacar OpenCV que es la librería más comúnmente usada en el tratamiento de imágenes. Por último se explicará brevemente el proceso de comunicación utilizado entre el sensor y la aplicación.

Python

El código de este proyecto ha sido desarrollado en el lenguaje de programación Python.
Python es un lenguaje de alto nivel, interpretado y de propósito general. Se destaca por su sintaxis legible y su enfoque en la legibilidad del código, lo que lo hace especialmente adecuado para principiantes y profesionales por igual. Entre sus características más importante podemos destacar las que nos hicieron decantarnos por este lenguaje.

- Tipado dinámico: Python es un lenguaje de tipado dinámico, lo que significa que no es necesario declarar explícitamente el tipo de una variable. Los tipos se asignan automáticamente en función del valor asignado a la variable.

- Amplia biblioteca estándar: Python cuenta con una amplia biblioteca estándar que abarca desde manipulación de cadenas y archivos hasta acceso a redes y servicios web. Esto facilita el desarrollo al proporcionar módulos y funciones preconstruidos para muchas tareas comunes.

- Multiplataforma: Python es compatible con diversas plataformas, lo que permite que el mismo código se ejecute en diferentes sistemas operativos sin necesidad de modificaciones sustanciales.

- Interpretado: Python es un lenguaje interpretado, lo que significa que no se compila directamente en código de máquina. En cambio, el intérprete de Python ejecuta el código fuente directamente, lo que facilita la depuración y la experimentación rápida.

- Aplicaciones variadas: Python se utiliza en una amplia gama de aplicaciones, como desarrollo web, operaciones matriciales (con bibliotecas como NumPy), tratamiento de imágenes, scripting, automatización, y más.

Picamera

Picamera es una interfaz de Python para Raspberry Pi que facilita el manejo y la accesibilidad de las cámaras.
(Picamera, s.f.)

OpenCV

La adquisición de imágenes y procesamiento de imágenes ha sido desarrollada haciendo uso principalmente de las funcionales de la librería OpenCV.
OpenCV, cuyas siglas significan Open Computer Vision (Visión Artificial Abierta), es una de las bibliotecas libres de visión artificial más utilizadas en la actualidad.
Las principales características por las que hemos elegido esta librería se resumen en las siguientes.

- Es libre, publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación.

- Multiplataforma, para los sistemas operativos GNU/Linux, Mac OS X, Windows y Android, y para diversas arquitecturas de hardware como x86, x64 (PC), ARM (celulares y Raspberry Pi).

- Documentada y explicada: la organización tiene una preocupación activa de mantener la documentación de referencia para desarrolladores lo más completa y actualizada posible, ejemplos de uso de sus funciones y tutoriales accesibles al público no iniciado en visión artificial, además de difundir y fomentar libros y sitios de formación.

Paquete StereoVison

El proceso de calibración que veremos posteriormente se ha llevado a cabo en gran parte haciendo uso de las funcionalidades proporcionadas por este paquete.

El paquete de librerías StereoVision nos permite una serie de utilidades para trabajar con cámaras estéreo. El objetivo de estas librerías se centra en el rendimiento, la facilidad de uso y la capacidad de construir configuraciones de imágenes 3D de forma sencilla. Las funcionalidades proporcionadas en este paquete se basan o utilizan de funciones nativas de OpenCV.

(Stereovisión, s.f.)

Paquete Tkinter

La interfaz gráfica con la que interactuamos al ejecutar la aplicación de usuario ha sido desarrollada a partir de la librería Tkinter. Este paquete de librerías proporciona un conjunto de herramientas y utilizades para administrar ventanas, botones, y en definitiva para

generar interfaces gráficas. Este paquete es una fina capa orientada a objetos encima de Tcl/Tk. Tkinter es un conjunto de envoltorios que implementan los widgets Tk como clases de Python.

(python, s.f.)

Comunicación TCP/IP

El proceso de comunicación entre el sensor de visión y la aplicación de usuario ha sido implementado bajo un protocolo TCP/IP.

Este modelo permite un intercambio de datos fiable dentro de una red, definiendo los pasos a seguir desde que se envían los datos (en paquetes) hasta que son recibidos. Para lograrlo utiliza un sistema de capas con jerarquías (se construye una capa a continuación de la anterior) que se comunican únicamente con su capa superior (a la que envía resultados) y su capa inferior (a la que solicita servicios). La importancia del protocolo TCP/IP radica en que nos permite que los datos enviados lleguen a su destino sin errores y bajo la misma forma en la que fueron enviados.

TCP es un protocolo orientado a conexión. Las aplicaciones utilizan un modelo cliente/servidor en las comunicaciones. Una aplicación consta de una parte de servidor y una de cliente, que se pueden ejecutar en el mismo o en diferentes sistemas y donde el servidor ofrece un servicio y el cliente realiza peticiones al mismo. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.

A la hora de trabajar con este protocolo debemos se debe tener especialmente en cuanta que durante la comunicación se establecen ciertas llamadas bloqueantes que hacen que el sistema quede a la espera de una respuesta y no siga ejecutando el programa hasta que esta se haya resuelto.
(Blanes, 2022)

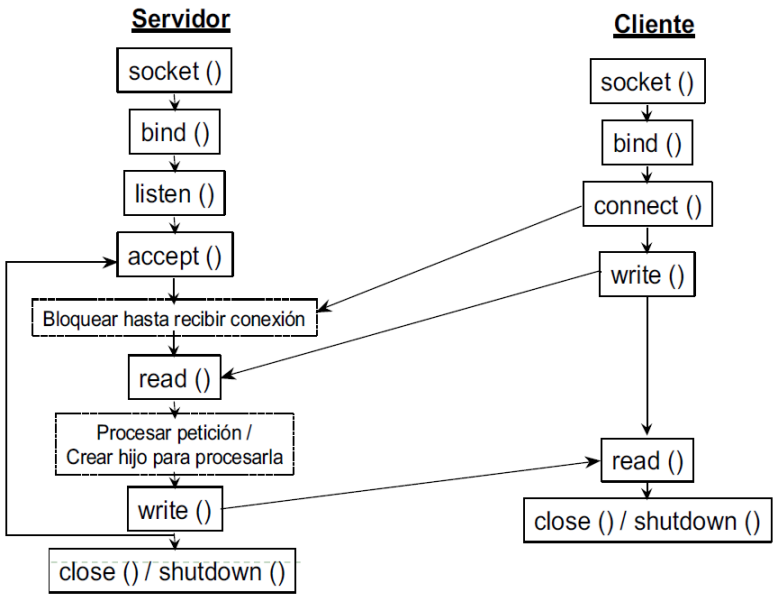


Figura. Esquema de una conexión TCP cliente-servidor

MARCO TEÓRICO

En este apartado estudiaremos en profundidad las bases de los sistemas de visión estereo analizando tanto su trasfondo matemático como los pasos seguidos para su implementación en nuestro sensor.

Modelo de cámara pinhole

El modelo de cámara pinhole se caracteriza por ser el más simplificado aplicable al proceso de adquisición de imágenes. En este modelo suponemos que los haces de luz que tomamos de la escena pasan por un punto infinitesimal al que denominamos $centro\ ópitco\ C$. Para cada punto $M(X,Y,Z)$ de la escena se obtiene una proyección $m(x,y)$ en el $plano\ de\ la\ imagen$ R, que se encuentra a una $distancai\ focal\ f$ del centro óptico y que es perpendicular al $eje\ óptico\ Z$.

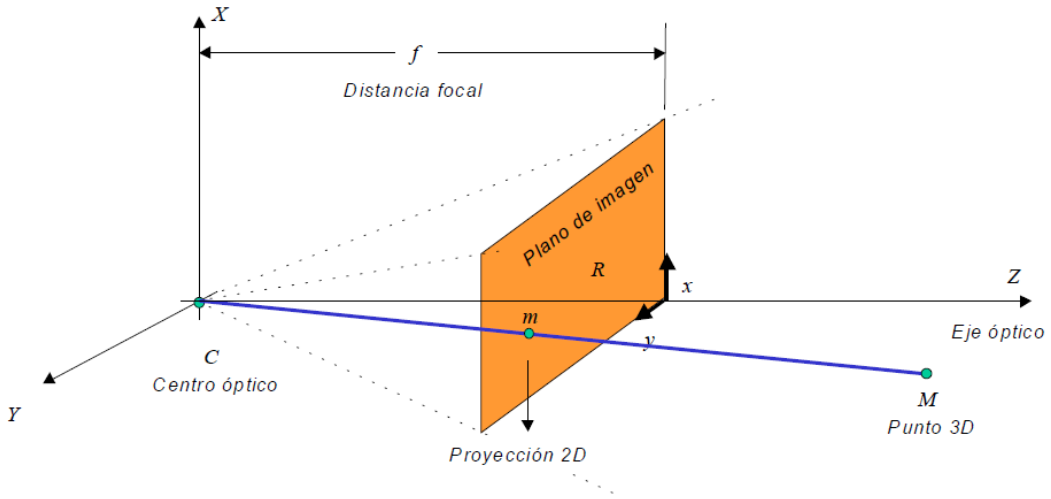


Figura. Modelo de cámara pinhole

Definimos el punto m como la intersección de la recta formada por la unión de los puntos C y M con el plano de imagen R.

$$m = \langle C, M \rangle \cap R$$

Aplicando el teorema de Thales a las coordenadas de los puntos m y M se obtiene la siguiente relación

$$\frac{X}{Z} = \frac{x}{f} \rightarrow Zx = fX$$

$$\frac{Y}{Z} = \frac{y}{f} \rightarrow Zy = fY$$

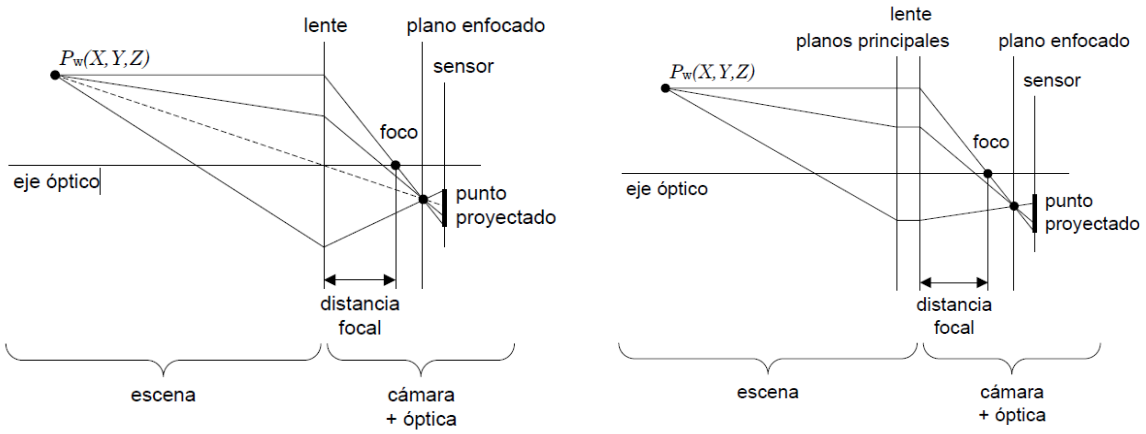
Que transformando a coordenadas homogéneas queda de la siguiente manera.

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

La matriz de tamaño 3x4 resultante se denomina $matriz\ de\ proyección\ perspectiva$ de la cámara.

Como podemos observar este modelo matemático no es mas que una simple transformación de perspectiva en la que contamos como único parámetro la distancia focal de la cámara, que es la distancia entre el $centro\ ópitco$ y el $plano\ de\ la\ imagen$.

Este modelo, sin embargo, tiene algunos inconvenientes, pues al requerir que el centro óptico tenga tamaño infinitesimal se pierden algunas características en el sistema de visión. Para solventar estos problemas se pueden recurrir a modelos más complejos de ópticas como el modelo de lente delgada o el modelo de lente gruesa.



(a) (b)

Figura. Modelos de cámara alternativos (a) Modelo de lente delgada; (b) Modelo de lente gruesa.

Parámetros de una cámara

(Cristian Daniel Ortega, 2013)

Todo modelo de cámara está determinado por dos tipos de parámetros.

Parámetros **intrínsecos** son aquellos determinados por las características físicas de la propia cámara, es decir las características de fabricación del sensor y su óptica.

- Distancia focal.
- Desplazamiento del centro de la imagen
- Coeficiente de distorsión radial
- Coeficiente de distorsión tangencial

A partir de la distancia focal se pueden obtener las longitudes focales de la lente de la cámara para cada uno de los ejes. Estos parámetros se miden en píxeles y se obtiene mediante las siguientes ecuaciones

$$f_x = fS_x$$

$$f_y = fS_y$$

Donde los coeficientes S_x y S_y indican el número de píxeles por unidad de longitud del sensor en cada eje. Estos parámetros junto a los referentes al desplazamiento del centro de la imagen suelen expresarse en la siguiente matriz

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Donde el coeficiente s indica el grado de perpendicularidad de la lente y suelen omitirse a efectos prácticos.

Por otro lado los parámetros **extrínsecos** determinan la orientación y la posición de la cámara respecto del sistema de coordenadas de la escena. No dependen por tanto de las características de las propias cámaras sino de su disposición espacial.

- Parámetros de traslación
- Parámetros de rotación

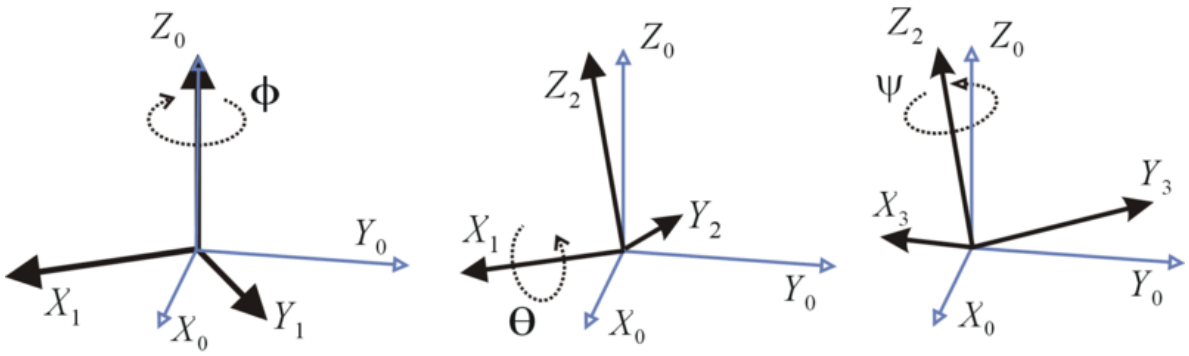


Figura. Rotaciones en los tres ejes de un sistema de referencia

Las matrices que definen las rotaciones de cada uno de los ejes y la traslación del sistema de referencia se expresan con las siguientes expresiones.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \quad R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

En ocasiones los parámetros intrínsecos son proporcionados por el fabricante a la hora de la adquisición de la cámara, aunque también pueden obtenerse junto a los parámetros extrínsecos mediante un proceso de calibración de la cámara.

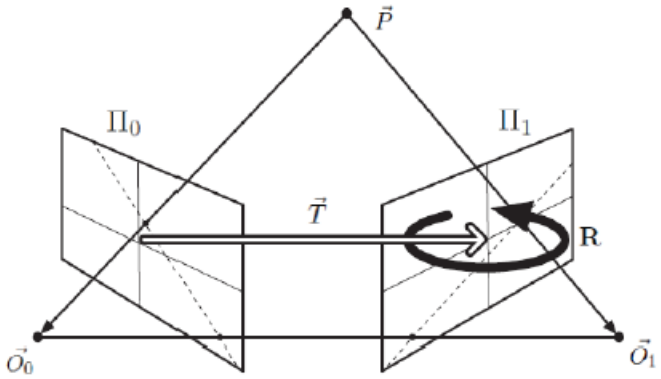
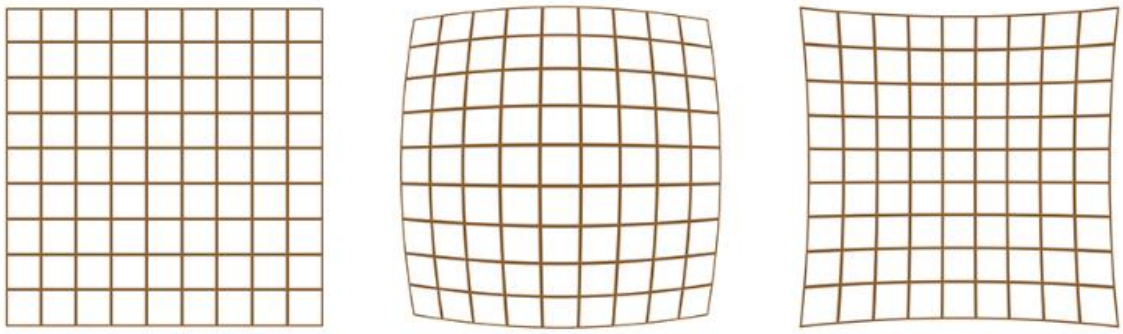


Figura. Ejemplo de rotación más traslación de una cámara

Defectos en la toma de imágenes

A la hora de trabajar con un modelo real de cámara se deben tener en cuenta que la incorporación de lentes para evitar las deficiencias del modelo pinhole conlleva una problemática adicional, las distorsiones ocasionadas por esta en la imagen recibida por el sensor. Esta distorsión se debe a que los rayos de luz no atraviesan la lente en las direcciones exactas que se esperaría. Dentro de estas distorsiones se distinguen dos tipos.

En la distorsión radial los puntos de la imagen se desplazan en direcciones radiales a partir del centro óptico favoreciendo que la imagen se "curve" conforme nos alejamos de este. Una causa de estos problemas puede ser causada por defectos en el pulimiento de la lente. El grado de distorsión radial viene determinado por unos coeficientes, cuyo signo determina si nos encontramos ante una distorsión de tipo cojín (positiva) o de tipo barril (negativa)



(a) (b) (c)

Figura. (a) Sin distorsión; (b) Distorsión tipo barril; (c) Distorsión tipo cojín.

Por otro lado encontramos la distorsión tangencial que se debe por una falta de paralelismo entre la lente y el sensor de la cámara. Su principal efecto es la generación de imágenes

trapezoidales. Sin embargo los efectos de la distorsión tangencial son menos acusados que la radial y por ello en muchas ocasiones pueden considerarse despreciables.

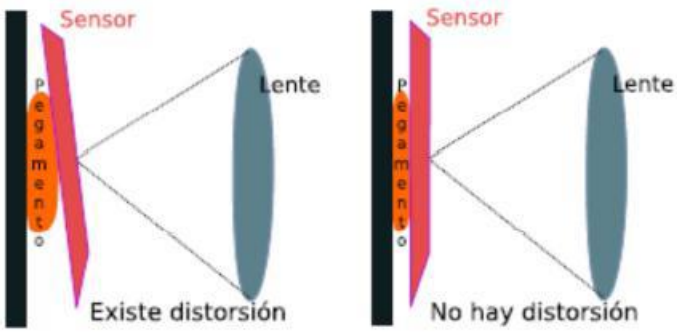


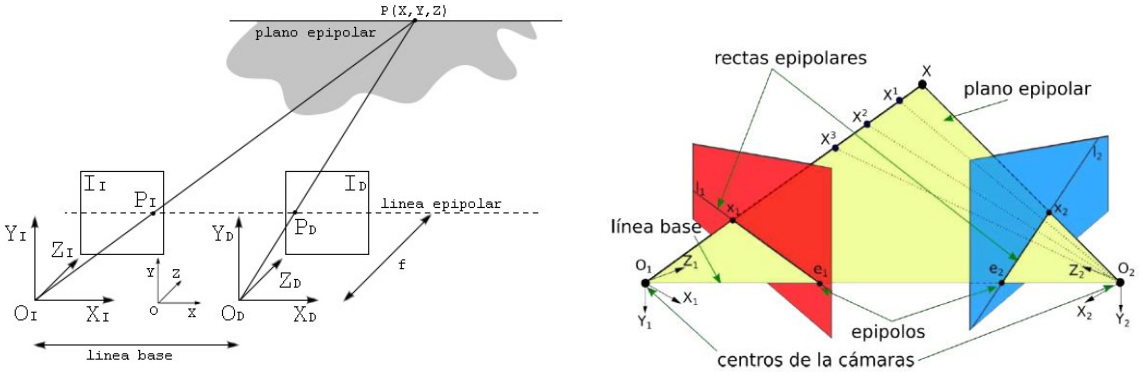
Figura. Ejemplo de distorsión tangencial

Modelo de cámara estereoscópica

Un sistema de visión estereo puede estar diseñado para modelos de cámara con diferentes atributos tanto físicos como geométricos. La características más importante de este diseño es la situación entre los ejes ópticos de las cámaras que lo componen.

El modelo más utilizado en los sistemas estereoscópicos es el modelo con cámaras cuyos ejes ópticos son paralelos. Con este modelo la problemática del cálculo para la obtención de la información tridimensional se reduce considerablemente. Los conceptos esenciales de este modelo son:

- Longitud focal. Es la distancia entre el eje óptico y el plano de proyección de la imagen.
- Línea base. Determina la distancia entre los ejes ópticos de ambas cámaras.
- Línea epipolar. Se define como la línea que une un mismo punto en las imágenes tomadas por ambas cámaras izquierda y derecha.
- Plano epipolar. Es el plano formado por los siguientes tres puntos. Un punto real de la escena. Los dos puntos correspondientes a los centros de proyección de las cada una de las cámaras.
- Línea epipolar. Se define como la intersección del plano epipolar con el plano de proyección de una cámara. Recordemos que en un supuesto ideal los planos de las cámaras son el mismo.



(a) (b)

Figura. Líneas y planos epipolares en sistemas estereo: (a) Con líneas epipolares coincidentes, (b) Con líneas epipolares diferentes

La característica que hace óptico a este modelo es la correspondencia entre las líneas epipolares y los puntos. Las proyecciones de todos los puntos situadas en una misma línea epipolar de la imagen derecha estarán en la misma línea epipolar de la imagen izquierda, y viceversa.
(Gonzalez., 2003)

La obtención de la profundidad en los sistemas estereo se basa en las correspondencias de puntos entre las distintas imágenes. Se analiza el desplazamiento de un conjunto de píxeles entre las imágenes tomadas, este desplazamiento puede ser por tanto horizontal como vertical. Esta tarea puede ser compleja y necesitar de un alto grado de computación para resolverse ya que la búsqueda de estas correspondencias es en las dos dimensiones del plano.

La solución que proporciona este modelo para minimizar la complejidad de esta tarea es el establecimiento de los ejes ópticos de las cámaras de tal modo que se consiga que ese desplazamiento o desviación de los píxeles se produzca tan solo en un plano horizontal. Para ello se hace que los ejes ópticos de las cámaras se sitúen paralelamente y que los ejes de abscisas sean coincidentes. Con esta configuración conseguimos que las imágenes estén completamente alineadas horizontalmente, es decir, las líneas epipolares que definen el plano epipolar son coincidentes.

A esta configuración se le denomina restricción epipolar. Gracias a ella los valores de disparidad de un punto estarán únicamente determinados por la diferencia entre las componentes horizontales de las representaciones de dicho punto en la imagen izquierda y derecha. A partir de dichos valores de disparidad se puede generar un mapa de disparidades. Y a partir de este podemos calcular el mapa de profundidad que ocupa el objetivo de este trabajo.
(Castedo hernandez, 2017)

Calibración de un sistema de visión estereoscópica

Queremos implementar un modelo de cámara con restricción epipolar, como el visto en el apartado anterior, sin embargo, el montaje de las cámaras como sus propias características de fabricación hacen que haya pequeños errores en el paralaje y orientación del modelo que son imposibles de corregir físicamente.

Todo sistema de visión estereo debe pasar por un proceso de calibración que permitirá ajustar ciertos parámetros con el de eliminar estos defectos.

Para la calibración de nuestro par estereo se han seguido los pasos proporcionados en el tutorial del proyecto StereoPi
(Pomazov, 2019)

Esta calibración calcula los parámetros de la cámara a partir de imágenes tomadas sobre un tablero de ajedrez con un número determinado de cuadros blancos y negros.

En nuestro caso hemos utilizado un tablero de 7 x 10 cuadrados con un tamaño de cuadrado de 26mm de lado.

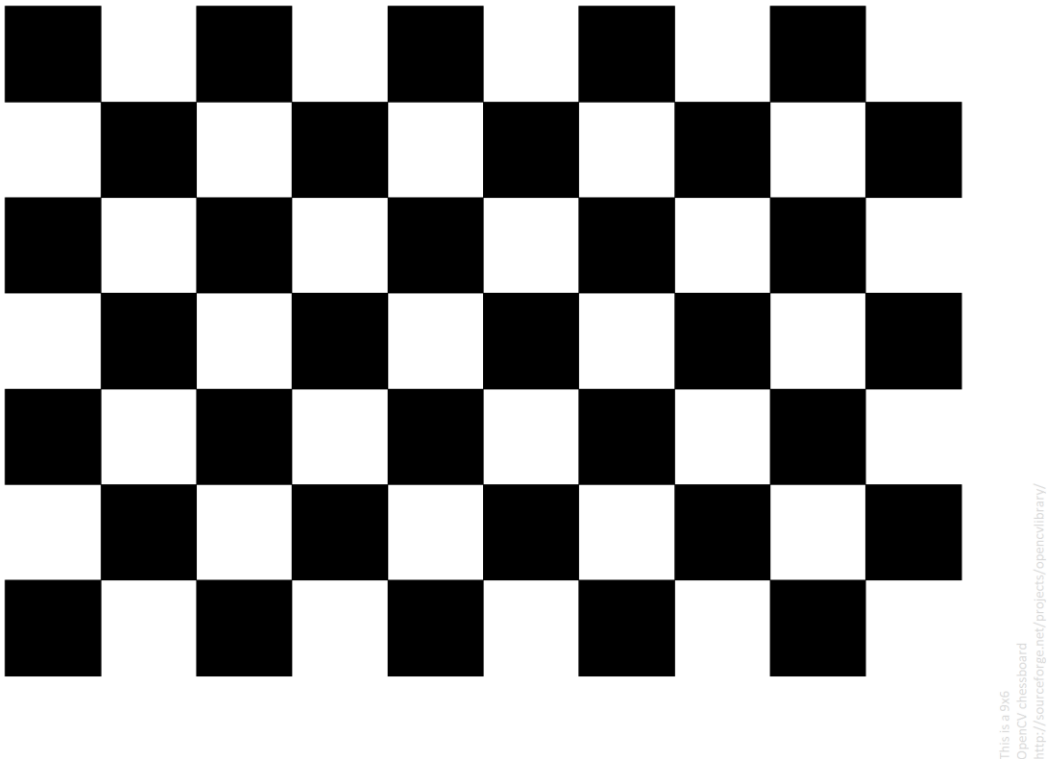


Figura. Tablero de calibración

En primer lugar necesitamos una batería de imágenes del tablero en diferentes posiciones y orientaciones. A mayor variabilidad de posición y rotación mejores resultados

obtendremos en la obtención de parámetros. También es importante apuntar que el tablero debe verse por completo por ambas cámaras en cada par de imágenes, de lo contrario e algoritmo de calibración no podrá encontrar todas las esquinas de los cuadrados.

Figura. Imágenes del tablero en diferentes orientaciones

- Inicialización de la cámara

Para la inicialización de la cámara y la captura de imágenes se hace uso del paquete de librerías *PiCamera*. Configuramos el modo estereo de las cámaras en un mismo plano horizontal, establecemos las dimensiones (ancho y alto) de las imágenes y el framerate.

```
cam_width = 1280
cam_height = 480
camera = PiCamera(stereo_mode='side-by-side',stereo_decimate=False)
camera.resolution=(cam_width, cam_height)
camera.framerate = 20
```

- Captura del patrón de calibración

Configuramos los parámetros del tablero de ajedrez que indicarán el algoritmo el número de cuadrados y esquinas que debe buscar. Las esquinas que buscará el algoritmo hacen referencia a “esquinas interiores”, es decir, las pertenecientes a un cuadrado cuyas cuatro esquinas sean también parte de otro cuadrado.

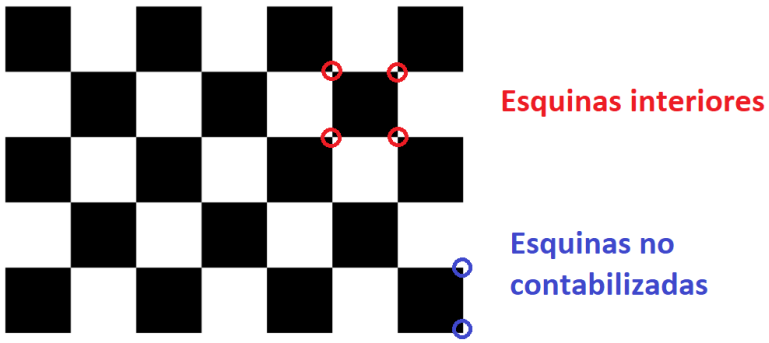


Figura. Ejemplo de esquinas interiores en el patrón

Para una correcta calibración se estima que el número mínimo de imágenes que requiere el algoritmo es de ocho. En nuestro caso tomaremos treinta imágenes para intentar optimizar los resultados en lo posible. El proceso de captura del patrón se realizará por tanto treinta veces dejando un lapso de tres segundos entre toma y toma para poder cambiar la posición del tablero. La función utilizada para este fin será *capture_continuos()*, proporcionada también por el paquete *PiCamera*. Además se aplica un coeficiente de escala de 0.5 a las dimensiones de la captura configuradas en la cámara, lo que nos dará como resultado una imagen capturada de dimensiones 640x240.

```
total_photos = 30
countdown = 3
scale_ratio = 0.5
img_width = int (cam_width * scale_ratio)
img_height = int (cam_height * scale_ratio)
camera.capture_continuous(capture, format="bgra", use_video_port=True,
resize=(img_width,img_height)):
```

Como nuestra cámara está configurada en modo estereo cada una de las imágenes estará compuesta a su vez por la unión de las imagenes tomadas por ambas cámaras.

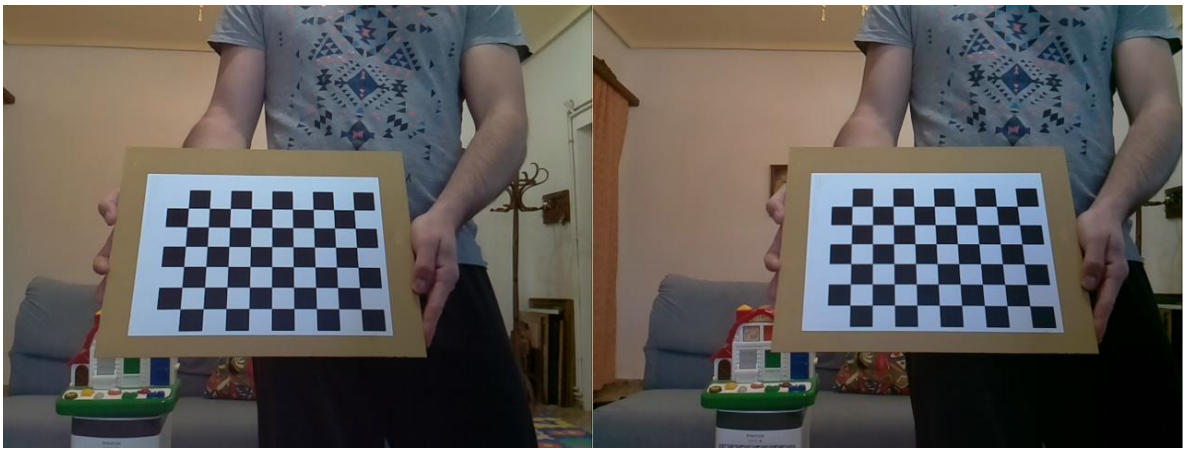


Figura. Ejemplo de fram con la dupla de imágenes tomada para la calibración

Será necesario por tanto separar ambas imágenes. Para ello, como conocemos las dimensiones de la imagen capturada por la cámara, dividimos dicha imagen por la mitad de la anchura y guardamos cada par de imágenes por separado.

```
photo_width = 640
img_height = 240
img_width = 320
imgLeft = pair_img [0:img_height,0:img_width]
imgRight = pair_img [0:img_height,img_width:photo_width]
```

El resultado son dos imágenes independientes de cada cámara para cada una de las tomas de calibración

Figura. Imágenes de calibración separadas por cámaras izquiera y derecha

- Calibración

En este punto ya disponemos de todos los elementos para comenzar la calibración. Proporcionamos al algoritmo los parámetros que necesita conocer del tablero para que pueda identificarlos en las imágenes que hemos tomado. Estos parámetros corresponden al número de esquinas por fila y columna del tablero y la distancia, en centímetros, que hay entre ellas, distancia que coincidirá con el lado de los cuadrados del tablero.

rows = 6
columns = 9
square_size = 2.63

El proceso de calibración se llevará a cabo utilizando las funciones del paquete *StereVision*. Para ello creamos un objeto de la clase StereoCalibrator() en la que le indicamos la información del patrón de calibración y el tamaño de las imágenes

calibrator = StereoCalibrator(rows, columns, square_size, image_size)

En primer lugar se importan cada par de imágenes por separado y se determina donde se encuentran las esquinas del tablero. Para todo este proceso se hace uso de la función de *get_corners()*

calibrator._get_corners(imgLeft)
calibrator._get_corners(imgRight)

Esta función trabaja internamente a partir de una serie de funciones de OpenCV. Los pasos que sigue son los siguientes.

Encuentra las esquinas internas de un tablero de ajedrez de la imagen mediante la función de OpenCV *findChessboardCorners()*.

ret, corners = cv2.findChessboardCorners(temp, (self.rows, self.columns))

(OpenCV, s.f.)

Una vez encontradas las esquinas, se refina la posición de estas mediante la función de OpenCV *cornerSubPix()*. El coste operaciones de esta función viene determinado por dos criterios, alcanzar un número máximo de iteraciones (*TERM_CRITERIA_MAX_ITER*) y alcanzar un nivel de precisión determinado (*TERM_CRITERIA_EPS*). En nuestro caso se finalizamos si se alcanzan las cien iteraciones o si la precisión epsilon es de 10^{-5}

cv2.cornerSubPix(temp, corners, (11, 11), (-1, -1), (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS, 30, 0.01))

(Opencv, Funciones Opencv, s.f.)

Una vez encontradas las esquinas internas del tablero se procede a dibujarlas encima del propio patrón. Gracias a esto podemos comprobar si en alguno de los pares de imágenes se ha producido algún error o si faltan esquinas por detectar. Para tal fin utilizamos la función del paquete StereoVision *add_corners()*

calibrator.add_corners((imgLeft, imgRight), True)

Internamente esta función llama a otras funciones del propio paquete de StereVision, llegando finalmente a la invocación de la función de OpenCV: drawChessboardCorners()

cv2.drawChessboardCorners(temp, (self.rows, self.columns), corners, True)

(Opencv, Funciones Opencv, s.f.)

En este punto ya tenemos identificados todas los puntos de interés de cada par de imágenes tomadas para la calibración. La búsqueda de los parámetros intrínsecos y extrínsecos de las cámaras se lleva a cabo mediante la función de Sterevision, *calibrate_cameras()*.

calibrator.calibrate_cameras()

Esta función trabaja a nivel interno invocando a otras dos funciones de OpenCV

cv2.stereoCalibrate()

(Opencv, Funciones Opencv, s.f.)

Esta función se encarga de encontrar los parámetros intrínsecos para cada una de las dos cámaras y los parámetros extrínsecos existentes entre ellas.

La función estima la transformación entre dos cámaras que forman el par estéreo. Si se calculan las poses de un objeto en relación con ambas cámaras, (R_1, T_1) y (R_2, T_2) respectivamente, si la posición relativa y la orientación entre las dos cámaras son fijas, entonces esas poses están relacionadas entre sí. Esto significa que, si se conocen la posición relativa y la orientación (R, T) de las dos cámaras, es posible calcular (R_2, T_2) a partir de (R_1, T_1).

$$R_2 = RR_1$$
$$T_2 = RT_1 + T$$

Por lo tanto, se puede calcular la representación de coordenadas de un punto 3D para el sistema de coordenadas de la segunda cámara cuando se da la representación de coordenadas del punto en el sistema de coordenadas de la primera cámara:

Además de la información relacionada con el estéreo, la función también puede realizar una calibración completa de cada una de las dos cámaras.

Aunque a la hora de estimar los parámetros intrínsecos de las cámaras es posible hacerlo con una mayor precisión si se obtienen con anterioridad y de forma individual para cada cámara, en nuestro caso obtendremos los parámetros intrínsecos a partir de la propia función de calibración. Con el fin de evitar problemas derivados por ruidos en la imagen o la alta dimensionalidad de esta, se fijan ciertas características del sistema estereo que restringen algunos parámetros como son *CALIB_FIX_ASPECT_RATIO*, *CALIB_SAME_FOCAL_LENGTH* y *CALIB_ZERO_TANGENT_DIST*.

Por otro lado los criterios que marcan la finzalización del algoritmo vienen determinados por las mismas variables y valores que para la detección de esquinas *TERM_CRITERIA_MAX_ITER*, y *TERM_CRITERIA_EPS*.

- Rectificación

La siguiente función de OpenCV invocada es la encargada rectificar virtualmente la orientación de las cámaras con el fin de conseguir que los planos de imagen de ambas cámaras sean coincidentes. Es decir, simular que las imágenes han sido tomadas desde la misma posición y orientación de la cámara. Para ello calcula dos matrices de rotación. En consecuencia de esta rectificación se consigue que todas las líneas epipolares sean paralelas y, por tanto, simplifica el problema de la correspondencia estéreo.

cv2.stereoRectify()

(Opencv, Funciones Opencv, s.f.)

La función utiliza como entrada las matrices calculadas por la función anteriormente explicada (stereoCalibrate) y proporciona como salida dos matrices de rotación y dos matrices de proyección en las nuevas coordenadas. Esta función distingue dos modos de visión estereo, vertical y horizontal.

En nuestro caso utilizamos la rectificación horizontal por lo que las proyecciones de las cámaras se desplazan entre si a lo largo del eje X. En las imágenes rectificadas las líneas epipolares correspondientes entre ambas cámaras son todas horizontales y tienen la misma coordenada Y. Las matrices de proyección resultantes quedarían de la siguiente forma:

$$P_1 = \begin{bmatrix} f & 0 & cx_1 & 0 \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad P_2 = \begin{bmatrix} f & 0 & cx_2 & T_x f \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Donde T_x es el desplazamiento lateral entre las cámaras.

En último lugar se invoca la función de OpenCV *initUndistortRectifyMap*, la cual permite eliminar los errores debidos a la distorsión de la cámara. Esta función deberá ser llamada dos veces, una para cada cámara y tomará como valores de entradas la matriz de la cámara y los coeficientes de distorsión.

cv2.initUndistortRectifyMap

(Opencv, Funciones Opencv, s.f.)

Esta función crea los mapas para el algoritmo de mapeo inverso que utiliza la función de OpenCV *cv2.remap* que será utilizada mas adelante. Es decir, a cada píxel (u,v) en la imagen corregida y rectificada la función calcula las coordenadas correspondientes en la imagen de origen es decir, en la imagen original de la cámara.

Una vez finalizado el proceso de calibración y rectificación se guardan en un fichero de configuración todos los valores resultantes mediante la función

calibration.export('calib_result')

Los resultados se guardan en un fichero con formado JSON como se muestra en la siguiente estructura de ejemplo

```
{
   "SADWindowSize":9,
   "minDisparity":4,
   "numberOfDisparities":32,
```

```
    "preFilterCap":27,
    "preFilterSize":9,
    "speckleRange":0,
    "speckleWindowSize":0,
    "textureThreshold":11,
    "uniquenessRatio":1
}
```

Estos parámetros se serán invariables para el sistema de visión estereo y no será necesaria una nueva calibración a no ser que se cambien las propiedades físicas de este, como la orientación o posición entre cámaras.

Finalmente se carga la configuración de estos parámetros y se rectifica uno de los pares de imágenes de calibración.

```
calibration = StereoCalibration(input_folder='calib_result')
rectified_pair = calibration.rectify((imgLeft, imgRight))
```

La función rectificar del paquete *StereoVisión* llama internamente a la función *remap()* de OpenCV. Esta función tiene como entrada la imagen original y los mapas de distorsión y rectificación resultantes de la función *cv2.initUndistortRectifyMap()* y nos devuelve como salida la imagen rectificada.

```
cv2.remap(frames[i], self.undistortion_map[side], self.rectification_map[side],
cv2.INTER_NEAREST)
```

El resultado como se puede observar son dos imágenes “deformadas” cuyas lineas epipolares son las mismas, es decir, la proyección de un punto en las imágenes de cada una de las dos cámaras se encuentra en una misma línea horizontal.

Figura. Imágenes izquierda y derecha rectificadas.

Cálculo de la disparidad

Algoritmo Block Matching

(Martín)
(Wikipedia, s.f.)

Dentro de los algoritmos de estimación de movimiento podemos distinguir dos tipos.
Los algoritmos en el dominio de la frecuencia y los algoritmos en el dominio del tiempo.

Dentro de este último tipo encontramos los algoritmos basados en coincidecia, o black matching. Es el utilizado por la mayoría de herramientas software y hardware.

El funcionamiento de este algoritmo es el siguiente. Cada par de imágenes es dividido en bloques denominados macrobloques. El algoritmo pretende detectar el movimiento entre imágenes con respecto a estos macrobloques que las constituyen.

Los bloques de una imagen son cotejados con los bloques de la otra imagen de referencia, deslizando el actual a lo largo de una región concreta de píxeles de la imagen destino.

A partir de un criterio de semejanza se determina cuál de los otros bloques candidatos presenta una mayor similitud o minimiza un error medido. Este proceso se lleva a cabo dentro de una ventana de búsqueda de tamaño fijo. Si el bloque elegido no se encuentra en la misma posición en ambas imágenes significa que se ha desplazado. La distancia del bloque coincidente entre ambas imágenes se define como vector de desplazamiento estimado y será el que se les asigne a todos los píxeles del macrobloque.

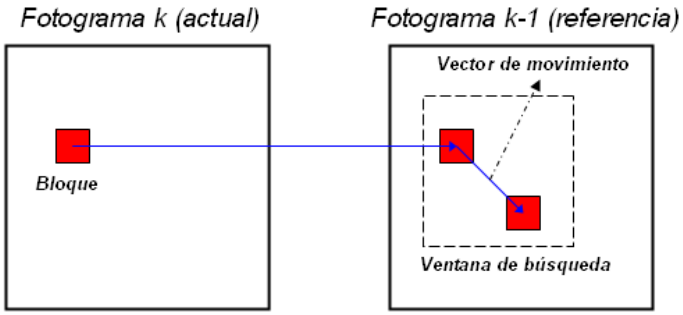


Figura. Representación de desplazamiento de un bloque en la ventana de búsqueda

A la hora de configurar el algoritmo de Block Matching se deben tener en cuenta dos cuestiones fundamentales que afectan tanto a la precisión y error de los resultados como al coste computacional de obtenerlos.
(Hirschmuller, Accurate and efficient Stereo Processing by Semi-Global Matching and mutual information)

- *Región de exploración*. Determina el tamaño de la región donde se va a llevar a cabo la búsqueda del bloque, es decir, el tamaño de la ventana de búsqueda. Aunque a primera vista podríamos suponer que mejor cuanto más grande, el coste computacional del algoritmo aumenta casi de forma cuadrática con el incremento de esta ventana.

  Un campo de exploración pequeño supondrá que los desplazamientos del bloque deben ser también pequeños, por tanto esta opción es interesante para desplazamientos lentos.

  Una solución común es por tanto es elegir una ventana de superficie ligeramente superior al tamaño máximo de los posibles objetos móviles.

- *Elección del bloque.* Determina la posición, tamaño y ubicación del inicio de la búsqueda, así como la escala de los bloques con los que trabajará el algoritmo. La elección de un tamaño de bloque adecuado vendrá determinada por los requisitos de nuestro sistema. Los bloques mayores son menos sensibles a posibles ruidos, mientras que bloques de dimensiones reducidas presentan una mejor definición de contornos. El principal factor sin embargo vendrá determinado por el tamaño de los objetos que pretendemos rastrear.

  El problema de apertura también debe tenerse en cuenta cuando trabajamos con objetos cuyo color es uniforme. En estos casos, si el tamaño elegido es demasiado pequeño, los bloques del interior del objeto parecen no moverse porque todo a su alrededor es del mismo color.

Clase StereoBM

Para este proyecto se ha utilizado un algoritmo de Block Matching perteneciente a OpenCV. Todas las funcionalidades se agrupan en la clase StereoBM y sus descendientes.
(Opencv, Funciones opencv, s.f.)

Para obtener el mapa de disparidad hacemos uso de la función propia *stereo_depth_map()* a la que pasamos como argumentos las imágenes rectificadas de la calibración.

disparity = stereo_depth_map(rectified_pair)

Dentro de esta función se llevan a cabo los siguientes procesos.

Creamos un objeto de tipo StereoBM para trabajar con las funcionalidades de Block Matching que proporciona esta clase de OpenCV.

sbm = cv2.StereoBM_create(numDisparities=16, blockSize=15)

A este objeto se le debe indicar dos parámetros de configuración:

- *numDisparities.* Configura el rango de búsqueda de disparidades. Para cada píxel, el algoritmo encontrará la mejor disparidad en un rango desde [0-numDisparities].

- *blockSize*. Configura el tamaño lineal de los bloques comparados por el algoritmo. El tamaño debe ser impar ya que el bloque está centrado en el píxel actual. Un tamaño de bloque más grande implica un mapa de disparidad más fluido, aunque menos preciso. Un tamaño de bloque más pequeño proporciona un mapa de disparidad más detallado, pero existe una mayor probabilidad de que el algoritmo encuentre una correspondencia incorrecta.

Una vez creado nuestro objeto configuramos todos los parámetros obtenidos de la calibración y llamamos a la función que calcula la disparidad.

disparity = sbm.compute(dmLeft, dmRight)

La función *compute()* calcula la disparidad entre dos imágenes que se le pasan como entrada y nos devuelve como salida el mapa de disparidad entre ambas. Este mapa tendrá el mismo tamaño que las imágenes de entrada y asignará un valor de disparidad a cada pixel.

- Escalado de resultados

Finalmente se escalan los valores del mapa de disparidad para obtener resultados en un rango de [0-1]

```
local_max = disparity.max()
local_min = disparity.min()
disparity_visual = (disparity-local_min)*(1.0/(local_max-local_min))
local_max = disparity_visual.max()
local_min = disparity_visual.min()
```

Cálculo de la profundidad

Como ya comentamos en el apartado de visión estereoscópica, para determinar la distancia a la que se encuentra un objeto de la escena se realiza una correspondencia entre las imágenes capturadas por las cámaras y se analiza el desplazamiento de puntos comunes que las componen.

Debido a la restricción epipolar que caracteriza nuestro sistema, así como la mayor parte del resto de sistemas estereoscópicos, las imágenes presentan un desplazamiento

horizontal que hace que las proyecciones de los puntos de la escena estén desplazados ligeramente. Como ya hemos mencionado, a esta diferencia de posición en el eje horizontal se le denomina dispardidad.
(Hirschmuller, Stereo processing by semiglobal matching and mutual information. Patter analysis and machine intelligence, 2008)

La conversión de la disparidad a distancias reales se lleva a cabo mediante una sencilla operación de semejanza de triángulos. En la siguiente imagen se muestra desde una vista de águila nuestro sistema de visión estereoscópico y un punto de la escena $P(X, Y, Z)$. La distancia b es la línea base correspondiente con la separación entre centro ópticos, f es la distancia focal y $P_I(x_I, y_I)$ y $P_D(x_D, y_D)$ son las proyecciones del punto en las imágenes izquierda y derecha respectivamente.

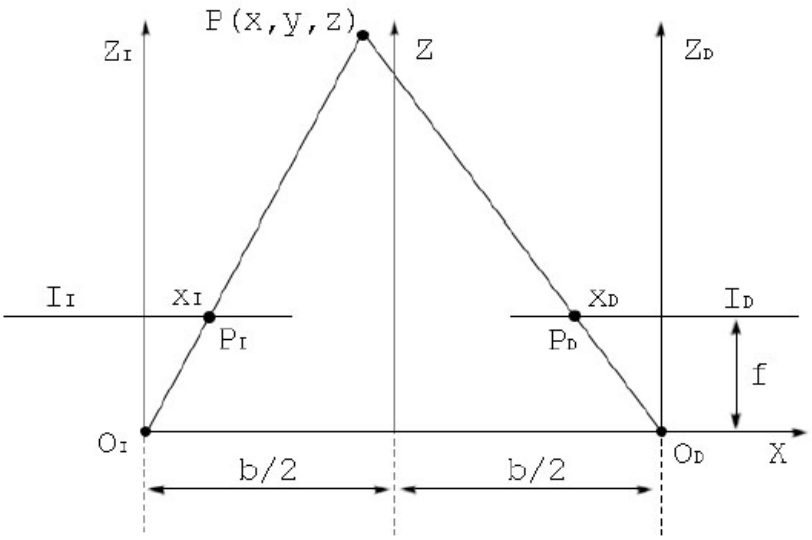


Figura. Geometría del cálculo de la profundidad a partir de la disparidad

Si aplicamos semejanza de triángulos para cada una de las proyecciones podemos obtener los valores de x_I y x_D.

$$Proyección\ izquierda: \frac{\frac{b}{2}+x}{z} = \frac{x_I}{f} \rightarrow x_I = \frac{f}{z}\left(x + \frac{b}{2}\right)$$

$$Proyección\ derecha: \frac{\frac{b}{2}-x}{z} = \frac{x_D}{f} \rightarrow x_I = \frac{f}{z}\left(x - \frac{b}{2}\right)$$

La disparidad se calcula como la diferencia de estos valores, por lo que a partir de esta podemos despejar el valor de la coordenada z del punto de la escena.

$$d = x_I - x_D = \frac{f}{z}\left(x + \frac{b}{2}\right) - \frac{f}{z}\left(x - \frac{b}{2}\right) = \frac{f}{z}b$$

$$z = \frac{f}{d}b$$

La profundidad es inversamente proporcional a la disparidad por lo que existe una relación no lineal entre ambas.

Cuando la disparidad es cercana a 0, pequeñas diferencias de disparidad producen grandes diferencias en profundidad. Cuando la disparidad es grande, pequeñas diferencias en disparidad casi no producen cambios en el valor de profundidad. Como consecuencia los sistemas de visión estéreo tienen una alta resolución en profundidad solo para objetos relativamente cercanos a la cámara.

Conociendo el mínimo incremento de disparidad posible Δd es posible calcular el mínimo valor de profundidad que el sistema puede medir, es decir, su resolución ΔZ.

$$\Delta Z = \frac{Z^2}{f}\Delta d$$

(Jesús Arturo Escobedo-Cabello)

APLICACIÓN DE USUARIO

A través de la aplicación de usuario controlamos los modos de trabajo que queremos visualizar y obtenemos la información proporcionada por el sensor. Esta aplicación se encarga también de realizar las peticiones de conexión al sensor. De forma adicional nos muestra en un cuadro de texto información referente a la conexión o los modos de trabajo.

Interfaz de usuario

La aplicación de usuario se presenta mediante una sencilla interfaz que cuenta de una barra de opciones, botones dedicados a la conexión/desconexión, botones de selección de modo de trabajo y un log de texto.

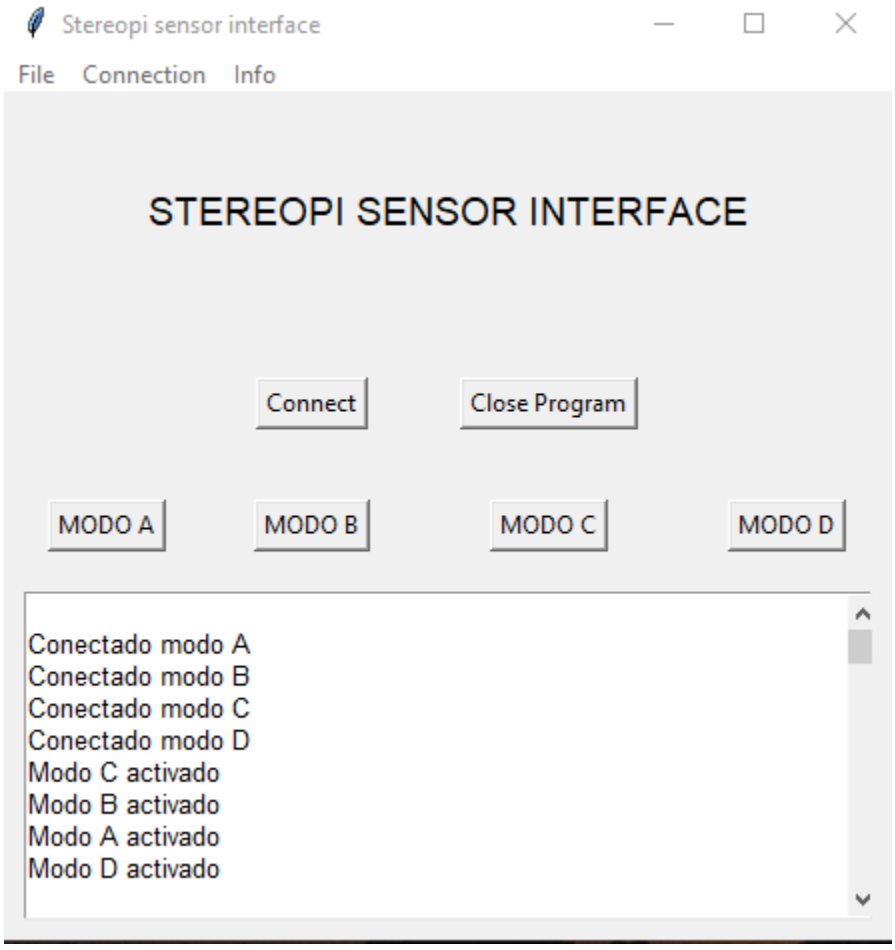


Figura. Interfaz de usuario

Las funcionalidades de las opciones de la barra y de los botones son las siguientes.

- ***Connect*** Conecta la aplicación con el sensor e inicia la comunicación.

- ***MODO A*** Abre o cierra la ventana del modo imagen original.
- ***MODO B*** Abre o cierra la ventana del modo de imagen rectificada y matriz de profundidad.
- ***MODO C*** Abre o cierra la ventana del modo mapa de profundidad.
- ***MODO D*** Abre o cierra la ventana del modo imagen rectificada y distancia al punto central.
- ***Disconnect*** Termina el programa.
- ***Info***. Muestra la información básica de la aplicación, versión, fecha, etc.

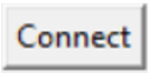
Arquitectura de la aplicación de usuario

En este apartado vamos a explicar la estructura de la aplicación de usuario a nivel de código. Nuestra aplicación se compone fundamentalmente de dos partes, la interfaz gráfica mediante la cual manejamos la aplicación y los hilos de comunicación que reciben las imágenes del sensor.

- Interfaz gráfica.

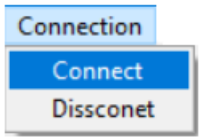
Toda la interfaz visible así como el manejo de las acciones mediante los botones o acciones de la barra han sido programaos haciendo uso de la librería *tkinter*

Cada botón de la interfaz tiene tres parámetros. El primero es común a todos, e indica que es una objeto de la interfaz en uso. El segundo es el texto que se mostrará encima del botón. Por último tenemos la función que se invoca al pulsar el botón. Ejemplo del botón de conexión.



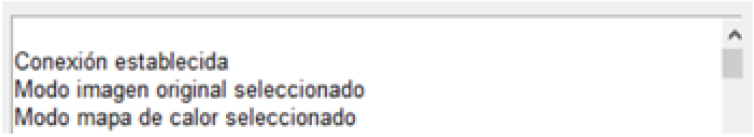
connectButton = Button(myFrame, text="Connect", command=tryConnection)

Las opciones de la barra se configuran de un modo muy similar



menuFile.add_command(label="Exit", command=exitApp)

El cuadro de texto en el que se muestra la información a tiempo real sobre las operaciones del sensor requiere como parámetros la configuración del tamaño, color y fuente de dicho texto.



txt = Text(myFrame, bg=BG_COLOR, fg=TEXT_COLOR, font=FONT, width=60, height=10)

- Hilos de comunicación.

Nuestra aplicación cuenta con cuatro hilos independientes de comunicación, uno para modo de trabajo. Es en estos hilos donde se lleva a cabo la recepción de las imágenes enviadas por el sensor y se comprueba si el modo correspondiente al hilo está activado, en cuyo caso se abre una ventana y se muestra la imagen recibida. De este modo es posible operar con varios modos de trabajo en simultáneo.

A continuación se muestra un esquema con la arquitectura y las comunicaciones internas de la aplicación donde cada recuadro turquesa es un hilo de ejecución.

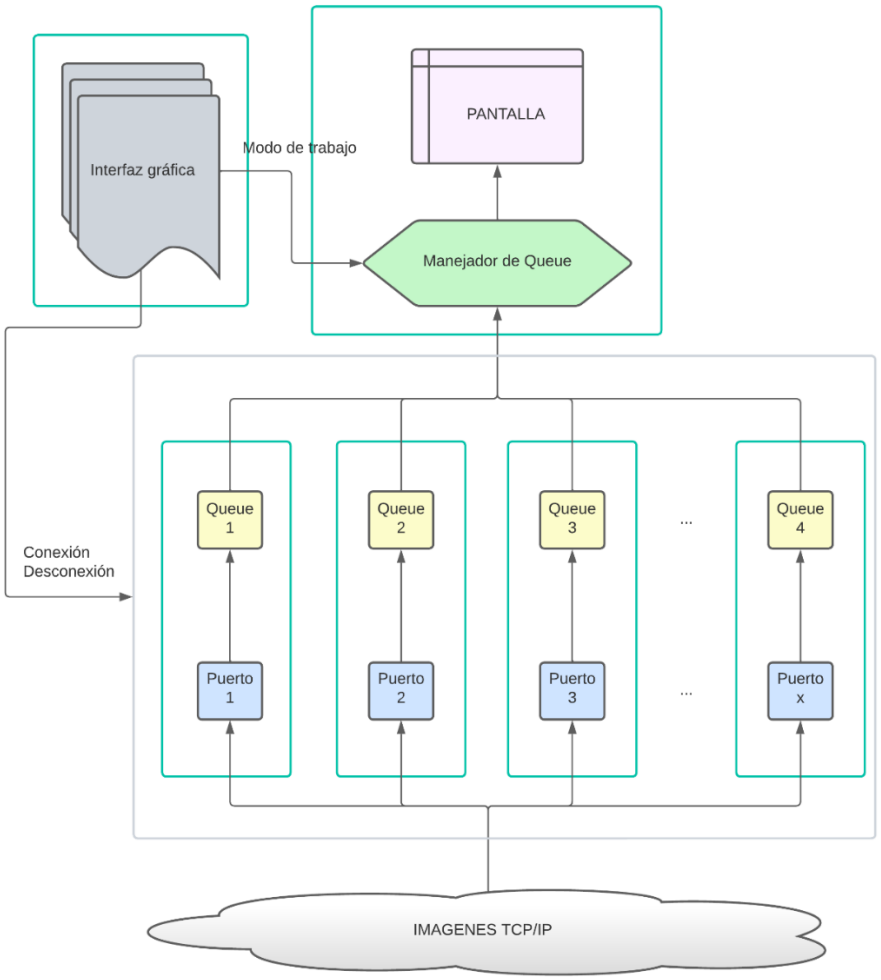


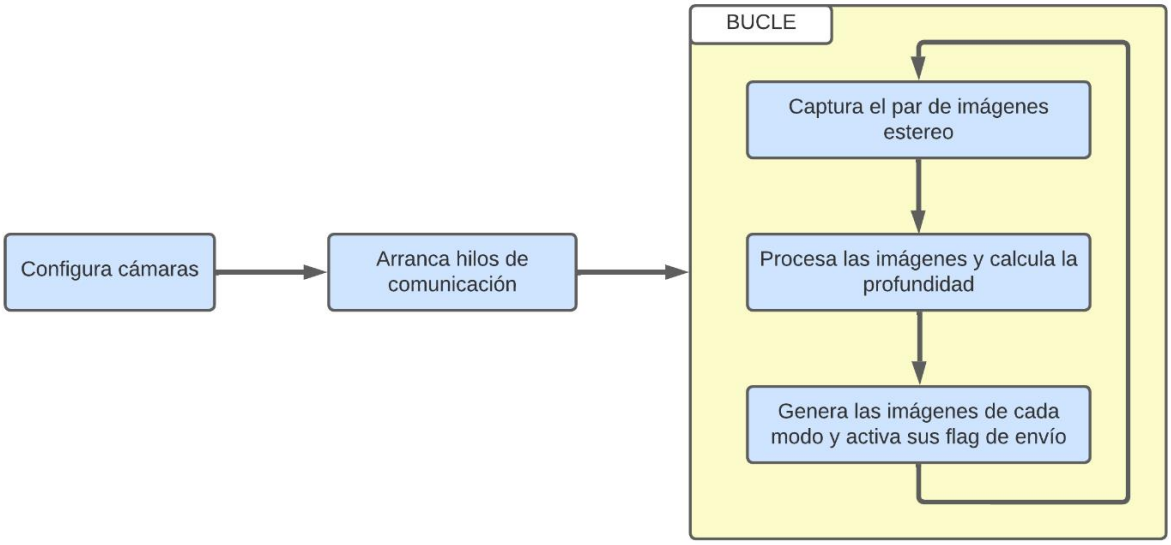
Figura. Arquitectura de la aplicación de usuario

APLICACIÓN DEL SENSOR

La aplicación del sensor es la encargada de la captura, procesamiento y envío de las imágenes de la cámara a la aplicación de usuario. Su arquitectura interna se basa en cuatro hilos, uno hilo principal y cuatro hilos de comunicación, uno por cada modo de trabajo.

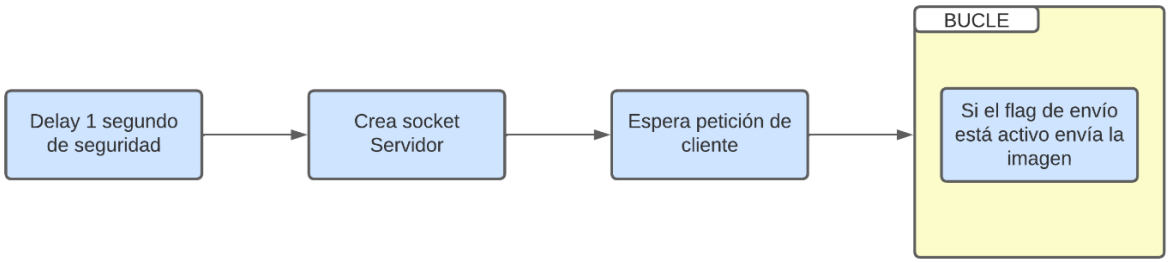
- Hilo principal

En el hilo principal se llevan a cabo todos los pasos referentes a la captura y el procesamiento del sensor. Además, es el encargado de arrancar los hilos de comunicación una vez se ha configurado la cámara. Su funcionamiento interno consta de los siguientes pasos.

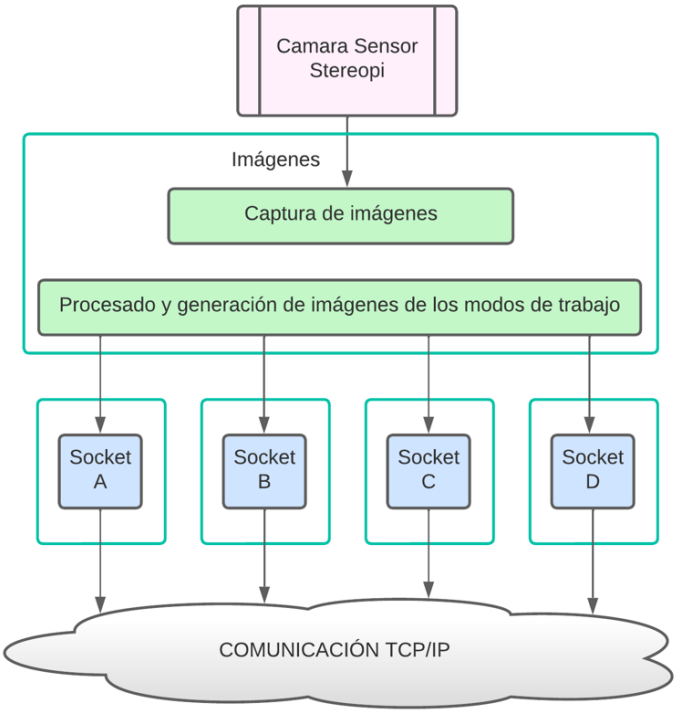


- Hilos de comunicación

Cada modo de trabajo cuenta con un hilo de comunicación independiente sin embargo en todos ellos el funcionamiento interno es el mismo. La conexión cliente-servidor con la aplicación de usuario se lleva a cabo mediante una comunicación TCP, para la cual se utiliza la librería socket que nos brinda las funcionalidades necesarias para crear y configurar los distintos sockets, y enviar la información. El funcionamiento interno de estos hilos es muy sencillo.



La estructura a nivel de hilos y sockets de comunicación se puede ver en el siguiente diagrama, donde cada recuadro turquesa encierra los procesos de un hilo.



Arranque y puesta en marcha

El sistema de visión estereoscópico debe ser accesible sin ningún tipo de intervención una vez esté encendido, es decir, debe arrancar y ejecutar los programas necesarios para trabajar de forma independiente y opaca para el usuario.

Para tal fin se ha introducido un proceso en el arranque del sistema operativo que ejecuta la aplicación del sensor cada vez que arranca el dispositivo. Este proceso se ha implementado mediante el uso de *daemons* que proporciona el sistema operativo Raspbian. Los daemons son un tipo especial de programa que se ejecuta en segundo plano, en vez de ser controlado directamente por el usuario, es decir, no disponen de una interfaz directa con el usuario, ya sea gráfica o textual.

A la hora de generar un daemon es necesario indicarle algunos parámetros. Entre los más importantes destacamos los siguientes:

ExecStart.	Ruta del ejecutable o script que queremos arrancar
Type.	Configura el inicio de nuestro servicio
Restart.	Indica si el servicio debe reiniciarse o no y en qué condiciones
RestartSec.	Define el tiempo que debe transcurrir hasta que se intenta el reinicio

(Atareado, s.f.)

Los parámetros de configuración del servicio son los siguientes:

```
[Unit]
Description= My StereoPi service
After=multi-user.target

[Service]
Type=simple
ExecStart=/usr/bin/python3 /home/pi/myservice.py

Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Modos de Trabajo

El sensor proporciona a la aplicación cuatro imágenes diferentes de manera simultánea, una para cada modo de trabajo. A continuación se detallan las características de cada uno.

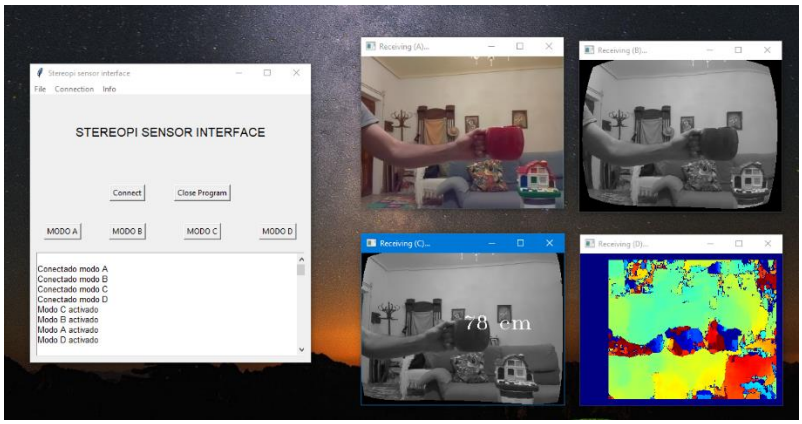


Figura. Interfaz y los cuatro modos de trabajo simultáneos

- Modo A. Imagen original

En este modo de trabajo se envía la imagen original tomada por el sistema de visión sin ningún otro tipo de procesamiento. Nos proporciona una guía para estimar si el resto de modos trabajan correctamente. La configuración de la cámara ya ha sido vista en el apartado de calibración por lo que no incidimos nuevamente en ella. Para la captura de frames se utiliza nuevamente la función capture_continuos.

capture_continuous(capture, format="bgra", use_video_port=True, resize=(img_width,img_height)):

La imagen recibida por la aplicación se muestra en la siguiente figura.

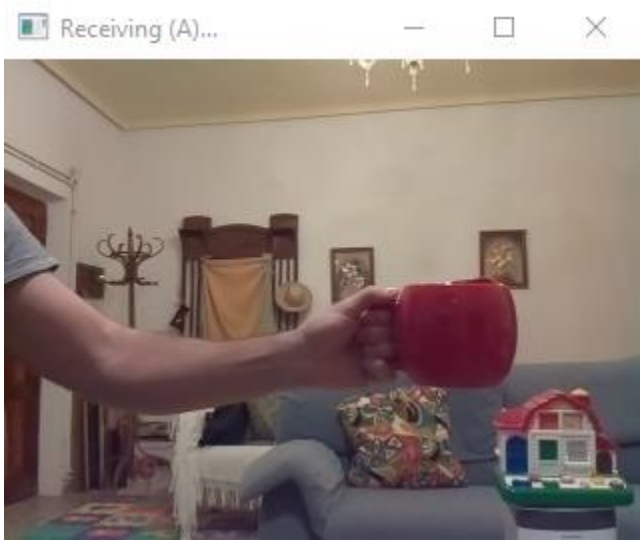


Figura. Imagen modo A

- Modo B. Imagen rectificada y matriz de disparidad.

En este modo de trabajo se envía la imagen rectificada en escala de grises y posteriormente un objeto matriz con los datos de la disparidad. No es objeto de este trabajo dar una utilidad a esos datos, por lo que estos datos no se almacenan en ningún fichero en la versión final de la aplicación. Sin embargo fueron utilizados para los cálculos de la curva de calibración durante las fases de medición.

La función que se encarga de la rectificación de la imagen es la misma utilizada para la calibración.

rectified_pair = calibration.rectify((imgLeft, imgRight))

La imagen recibida por la aplicación se muestra en la siguiente figura.

Figura. Imagen modo B

- Modo C. Imagen rectificada y distancia al punto central

En este modo de trabajo se envía la imagen rectificada en escala de grises en la que se muestra la distancia del punto central de la imagen.
Esta distancia se obtiene de calcular la media de las distancias obtenidas para un subgrupo de píxeles del punto central de la imagen.

Figura. Imagen modo C

- Modo D. Mapa de profundidad

Este modo de trabajo envía la imagen del mapa de profundidad calculado a partir del par estéreo.

```
rectified_pair = calibration.rectify((imgLeft, imgRight))
disparity = stereo_depth_map(rectified_pair)
```

Esta función también fue explicada en el apartado Clase StereoBM por lo que solo se analizarán las siguientes líneas que normalizan y adaptan el mapa de profundidad.

```
disparity = sbm.compute(dmLeft, dmRight)
local_max = disparity.max()
local_min = disparity.min()
disparity_grayscale = (disparity-local_min)*(65535.0/(local_max-local_min))
disparity_fixtype = cv2.convertScaleAbs(disparity_grayscale, alpha=(255.0/65535.0))
disparity_color = cv2.applyColorMap(disparity_fixtype, cv2.COLORMAP_JET)
```

En primera lugar normalizamos los valores obtenidos para que la diferencia máxima de disparidad sea de 65535 y la mínima 0, resultados que guardamos en disparity_grayscale
Mediante la función convertScaleAbs adecuamos los valores del mapa en el rango [0-255] el cual es el rango indicado para obtener una imagen en escala de grises.
Por último la función applyColorMap implementamos la gama de colores asociada a cada valor en escala de grises.

La función *stereo_depth_map* devolverá la imagen del mapa de disparidad en la gama de colores seleccionada

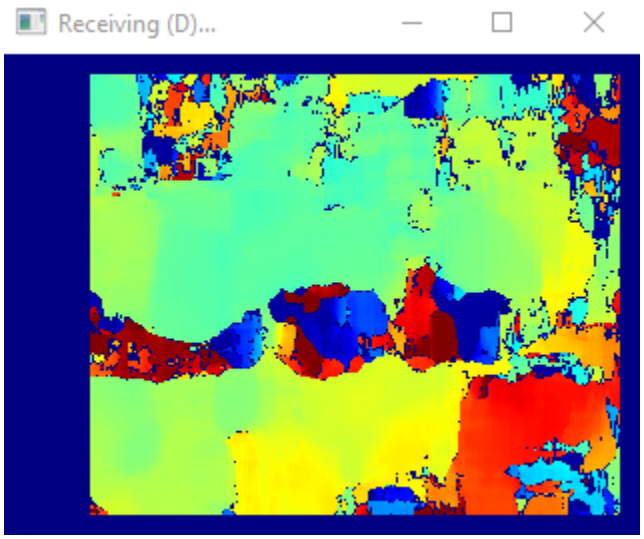


Figura. Imagen modo D

RESULTADOS

Curva de calibración

Como mencionamos en los apartados “Cálculo de la disparidad” y “Cálculo de la profundidad”, cada píxel de la imagen tiene asignado un valor de disparidad, que es la distancia horizontal que se ha desplazado el pixel entre las imágenes de cada cámara.

$$d = x_I - x_D$$

A partir de la configuración de nuestro sistema estéreo, y conociendo la disparidad y los valores de calibración de la cámara, es posible determinar la distancia Z real de un punto de la imagen a partir de la siguiente ecuación

$$z = \frac{f}{d} b$$

Donde *z* es la profundidad real de la escena en unidades de longitud, *d* la disparidad en unidades de píxeles, *f* la distancia focal en píxeles y *b* la distancia entre centros ópticos en unidades de longitud. En la etapa de calibración calculamos los valores de *b* y *f* por lo que a partir de estos datos podemos obtener la curva de calibración del sistema.

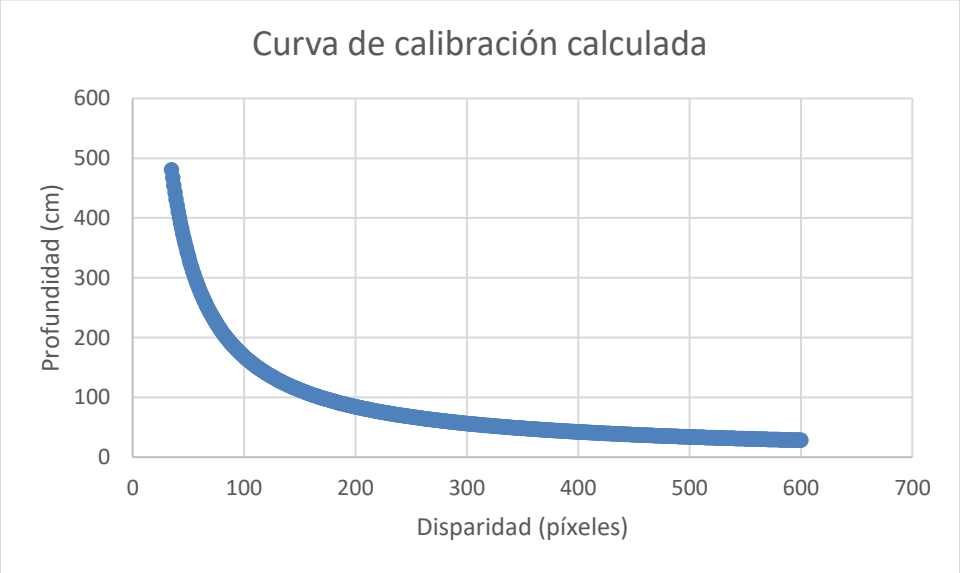


Figura. Calibración calculada

Como podemos comprobar la curva se ajusta a la ecuación de dos variables inversamente proporcionales, disparidad y profundidad. Se puede corroborar a su vez como pequeños cambios de disparidad en las cercanías del sensor producen grandes cambios en su profundidad, mientras que para los objetos más alejados los cambios de disparidad pasan a ser imperceptibles.

Pasamos ahora a obtener experimentalmente la curva de calibración con nuestro sensor. Debemos tener en cuenta que por defectos en la calibración, para muchos de los píxeles los valores de disparidad serán erróneos, por lo que en la práctica es recomendable trabajar con los valores medios de un grupo de píxeles. Por otro lado si estos grupos de píxeles son muy grandes, se producirán errores en los bordes de los objetos.

(Juan Musuña Toapanta)

Para calcular la curva de calibración experimental se han llevado a cabo una serie de medidas de un mismo objeto a diferentes distancias, anotando los valores de disparidad que calculaba el sensor en cada caso.

Distancia Real (cm)	Disparidad (píxeles)	Distancia experimental (cm)	Error Absoluto (cm)	Error Relativo (%)
300	46	365.53	65.53	21.84
200	75	224.19	24.19	12.10
150	121	138.96	11.04	7.36
100	156	107.78	7.78	7.78
50	384	43.79	6.21	12.42

Con estos datos construimos la curva de calibración experimental que nos muestra el modelo real que el sensor está calculando.

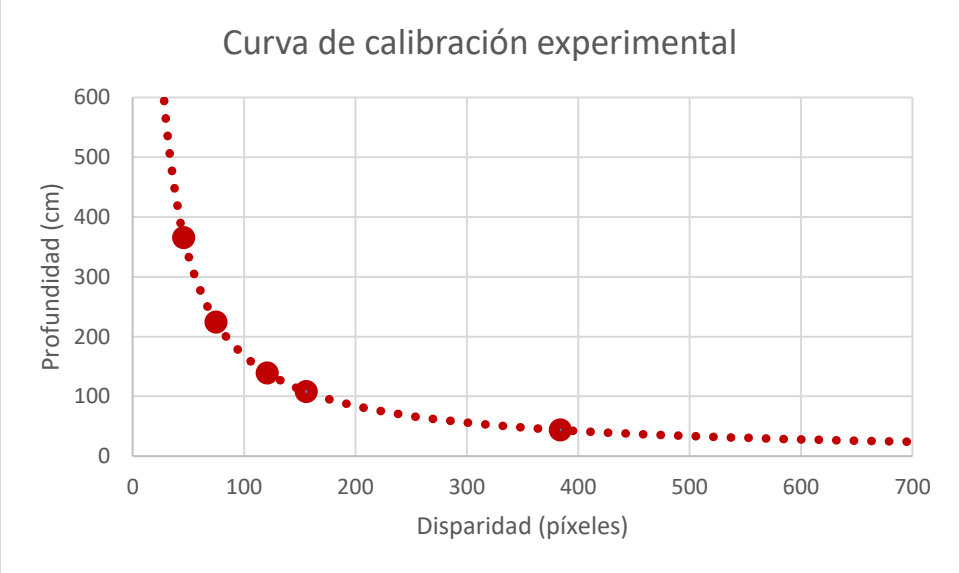


Figura. Calibración obtenida mediante resultados experimentales

A partir de los 300cm el sistema era prácticamente incapaz de detectar cambios en los objetos y cuando lo hacía los resultados no eran estables ni concluyentes.

La razón por la que no se han utilizado más medidas en la construcción de la curva de calibración ha sido la variabilidad en los resultados obtenidos. Se han realizado numerosas pruebas y los resultados siempre eran dispares entre sí, no consiguiendo un patrón estable de resultados, por lo que un número elevado de medidas acababa empeorando los resultados a la hora de calcular una línea de tendencia.

A la vista de ambas curvas de calibración podemos afirmar que nuestro sensor se comporta dentro de unos términos "aceptables" con unos errores relativos máximo inferiores en términos generales al *25%* y una media del error relativo del *12.3%*

Cabe destacar que durante todo el desarrollo del proyecto se han obtenido resultados dispares tanto en la etapa de calibración como en las medidas de profundidad. En muchas ocasiones obteniendo resultados dispares con unos mismos parámetros dependiendo del día e incluso de la hora a la que se practicaban. De este comportamiento concluyo que la iluminación tanto durante el proceso de calibración como durante el proceso de medición es una parte mucho más fundamental e importante de lo que hubiese imaginado en un primer momento.

Análisis experimental de tiempos

Una parte fundamental en los resultados es el análisis la frecuencia de imágenes a la que es capaz de proporcionar el sensor. Esta frecuencia determina directamente las posibles aplicaciones que puede tener nuestro sensor y si puede ser considerado como un sistema de visión en tiempo real.

El sensor de las cámaras está configurado para limitar el framerate a 20 frames por segundo por lo que no podemos esperar mejor resultado que este. Se midieron los tiempos entre frames recibidos en la aplicación cliente para cada modo de trabajo obteniendo los siguientes resultados.

Modo A	17
Modo B	15
Modo C	14
Modo D	17

Estos valores han sido calculados tomando el valor medio de tres pruebas en las que se introducían y movían objetos en la escena a diferentes distancias del sensor.

Problemas en el desarrollo del proyecto

Durante el desarrollo del trabajo hemos enfrentado una serie de problemas que han afectado tanto a los resultados de medición del sensor como a sus futuras posibles aplicaciones. Señalamos a continuación algunos de los más relevantes con el fin de prevenir su aparición en proyectos relacionados.

- Alimentación insuficiente

Uno de los primeros problemas que enfrentamos fue de los más complicados de identificar y de los más fáciles de solucionar. Durante varias semanas el sistema se apagaba y reiniciaba sin razón aparente. Los procesos de instalación de paquetes y librerías tenían que ser repetidos en numerosas ocasiones porque estos reinicios se producían antes de completarse. Este problema supuso muchas horas dedicadas a buscar información sobre reinicios y cuelgues del sistema sin encontrar solución ninguna.

Finalmente se midió la alimentación que llegaba a la Raspberry a través de un lector USB y se comprobó que los niveles de tensión no eran del todo estables.

Figura. Voltímetro digital USB

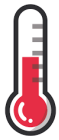
La solución fue tan sencilla como cambiar el trasformador de alimentación por otro más potente y fiable, con lo que todos los problemas de reinicio quedaron solucionados.

- Sobrecalentamiento del sensor

En el trascurso de los procesos de calibración, medición y testeo, el sensor estuvo en funcionamiento procesando imágenes durante largos periodos de tiempo. Durante estos periodos ocasionalmente el sistema se “colgaba” y la pantalla quedaba congelada, no respondiendo a ningún periférico ni a comandos de reinicio. En principio se achacó al citado problema de alimentación deficiente, pero finalmente se comprobó como siempre que ocurría este problema, el microprocesador de la placa Raspberry estaba muy caliente.

En pruebas posteriores se comprobó las temperaturas de CPU y GPU mediante el comando *vcgencm measure_temp* para la temperatura de la GPU y el comando *cat /sys/class/thermal/thermal_zone0/temp* para la temperatura de la CPU. Pudimos observar

que los valores devueltos daban temperaturas de funcionamiento críticas, cercanas a las 80ºC. En algunas ocasiones se llegó a visualizar en la propia interfaz de la raspberry el icono de aviso de temperatura elevada, situación ante la que rápidamente se apagó el sistema y se dejó enfriar.



Se aconseja por tanto el uso de un disipador en futuros proyectos que requieran una alta carga de CPU o GPU.

- Iluminación

Como ya hemos mencionado el proceso de calibración de las cámaras estéreo es la parte más importante de todo el sistema de visión. Cualquier error inducido en este paso genera grandes errores en las mediciones de disparidad. Este paso se repitió decenas de veces a lo largo del proyecto intentando obtener los mejores resultados posibles variando las condiciones ambientales, patrones de calibración, etc. Durante los procesos de calibración identificamos dos aspectos determinantes para la obtención de buenos resultados.

El primer aspecto es la iluminación de la escena. Es imprescindible contar con una iluminación suficiente y estable a la hora de calibrar los patrones. Sin embargo las iluminaciones fluorescentes o LED pueden tener efectos adversos en este sentido ya que la luz que emiten no es constante sino que parpadean a una alta frecuencia, siendo posible por tanto que las imágenes capturadas durante una misma sesión de calibración tengan iluminaciones dispares. Por otro lado este tipo de iluminaciones directas provocan brillos y reflejos en el patrón, lo que en ocasiones conlleva con errores en la detección de esquinas en el mismo.

Durante nuestras pruebas de calibración obtuvimos resultados dispares cuando trabajamos con una iluminación proveniente de varias lámparas, todas ellas LED, siendo los resultados más homogéneos al calibrar durante las mañanas con iluminación natural.
Es por ello que se aconseja de una abundante iluminación de luz natural o de bombilla de filamento y si fuese posible reflectores de luz blancos en los lados del campo de visión.

- Patrón de calibración

En segundo lugar debemos considerar la calidad del patrón de calibración. Todo el proceso de calibración se basa en encontrar correspondencias entre puntos clave del patrón de calibración, por lo que los defectos físicos del mismo pueden dar lugar a problemas en la detección de esquinas y errores en la obtención de correspondencias.

Con frecuencia nuestro proceso de calibración no detectaba todas las esquinas del patrón o la posición de las mismas no era muy precisa. Para estas pruebas se utilizaba un patrón

impreso en un papel no mate, de bajo gramaje y pegado sobre un trozo de cartón recortado de una caja. Con el tiempo se descubrió que las pequeñas dobleces del cartón, el reflejo del papel y la baja calidad de impresión podían estar afectando negativamente.
So optó por tanto en fabricar otro nuevo tablero de calibración sobre una superficie maciza y completamente lisa para lo que utilizamos la parte trasera de un espejo. El patrón de ajedrez fue impreso en un papel de alto gramaje y de brillo mate.

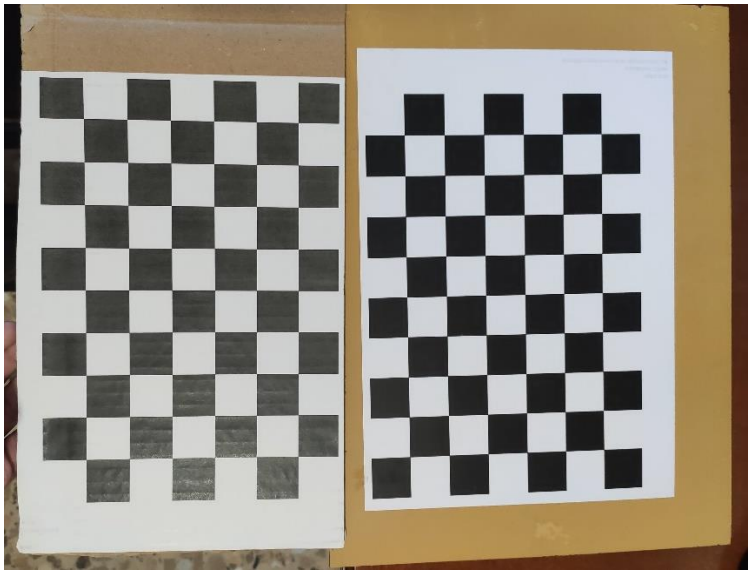


Figura. Comparación entre los tableros de calibración bajo la misma luz. A la izquierda el original, a la derecha el mejorado.

Con este nuevo patrón se redujeron las imágenes de calibración que fallaban en la detección de todas las esquinas aproximadamente de un 15% a un 7% y se mejoró la precisión en el posicionamiento del resto.

DISCUSIÓN DE RESULTADOS

A raíz de los resultados obtenidos realizamos un estudio acerca de las características finales de nuestro sensor y que campos y aplicaciones podría ser utilizado.

Analizaremos los resultados desde tres ópticas diferentes

- Aplicaciones en función de la precisión

De las pruebas experimentales obtuvimos un error relativo inferior al 25% y de valor medio cercano al 13%. Estos errores corresponden a un valor demasiado elevado para cualquier sistema que necesite una precisión media-alta o en los que los errores en las distancias puedan suponer consecuencias negativas. También cabe destacar el corto rango de profundidad en el que trabajo el sensor por lo que no sería útil para aplicaciones que requieran largas distancias.

Por tanto, nuestro sistema de visión queda fuera de lugar para actividades que requieran una alta precisión. Por otro lado el error es lo bastante pequeño para utilizarse en funciones que requieran detección de presencias en rangos de cercanía (cerca, media distancia, lejos).

- Análisis en función del tiempo de ejecución

El análisis del framerate ha dado como resultado una media de 15 FPS estables en el conjunto de los modos de trabajo. Esta tasa de refresco aun no siendo muy elevada es suficiente para considerar nuestro sistema de visión como un sistema a tiempo real.

Como es obvio esta tasa de refresco no será adecuada para ningún sistema crítico en el que se necesite de información inmediata. Sería el caso, por ejemplo, de detecciones de elementos móviles a alta velocidad. Por el contrario, nuestro sensor no queda lejos del ratio de refresco a partir del cual el ojo humano percibe movimiento fluido (24FPS), por lo que nuestro sistema podrá ser utilizado en aplicaciones sustitutivas de la visión humana. Pueden servir como ejemplo el sistema de visión de un robot móvil, una cámara de vigilancia que aporte información de los elementos que aparecen en la imagen, etc.

- Análisis en función del coste

Este es sin duda el punto fuerte de nuestro sistema de visión y por el cual se le puede restar importancia al resto de inconvenientes. Los costes totales de nuestro sistema son muy inferiores a los de cual otra solución comercial del mercado. El comedido precio del sistema lo hace oportuno para aprendizaje autodidacta en pequeños proyectos, donde el sector de la educación puede ser especialmente provechoso.

EJEMPLOS DE APLICACIÓN

En el apartado de discusión de resultados hemos delimitado las ventajas y limitaciones de nuestro dispositivo. El objetivo de este trabajo abarca el estudio de posibles campos de aplicación reales en los que el sistema de visión podría ser aprovechado. A continuación se analizan una serie de aplicaciones en las que nuestro sistema de visión podría ser integrado.

Sensor de aparcamiento en vehículo industrial

Los vehículos actuales ya cuentan con diferentes sistemas de ayuda al aparcamiento, como sensores de proximidad y cámaras, incluso en los paquetes de gama alta cámaras 360º. Estos dispositivos suelen operar de forma redundante, dando una imagen del entorno a partir de las cámaras y midiendo con sensores de proximidad los objetos colindantes.

Un sistema de visión estereoscópico como el nuestro tendría la capacidad de dar simultáneamente imagen y profundidad con un solo dispositivo lo que conllevaría al ahorro de costes de fabricación.

Los sistemas de control de distancia de aparcamiento más comunes son los que operan por ultrasonidos. Estos sistemas tienen algunas desventajas que podrían ser solucionadas por un sistema de visión estereoscópica.

- Requieren de varios sensores de ultrasonidos para una correcta identificación de objetos, mínimo tres.

- Error en la detección de ciertos objetos. Los sensores de ultrasonidos pueden fallar en la detección de objetos de determinadas formas o si se encuentran posiciones no alcanzables a las ondas de ultrasonidos. Podría ser el caso de un objeto saliente en posición horizontal como una rama de un árbol bajo, un hierro fino clavado en el suelo. Otro caso que pueden dar resultados erróneos son elementos donde los ultrasonidos no reboten correctamente, como pueden ser vallas de alambre u objetos muy por debajo de la línea del sensor.

En todos estos casos un sistema de visión estéreo solucionaría el problema ya que además de la distancia proporciona una imagen de la escena, por tanto el usuario sería capaz de identificar por sí mismo potenciales peligros para la maniobra.

Por otro lado nuestro sistema de visión también presenta inconvenientes. Nuestro sistema calcula la profundidad a partir de la comparación de las imágenes tomadas por cada cámara, es por ello que el objeto a medir debe estar contemplado en ambas imágenes, y esto solo se cumple a partir de una distancia mínima. Esto limita la ubicación de la cámara a posiciones donde se pueda encuadrar con ambas cámaras un objeto que esté tocando el vehículo. Nuestro sistema debería ubicarse idealmente a una altura suficiente y en la parte más posterior del vehículo, lo que nos plantea un problema a la hora de implementar nuestro sistema en vehículos personales como son los coches.

La mayor parte de los coches están diseñados para albergar un maletero en su parte trasera, este maletero suele, en el mejor de los casos, tener formas irregulares que imposibilitan la visión de la parte baja del coche desde la parte alta; en el peor de los casos, como el segmento de las berlinas o lo sedán, el maletero directamente sobresale ampliamente del resto del coche. En estos casos nuestro sistema no podría ser implementado ya que perderíamos la capacidad de visión cercana a la parte trasera bien por ocultación del maletero, o por el encuadre de objetos.

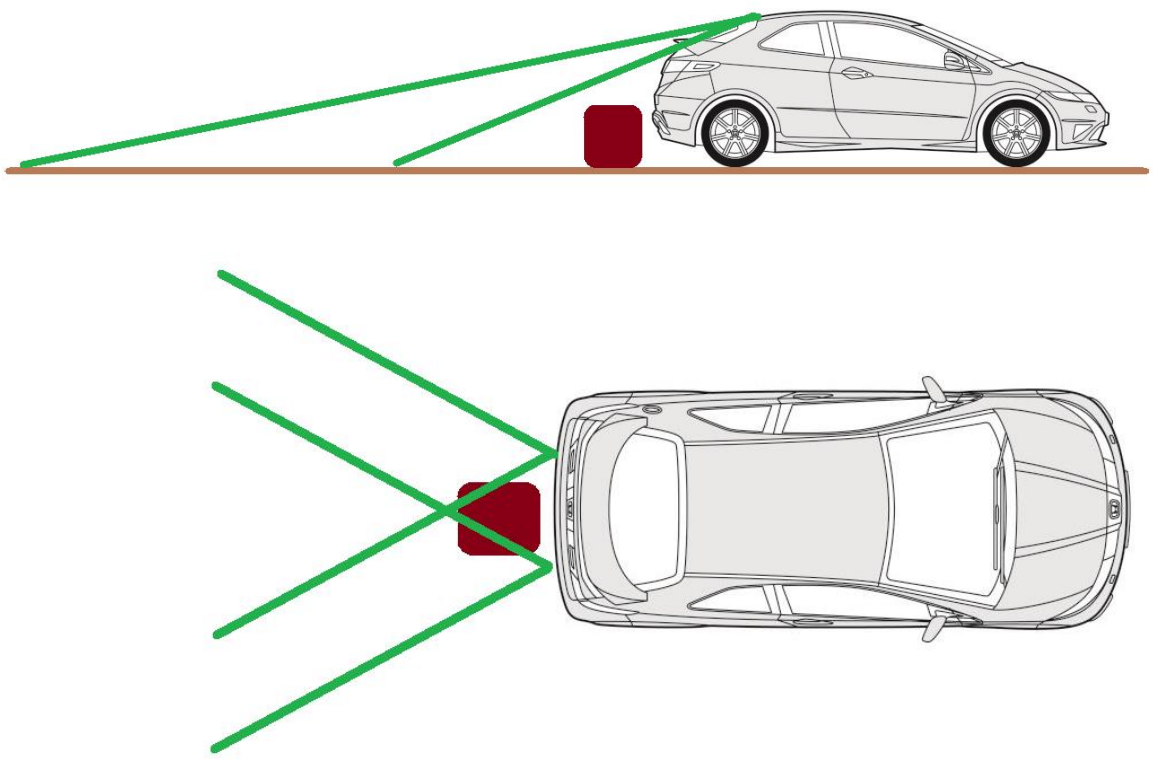


Figura. Ejemplos de limitación de visión por ocultación de objetos y encuadre por cercanía al sensor.

Por esta razón se ha delimitado el segmento de aplicación a vehículos industriales los cuales suelen tener un diseño para el aprovechamiento óptimo del espacio, lo que se traduce en partes traseras completamente verticales, donde la zona de carga carece de ocultación de ningún elemento.

Si bien es cierto que esta idea puede ser interesante, deberían realizarse algunas modificaciones al sistema de visión. La principal limitación para la implementación de nuestro sensor como sistema de control de distancia es el reducido campo de visión con el que podemos operar. Si quisiéramos tener una imagen de toda la parte trasera de un vehículo sería necesario utilizar cámaras de gran angular o de "ojo de pez". Estas cámaras están incluidas en nuestro Kit StereoPi y permiten un ángulo de visión de hasta 160º, ofreciendo las cámaras actualmente utilizadas un ángulo de tan solo 62º.

En el caso de cambiar a las cámaras de gran angular las únicas modificaciones que se realizarían en el proyecto sería el proceso de calibración, siendo funcionales por tanto la aplicación cliente como la del sensor.

Otra aplicación posible en esta línea sería la detección de ángulos muertos, en cuyo caso los sensores de visión se debería situar en los costados del vehículo.

Figura. Ángulos muertos en un camión

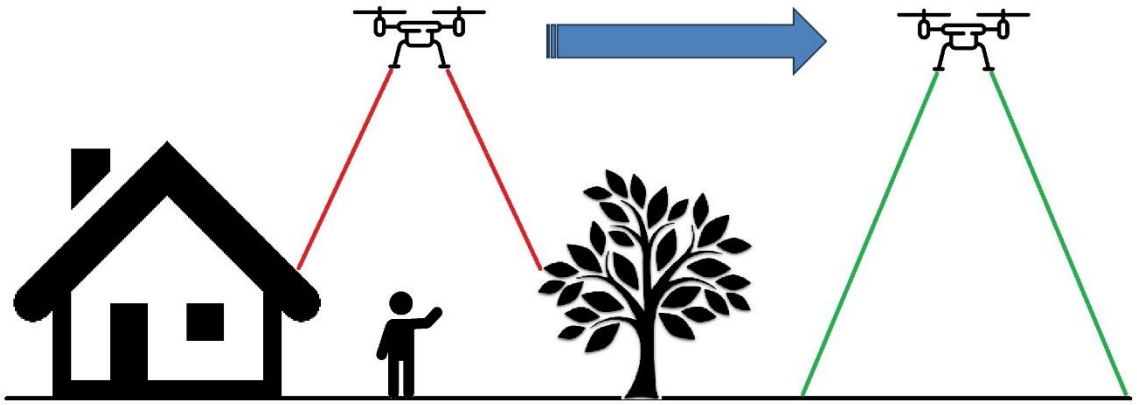
Sistema anticolisión para drones de paquetería

Los vehículos aéreos no tripulados, más comúnmente denominados drones, constituyen un sector tecnológico cada vez más afianzado en todo tipo de actividades, desde sus orígenes militares, pasando por industria del cine, hasta a pequeños drones destinados a juguetería.

Uno de los sectores con más potencial para el uso de drones es de la paquetería de corta y mediana distancia. Este tipo de soluciones ya están puestas en marcha por grandes empresas como Amazon con proyectos piloto para el envío de mercancías en entornos urbanos.

Los drones disponen de sistemas de guiado GPS con el cual son capaces de volar de manera autónoma dadas unas coordenadas de destino. Sin embargo, para llevar a cabo un envío no vale con conocer con precisión la coordenada GPS donde se debe depositar el paquete, el dron debe disponer de la capacidad de reconocer el entorno donde piensa aproximarse o incluso aterrizar para evitar así obstáculos e incluso personas.

Es aquí donde nuestro sistema de visión estereoscópica puede ser aprovechada. Una vez el dron haya llegado a la posición GPS indicada, el sistema de visión será capaz de reconocer objetos en la escena bajo el dron y calcular la distancia a cada uno de ellos. A partir de esta información se puede programar el dron para que evitar los elementos peligrosos y encontrar la mejor ruta para aterrizar o desplegar el paquete.



Adicionalmente se podría implementar un eje de movilidad a las cámaras para que durante el vuelo apuntaran en la dirección en la que vuela el dron. Así se permitiría la detección de elementos peligrosos (antenas, postes de telefonía, torres eléctricas, etc.) también durante el vuelo.

El bajo peso y tamaño del sistema StereoPi lo hacen ideal para este tipo de labores en las que el volumen y peso de la carga es fundamental.

RECOMENDACIONES Y TRABAJOS FUTUROS

El desarrollo de este proyecto me ha llevado a analizar el amplio abanico de posibilidades que proporciona un sistema de visión estereoscópico. Sin embargo, como he podido comprobar bajo mis propios resultados, la calibración y puesta a punto de estos sistemas no es para nada trivial, pues es necesario tener amplios conocimientos en el campo de la visión y la programación para conseguir resultados óptimos que permitan al sensor trabajar con precisión y rapidez.

Es por esto que en una primera línea de trabajo futura que recomiendo y que espero llevar a cabo es una mejora de las características del sensor a nivel tanto de hardware como de software y un control de calidad exhaustivo en el proceso de calibración.

- Hardware. El kit *Stereopi* con el que se ha desarrollado este trabajo combina un módulo *Raspberry Pi 3* Lite junto a un par de cámaras *Raspberry Pi Camera* de primera generación *(V1)*. Actualmente se disponen de versiones más modernas con mas resolución que permitirían un proceso de calibración más efectivo, así como un módulo de procesamiento más potente permitiría obtener mejores rendimientos en la tasa de refresco.

- Calibración. Como se ha visto a lo largo del trabajo, el proceso de calibración es la parte fundamental a tener en cuenta. La calidad de la iluminación de la escena debe tenerse en cuenta como una prioridad y no se debe escatimar en recursos. Por ello se plantea como medida utilizar un set de iluminación fotográfico. Por otro lado el patrón de calibración también es una pieza fundamental. En este proyecto se utilizó una impresión del tablero de ajedrez pegada sobre una superficie, sin embargo se recomienda comprar un tablero ya fabricado para minimizar errores.

- Software. En este proyecto se ha programado una aplicación de usuario para controlar el sensor, esta aplicación debe estar conectada por red al sensor para ser utilizada. Sin embargo, sería de gran utilidad poder manejar el sensor sin necesidad de estar físicamente cerca. Por ello se propone la migración de la aplicación de usuario a un web a partir de la cual podamos conectar vía internet al sensor.

Como segunda línea de trabajo, la cual estaba originalmente pensada para este mismo proyecto, se ha planteado la implantación de un sistema de detección de objetos a nuestro sensor. Actualmente existen diversas soluciones para detección de objetos en imágenes a tiempo real, como pueden ser de ejemplo YOLO basado en Darknet, o efficientDet basado en TensorFlow.

Un sensor capaz de proporcionar imagen, posición e identificación de un elemento del entorno a tiempo real abriría un sinfín de nuevas aplicaciones en las que integrar el sensor

de visión estereoscópica. Es esta sin duda la línea de trabajo más llamativa y en la que me gustaría continuar avanzando a la finalización de este trabajo.

(Wes Archer)
(Phil King)
(Russell Barnes)

REFERENCIAS BIBLIOGRÁFICAS

Atareado. (s.f.). *Atareado*. Obtenido de Atareado: https://atareao.es/tutorial/trabajando-con-systemd/como-crear-un-servicio-con-systemd/

Blanes, N. J. (2022). *Apuntes UT04 de Redes y sistemas distribuidos. MAII - UPV.* Valencia.

Castedo hernandez, C. (Mayo de 2017). Estimacion de la posicion 3d de objetos deformables mediante marcas en color. *Estimacion de la posicion 3d de objetos deformables mediante marcas en color*. Universidad de valladolidad.

Cristian Daniel Ortega, F. E. (2013). *Tecnicas de implementacion de visión estereoscópica en robótica.*

Ingeniería al día, R. I. (Número 10). Visión por computador, su historia y algunos principales hitos. *Revista Ingeniería al día*.

Escalera, A. D. (s.f.). *Visión por computador Fundamentos y Métodos.* Pearson.

Gonzalez., J. I. (2003). *Estudio experimental de metodos de calibracion y autocalibracion de cámaras.*

Hirschmuller, H. (2008). *Stereo processing by semiglobal matching and mutual information. Patter analysis and machine intelligence.*

Hirschmuller, H. (s.f.). *Accurate and efficient Stereo Processing by Semi-Global Matching and mutual information.*

Phil King, D. A. (s.f.). *the camera module guida.*

Picamera. (s.f.). *Picamera web oficial*. Obtenido de https://picamera.readthedocs.io/en/release-1.13/

Pomazov, E. (2019). *OpenCV and Depth Map on StereoPi tutorial*. Obtenido de OpenCV and Depth Map on StereoPi tutorial: https://stereopi.com/blog/opencv-and-depth-map-stereopi-tutorial

J. Battle, E. M. (1998). *Recent progress in coded structured light as a technique to solve the correspondence problem.*

J. M. López-Vallés, A. F.-C. (2006). *Conceptos y técnicas de estereovisión por computador. Inteligencia artificial.*

Jesús Arturo Escobedo-Cabello, V. P.-H. (s.f.). *Aplicación de un algoritmo de cálculo de disparidad para la estimación de profundidades usando cámaras estéreo.*

Juan Musuña Toapanta, B. Z. (s.f.). Diseño y construcción de un robot móvil que permita la obtención de una nube de puntos del escaneo de habitaciones utilizando láser y webcams.

Martín, J. A. (s.f.). Compresión de vídeo. *Compresión de vídeo*. Universidad de Cantabria.

Russell Barnes, D. A. (s.f.). *Dive in python.*

Stereovisión. (s.f.). *Stereovision*. Obtenido de https://github.com/erget/StereoVision

Wes Archer, D. C. (s.f.). Raspberry Pi projects book volume 5.

Wikipedia. (s.f.). *Block Matching*. Obtenido de Block Matching: https://es.wikipedia.org/wiki/Block_matching

python. (s.f.). *Documentación Pyhton*. Obtenido de Documentación Pyhton: https://docs.python.org/es/3/library/tk.html

https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a. (s.f.). Obtenido de https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a

Opencv. (s.f.). *Funciones opencv*. Obtenido de https://docs.opencv.org/3.4/d9/dba/classcv_1_1StereoBM.html

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://docs.opencv.org/4.x/dd/d92/tutorial_corner_subpixels.html

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga6a10b0bb120c4907e5eabbcd22319022

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga9d2539c1ebcda647487a616bdf0fc716

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga617b1685d4059c6040827800e72ad2b6

Opencv. (s.f.). *Funciones Opencv*. Obtenido de https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga7dfb72c9cf9780a347fbe3d1c47e5d5a

OpenCV. (s.f.). *Funciones openCV*. Obtenido de https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a