Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

ESTIMACIÓN DE LA POSICIÓN 3D DE OBJETOS DEFORMABLES MEDIANTE MARCAS EN COLOR

Autor:
Castedo Hernández, Carlos

Tutor:
Félix Miguel Trespaderne
Eusebio de la Fuente Lopez
Departamento de Ingeniería de Sistemas y Automática

Valladolid, Mayo de 2017

Resumen

El objetivo principal del Trabajo de Fin de Grado es la realización de una aplicación de visión artificial destinada a la cirugía laparoscópica. El sistema desarrollado deberá de identificar la mano del cirujano, empleando la información de color captada por las cámaras, y asimismo extraer la información tridimensional de la escena. Esta información será posteriormente enviada a un robot de asistencia quirúrgica para su guiado automático.

Palabras clave

Visión Artificial - C++ - OpenCV - Laparoscopia

Índice general

Índice de figuras

Capítulo 1

Introducción

Con este trabajo de fin de grado se pretende integrar un sistema de visión artificial en un entorno robotizado que asista al cirujano en operaciones de cirugía lapparoscópica asistida por la mano. A continuación, se hará una pequeña introducción a cada uno de los campos comprendidos en este trabajo: la cirugía laparoscópica, la visión artificial y OpenCV.

1.1. Cirugía laparoscópica

La cirugía laparoscópica está considerada como uno de los grandes avances en el campo de la cirugía en el siglo XX, ya que permite la realización de las mismas intervenciones que en la cirugía convencional, pero reduciendo el tamaño de las incisiones.

La cirugía laparoscópica es un tipo de cirugía mínimamente invasiva. La cirugía laparoscópica se basa en la realización de varias incisiones en el paciente de un tamaño que oscila entre los 0.5 cm y 1 cm. Cada una de estas incisiones se conoce como puerto y el instrumento que se insertan en los puertos se conoce como trocar. A través de uno de estos puertos se introduce una cámara conocida con el nombre de laparoscopio. Al iniciar la cirugía se infla el abdomen del paciente haciendo uso del gas, dióxido de carbono, proporcionando al cirujano un espacio de trabajo y visibilidad. El cirujano puede observar el interior de la cavidad abdominal del paciente mediante las imágenes transmitidas por el laparoscopio a unos monitores. Estas operaciones necesitan de dos cirujanos uno encargado de realizar la cirugía como tal y otro encargado de mover el laparoscopio por el interior de la cavidad abdominal del paciente.

Como ya se ha citado anteriormente, este trabajo se va a centrar en la cirugía laparoscópica asistida con la mano. En estos casos se utiliza otro tipo de puerto lo suficientemente amplio para que el cirujano pueda insertar la mano, siendo esta incisión más grande que el resto de incisiones de laparoscopia.

Algunas de las ventajas de la cirugía laparoscópica respecto a la cirugía tradicional son:

- Menor dolor postoperatorio.
- Recuperación más rápida, lo que conlleva una estancia en el hospital más corta.

- Retorno más rápido a su actividad habitual, laboral y física.
- Mejor resultado estético.
- Debido a la delicada y reducida manipulación de los tejidos el edema de estos, tras la cirugía, y las pérdidas de sangre son menores.
- Menor incidencia de complicaciones de las heridas.
- Alteración menor de los mecanismos de defensa del paciente.

Las desventajas de este tipo de cirugías son:

- Elevado coste de la cirugía, debido a la tecnología empleada.
- Necesidad de personal especializado, en cada operación se necesitan dos cirujanos altamente cualificados. Uno de ellos será el encargado de realizar la cirugía como tal y el otro se encargará del posicionamiento del laparoscopio.
- En determinadas enfermedades puede haber desventajas más técnicas.

1.2. Visión artificial

La visión artificial, también conocida con el nombre de visión por computador, es una disciplina que estudia los métodos para la adquisición, procesamiento, análisis y comprensión de las imágenes para la obtención de información tanto numérica como simbólica para su posterior tratamiento por un ordenador.

La visión artificial pretende imitar a la visión humana. Este objetivo es muy difícil de conseguir ya que la capacidad de comprensión de imágenes del ser humano es muy sofisticada. Un ejemplo de esto es la cantidad de suposiciones que realiza el ser humano a la hora de interpretar imágenes. El sistema de visión del ser humano no necesita ver un objeto al completo para identificarlo, esto es muy difícil de implementar mediante la visión artificial. Algunos de los usos dados a la visión artificial son:

- Aprendizaje automático, utilizado para la clasificación y segmentación de imágenes.
- Detección de objetos, se basa en detectar la presencia de objetos en una imagen.
- Análisis de video, utilizado en el sector de video vigilancia y seguridad. Alertando ante posibles movimientos.
- Visión 3D, permite emular la visión humana mediante un ordenador. Generando modelos tridimensionales a partir de imágenes bidimensionales.

1.3. OpenCV

OpenCV es una biblioteca de funciones de código abierto para C y C++ que emplearemos en el desarrollo de nuestro sistema de visión estereoscópico, originalmente desarrollada por Intel. El uso dado a esta biblioteca va desde la detección de movimiento en sistemas

de vigilancia hasta el reconocimiento de objetos en aplicaciones de control de procesos.

El objetivo principal de OpenCV es la creación de un entorno de desarrollo fácil de utilizar y altamente eficiente.

OpenCV es multiplataforma, presentando diferentes versiones dependiendo del sistema operativo, los sistemas operativos que tienen sus versiones correspondientes son GNU/LINUX, Windows y Mac OS X.

1.4. Justificación

Uno de los motivos que me hizo decantarme por este trabajo de fin de grado es el hecho de que sea un trabajo de investigación. Siendo un trabajo con vías a un posible desarrollo posterior. Considero que todos los trabajos de fin de grado tendrían que poder ser implementados en el futuro y no solamente quedarse en una idea desarrollada, como ocurre con una gran parte.

Otro de los motivos principales fue el mismo que me hizo cursar este grado, el interés por la robótica. Dentro del campo de la robótica mis intereses siempre se han centrado en la inteligencia artificial. La visión artificial es uno de los campos más importantes y comunes a la hora de desarrollar robots con inteligencia artificial.

Considero que los sistemas de visión artificial inteligentes es uno de los campos con mayor proyección de futuro debido a su mínimo desarrollo, y a su complejidad en comparación con otros campos de la inteligencia artificial.

Debido a la mejora supuesta por el uso de las técnicas de cirugía laparoscópica, se ha visto necesario el apoyo de estas. Este trabajo de fin de grado va a buscar servir de ayuda a este tipo de cirugías permitiendo una facilidad en su realización y una reducción de costes. En la actualidad las cirugías laparocópicas asistidas mediante mano requieren de dos o más cirujanos, provocando un encarecimiento de la cirugía en si, haciendo que sean inaccesibles económicamente para una gran parte de la población. La idea de este trabajo de fin de grado consiste en conseguir sustituir al cirujano encargado de iluminar la cavidad abdominal del paciente por un robot. Esto abarataría los costes y haría más accesibles este tipo de cirugías al público.

1.5. Objetivo

El objetivo principal del Trabajo de Fin de Grado es la realización de un software capaz de identificar la mano del cirujano, introducida en las cirugías laparoscópicas asistidas con la mano e identificar la posición en el espacio de esta respecto a un eje de coordenadas situado en el centro de la lente de una de las cámaras utilizadas. Los datos obtenidos se enviarán a un robot, el cual es el encargado de iluminar la mano del cirujano dentro del abdomen del paciente. Por lo que el robot tiene que realizar los movimientos oportunos para conseguir que la mano del cirujano este siempre iluminada.

Durante el desarrollo del trabajo se han ido fijando una serie de objetivos intermedios. Además, se han añadido una serie de objetivos complementarios que han facilitado el cumplimiento del objetivo final. Esta serie de objetivos complementarios han ido añadiéndose

según mis conocimientos de la librería de OpenCV han ido aumentando.

La serie de objetivos que se han planteado durante la realización del trabajo han sido los siguientes:

- Realización de un estudio acerca de los posibles métodos para la obtención de mapeados 3D.
- Toma de imágenes de dos cámaras de forma simultánea, y almacenamiento de estas.
- Extracción de las matrices de calibración de las cámaras y el almacenado de estas en un archivo. Permitiendo la lectura de las matrices para su posterior utilización.
- Rectificar, remapear y corregir la distorsión de las imágenes obtenidas por las cámaras mediante el uso de las matrices de calibración, permitiendo corregir los efectos provocados por la curvatura de la lente.
- Obtención de los valores correspondientes al color a discriminar y el rango de tolerancia utilizado, almacenando estos valores en un archivo para su posterior utilización.
- Adquisición del valor de los parámetros correspondientes al emparejamiento de imágenes. El emparejamiento se lleva a cabo mediante la técnica de block matching y esta requiere de una serie de parámetros. El valor de los parámetros a de almacenarse en un fichero para su posterior utilización.
- Obtención de las coordenadas de los elementos en la escena a partir de los valores anteriormente obtenidos, mediante un procedimiento de conversión de 2D a 3D.

Para lograr alcanzar los objetivos marcados se han utilizado la documentación existente en la red acerca de la librería OpenCV y los libros citados en la bibliografía.

A lo largo del trabajo se explicará el proceso realizado para lograr los objetivos fijados, el cual incluye un estudio de las posibles opciones para alcanzar el objetivo junto con la justificación de la opción elegida.

Capítulo 2

Imagen digital

En este capítulo se pretende llevar a cabo una introducción de los conceptos básicos y necesarios que se han de conocer sobre las imágenes, formación y procesamiento, para poder seguir el desarrollo de este trabajo de fin de grado.

Una imagen es la representación óptica de uno o más objetos iluminados por una o más fuentes de radiación. Así, en general, los elementos que forman parte del proceso de formación de una imagen son los siguientes: objeto u objetos, fuente o fuentes de iluminación y sistema de formación de la imagen [33].

2.1. Historia

Desde el comienzo de los tiempos ha existido un interés en la representación del mundo tridimensional mediante imágenes planas, bidimensionales, como las obtenidas mediante cámaras. En un principio este interés se puede observar en la pintura y posteriormente en la fotografía.

Durante la época de la Grecia clásica, se descubren muchas propiedades geométricas de los objetos, sobre todo proyectivas, permitiendo un gran avance en la representación de estos en planos bidimensionales. Thales de Mileto (623 a.C – 546 a.C), conocedor de estas propiedades consiguió grandes logros como la predicción de un eclipse solar y la medición de una pirámide a partir de la sobra que proyectaba.

Otra gran figura fue Euclides (325 a.C – 265 a.C), matemático y geómetra griego, conocido como "El Padre de la Geometría". Desarrollando el concepto de geometría plana, concibiendo esta como un conjunto de líneas y puntos, independientes de un sistema de coordenadas.

Figura 2.1: La escuela de Atenas, Rafael Sanzio, 1512 d.C.

En el Renacimiento tienen lugar avances en el campo de la geometría. Los pintores italianos llevan a cabo un estudio acerca de la formación de las imágenes y la geometría de los objetos con el fin de poder alcanzar un mayor grado de realismo en sus obras. Gracias a esto consiguen plasmar los efectos de la perspectiva, mediante las diferencias de profundidad en los objetos,

como se puede observar en la figura 2.1. Fue Filippo Bruneleschi (1377 d.C – 1446 d.C) el inventor de la perspectiva.

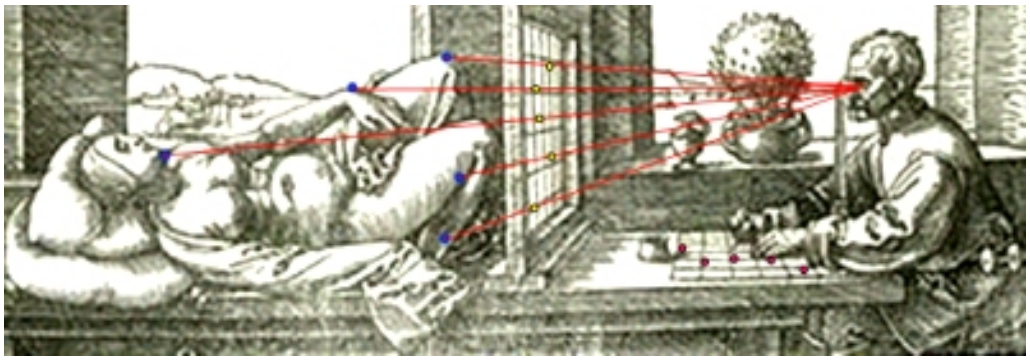


Figura 2.2: Máquina de perspectiva.

A partir de esta época se empieza a utilizar el punto de fuga, en el cual convergen las rectas paralelas que se alejan del observador. Hasta entonces, en las pinturas prerenacentistas, se utiliza la perspectiva para dar importancia a los objetos o personajes llevando a un plano más cercano al observador lo importante y alejando lo menos importante.

En el siglo XVI, se desarrolla la teoría de la perspectiva. Promoviendo la invención de las Máquinas de Perspectiva, las cuales facilitaban la reproducción de la perspectiva sin llevar a cabo cálculos matemáticos. La máquina de la perspectiva se basaba en un marco de forma rectangular, probablemente su formato se correspondería a la proporción áurea, el cual estaba dividido en cuadrados iguales mediante hilos tensados. Esta se situaba entre el pintor y el ente a dibujar, dividiendo la escena, como se puede observar en la figura 2.2. Esta misma división la hacía el pintor sobre el lienzo, permitiéndole llevar a cabo una correspondencia entre la escena y lo dibujado mucho más realista.

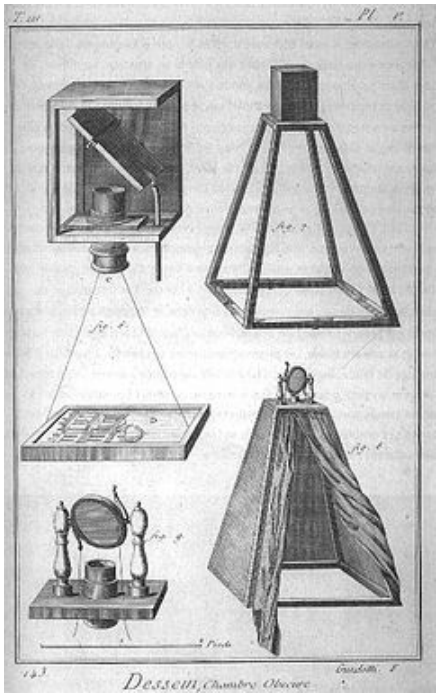


Figura 2.3: Esquema de los componentes de una cámara oscura.

Las máquinas de perspectiva pueden ser consideradas como el primer intento de una cámara fotográfica. Utilizando un plano donde se forma la imagen, la rejilla, y un punto como centro óptico, el ojo del dibujante, no perteneciente al plano anterior, y dónde se intersectan los rayos que forman la imagen.

En el año 1604, el nombre de cámara oscura es acuñado por Johannes Kepler en el tratado Ad Vitellionem Paralipomena. En este se explica el funcionamiento de la cámara

oscura. La cámara oscura permite obtener la proyección plana de una imagen externa sobre la zona interior de su superficie. Consiste en una caja cerrada y un pequeño agujero por el que entraba una mínima cantidad de luz que proyectaba en la pared opuesta la imagen del exterior. En la figura 2.3 se puede observar un esquema de la cámara oscura.

Originalmente consistía en una sala cerrada cuya única fuente de luz era un pequeño orificio practicado en uno de los muros, por donde entran los rayos luminosos reflejando los objetos del exterior en una de sus paredes. El orificio funciona como una lente convergente y proyecta, en la pared opuesta, la imagen del exterior invertida tanto vertical como horizontalmente.

Uno de los usos fue como ayuda para los pintores. La imagen se proyectaba sobre un papel y servía de pauta para dibujar sobre ella. Uno de los pintores que uso la cámara oscura fue Vermeer (1632 d.C – 1675 d.C).

El químico Niepce (1765 d.C – 1833 d.C) toma la primera fotografía en 1826 d.C. Esto lo consiguió utilizando la cámara oscura y una superficie fotosensible. El método utilizado por Niepce fue perfeccionado y en 1833, el químico Daguerre (1787 d.C – 1851 d.C) lleva a cabo el primer proceso fotográfico práctico. En este se utilizaba una placa fotográfica revelada mediante vapor de mercurio y fijada con trisulfato de sodio.


Figura 2.4: Primera imagen digital.

En 1955, Russel Kirsch crea la primera imagen digital. Para ello emplea un dispositivo que transformaba las imágenes en matrices de ceros y unos. En la figura 2.4, se puede observar el resultado de utilizar el dispositivo creado por Russel Kirsch., una imagen escaneada del hijo de Russel Kirsch. Se puede observar la baja resolución de la imagen en las líneas que aparecen en la propia imagen, debidas a la baja capacidad de almacenamiento que tenían los ordenadores en aquellos años.

En 1975, aparece la primera cámara digital de la historia, permitiendo la obtención de imágenes digitales de forma directa.

2.2. Tipos de imágenes digitales

Las imágenes digitales se pueden dividir en dos tipos: vectorial y bitmap. Cada uno de estos tipos presenta unas características determinadas que hacen que tengan aplicaciones específicas, además de necesitar programas diferentes para la edición de estas.

2.2.1. Imagen vectorial

Es un tipo de imagen compuesta por objetos geométricos simples e independientes, por ejemplo: segmentos, polígonos, arcos, etc. Cada uno de estos objetos está definido por una serie de parámetros matemáticos como posición, forma, color, grosor. Cualquier imagen por compleja que parezca siempre se puede reducir a un conjunto de entidades geométri-

cas simples. La figura 2.5 es un ejemplo de imagen vectorial. La principal ventaja de las imágenes vectoriales es la posibilidad de modificar la escala, permitiendo la ampliación o reducción de la imagen sin perder la calidad de esta.

Las principales aplicaciones son:

- Generación de gráficos, como creación de logos mediante programas de diseño asistido por ordenador, computer Aided Desing (CAD).
- Lenguajes de descripción de documentos, permitiendo representar un documento correctamente independientemente del tamaño de la resolución del dispositivo de salida.
- Tipografías.
- Videojuegos, empleados en aquellos que se desarrollan en 3D.
- Internet.

Figura 2.5: Imagen digital vectorial.

2.2.2. Imagen Bitmap

Esta imagen consiste en una matriz en la que cada posición es conocida con el nombre de pixel. Siendo este la superficie homogénea más diminuta que forma parte de una imagen, conteniendo un color uniforme. La formación de estas imágenes se realiza en la retina del observador, la cual integra los cambios de color y luminosidad entre pixeles próximos. La figura 2.6 se corresponde con una imagen bitmap.

Las imágenes bitmap se definen mediante sus dimensiones, altura y anchura expresadas en pixeles, y la profundidad del color. La calidad de una imagen es directamente proporcional a la densidad de pixeles, la resolución de la imagen.

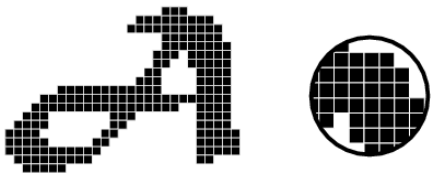


Figura 2.6: Imagen digital bitmap.

En este tipo de imágenes no ofrecen buenos resultados cuando se cambia su escala. Sin embargo, permiten una reproducción de objetos sutilmente iluminados y escenas con grandes variedades tonales.

Las imágenes bitmap es el tipo de imagen más utilizado en el ámbito de la visión artificial, por lo que se llevara a cabo un estudio de este tipo de imágenes de forma exhaustiva. Desde este momento, siempre que se haga referencia a una imagen digital será una imagen bitmap.

Formatos de almacenamiento

El almacenamiento de una imagen no sólo consiste en guardar la información correspondiente a cada pixel, también se guarda una cabecera la cual contiene la información destinada al programa que la abre y la muestra.

En los primeros años del procesamiento de imágenes los formatos de almacenamiento de estas no estaban estandarizados. De forma que las aplicaciones de procesamiento de imágenes almacenaban las imágenes en formatos propios de cada aplicación. Las incompatibilidades entre formatos y por lo tanto entre aplicaciones eran un impedimento para la mejora. La solución fue la creación de una serie de formatos de almacenamiento estandarizados para el almacenamiento de las imágenes digitales.

El tamaño de las imágenes suele ser bastante elevado, por lo tanto, los formatos de almacenamiento de imágenes realizan una compresión de la imagen guardada. Esta compresión se puede realizar de dos formas:

- Compresión sin pérdida de información (LOSSLESS).
- Compresión con pérdida de información (LOSSY).

Algunos formatos de almacenamiento han sido desarrollados por empresas y es necesario pagar para poder utilizarlos. Existen otros que son de dominio público y pueden utilizarse libremente. Los más importantes son los siguientes:

- JPEG. Es un formato de compresión con pérdidas, pero es el mejor para imágenes de tonos continuos similares que contienen muchos colores. Presenta una gran capacidad de compresión manteniendo la calidad de la imagen. Hoy en día es el formato más utilizado.

  Se basa en la eliminación de la información no apreciable, basándose en el hecho de que el ojo humano distingue mejor las variaciones de luminosidad que los cambios de color. Por lo que este algoritmo separa la información de la imagen en dos partes: color y luminosidad y las comprime por separado.

  Permite almacenar imágenes de 16 millones de colores y 256 variaciones de gris, lo que equivale a una profundidad de color de 24 bits y 8 bits respectivamente. Además, permite la carga progresiva en un navegador, convirtiéndolo en un formato estándar en la web.

- Tagged Image File Format (TIFF). Este formato es ampliamente utilizado debido a su gran flexibilidad. Proporciona un conjunto de métodos de compresión y de espacios de colores, por lo que cabe la posibilidad de almacenar en un mismo archivo TIFF una misma imagen con diferentes propiedades. Debido a esto los archivos con este formato pueden ser muy grandes. Este formato realiza una compresión sin pérdida.

- Graphics Interchange Format (GIF). Este formato crea una equivalencia de colores, convirtiendo una imagen de 16 millones de colores a una imagen de 256. Reduciendo la profundidad del color de 24 bits a 8 consiguiendo una reducción en el tamaño de la imagen considerable. En el caso de que la imagen tenga una profundidad de color menor que 8 bits, en la compresión no se pierde información. En los casos donde la imagen a comprimir tiene muchos colores, aplica un algoritmo para aproximar los colores de la imagen a una paleta de 256 colores. La compresión de este formato se realiza en dos pasos; el primero consiste en reducir el número de colores de la imagen a 256 y en el segundo se remplazan áreas de color uniforme usando código de secuencias. Debido al pequeño tamaño de los archivos de este tipo fue rápidamente extendido en los primeros años de Internet. Los dos grandes inconvenientes de este formato son la posible pérdida de colores, la cual puede llegar a ser de un 99.998 % de estos y el hecho de ser un formato propietario.

- Portable Network Graphics (PNG). Este formato combina lo mejor de los formatos JPG y GIF. Se trata de un formato de almacenamiento sin pérdida, con una profundidad de color de 24 bits. La compresión es totalmente reversible por lo que se puede recuperar la imagen original. Los inconvenientes principales son: el mayor tamaño de los archivos en comparación con el JPG y la incapacidad de soportar animaciones.
- Windows Bitmap (BMP). Es un formato sin pérdidas con una profundidad de color máxima de 24 bits. Su tasa de compresión es bastante mala pero debido a ser el formato nativo de Microsoft suele ser frecuente encontrar archivos con este formato.

2.3. Formación

El concepto de imagen digital está asociado a una función bidimensional f(x, y), cuya amplitud o valor será el grado de iluminación (intensidad de la luz) en el espacio de coordenadas (x, y) de la imagen para cada punto. El valor de esta función depende de la cantidad de luz que incide sobre la escena, así como de la parte que sea reflejada por los objetos que componen dicha escena [4].

La función bidimensional, *f(x, y)*, se determina mediante la ecuación 2.1

$$f(x,y) = i(x,y) \cdot r(x,y) \tag{2.1}$$

Donde i representa la iluminación y r la reflexión, del punto con las coordenadas x e y en la escena. Los valores correspondientes a la iluminación se encuentran en el intervalo (0, infinito) mientras que los valores de la reflexión se encuentran acotados en el intervalo (0,1). Debido a esto, los valores tomados por la función f están acotados entre cero e infinito, estos valores se corresponden con la cantidad de luz reflejada.

2.4. Captura

La adquisición de imágenes digitales necesita tres componentes, un sistema óptico, un captor y un digitalizador, además de una entrada que es la función *f*, o lo que es lo mismo, la cantidad de luz reflejada. Estos tres componentes forman el sistema de formación de imágenes que se encuentra implementado en cualquier cámara fotográfica o de vídeo.

La función del sistema óptico es la captación de los rayos luminosos y su concentración sobre el sensor sensible de la cámara. El sistema óptico está formado por una o más lentes convexas. Los principales parámetros de un sistema óptico son:

- Punto focal, es el punto donde los rayos luminosos convergen después de pasar a través de la lente convexa.
- Distancia focal (f), es la distancia entre la lente y el punto focal. Este parámetro será de gran importancia para la determinación de la posición y el tamaño de objetos en la escena.
- El factor de potencia (D), es la inversa de la distancia focal.
- El número F relaciona la distancia focal con el diámetro del diafragma, indicando la cantidad de luz que puede pasar a través de una lente. Se obtiene mediante la división de la distancia focal entre el factor de potencia.

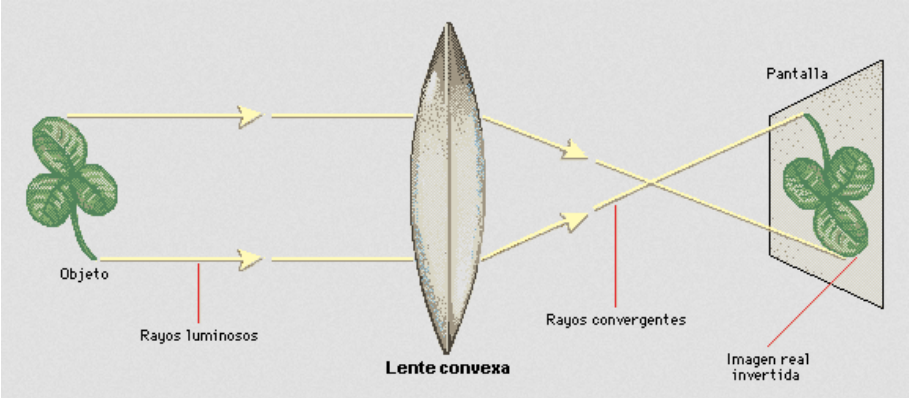


Figura 2.7: Formación de una imagen mediante una lente convergente.

El captor fotoeléctrico es el dispositivo encargado de convertir la luz en una señal eléctrica. El captor más utilizado y en el que nos centraremos será el dispositivo de cargas acopladas (Charge Coupe Device, CCD).

Los CCDs son circuitos integrados formados por elementos semiconductores llamados photosites. Estos se encuentran dispuestos en una matriz, y cada photosites se corresponde con un pixel de la imagen. Cada elemento semiconductor es excitado por una serie de fotones procedentes de la escena, la cantidad de carga almacenada en cada photosites depende del grado de excitación de este y se corresponde con la reflectancia en ese punto. Sus principales ventajas frente a sus predecesores son: su reducido tamaño y su bajo consumo de potencia.

Figura 2.8: Sensor CCD.

Las funciones realizadas por los CCDs son tres:

- Registro de la imagen, se realiza la conversión fotoeléctrica.
- Registro temporal, se almacena momentáneamente la carga de las photosites.
- El drenaje, se efectúa la salida de la carga.

La salida de captador es una señal analógica bidimensional, *s(x, y)*. Para la conversión de la señal analógica a digital se hace uso de un convertidor A/D, también conocido con el nombre de digitalizador.

Dependiendo del tipo de imagen que se quiera obtener se implementaran diferentes configuraciones en cuanto al captor. Si se quiere obtener una imagen en blanco y negro tan solo se necesita un CCD, al cual no hace falta realizar ningún tipo de modificación. En el caso de querer obtener imágenes en color existen dos posibilidades: utilizar tres CCD físicamente separadas, añadiéndoles un filtro de color a cada una obteniendo el color rojo, verde y azul. Juntando la información de los tres CCDs se obtendría una imagen en color. La segunda opción es utilizar un CCD modificado donde cada pixel de la imagen este representado por tres photosites, y en cada uno de ellos se filtrase un color. El resultado

de esta segunda técnica en comparación con la primera es mucho peor ya que en este caso cada photosite representa un punto de la escena y no el mismo.

Para crear una imagen digital es necesario convertir esa señal continua a forma digital, lo cual involucra los procesos de cuantificación y muestreo. Como una señal analógica es continua tanto con respecto a las coordenadas x e y como en amplitud, para convertirla a forma digital es necesario realizar una discretización en ambas coordenadas y también de su amplitud. La digitalización de las coordenadas se denomina muestreo de la imagen, mientras que la de la amplitud se conoce como cuantificación de los niveles grises.[33]

2.5. Representación

Las imágenes digitales como se ha dicho anteriormente están representadas por una función bidimensional, en el caso de las imágenes en blanco y negro. Las imágenes digitales en color están representadas mediante tres funciones bidimensionales a partir de las cuales se forma el color, posteriormente se explicará el concepto de imágenes en color y su formación. Una función bidimensional no es más que una matriz, por lo que todas las imágenes digitales se pueden representar mediante matrices. Los valores de los diferentes elementos de la matriz, denominados pixeles, son determinados mediante la cuantización de intensidad.

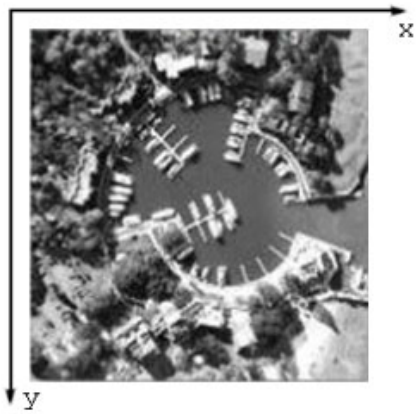


Figura 2.9: Convención de ejes en la representación de una imagen digital.

En las imágenes en blanco y negro solamente se almacena un valor por pixel, correspondiente con el nivel de intensidad. Para la representación de este valor se emplean una serie de bits que determinan el rango de valores que puede tomar el pixel. El número de bits más común es 8, por lo que el rango de valores varía entre 0 y 255. Mediante 8 bits el mayor número en binario es el 1111111 que se corresponde con el 255 en decimal y el mínimo es el 00000000 que se corresponde con el 0 en decimal. Donde el 0 representa el negro y el 255 representa el blanco.

Las imágenes en color necesitan de tres valores para la formación del color y cada uno de estos valores representa un componente básico del color. Uno de los modelos de color más utilizados es el RGB (Red Green Yellow), las imágenes formadas mediante RGB tienen tres matrices. Cada una de las matrices se corresponde a un color: rojo, verde y amarillo. La representación de los valores se hace de la misma manera que en las imágenes en blanco y negro. En este caso cada pixel tiene tres valores, la nomenclatura de los valores de un pixel es: (R, G, B), por ejemplo, un pixel rojo tendrá los siguientes valores: (255, 0, 0). Posteriormente se llevará a cabo una explicación más extensa del color y los modelos de color.

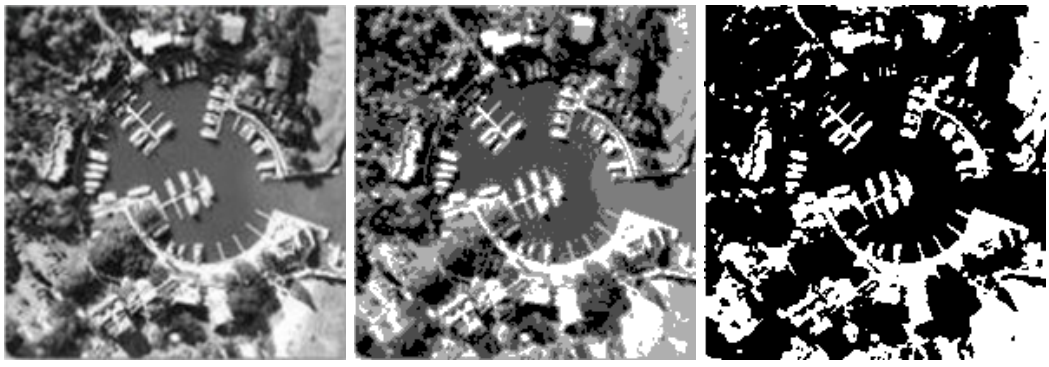
2.6. Resolución espacial y en amplitud

La resolución espacial está determinada por la cantidad de pixeles contenidos en la imagen digital, es decir, cuanto mayor sea el número de filas o columnas de la matriz mayor será la resolución espacial. La calidad de una imagen es directamente proporcional a la resolución espacial, cuanto menor sea la resolución espacial menor será la calidad de la imagen. Esto se puede observar en la figura 2.10.

(a) 512x512 (b) 64x64 (c) 16x16

Figura 2.10: Variación de la resolución espacial en una misma imagen.

La resolución en amplitud se corresponde con el rango de valores alcanzable por un pixel. El valor de un pixel es obtenido en la cuantificación, explicada anteriormente. Normalmente las imágenes en blanco y negro tienen una resolución en amplitud de 256 valores, es decir, 256 niveles de gris, correspondientes con los 8 bits asignados a cada pixel. En el caso de que solo se asignase un bit a cada pixel la resolución en amplitud se vería disminuida a dos valores, el cero y el uno. El cero se correspondería a con el negro mientras que el uno lo haría con el blanco. Un ejemplo de diferentes resoluciones en amplitud se puede observar en la figura 2.11.



(a) 256 niveles de grises (b) 5 niveles de grises (c) 2 niveles de grises

Figura 2.11: Variación de la resolución en amplitud en una misma imagen.

2.7. Segmentación

La segmentación es el proceso por el cual se extrae de la imagen cierta información subyacente para su posterior uso. La segmentación está basada en dos principios fundamentales discontinuidad y similitud. [36]

La segmentación basada en discontinuidades, orientada a bordes, se lleva a cabo mediante un análisis de la imagen digital en busca de zonas de pixeles donde las propiedades, como iluminación o color, varían. Por ejemplo, en el contorno de los objetos suele haber variaciones en las propiedades por el hecho de que los pixeles de la zona pertenecen a objetos distintos. Esto permite identificar el contorno de los objetos presentes en la escena.

Los operadores utilizados para llevar a cabo una segmentación orientada a bordes son:

- Primera derivada.
- Segunda derivada.
- Morfológicos.

La segmentación basada en continuidades, orientada a áreas, busca la identificación de vecindades de pixeles con las mismas propiedades de forma que se correspondan con un mismo objeto de la escena. Los operadores utilizados para llevar a cabo una segmentación orientada a áreas son:

- Binarización mediante el uso de umbrales.
- División de regiones.
- Similitud de las propiedades como textura, color, etc.

En ambas segmentaciones se ha de manipular la imagen llevando a cabo una transformación de esta, mediante la variación de los valores de los pixeles.

En algunos casos la calidad de las imágenes digitales tomadas no es lo suficientemente buena obteniendo unos resultados erróneos en el proceso de segmentación. Para llevar a cabo una mejora en la calidad de las imágenes se realizan ciertos procesos, consiguiendo la eliminación de ruidos en la imagen.

2.8. Iluminación y reflexión

Debido a la importancia de la iluminación y reflexión en la formación de imágenes se ha considerado necesario explicar brevemente algunos conceptos básicos.

La iluminación natural de las escenas no suele ser la más adecuada, esto conlleva a la implementación de sistemas de iluminación propios. Para el desarrollo de estos se han de conocer las diferentes fuentes de luz y como pueden ser iluminados en función del tipo de superficie.

2.8.1. Propiedades materiales

Los materiales tienen diferentes propiedades reflexivas, absorbente y transmitivas dependiendo del comportamiento cuando incide un haz de luz sobre estos. Estas propiedades están basadas en los tres comportamientos principales de un material cuando incide sobre el un haz luminoso: que todo él se refleje, se absorba o se transmita a través de él.

Dependiendo de cómo se refleja el haz de luz sobre la superficie del material, se clasifican en los siguientes tipos:

- Especulares, el haz luminoso es reflejado en su totalidad, siendo el ángulo de incidencia del haz de luz el mismo que el del rayo reflejado.
- Difusos, el haz luminoso es reflejado en todas las direcciones.
- Reflectores, el rayo reflejado tiene la misma dirección que el haz de luz, pero sentido contrario.
- Selectivos al espectro, dependiendo de la longitud de onda del haz de luz este será absorbido o reflejado de forma difusa o especular.
- No selectivos al espectro, todas las longitudes de onda del haz son reflejadas.

Los diferentes tipos de materiales según sus propiedades de absorción son: selectivos al espectro o no selectivos al espectro, pero en este caso se refieren a las longitudes de onda que son absorbidas y no reflejadas. Los materiales de color negro son no selectivos al espectro mientras que los materiales de color blanco son selectivos al espectro, no absorbiendo ninguna longitud de onda.

También se pueden clasificar los materiales dependiendo de si el haz de luz es capaz de pasar a través de ellos, obteniendo la siguiente clasificación de los materiales:

- Transparentes, permiten el paso de la luz sin absorberla o reflejarla.
- Translucidos, la luz es capaz de atravesar el material, pero de forma difusa.
- Selectivos al espectro, dejan pasar dependiendo de su longitud de onda.

2.8.2. Tipos iluminación

Iluminación direccional. En este caso la iluminación se realiza orientada al objeto. Algunas de las aplicaciones donde se utiliza este tipo de iluminación es localización y reconocimiento de objetos.

Iluminación difusa. El objetivo de esta es iluminar al objeto desde todas las direcciones evitando así la formación de sombras.

Iluminación a contraluz. Se realiza por detrás del objeto, de forma que el objeto se encuentra entre la cámara y la fuente de luz. Este tipo de iluminación se utiliza en aplicaciones para el cálculo del tamaño de objetos o el conteo de agujeros en piezas.

Iluminación oblicua. Se utiliza para resaltar las partes tridimensionales para su posterior análisis.

Iluminación estructurada. Se basa en la proyección de puntos sobre una superficie, permitiendo la detección de objetos y analizando la distorsión en el patrón de luz obtener características tridimensionales del objeto.

Iluminación coaxial. Permite obtener imágenes sin reflejos, los objetos son iluminados de forma que la luz incide sobre el objeto con la misma dirección que entra a la cámara.

Iluminación dark-field. Utilizada para resaltar los bordes de los objetos transparentes, esto se consigue colocando un material opaco del mismo tamaño que el objeto en la fuente de luz. De esta forma los rayos captados por el sistema de visión se corresponden con aquellos que han cambiado de dirección al pasar a través del objeto.

2.9. Color

Las imágenes a color se encuentran en muchos aspectos cotidianos de nuestra vida. La percepción del color ha sido de gran interés para científicos, psicólogos y artistas, durante años. Este capítulo tratara los aspectos técnicos necesarios para poder entender y trabajar con imágenes a color.

El uso del color en el procesamiento de datos no se le ha prestado mucha atención hasta hace unos años debido a la gran carga computacional y a la memoria necesaria que requieren las imágenes a color. El color es una de las características más importantes en un objeto, permitiendo llevar a cabo una identificación y extracción de objetos de una escena más fácilmente.

En el tratamiento de imágenes el procesamiento de las imágenes en color se divide en dos áreas fundamentales: color y pseudocolor. En la primera categoría se procesan las imágenes obtenidas con un sensor de color multiespectral, mientras que en la segunda las imágenes monocromas son coloreadas por asignación de un color a un determinado nivel de intensidad.[12]

2.9.1. Fundamentos del color

La luz está formada por una serie de ondas electromagnéticas con longitudes de onda específicas obteniendo la luz infrarroja, ultravioleta o el espectro visible. El espectro visible, véase figura 2.12, comprende el conjunto de longitudes de onda que pueden distinguir los seres humanos.

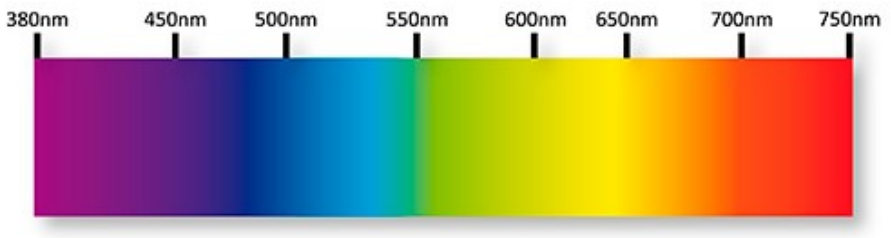


Figura 2.12: Espectro visible por el ojo humano

Cuando se ilumina un cuerpo con un haz de luz, este refleja parte de las ondas electromagnéticas absorbiendo el resto. Las ondas electromagnéticas reflejadas producen una

serie de señales nerviosas que son identificadas con un color determinado. Un cuerpo que refleje todas las longitudes de onda será blanco, mientras que un cuerpo que absorba todas las longitudes de ondas será negro. Para que un cuerpo se muestre con un color ha de ser capaz de absorber todas las ondas electromagnéticas menos aquella que represente dicho color. Por ejemplo, los objetos de color verde son aquellos que reflejan longitudes de onda de 500nm a 570nm y absorben el resto.

La síntesis de los colores del espectro visible se puede realizar de dos formas diferentes. La primera es mediante la síntesis sustractiva del color, la cual se basa en la formación del color mediante el filtrado de algunas longitudes de ondas contenidas en la luz blanca. La segunda es mediante la síntesis aditiva del color, la cual se basa en la obtención de color mediante la suma de otros colores conocidos como colores primarios, rojo, verde y azul. Las diferentes síntesis de color se explicarán en mayor profundidad a continuación.

El ser humano ve todos los colores como combinación de tres colores primarios: rojo(R), verde(G) y azul(B), es decir, el ser humano emplea la síntesis aditiva para la formación de los colores. Las longitudes de onda correspondientes a los colores primarios están estandarizadas. El color rojo tiene una longitud de onda de 700nm, el verde de 546.1nm y el azul de 435.8nm.

Las características principales de un color son: brillo, matiz y saturación. El brillo incorpora la noción cromática de intensidad. El matiz es un atributo asociado con la longitud de onda dominante en la mezcla de longitudes de onda de la luz, viene a ser el color percibido por el observador, cuando se dice de un objeto que es rojo, naranja o amarillo, realmente se está especificando el matiz, el grado de saturación es inversamente proporcional a la cantidad de luz blanca añadida. [12]

La cromaticidad es la combinación del matiz y la saturación, pudiendo identificar un color mediante su brillo y cromaticidad.

Una forma para la obtención de los colores del espectro visible es mediante la utilización del diagrama de cromaticidad, véase figura 2.13. Todos los puntos contenidos en la curva se corresponden con colores del espectro visible, estos colores están completamente saturados. Mientras que los puntos no contenidos en esta se corresponden con la mezcla de colores del espectro visible y cuanto más cercanos estén de W menor será el grado de saturación. El punto W se corresponde con el color blanco.

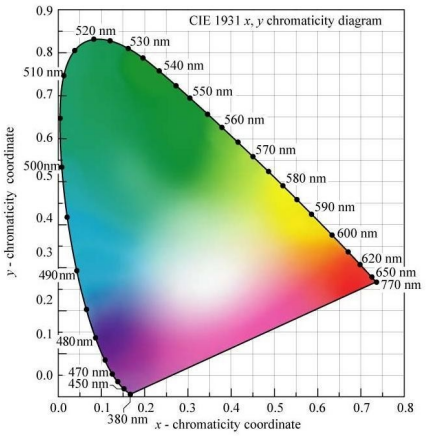


Figura 2.13: Diagrama de cromaticidad.

2.9.2. Síntesis del color

Durante siglos se ha experimentado con los colores, con el fin de obtener la mayor gama de colores posible a partir del mínimo número de colores, conocidos como colores primarios. Se propusieron varias teorías acerca de cuáles eran estos colores primarios a partir de los cuales se podría obtener cualquier otro.

Los colores primarios dependen de la fuente de color, pudiendo ser una fuente luminosa o un objeto que refleja luz. Teniendo en cuenta estos dos tipos de fuentes, los modelos difundidos para la síntesis del color son:

- Síntesis sustractiva.
- Síntesis aditiva.

Síntesis sustractiva

Permite explicar la creación de colores en pinturas, tintas, etc. Donde se crean colores que absorben algunas longitudes de onda y reflejan otras. Como se ha dicho anteriormente, el color de un objeto se corresponde con las longitudes de onda reflejadas por este.

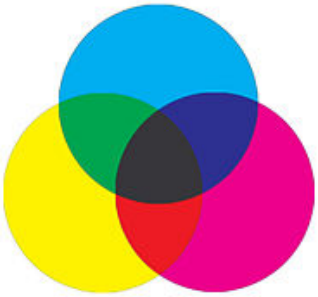


Figura 2.14: Mezcla sustractiva de colores primarios.

El color de partida es el aportado por la luz, en este caso, blanco. En la síntesis sustractiva los colores primarios son el amarillo, el magenta y el cian, absorbiendo cada una uno una serie de longitudes de onda, es decir, actuando como filtros. El color amarillo refleja las longitudes de onda correspondientes al color azul, el magenta refleja las del color verde y el cian las del rojo.

La mezcla de dos colores primarios crea un nuevo color que sirve como filtro para otras longitudes de onda. Dependiendo de las cantidades de cada color utilizadas en la mezcla se obtendrán diferentes colores. Por ejemplo, realizando una mezcla del color amarillo y magenta a partes iguales, el resultado será un color que impedirá el paso de la longitud de onda correspondiente al color rojo.

Los principales usos de la síntesis sustractiva son:

- Impresión a color.
- Fotografía a color.
- Artes plásticas.
- Pintura decorativa.

Algunos de los modelos de color basados en la síntesis sustractiva son: CMY (Cyan-Magenta-Yellow), CMYK (Cyan-Magenta-Yellow-Key) y RYB (Red-Yellow-Blue).

Síntesis aditiva

Basada en la obtención de un color de lux mediante la suma de otros. Para ello, se necesita la emisión de luz desde una fuente de alimentación de algún tipo. Newton descubrió que la suma de todos los colores del espectro visible daba como resultado la luz blanca. Basándose en este descubrimiento Thomas Young descubrió que solo necesitaban tres colores del espectro visible para obtener el resto. Además, la suma de estos tres colores era el blanco.

El proceso de creación de colores utiliza luz roja, verde y azul. De la combinación de dos colores primarios se obtiene un color secundario. Los colores secundarios son: cian, magenta y amarillo. La obtención del espectro completo se consigue variando la intensidad de las tres luces.

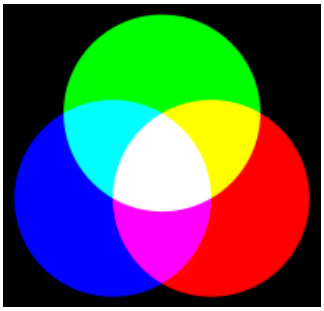


Figura 2.15: Mezcla aditiva de colores primarios.

Esta forma de síntesis se utiliza en aquellos métodos que reproducen y/o capturan imágenes que dependen de la emisión directa de luz. Algunos de sus usos son:

- Proceso de captura de imágenes de una cámara de fotografía en color.
- Proceso de captura de imágenes de una cámara de vídeo.
- Pantalla de televisor en sus diversos sistemas.
- Monitor de computadora,
- Pantallas de teléfonos móvil y otros aparatos electrónicos.
- Proyector de diapositivas.
- Proyector cinematográfico y de vídeo.

La síntesis de color aditiva es el modelo más utilizado en la actualidad debido a la gran cantidad de usos que tiene. Algunos de los modelos utilizados por la síntesis aditiva son RGB y HSL/HSV.

2.9.3. Modelos del color

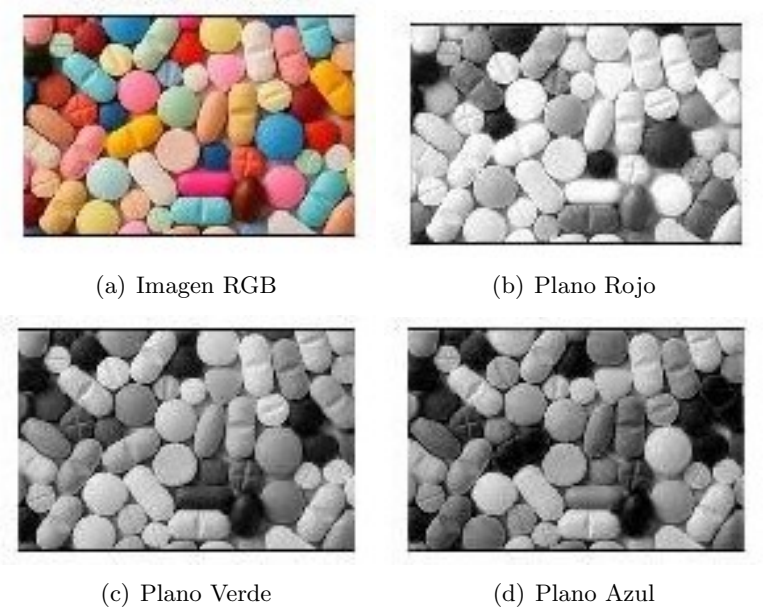
La representación de los colores se puede realizar de diferentes formas. Para la representación de un color se ha de especificar las características principales de este: brillo, matiz y saturación. Un modelo de colores es un modelo matemático mediante el cual se pueden representar todos los colores de forma numérica, asignado una posición específica a cada color dentro de un sistema de coordenadas en tres dimensiones.

Dependiendo del tipo de sensor y la aplicación se han desarrollado diferentes modelos de color. Los modelos más utilizados en el procesamiento de imágenes son: RGB, HSV, HSI, YIQ.

Modelo RGB

Se basa en la combinación de los tres colores primarios, rojo(R), verde(G) y azul(B), para la formación del resto de colores. Este modelo es usado para la transmisión, representación y guardado de imágenes en dispositivos digitales como ordenadores, cámaras digitales y escáneres. Este modelo no define los colores rojo, verde y azul, debido a esto los mismos valores RGB pueden mostrar colores diferentes dependiendo del dispositivo. Aunque utilicen el mismo modelo de color, sus espacios de color pueden variar.

Como se ha comentado anteriormente, este modelo usa la síntesis aditiva para la formación de colores. El color está representado por un vector de tres elementos correspondientes con la cantidad de rojo, verde y azul, en ese orden. El rango de los valores tomados por los elementos del vector depende de la profundidad del color. Asumiendo que los vectores son unitarios, el rango de valores es [0,1]. Cuanto más próximo a cero sea el valor, menor será la intensidad de ese color primario en la mezcla, cuanto más próximo a uno, mayor será la intensidad. Por ejemplo, el color rojo se corresponde con el vector (1,0,0), el verde con el vector (0,1,0) y el azul con el (0,0,1).



(a) Imagen RGB (b) Plano Rojo

(c) Plano Verde (d) Plano Azul

Figura 2.16: Planos rojo, verde y azul correspondientes a una imagen RGB.

El subespacio de este modelo es un tetraedro, véase figura 2.17. Los colores primarios, rojo, verde y azul, se encuentran en tres de los vértices del tetraedro. Los colores secundarios, cyan, magenta y amarillo, se encuentran en los otros tres vértices del tetraedro. El color negro se encuentra en el vértice situado en el origen del sistema de coordenadas, mientras que el color blanco se encuentra en el vértice opuesto. Sabiendo la posición del negro y blanco se deduce que la escala de grises se corresponde con la diagonal que une ambos vértices.

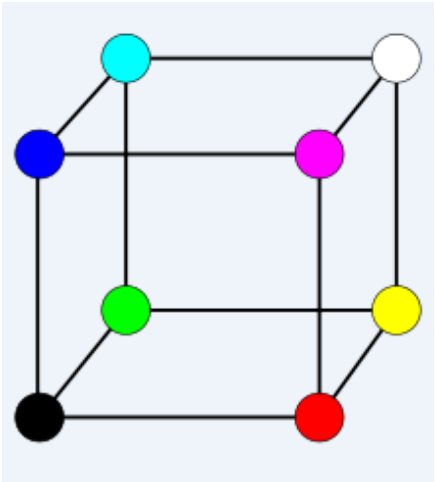


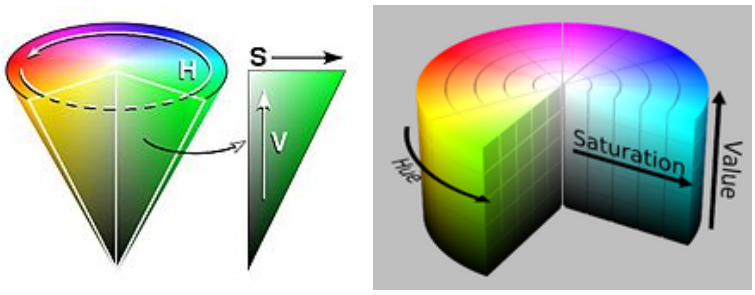
Figura 2.17: Subespacio RGB formado por un tetraedro.

Una imagen formada mediante este modelo de color está basada en la superposición de tres planos. Cada uno de estos se corresponde con uno de los colores primarios, véase figura 2.16. El color de un punto de la imagen está determinado por el valor en ese punto en los tres planos, formando así el vector que determine el color formado por síntesis aditiva de los colores primarios.

También existe una representación hexadecimal, permitiendo expresar fácilmente el color. Este sistema utiliza tres códigos de dos dígitos para expresar la intensidad de cada uno de los colores primarios. Por ejemplo, el color rojo se representaría como #ff0000, el verde como #00ff00 y el azul como #0000ff.

Modelo HSV

Basado en el modo de percepción de los colores que tienen los humanos. Este modelo define el color en términos de sus componentes: hue o matiz, saturation o saturación y value o valor. El subespacio de este modelo es un cono, véase la figura 2.18(a). Donde el matiz está representado por la región circular del cono, la saturación se corresponde con el radio de la región circular y el valor se representa mediante la altura del cono. También se puede representar mediante un cilindro, véase figura 2.18(b).



(a) Cono (b) Cilindro

Figura 2.18: Distribuciones 3D del modelo HSV.

El matiz toma valores que van desde 0 a 360°. Cada valor se corresponde con un color, por ejemplo, el valor 0 se corresponde con el rojo, el 60 con el amarillo y el 120 con el verde. Para identificar las posiciones de los colores primarios: rojo, verde y azul, se ha de dividir los 360 grados en tres partes. Sabiendo que el color rojo se encuentra en 0°, se deduce que el color verde se encuentra en 120°, y el azul en 240°. Los colores secundarios, amarillo, cian y magenta, se encuentran en las posiciones intermedias, el amarillo en 60°, el cian en 180°y el magenta en 300°.

La obtención del color blanco se puede elegir cualquier color, la saturación tiene que ser mínima y el valor tiene que ser el máximo. Para la obtención del color negro ocurre algo similar, se puede elegir cualquier color y saturación, pero el valor tiene que ser el mínimo.

La saturación se corresponde con la pureza colorimétrica, es decir, la cantidad de gris en el espacio del color. Toma valores comprendidos entre 0 y 100 %. Cuanto menor sea la saturación de un color mayor tonalidad grisácea habrá en este y más decolorado estará.

El valor se corresponde con el brillo de un color y varia con la saturación de un color. Los valores que toma están comprendidos en un rango de 0 a 100 %. Cuando el valor es 0 el color es negro.

Aunque el modelo de color RGB sea muy popular, cuando se ha de trabajar con el color se suele utilizar el modelo HSV. Esto se debe a el proceso de formación de color,

mientras que en el modelo RGB el color se forma mediante los valores de rojo, verde y azul, en el modelo HSV el color se corresponde con un solo parámetro, el matiz. Esto hace que sea mucho más sencillo seleccionar colores y discriminarlos en una imagen utilizando el modelo HSV.

Algunas de las aplicaciones donde se utiliza este modelo de color son: sistemas para la detección del grado de madurez de las frutas o la inspección del acabado de color de determinados productos.

Modelo CMY

Este modelo se basa en la síntesis sustractiva, por lo cual sus colores primarios son el cyan, magenta y amarillo. Los colores son formados mediante la absorción de determinadas longitudes de onda.

Este modelo es utilizado especialmente para la impresión, ya que las impresoras depositan pigmentos coloreados los cuales absorben las longitudes de onda de los colores que no se necesitan para la formación del color en ese punto. Para realizar la conversión de RGB a CMY se han de utilizar las ecuaciones: 2.2, 2.3 y 2.4.

$$C = 1 - R \tag{2.2}$$

$$M = 1 - G \tag{2.3}$$

$$Y = 1 - B \tag{2.4}$$

Modelo YIQ

Este modelo presenta mayor sensibilidad a cambios de reflectancia que a cambios de matiz o saturación. Además, permite el tratamiento de la reflectancia por separado, de forma que la reflectancia de una imagen puede ser procesada sin afectar a su contenido en color.

El modelo YIQ se utiliza en la televisión. Siendo una combinación de los valores del modelo RGB, cuya conversión es expresada mediante las ecuaciones: 2.5, 2.6 y 2.7.

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \tag{2.5}$$

$$I = 0{,}596 \cdot R - 0{,}275 \cdot -0{,}321 \cdot B \tag{2.6}$$

$$Q = 0{,}212 \cdot R - 0{,}523 \cdot G + 0{,}311 \cdot B \tag{2.7}$$

Capítulo 3

Visión artificial

La visión artificial o visión por computador es una disciplina científica que permite la adquisición, procesamiento, análisis y comprensión de las imágenes del mundo real. Con el objetivo de extraer información numérica o simbólica para posteriormente ser tratadas mediante un ordenador. El ser humano utiliza los ojos y el cerebro para comprender el mundo que le rodea. La visión por computador trata de proporcionar a los ordenadores la capacidad de percibir y comprender una imagen o secuencia de imágenes y actuar en función a estas. Esto se consigue gracias a distintos campos como la geometría, la estadística, la física y otras disciplinas. La adquisición de los datos se realiza mediante varios medios como secuencias de imágenes, vistas desde varias cámaras de vídeo o datos multidimensionales desde un escáner médico.

Hay muchas tecnologías que utilizan la visión por computador, entre las cuáles:

- Reconocimiento de objetos.
- Detección de eventos.
- Reconstrucción de una escena (mapping).
- Restauración de imágenes.

3.1. Historia

Una de las primeras aplicaciones en la mejora de la calidad de las imágenes fue el tratamiento de las mismas para su publicación en los periódicos enviadas por cable submarino entre Londres y Nueva York. La introducción del sistema de transmisión de imágenes por cable a través del Atlántico, conocido como sistema Bartlane, a principio de los años 20 redujo el tiempo de transmisión desde más de una semana a menos de tres horas. Inicialmente, el sistema era capaz de codificar imágenes en 5 niveles de gris distintos, capacidad que se vio aumentada a 15 niveles hacia 1929. Las mejoras en los métodos de transmisión de imágenes continuaron durante los siguientes 35 años.

La aparición de los primeros computadores digitales y los primeros programas espaciales en USA impulsaron los conceptos de procesamiento de imágenes de una forma significativas.

En los últimos años de la década de los 60, comienza a desarrollarse la visión artificial en aquellas universidades punteras en el campo de la inteligencia artificial. El objetivo de la

visión artificial es imitar el sistema de visión que poseen los seres humanos, permitiendo actuar de forma inteligente a los robots que tuviesen implantados sistemas de visión artificial.

La diferencia de la visión por computador respecto a los sistemas digitales de procesamiento de imágenes de la época, era el objetivo de obtener estructuras tridimensionales a partir de imágenes obteniendo un entendimiento pleno de la escena. Los avances realizados en la década de los 70 asentaron las bases de muchos de los algoritmos utilizados hoy en día en sistemas de visión artificial, algunos de estos algoritmos son:

- Extracción de contornos.
- Etiquetado de lineas.
- Estimación de movimiento.
- Optical flow.

Durante la siguiente década los estudios realizados se basaron en rigurosos análisis matemáticos y aspectos cuantitativos de la visión por computador. Estos incluyen el concepto de *scale-space*, la interferencia en la forma de varias señales como sombra, textura y foco, y el modelado de contornos. Los investigadores se dieron cuenta de que muchos de estos conceptos matemáticos podían ser tratados con el mismo marco de optimización que la *regularización* y los *campos aleatorios de Markov*.

En la década de los 90, algunos de los aspectos investigados se volvieron más populares que otros. La investigación en reconstrucciónes 3D proyectivas permitió un mejor entendimiento de la calibración de las cámaras. Con la optimización de los métodos para la calibración de cámaras, se percataron de que muchos de las ideas habían sido ya desarrolladas en la teoría *bundle adjustment* del campo de la fotogrametría. Esto condujo a la aparición de métodos para la reconstrucción 3D de escenas a partir de múltiples imágenes, obteniendo progresos en cuanto a la problemática de la correspondencia estéreo y las técnicas de *multi-view stereo*. En el mismo periodo de tiempo se produjeron variaciones en el *graphic cut* que permitieron resolver los problemas de segmentación de imágenes. En esta década se utilizaron por primera vez técnicas de aprendizaje estadístico en el reconocimiento de caras en imágenes. En los últimos años de la década de los 90, hubo un cambio significativo cuando aumento la interacción entre los campos de visión por computador y gráficos por computador, permitiendo:

- Image-based rendering.
- Image morphing.
- View interpolatión.
- Panoramic image stiching.
- Light-field rendering.

Actualmente se ha producido un resurgimiento de los métodos basados en características, usados junto con técnicas de aprendizaje automático y marcos complejos de optimización.

3.2. Campos relacionados

Uno de los usos más importantes de la visión por computador es la navegación de un robot en un entorno. Para que un robot pueda desplazarse en un entorno se necesita comprender de forma detallada el entorno. La información sobre el entorno puede ser proporcionada por un sistema de visión artificial, el cual actuaría como un sensor y proporcionaría información de alto nivel tanto del entorno como del robot.

La inteligencia artificial y la visión por computador comparten otros campos como el reconocimiento de patrones y técnicas de aprendizaje. En consecuencia, la visión por ordenador algunas veces es considerada como una parte del campo de la inteligencia artificial o del campo de la informática en general.

La física de *solid-state* es otro de los campos que esta estrechamente relacionado con la visión por computador. La mayoría de los sistemas de visión por computador están basados en la utilización de sensores de imágenes, los cuales detectan radiaciones electromagnéticas, que típicamente se encuentran en forma de luz visible o infrarroja. Los sensores están diseñados utilizando los conocimientos proporcionados por la física cuántica. El proceso por el cual la luz interactúa con las superficies se explica usando la física. La física explica el comportamiento de la óptica, que es una parte fundamental en la mayoría de los sensores de imágenes. La sofisticación de estos sensores hace que incluso se necesite la mecánica cuántica para proporcionar una comprensión completa del proceso de formación de la imagen. Además, varios problemas de medición en la física pueden ser resueltos mediante los sistemas de visión por computador, por ejemplo el movimiento en fluidos.

Un tercer campo que juega un papel importante es la neurobiología, especialmente el estudio de sistema de visión biológica. Durante el siglo pasado, ha habido una gran cantidad de estudios de los ojos, neuronas y de las estructuras cerebrales encargadas del procesamiento de los estímulos visuales tanto en humanos como en animales. Esto ha llevado a una descripción aproximada, pero complicada, de cómo funcionan los sistemas de visión en seres vivos para resolver ciertas tareas relacionas con la visión por computador. Creando un subcampo dentro de la visión por ordenador donde los sistemas de vision artificial son diseñados para imitar el procesamiento y comportamiento de los sistemas de visión biológicos, a diferentes grados de complejidad. Además, algunos de los métodos de aprendizaje desarrollados en el campo de la visión artificial tienen su origen en la biología.

Algunas líneas de la visión por computador están estrechamente relacionadas con el estudio de la visión biológica. Al igual que muchas líneas de investigación de la inteligencia artificial están estrechamente vinculadas con la investigación sobre la conciencia humana, y el uso del conocimiento almacenado para interpretar, integrar y utilizar la información visual. El campo de la visión biológica estudia y modela los procesos fisiológicos detrás de la percepción visual tanto en animales como en personas. La visión por ordenador estudia y describe los procesos de implementación en software y hardware detrás de sistemas de visión artificial. El intercambio interdisciplinario entre la visión biológica y la visión por computador ha demostrado ser beneficioso para ambos campos.

Otro de los campos relacionados con la visión por ordenador es el procesamiento de señales. Muchos métodos para procesar señales de una variable, típicamente señales tem-

porales, se pueden extender de una manera natural al procesamiento de señales de dos variables o señales multi-variables en visión por computador. Sin embargo, debido a la naturaleza especifica de las imágenes hay muchos métodos desarrollados dentro de la visión por computador que no tienen una contrapartida en el procesamiento de señales de una variable. Junto con la multidimensionalidad de la señal, esto define un subcampo en el procesamiento de señales como parte de la visión por computador.

Además de los puntos de vista acerca de la visión por computador expuestos anteriormente, muchos de los temas de investigación relacionados también pueden ser estudiados desde un punto de vista puramente matemático. Por ejemplo, muchos métodos utilizados en la visión por computador están basados en aspectos estadísticos, geométricos u optimizativos.

Los campos más estrechamente relacionados con la visión por computador son el procesamiento de imágenes, análisis de imágenes y visión artificial. Hay una superposición significativa en el rango de técnicas y aplicaciones que cubre. Esto implica que las técnicas básicas que se utilizan y desarrollan en estos campos son más o menos idénticas, algo que se puede interpretar como un sólo campo con nombres diferentes.

La visión por computador es de alguna forma la inversa de los gráficos por ordenador. Mientras que los gráficos por ordenador producen datos de imágenes de los modelos 3D, la visión por computador a menudo produce modelos 3D a partir de datos de imágenes. También existe la tendencia hacia la combinación de las dos disciplinas, por ejemplo, como la realidad aumentada.

Las siguientes caracterizaciones son relevantes pero no deben tomarse como universalmente aceptadas:

- El procesamiento de imágenes y análisis de imágenes tienden a centrarse en imágenes 2D, cómo transformar una imagen en otra, por ejemplo mediante las operaciones *pixel-wise* como la mejora de contraste, las operaciones *locales* como la extracción de bordes o eliminación de ruido, o transformaciones geométricas tales como la rotación de imágenes. Esta caracterización implica que el procesamiento o análisis de imágenes no requiera suposiciones ni produzca interpretaciones sobre el contenido de la imagen.

- La visión por computador incluye el análisis 3D a partir de imágenes 2D. Este análisis de la escena 3D proyectada sobre una o varias imágenes, por ejemplo, la forma de reconstruir la estructura u otra información sobre la escena 3D a partir de una o varias imágenes. La visión por computador suele depender de las suposiciones más o menos complejas sobre la escena representada en la imagen.

- La visión máquina es el proceso de aplicación de una serie de tecnologías y métodos para proporcionar la inspección automática de imágenes., control de procesos y guiado de robots en aplicaciones industriales. La visión artificial tiende a centrarse en aplicaciones, principalmente en la fabricación, por ejemplo, robots y sistemas autónomos basados en la visión para la inspección basada en la visión o la medición. Esto implica que las tecnologías de sensores de imagen y la teoría de control a menudo están integradas con el procesamiento de datos de imágenes para controlar un robot y que el procesamiento en tiempo real se enfatiza mediante implementaciones eficientes en hardware y software. También implica que las condiciones externas co-

mo la iluminación pueden ser y son a menudo más controladas en la visión artificial que en la visión por computador, lo que permite el uso de diferentes algoritmos.

- La formación de imágenes conocida con el nombre de *imaging*, pero a veces también se ocupa del procesamiento y análisis de imágenes. Por ejemplo, la creación de imágenes médicas, *meducal imaging*, requiere de un trabajo de análisis de datos de imágenes para la utilización en aplicaciones médicas.
- El reconocimiento de patrones es un campo que utiliza varios métodos para extraer información de señales en general, generalmente basados en enfoques estadísticos y redes neuronales artificiales. Una parte significativa de este campo se dedica a aplicar estos métodos a los datos de imágenes.

3.3. Aplicaciones

Las aplicaciones van desde sistemas de visión artificial en la industria, como por ejemplo la comprobación del etiquetado de latas en una linea de producción, a la investigación en el campo de la inteligencia artificial en robots. La visión por computador y la visión industrial artificial tienen muchos aspectos en común. La visión por computador esta formada por las principales tecnologías que permiten el análisis automático de imágenes. La visión industrial artificial se refiere a la combinación de varios métodos de análisis automático de imágenes junto con otros métodos y tecnologías que permiten la inspección automatizada y la orientación de robots en aplicaciones industriales. En muchas aplicaciones de visión por computador, los computadores están preprogramadas para resolver una tarea en particular, pero los métodos basados en el aprendizaje del sistema están ganando popularidad. Algunos ejemplos de aplicaciones de visión por computador incluyen sistemas para:

- Control de procesos, por ejemplo el control de un robot industrial.
- De navegación, por ejemplo vehículos autónomos.
- La detección de eventos, por ejemplo, para el conteo de personas o la vigilancia visual.
- La organización de la información, por ejemplo las bases de datos de indexación de imágenes y secuencias de imágenes.
- Modelado de objetos o entornos, por ejemplo, análisis de imágenes médicas o de modelado topográfico.
- La interacción, por ejemplo, como la entrada a un dispositivo para una interacción hombre-maquina.
- Inspección automática, por ejemplo, aplicaciones de fabricación.

Uno de los campos de aplicación más prominentes es la visión médica por ordenador o el procesamiento de imágenes médicas. Esta área se caracteriza por la extracción de información a partir de datos de imágenes con el propósito de hacer un diagnóstico médico de un paciente. En general, los tipos de imágenes con los que se trabaja en este campo son imágenes microscópicas, imágenes de rayos X, imágenes de angiografía, imágenes ultrasónicas, e imágenes de tomografía. Un ejemplo aplicaciones médicas son la detección de tumores, arteriosclerosis. También se pueden llevar a cabo mediciones de órganos, el

flujo de sangre, etc. Esta área de aplicación también apoya la investigación médica, proporcionando nueva información, por ejemplo, acerca de la estructura del cerebro, o sobre la calidad de los tratamientos médicos. Las aplicaciones de la visión por computador en el área médica también incluyen la mejora de las imágenes que son interpretadas por los seres humanos, por ejemplo imágenes ultrasónicas o imágenes de rayos X, para reducir la influencia del ruido.

Otra de las áreas más importantes en cuanto a la aplicación de la visión por computador es la industria, conocida algunas veces como visión artificial, donde se extrae información con el fin de facilitar el proceso de fabricación. Un ejemplo es el control de calidad en el que los detalles o productos finales se inspeccionan automáticamente para detectar defectos. Otro ejemplos es la medición de la posición y la orientación de los objetos a ser recogidos por un brazo robot. La visión artificial también se encuentra muy extendida en los procesos agrícolas para la detección y eliminación de alimentos en mal estado, este proceso se llama selección óptica.

Las aplicaciones militares son probablemente una de las áreas más grandes para la visión por computador. Algunas de estas aplicaciones son la detección de soldado o vehículos enemigos y el guiado de misiles. Los sistemas más avanzados de direccionamiento de misiles envían el misil a un área en vez de a un objetivo específico, y la selección de blancos se hace cuando el misil alcanza el área. Los conceptos modernos, como el reconocimiento del campo de batalla, implican la utilización de varios sensores, incluyendo sensores de visión, proporcionando información sobre el campo de batalla que puede usarse para apoyar decisiones estratégicas. En este caso, el procesamiento automático de los datos se utiliza para reducir la complejidad y para fusionar información de múltiples sensores para aumentar la confiabilidad.

Una de las áreas de aplicación más recientes es la de vehículos autónomos, incluyendo vehículos sumergibles, terrestres (pequeños robots con ruedas, coches o camiones), aéreos, y vehículos aéreos no tripulados (UAV). El nivel de autonomía varía desde vehículos totalmente autónomos (no tripulados) hasta vehículos en los que los sistemas basados en la visión por computador soportan necesitan un conductor o piloto en diversas situaciones. Los vehículos totalmente autónomos suelen utilizar la visión por computador para la navegación, es decir, para saber donde está, o para producir un mapa de su entorno(SLAM) y para detectar obstáculos. También se puede utilizar para la detección de ciertos eventos como por ejemplo la detección de incendios forestales. Un ejemplo de sistemas de apoyo son los sistema de advertencia de obstáculos implementados en algunos automóviles y los sistemas de aterrizaje autónomo de aeronaves. Varios fabricantes de automóviles han implementado en sus vehículos sistemas de conducción autónoma, como por ejemplo Tesla y Mercedes-Benz. Hay una amplia variedad de ejemplos de vehículos militares autónomos, desde misiles avanzados hasta vehículos aéreos no tripulados para misiones de reconocimiento. La exploración del espacio esta siendo llevada a cabo por vehículos autónomos utilizando la visión por computador, por ejemplo, el Mars Exploration Rover de la NASA y el ExoMars Rover de la ESA.

Otras áreas de aplicación incluyen:

- Apoyo de efectos visuales en el mundo cinematográfico, por ejemplo el seguimiento de la cámara (matchmoving).

- Vigilancia.

3.4. Tareas típicas

Cada una de las áreas de aplicación descritas anteriormente emplea un conjunto de tareas propias de la visión por computador. A continuación se presentan algunos ejemplos de tareas típicas de visión por ordenador.

3.4.1. Reconocimiento

El problema clásico en la visión por ordenador. El procesamiento de imágenes y la visión artificial es el de determinar si los datos de la imagen contienen algún objeto, característica o actividad especifica. En la literatura se describen diferentes variedades del problema de reconocimiento:

- Reconocimiento de objetos, también llamado clasificación de objetos. Uno o varios objetos son especificados con anterioridad o aprendidos de forma que pueden ser reconocidos, por lo general junto con sus posiciones en 2D en la imagen o su posicionamiento 3D en la escena. Blippar, Google Goggles tienen integrados sistemas de reconocimiento de objetos.

- Identificación, la imagen de un objeto es reconocida dentro de otra imagen. Por ejemplo la identificación de la huella dactilar.

- Detección, los datos de una imagen son analizados en busca de una condición especifica. Los ejemplos incluyen la detección de posibles células o tejidos anormales en imágenes médicas o la detección de un vehículo en un sistema de peaje automático. La detección basada en cálculos relativamente simples y rápidos se utiliza a veces para encontrar regiones más pequeñas de datos de imagen interesantes que pueden analizarse adicionalmente mediante técnicas más exigentes desde el punto de vista computacional para producir una interpretación correcta.

En la actualidad, los mejores algoritmos para estas tareas se basan en redes neuronales convolucionales. Una ilustración de las capacidades de los algoritmos se pueden observar en la ImageNet Large Scale Visual Recognition Challange, siendo este el punto de referencia en los algoritmos de clasificación y detección de objetos. El rendimiento de las redes neuronales convolucionales, en las pruebas del ImagNet, se encuentra cerca al de los humanos. Los mejores algoritmos todavía siguen teniendo problemas en la identificación de objetos pequeños o delgados, como hormigas en el tallo de una flor. También tienen problemas con imágenes que han sido distorsionadas con filtros (un fenómeno cada vez más común con cámaras digitales modernas). Por el contrario, este tipo de imágenes no suponen ningún problema para el ser humano. Los seres humanos suelen tener problemas en la clasificación de objetos en clases de grano fino, como la raza particular de un perro o especie de pájaro, mientras que las redes neuronales convolucionales manejan esto con gran facilidad.

Existen varias tareas especializadas basadas en el reconocimiento, tales como:

- Recuperación de imágenes por contenido, consiste en la identificación entre un amplio conjunto de imágenes, aquellas que tienen un algo específico. El contenido se puede especificar de varias manera, en términos de similitud respecto a una imagen de muestra (obtener todas las imágenes similares a una imagen), o en términos

de búsqueda de alto nivel(obtener todas las imágenes con casas, coches y tomadas durante el invierno).

- Estimación de posición, o la orientación de un objeto en relación específica a la cámara. Un ejemplo de aplicación de esta técnica sería ayudar al brazo de un robot a retirar objetos de una cinta transportadora en una cadena de montaje.
- Reconocimiento óptico de caracteres (OCR), la identificación de los caracteres en las imágenes de texto impreso o escrito a mano, por lo general, con vistas a la codificación del texto en un formato con mayor facilidad para la edición.
- Lectura de códigos como matrices de datas o códigos QR.
- Reconocimiento facial.
- Tecnología de reconocimiento de formas(SRT), en los sistemas encargados de contra personas, la diferenciación de estas respecto a objetos se puede conseguir mediante la identificación de patrones de cabeza y hombros.

3.4.2. Análisis de movimiento

Varias tareas se relacionan con la estimación de movimiento donde se procesa una secuencia de imágenes para producir una estimación de la velocidad, ya sea en cada punto de la imagen o en la escena 3D, o incluso de la cámara que produce las imágenes. Ejemplos de estas tareas son:

- Egomotion, determinar el movimiento 3D, rotación y traslación, de la cámara a partir de una secuencia de imágenes producidas por la cámara.
- Tracking, seguimiento de los movimientos de un conjunto de puntos de interés u objetos, por ejemplo vehículos o seres humanos, en la secuencia de imágenes.
- Optical flow, determinación para cada punto de la imagen, cómo este se mueve en relación con el plano de la imagen, es decir, su movimiento aparente. Este movimiento es el resultado de la combinación del movimiento realizado por el punto 3D correspondiente respecto a la escena y cómo se mueve la cámara con relación a la escena.

3.4.3. Reconstrucción de la escena

Proporcionando una o más imágenes correspondientes a un escena intenta realizar un modelo 3D de la escena. En la forma más sencilla el modelo es un conjunto de puntos 3D. Algunos métodos más sofisticados producen una superficie 3D. Existen sensores que utilizan un patrón de puntos para poder crear un modelo 3D a partir de la imagen. Actualmente se dispone de algoritmos que permiten superponer imágenes 3D formando una nube de puntos.

3.4.4. Restauración de imágenes

El objetivo de la restauración de imágenes es la eliminación de ruido (ruido del sensor, desenfoque de movimiento, etc.) de las imágenes. El enfoque más simple para la eliminación del ruido consiste en la utilización de varios tipos de filtros, tales como los filtros de paso bajo o filtros medianos. Métodos más sofisticados suponen el modelo de estructura

de la imagen, un modelo que distingue la imagen del ruido. Primero se analizan los datos obtenidos a partir de la imagen en condiciones de la estructura de la imagen, tales como líneas o bordes, y después se controla el filtrado basándose en la información obtenida en el paso anterior, obteniendo una mejor eliminación del ruido en comparación con los enfoques más simples.

3.4.5. Métodos del sistema

La organización de un sistema de visión por computador depende en gran medida de la aplicación a desarrollar por este. Algunos sistemas tienen como cometido la resolución de un problema específico de detección o medición, mientas que otros son subsistemas dentro de un sistema mayor que a su vez contiene otros subsistemas encargados del control de actuadores mecánicos, planificación, bases de datos de información, interfaces máquina, etc. La implementación especifica de un sistema de visión por computador también depende de si su funcionalidad esta especificada o si parte de ella puede ser aprendida o modificada durante el funcionamiento. Muchas funciones suelen ser exclusivas de la aplicación, sin embargo hay una serie de funciones típicas que se encuentran en muchos sistemas de visión por ordenador.

- Adquisición de imágenes. Una imagen digital es producida por uno o varios sensores de imagen, además de diversos tipos de cámaras sensibles a la luz, incluyen sensores de rango, aparatos de tomografía, radares, etc. Dependiendo del tipo de sensor, la imagen obtenida puede ser una simple imagen 2D, un volumen 3D, o una secuencia de imágenes. Los valores de los pixeles normalmentºe corresponden a la intensidad de luz en una o varias bandas espectrales (imágenes grises o de imágenes en color), pero también pueden estar relacionados con diferentes medidas físicas, tales como la profundidad, la absorción o reflactancía de las ondas ultrasónicas o electormagnéticas, o de resonancia magnética nuclear.

- Pre-procesamiento. Antes de que un método de visión por computador pueda trabajar con la imagen para extraer información de esta, suele ser necesario el procesamiento de la imagen para asegurar que la imagen cumple ciertos aspectos asumidos por el sistema de visión por computador. Algunos ejemplos son:

  - Remuestreo para asegurar que el sistema de coordenadas de la imagen es el correcto.
  - Reducción del ruido para evitar que este falsee el resultado.
  - Mejorar el contraste para asegurar que se puede detectar la información relevante.

- Extracción de características. Las imágenes poseen una serie de características que se pueden clasificar en varios niveles de complejidad. Los ejemplos típicos son:

  - Las líneas, los bordes y aristas.
  - Localización de puntos de interés tales como esquinas, manchas o puntos.

  Las características más complejas pueden estar relacionadas con la textura, la forma o el movimiento.

- Detección/segmentación. En un cierto punto en el procesamiento de la imagen se toma la decisión de que puntos de la imagen o regiones son relevantes para su posterior procesamiento. Algunos ejemplos son:

  - Selección de un conjunto especifico de puntos de interés.
  - Segmentación de una o varias regiones de imagen que contienen un objeto específico de interés.

- Procesamiento de Alto Nivel. En este paso la entrada es típicamente un pequeño conjunto de datos, por ejemplo un conjunto de puntos o una región de la imagen que se supone que contiene un objeto específico. El procesamiento restante tiene que lidiar con:

  - Comprobación de que los datos cumplen las suposiciones realizadas por la aplicación.
  - Estimación de parámetros específicos de la aplicación, como la posición del objeto o el tamaño.
  - Reconocimiento de imágenes, clasificar un objeto detectado en diferentes categorías.
  - El registro de imágenes, comprobar y combinar dos vistas diferentes del mismo objeto.

3.4.6. Toma de decisiones

Tomar la decisión final necesaria para la aplicación, como por ejemplo:

- Aprobación o desaprobación en aplicaciones de inspección automática.
- Encontrar o no un patrón en aplicaciones de reconocimiento.
- Obtención de posibles positivos para la posterior revisión humana en aplicaciones médicas, militares y de seguridad.

3.5. Sistema visual humano

Los sentidos nos permiten interactuar con el medio, la vista es considerado el sentido más importante y sofisticado. Se considera que el 75 % de la información procedente de los sentidos es captada por la vista, teniendo dos millones de fibras nerviosas. A través de la vista se recibe información del posicionamiento de objetos, distinción entre objetos móviles y fijos, reconocimiento de personas, etc.

Hasta el momento no se ha conseguido desarrollar un sistema de visión artificial tan complejo como el sistema visual humano, siendo este el objetivo a alcanzar. Para la implementación de sistemas de visión artificial es necesario conocer y entender el sistema visual humano.

3.5.1. Formación imágenes

Como se ha comentado en el capítulo dedicado al color. Los objetos reflejan ondas electromagnéticas dependiendo de su longitud de onda. El ojo actúa como un receptor de estas ondas electromagnéticas, transmitiendo estímulos nerviosos al cerebro, el cual produce una sensación de claridad y de color.

El ojo es el órgano más importante del sistema de visión humano, está compuesto por una serie de partes que permiten la realización de sus funciones, algunas de las más importantes son: pupila, iris, cámara interior, cristalino, músculo ciliar, esclerótica, coroides, cuerpo vítreo y retina.

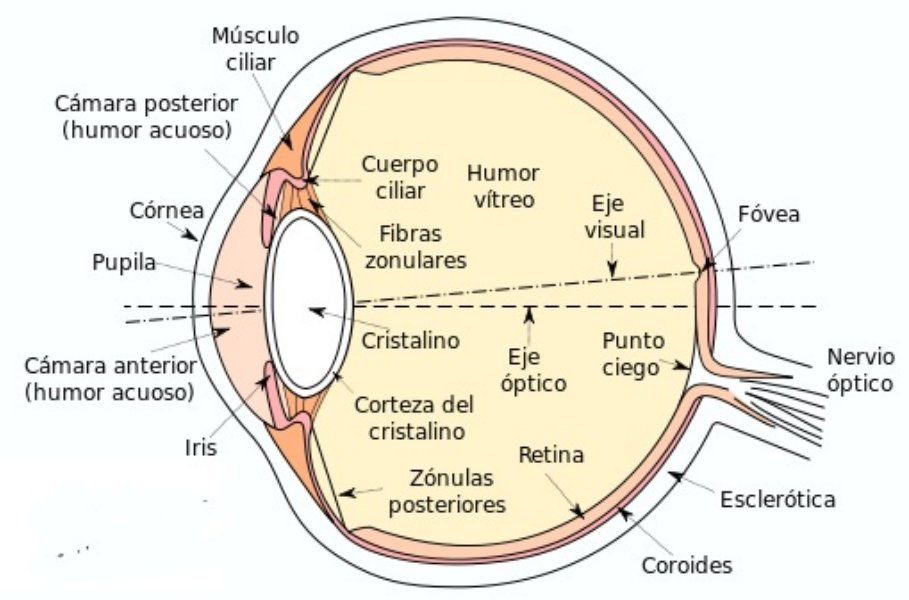


Figura 3.1: Corte transversal del ojo humano.

La luz entra al ojo por la pupila, la cual es transparente. El paso de luz es regulado por el iris y el músculo ciliar, abriendose y cerrandose en función de la cantidad de luz mediante la ayuda del músculo ciliar. Cuando la cantidad de luz que pasa por la pupila muy alta, el iris se cierra. En el caso de que la cantidad de luz sea insuficiente el iris se abre. El movimiento del iris es producido gracias al músculo ciliar.

La rayos de luz deben de concentrarse en la retina para que se puedan formar las señales correspondientes a las imágenes. Antes de llegar a la retina los rayos luminosos atraviesan la córnea, la cual es transparente y permite el paso de los rayos de luz. Mediante el iris se regula la cantidad de luz que pasa hacia la retina. El enfoque es llevado a cabo mediante el cristalino y los músculos ciliares, estos deforman el cristalino permitiendo que enfoque objetos a distintas distancias.

El elemento sensor lo forma la retina, que se encuentra en la parte posterior del glóbulo ocular. Se pueden distinguir dos partes: la fóveda y la mácula. La fóveda es la zona central donde se encuentra el mayor número de elementos sensibles a la luz y que dará una zona de la imagen con una alta resolución. Su área es muy pequeña, de 1.5mm de diámetro, y de unos dos grados de ángulo visual. La mácula constituye la mayor parte de la retina y es donde la resolución es peor. Además de por el tamaño y resolución ambas zonas se diferencian por las longitudes de onda de la luz a las que presentan una mayor actividad [4].

En la retina se pueden encontrar dos tipos de células fotoreceptoras: los bastones y los conos. La función de ambas es similar, aunque reaccionan a diferentes longitudes de onda. Los bastones se encuentran mayoritariamente en la mácula aunque se pueden encontrar por toda la retina. Estos fotoreceptores se activan para longitudes de onda comprendidas

entre 350 y 600 nanómetros, formando imágenes en blanco y negro debido a que no reaccionan al color. Los bastones transmiten los estímulos a través del nervio óptico, estando conectados varios a una misma neurona. Los conos se hayan en la fóveda de la retina. Existen tres tipos de conos dependiendo de las longitudes de onda a las que reaccionen, 380 a 500, de 500 a 600 y de 600 a 730 nanómetros siempre que la luz sea brillante, permitiendo distinguir los colores presentes en la escena observada. Cada cono se encuentra conectado mediante el nervio óptico a varias neuronas.

Las señales nerviosas enviadas a las neuronas son procesadas por el cerebro. Estas llegan al cortex visual primario. Después de recibir la información que se genera de forma continua, el cerebro lleva a cabo la extracción de características. Ciertas partes del cerebro responden mejora a determinadas características, el lóbulo inferior temporal responde a los colores, texturas y formas de objetos que hay en las imágenes. Mientras que las zonas parietales responden mejor a los cambios espaciales de los objetos en la imagen.

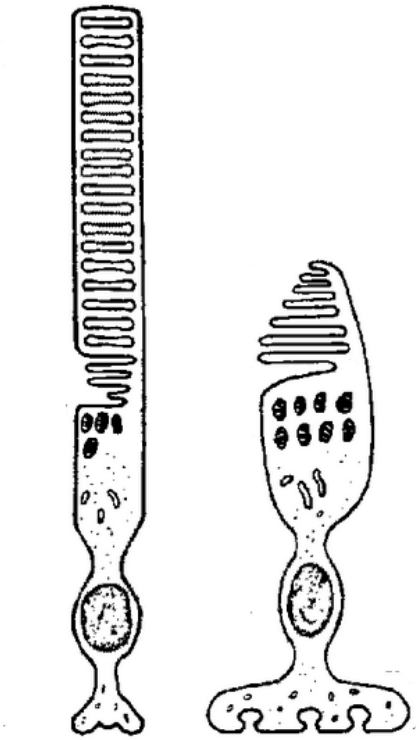


Figura 3.2: Representación de las células fotoreceptoras, donde se puede ver un bastón y un cono, a la izquierda y derecha respectivamente.

3.5.2. Diferencias y similitudes con la visión natural

La visión humana puede definirse como un sistema integrado, de forma genérica, por dos ojos, el nervio óptico y el cerebro. El ojo es una lente que forma una imagen óptica y detecta la imagen con su retina. Desde aquí, a través del nervio óptico, la información es transmitida al cerebro para que se procese y se extraiga la información necesaria.

El funcionamiento de un sistema de visión artificial es muy similar; una cámara es capaz de recoger cierta imagen a través de la lente, y transmitir la secuencia obtenida a un ordenador que analiza la información, que puede emplearse para el fin que se desee.

El ojo humano está formado por una serie de órganos y partes bien diferenciadas que, salvando la comparativa biológica-mecánica, se pueden hacer corresponder fácilmente con elementos de un sistema de visión automático.

- La pupila de ojo realiza la función del diafragma, al encargarse de controlar la cantidad de luz que entra al sensor.
- El cristalino posee unas propiedades similares a las de la lente. Logra enfocar la imagen y reproducir los objetos de forma nítida.

- La retina, al igual que la película, es la encargada de general la imagen que realmente se va a ver.
- El nervio óptico funciona como un bus de conexión, capaz de conectar la cámara a un sistema de procesamiento.
- El lóbulo parietal del cerebro es el equivalente al sistema de procesamiento de datos. Es decir, en un sistema automático de visión se aplica un algoritmo ya implementado de forma intrínseca que determina si la imagen cumple cierta serie de requisitos.

Actualmente, la reproducción de las capacidades de la visión humana en un sistema de visión sigue siendo una tarea utópica, dada la gran cantidad de formas y colores que es capaz de procesar el cerebro humano en ínfimos instantes de tiempo, el ojo humano puede detectar 100 millones de píxeles en el espectro de la luz, mientras que una cámara normal puede llegar a unos 250.000 elementos. Sin embargo, a través de la simplificación de ciertos elementos y la utilización de algoritmos de procesamiento, es posible reproducir en gran medida estas capacidades.

No obstante, dichas capacidades no son necesarias para todas las operaciones existentes. Si bien es cierto que, por ejemplo, un sistema de visión no puede conducir un coche, ya que es incapaz de reaccionar ante eventos imprevisibles, sí que puede realizar trabajos sencillos y repetitivos, como los que se encuentran en las cadenas de montaje, que pueden llegar a ser repetitivos. En este tipo de tareas, la efectividad del ser humano se ve reducida drásticamente, mientras que un sistema de visión implementado de forma lo suficientemente adecuada puede llegar a rozar efectividades superiores al 99 %.

3.6. Visión estereoscópica

Permite la obtención de información 3D a partir de imágenes digitales obtenidas mediante cámaras CCD, es decir, permite obtener la tercera dimensión de los objetos situados en las imágenes. La información se obtiene mediante la comparación de una misma escena desde dos puntos de vista distintos, obteniendo así la posición relativa de los objetos dentro de esta. Este proceso está basado en la estereoscopía.

Al trabajar con imágenes se representan las escenas de forma bidimensional, es decir, de forma directa podemos saber tanto la altura como anchura de un objeto en la escena pero se desconoce cualquier información acerca de la 3D. Esto ha supuesto un problema sobre todo para los sistemas de navegación en robótica, los cuales necesitan esta información para poder moverse por la escena. Para solucionar esta falta de información se han desarrollado algunos métodos como:

- Sensores de ultrasonidos, que utilizan energía acústica. Estos están compuestos de un emisor y un receptor de ultrasonidos. Su funcionamiento consiste en: el emisor emite el ultrasonido en un instante t, este rebota y es captado por el receptor. Sabiendo el instante en que se emitió el ultrasonido se puede calcular el tiempo que ha tardado en llegar. Conociendo la velocidad a la que viaja el sonido en el medio y el tiempo que ha tardado se calcula la distancia al objeto en el cual ha rebotado el ultrasonido.
- Triangulación, consiste en la emisión de un haz, mediante una fuente de luz, con un cierto ángulo. Al incidir este sobre una superficie se refleja, este haz reflejado es recogido por un detector. Estando la fuente de luz y el detector separados una

distancia se forma un triángulo, cuyos lados son: la distancia entre la fuente de luz y detector, la distancia entre el objeto donde se refleja el haz incidente y la fuente de luz y, por último, la distancia entre el detector y el objeto. De esta forma se puede conocer la distancia al objeto.

- Luz estructurada, se basa en la proyección de una configuración de luz, normalmente una malla de puntos. La distorsión de los puntos de la malla es obtenida mediante una cámara y a partir de esta distorsión se puede calcular las distancias.

- Visión estereoscópica artificial, donde se utilizan dos o más cámaras para obtener imágenes de la misma escena, pero con diferentes perspectivas, permitiendo mediante la triangulación obtener las distancias.

Los métodos expuestos anteriormente se pueden clasificar en dos tipos, pasivos y activos. Los métodos activos son aquellos que actúan sobre la escena, iluminándola o enviando un haz energético.

3.6.1. Historia

La visión binocular fue estudiada por Leonardo da Vinci y Euclides. Los principios de la misma fueron expuestos en una serie de estudios realizados por el Kepler.

En 1838, el físico Sir Charles Wheatstone construyó el primer sistema que permitía percibir la tridimensionalidad a partir de dos imágenes, considerándolo el padre de la visión estereoscópica. Este acontecimiento ocurrió antes del descubrimiento de la fotografía.

En 1849, Sir David Brewster crea la primera cámara estereoscópica. Esta permitía ver las imágenes tomadas por las lentes mediante un visor. Posteriormente, Oliver Wendell Holmes construyó el estereoscopio de mano más popular, véase figura 3.3, del siglo XIX.

En 1930, aparecen las cámaras 3D, como ViewMaster o Realist, resurgiendo con ellas la estereofotografía. En la actualidad dichas cámaras han quedado obsoletas.

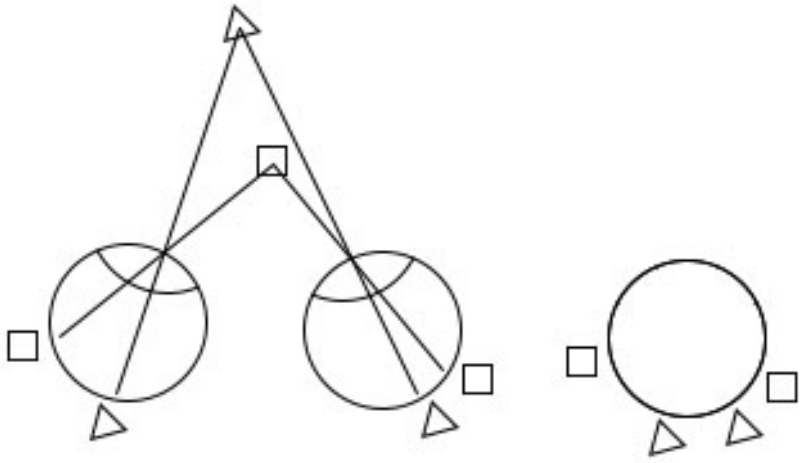
A mediados del siglo XX se intento la creación de películas 3D, debido a que las técnicas utilizadas causaban problemas en la vista no tuvieron éxito. En 1980, surgen las películas en alta resolución como IMAX3D.

Figura 3.3: Estereoscopio creado por Oliver Wendell Holmes.

3.6.2. El ser humano

La visión estereoscopica se basa en el modelo estereocópico biológico. Los seres humanos tenemos un sistema de visión estereoscópica natural, formado por los ojos y el cerebro. Debido al desplazamiento relativo de los ojos, situándose en posiciones diferentes, las imágenes recogidas en la retina de cada uno son ligeramente diferentes. Estas diferencias son procesadas por el cerebro, calculando las distancias a la que se encuentran los objetos mediante un proceso de triangulación conocido como paralaje.

El paralaje es la desviación angular de la posición aparente de un objeto, dependiendo del punto de vista elegido. Como se puede apreciar en la figura 3.4(a), la posición del triangulo varía dependiendo del punto de vista. En el ojo derecho estaría a la izquierda del cuadrado, mientras que en ojo izquierdo estaría a la derecha del cuadrado.



(a) Sistema de visión estereoscópica biológico.

(b) Resultado de superposición de las imágenes obtenidas por los dos ojos.

Figura 3.4: Ejemplo del proceso de visión estereoscópica biológico.

En la figura 3.4(b), se puede apreciar la desviación angular o separación relativa de ambos objetos. El triangulo presenta una desviación angular menor a la del cuadrado. Sabiendo que la desviación angular es inversamente proporcional a la distancia del objeto al ojo, se deduce que el cuadrado se encuentra más cerca que el triangulo en la escena. La disparidad es la separación relativa de los objetos en las imágenes obtenidas en cada ojo.

El cálculo de las distancias permite posicionar espacialmente a los objetos en la escena, obteniendo una sensación de profundidad o volumen. Las variaciones verticales no son utilizadas para la creación de la sensación de profundidad.

La estereoscopia es el método principal por el cual el cerebro crea imágenes tridimensionales de los objetos que estamos viendo, aunque no es la única. Algunos de los métodos utilizados por el cerebro para obtención de profundidad son:

- Distribución de luces y sombras. La iluminación es un factor muy importante a la hora de crear sensación de volumen. Mediante la sombra y el contraste se pueden obtener la sensación de relieve y volumen.

- Superposición de imágenes. Cuando se tienen dos objetos en una misma escena, el objeto, que oculta parcialmente al otro, es el más cercano. Mientras que el objeto oculto es el más lejano.

- Perspectiva. El efecto de profundidad obtenido mediante la perspectiva es muy claro. Un ejemplo es una imagen de una carretera, donde las líneas que delimitan los carriles son paralelas, pero en la imagen convergen en un punto conocido como el punto de fuga.

- Paralaje por movimiento. Cuando el observador se mueve se produce la sensación de movimiento de los objetos de la escena en el sentido del movimiento del observador y en el contrario dependiendo de su posición. Fijando la vista en un objeto, todos los objetos por detrás de este se moverán en sentido contrario al desplazamiento del observador mientras que los objetos situados delante se moverán en el mismo sentido.

3.6.3. Adquisición de imágenes

La toma de imágenes se puede realizar de formas muy diferentes. Esta va a estar influida por el tipo de aplicación para la cual se van a utilizar las imágenes. No se pueden tomar las imágenes para una aplicación basada en la detección de obstáculos de un vehículo, en la que se necesitan imágenes de alta resolución ya que se va a llevar a cabo la identificación de objetos. Sin embargo, para una aplicación basada en la obtención de mapas cartográficos no es necesario que las imágenes sean de alta resolución.

Además la toma de las imágenes puede ser simultanea mediante la utilización de varias cámaras, las cuales toman las imágenes en el mismo instante de tiempo. Las imágenes obtenidas presentan diferencias debidas al posicionamiento de las cámaras, siendo necesario conocer el posicionamiento de las cámaras para poder obtener la información 3D de la escena.

Otra forma de tomar imágenes es mediante una sola cámara. Esta forma se da en aplicaciones como la citada anteriormente de obtención de mapas cartográficos, donde se situaría en un vehículo en movimiento. En este caso es esencial conocer la velocidad de desplazamiento del vehículo y el intervalo de tiempo entre imágenes, para poder obtener la información 3D.

3.6.4. Modelo de cámara

El modelo de cámara consiste en una serie de atributos tanto geométricos como físicos, necesarios para la implementación de un sistema de visión estéreo. Una de las diferencias más importantes entre los diferentes modelos es la situación de los ejes ópticos de las cámaras. Los ejes ópticos pueden situarse de dos formas distintas, paralelos o convergentes.

Otro aspecto importante dentro de los modelo de cámaras es la relación entre las posiciones de los elementos en el sistema. Algunos modelos presentan una componente relativa, relacionando el sistema de coordenadas de una cámara con el del la otra, siendo independiente de la escena. También pueden presentar una componente absoluta, la cual relaciona el sistema de coordenadas de una cámara con el sistema de coordenadas fijo de la escena.

El modelo de dos cámaras con sus ejes ópticos paralelos es muy utilizado debido a la sencillez de los cálculos necesarios para la obtención de información 3D de la escena. Este modelo se puede observar en la figura 3.5. Los conceptos básicos de este modelo son:

- Longitud focal efectiva(f), siendo la distancia entre el eje óptico y el plano de formación de la imagen.

- Linea base, se corresponde con la distancia entre los ejes ópticos de ambas cámaras.

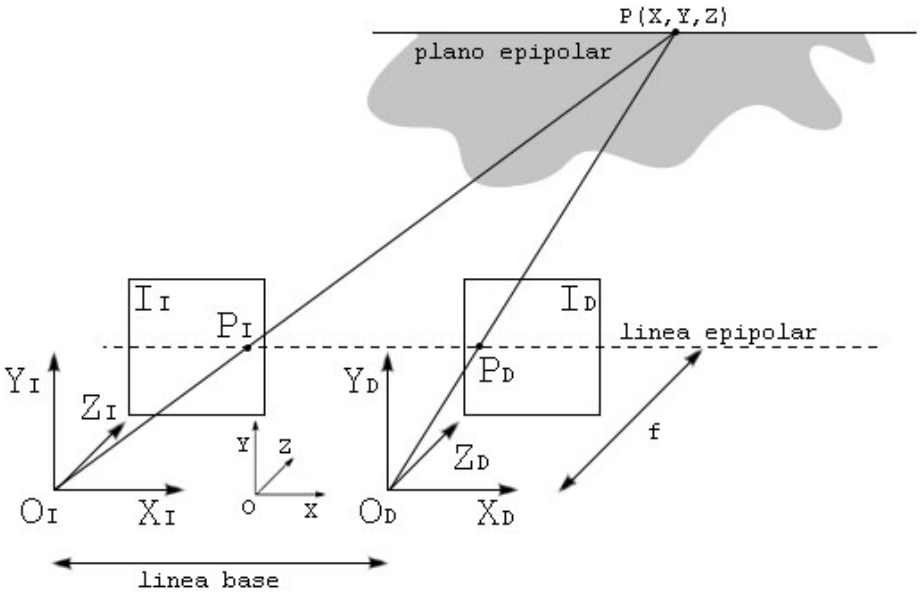


Figura 3.5: Representación del modelo geométrico de dos cámaras con los ejes ópticos paralelos.

- Linea epipolar, es una linea que une el mismo punto en la imagen izquierda y derecha.
- Plano epipolar, se forma mediante tres puntos, un punto de la escena, y los dos puntos correspondientes a cada uno de los centros de proyección de las cámaras. La intersección de este y el plano de proyección de una cámara es la linea epipolar.

Una característica de este modelo es la correspondencia entre los puntos y las líneas epipolares. Todos los puntos cuya proyección derecha esté en una misma línea epipolar en la imagen derecha, sus proyecciones izquierdas deben de estar en una misma línea epipolar en la imagen izquierda, y viceversa.

La búsqueda de los puntos correspondientes en las distintas imágenes puede ser un proceso muy complejo. Esta complejidad se debe a la necesidad de hacer una búsqueda en dos dimensiones. Para llevar a cabo el emparejamiento de dos puntos, se ha de seleccionar un conjunto de píxeles vecinos ya que las imágenes derecha e izquierda pueden presentar un desplazamiento tanto vertical como horizontal. Para reducir la complejidad de la búsqueda de los puntos correspondientes se establecen los ejes ópticos de las cámaras de forma que solo exista un desplazamiento horizontal entre estos. Esto se consigue haciendo que los ejes ópticos de las cámaras sean paralelos y los ejes de abscisas sean coincidentes, consiguiendo que las imágenes estén alineadas horizontalmente. En este caso las lineas epipolares que definen el plano epipolar son coincidentes. Esta geometría permite realizar una búsqueda de los puntos correspondientes recorriendo las imágenes por filas.

Con la geometría de un sistema estereocópico citada en el párrafo anterior, se obtiene la restricción epipolar. Esta restricción determina que los valores de disparidad de un punto solamente se deben a la diferencia entre las componentes horizontales de las representaciones de dicho punto en la imagen izquierda y derecha. A partir de los valores de

disparidad obtenidos para cada punto se puede crear un mapa de disparidades. Utilizando el mapa de disparidades se puede obtener un mapa de profundidad que es el objetivo de un sistema estereoscópico.

3.6.5. Extracción de las características

Una vez obtenidas las imágenes del sistema estereoscópico se ha de establecer la correspondencia entre ellas. Para ello se lleva a cabo un análisis de las imágenes con el fin de obtener elementos identificativos, facilitando el proceso de correspondencia.

La obtención de los elementos identificativos esta condicionada a la técnica elegida para llevar a cabo la correspondencia. Algunos de los elementos son: puntos de borde aislados, cadenas de puntos de bordes, regiones delimitadas por bordes. Los bordes de los objetos presentes en la escena fotografiada son de gran ayuda a la hora de realizar el emparejamiento, ya que permiten identificar tanto forma como tamaño del objeto. Para la obtención de los bordes se ha de realizar la primera derivada de la imagen obteniendo puntos de borde en las zonas donde hay un cambio brusco de color. Una vez obtenidos dichos puntos de borde y dependiendo del método utilizado se trabajara con segmentos rectos, no recots, cadenas cerradas formando estructuras geométricas, etc. Las regiones son elementos utilizados frecuentemente en los sistemas de visión estereoscópica, los cuales son zonas de las imágenes que se asocian a superficies en la escena y que están delimitadas por puntos de borde.

En algunos casos no se buscan elementos sino que se comparan patrones de luminosidad de un conjunto de píxeles en ambas imágenes.

Dependiendo de la técnica de correspondencia utilizada puede ser necesaria la realización de un paso más a la hora de obtener elementos identificativos de las imágenes. Este paso a mayores se conoce con el nombre de segmentación, y consiste en determinar una serie de atributos propios de cada elemento, como área, nivel de intensidad, etc.

3.6.6. Correspondencia estereoscópica

La correspondencia estereoscópica consiste en la obtención de las proyecciones de un punto de la escena en las imágenes del par estereocópico.

El proceso de correspondencia estereoscópica es el paso más complejo en la visión estereoscópica. El hecho de tener dos imágenes tomadas desde dos cámaras y desde posiciones o ángulos diferentes hace que la iluminación pueda ser diferente en las imágenes, pudiendo aparece reflejos. Además existe la posibilidad de que se produzcan oclusiones quedando ocultas partes de la escena en una imagen y visibles en la otra. También se ha de tener en cuenta que aunque las imágenes sean realizadas con cámars con las mismas características, el comportamiento de los componentes electrónicos y ópticos puede ser distinto.

Como ya se comento en el apartado anterior, existen varias técnicas de correspondencia estereoscópica que difieren en los elementos identificativos utilizados para lleva a cabo la correspondencia. Las dos grandes clases de técnicas son:

- Las técnicas basadas en el área(“area-based”), utilizan patrones de intensidad en los que engloban a la vecindad de un pixel de una de las imágenes del par estereoscópico

y los comparan con los patrones de intensidad correspondientes con un pixel de la otra imagen del par estereoscópico. Este tipo de técnicas se basan el la utilización de la intensidad de los píxeles.

La principal ventaja de este tipo de técnicas es la posibilidad de crear mapas de profundidad muy densos, esto se debe a que la correspondencia se realiza píxel a píxel, obteniendo un valor de disparidad para cada uno. También presentan una serie de desventajas como: el uso de la intensidad en cada pixel hace que sean más sensibles a modificaciones en la iluminación, y la posibilidad de obtener falsas correspondencias.

Algunas de las técnicas basadas en el área son:

- Suma de diferencias absolutas.
- Suma de diferencias al cuadrado.
- Correlación cruzada normalizada.
- Coeficiente de correlación de Pearson.

- Las técnicas basadas en las características("feature-based"), utilizan representaciones simbólicas obtenidas a partir de la imagen de intensidad. Algunas de estas representaciones simbólicas han sido citadas anteriormente, los puntos de borde aislados, cadenas de puntos de bordes, regiones delimitadas por bordes.

  Algunas de las principales ventajas que presentan estas técnicas son: estabilidad ante cambios en la iluminación, permiten realizar comparaciones entre atributos, mayor rapidez debida al procesamiento de menos puntos. La principal desventaja es la falta de densidad en los mapas de profundidad.

El proceso de correspondencia se puede dividir en dos partes, una local y una global. En la parte local se lleva a cabo una correspondencia de los valores de los atributos con el fin de determinar el valor de semejanza entre los vectores de atributos, estableciendo una correspondencia local. En la parte global se lleva a cabo una comprobación de las correspondencias locales. Ambas partes son llevadas a cabo aplicando una serie de restricciones.

3.6.7. Cálculo de la distancia

Para la determinación de las distancias a las que se encuentran los objetos en la escena se ha de realizar la correspondencia entre las imágenes izquierda y derecha. La forma de obtener las distancias esta ligada con la geometría del sistema estereoscópico. Debido a que en la gran mayoría de sistemas de visión estereoscópica se aplica la restricción de epipolaridad, el cálculo de las distancias es una simple triangulación.

Las proyecciones de un objeto en ambas imágenes, debido a la restricción de epipolaridad, tienen la misma posición en el eje vertical. Las imágenes presentan un desplazamiento horizontal que hace que las proyecciones de los objetos de la escena tengan posiciones en el eje horizontal ligeramente distintas. Esta diferencia en el posicionamiento en el eje horizontal es la disparidad. Todos los valores de disparidad de las imágenes del par estereoscópico se encuentran en el mapa de disparidades.

El paso de los valores de disparidades a distancias se realiza mediante una semejanza de triángulos. A continuación, nos fijamos en la figura 3.6, donde b es la linea base, f es la distancia focal, *P(x,y,z)* es un punto de la escena, P_I es la proyección de punto en la imagen izquierda, P_D es la proyección de punto en la imagen derecha, cada uno de los puntos tiene sus coordenadas correspondientes, (x_I, y_I) y (x_D, y_D) respectivamente.

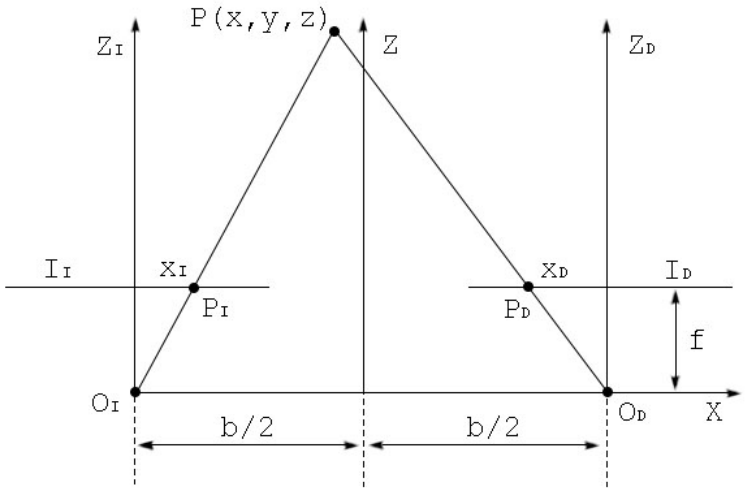


Figura 3.6: Perspectiva superior de sistema de visión estereoscópico con los ejes ópticos de las cámaras paralelos.

Aplicando la semejanza de triángulos para ambas proyecciones, se obtiene los valores de x_I, ecuación 3.1 y de x_D, ecuación 3.2. Sabiendo que la diferencia de estos valores se corresponden con la disparidad se obtiene la coordenada z del punto de la escena.

$$O_I : \frac{\frac{b}{2} + x}{z} = \frac{x_I}{f} \rightarrow x_I = \frac{f}{z}\left(x + \frac{b}{2}\right) \tag{3.1}$$

$$O_D : \frac{\frac{b}{2} - x}{z} = \frac{x_D}{f} \rightarrow x_D = \frac{f}{z}\left(x - \frac{b}{2}\right) \tag{3.2}$$

$$d = x_I - x_D = \frac{f}{z}\left(x + \frac{b}{2}\right) - \frac{f}{z}\left(x - \frac{b}{2}\right) = \frac{f}{z}\,b \rightarrow z = \frac{f}{d}\,b \tag{3.3}$$

Como se puede observar en la ecuación 3.3, la profundidad es inversamente proporcional a la disparidad, es decir, cuanto más cerca se encuentre un objeto en la escena mayor disparidad tendrá. Para llevar a cabo una mejora en la precisión de la medida de profundidad se ha de incrementar el valor de la linea base, haciendo que la disparidad aumente. Hay que tener especial precaución ya que puede ocurrir que debido al alto valor de disparidad apenas haya características comunes en ambas imágenes.

3.6.8. Interpolación

Como ya se comento anteriormente dependiendo de la aplicación del sistema estereoscópico se necesitan mapas de profundidad más o menos densos. La densidad de estos

mapas esta asociada al tipo de método, basado en el área o basado en las características. El método basado en el área proporciona un mapa de disparidades muy denso, haciendo innecesario llevar a cabo el proceso de interpolación. Sin embargo, cuando se utiliza el método basado en las características el mapa de disparidades que se obtiene presenta una densidad muy baja siendo necesario llevar a cabo una interpolación.

La interpolación se puede realizar de diversas formas, siendo las más comunes las dos siguientes:

- Mediante una función de profundidad continua. En este caso se considera el mapa de disparidades como el muestreo de una función de profundidad continua, y mediante la interpolación se hallaría la función continua que la aproxime. Permitiendo establecer una relación entre cualquier punto del espacio con un punto valor del mapa de disparidades. Algunos de los métodos de interpolación usados son: de Lagrange, de Hermite y mediante Splines.

- Mediante un modelo geomético. Este caso intenta ajustar el mapa de disparidades mediante un modelo geométrico calculado anteriormente.

3.6.9. Filtrado de mapas de disparidad

A la hora de obtener el mapa de disparidad mediante los métodos citados anteriormente cabe la posibilidad de encontrarnos con posibles errores. Estos se pueden deber a posibles errores en la toma de imágenes por parte del sistema estereoscópico, o directamente errores en el cálculo de los valores de disparidad. Para solucionar estos problemas se aplican una serie de filtros sobre el mapa de disparidades.

El tratamiento de los mapas de disparidad se realiza ya que se considera que la escena representada en ellos presenta una serie de relaciones. Es decir, un pixel propio de un objeto de la escena tendrá un valor de disparidad parecido al resto de píxeles pertenecientes al objeto. Los filtros más utilizados son:

- Filtro de la mediana. Este filtro se basa en la utilización de la mediana sobre una vecindad de píxeles para la determinación del valor de disparidad del pixel sobre el cual se esta aplicando el filtro. La determinación del valor correspondiente al valor de la mediana depende del si el número de valores comprendidos en el conjunto es par o impar.

  En ambos casos el primer paso es ordenar los valores en orden creciente o decreciente. En el caso de que el número de valores sea par el valor de la mediana es el valor del elemento que ocupe la posición intermedia en el conjunto de elementos ya ordenados, este valor se puede obtener aplicando la siguiente expresión: $M_e = x_{(n+1)/2}$. Sin embargo, si el número de valores es impar el valor de la mediana se expresa mediante: $M_e = \left(x_{n/2} + x_{n/2+1}\right)/2$, al no haber un valor intermedio se realiza la media de los dos valores centrales.

  Un ejemplo de este último caso sería el siguiente, se ha seleccionado una vecindad formada por nueve vecinos con los siguientes valores: *12,21,26,55,41,31,33,38,5,27*. El primer paso sería llevar a cabo una ordenación de estos valores, *5,12,21,26,27,31,33,38,41,55*.

El conjunto de píxeles anteriores es par de forma que aplicando la ecuación correspondiente se obtiene:

$$M_e = \frac{(x_{n/2} + x_{n/2+1})}{2} = \frac{x_5 + x_6}{2} = \frac{27 + 31}{2} = 29 \tag{3.4}$$

Con este tipo de filtro se consiguen eliminar valores erróneos ya que se asigna el valor intermedio de la vecindad. Para la aplicación de este filtro es recomendable tener una proporción mayor de píxeles correctos que incorrectos ya que podría darse la situación de dar un valor incorrecto a un pixel.

- Filtro de la media. Este filtro se basa en el cálculo del valor medio aritmético de una vecindad de píxeles para establecer el valor del pixel sobre el cual se esta aplicando el filtro. El valor medio aritmético se obtiene utilizando la expresión 3.5.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3.5}$$

Utilizando el mismo conjunto de datos que en el ejemplo del filtro de la mediana y aplicando la expresión 3.5 se obtiene:

$$\bar{x} = \frac{12 + 21 + 26 + 55 + 41 + 31 + 33 + 38 + 5 + 27}{10} = \frac{289}{10} = 28{,}9$$

3.7. OpenCV

El 13 de Junio del 2000, Intel Corporation anunció que estaba trabajando con un grupo de reconocidos investigadores en visión por computador para realizar una nueva librería de estructuras/funciones en lenguaje C. Esta librería proporcionaría un marco de trabajo de nivel medio-alto que ayudaría al personal docente e investigador a desarrollar nuevas formas de interactuar con los ordenadores. Este anuncio tuvo lugar en la apertura del IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Había nacido The Open Computer Vision Library y lo hacía bajo licencia BSD (Software Libre).

La librería OpenCV es una API de aproximadamente 300 funciones escritas en lenguaje C que se caracterizan por lo siguiente:

- Su uso es libre tanto para su uso comercial como no comercial.
- No utiliza librerías numéricas externas, aunque puede hacer uso de alguna de ellas, si están disponibles, en tiempo de ejecución.
- Es compatible con The Intel® Processing Library (IPL) y utiliza The Intel® Integrated Performance Primitives (IPP) para mejorar su rendimiento, si están disponibles en el sistema.
- Dispone de interfaces para algunos otros lenguajes y entornos: EiC - intérprete ANSI C escrito por Ed Breen. Hawk y CvEnv son entornos interactivos (escritos en MFC y TCL, respectivamente) que utilizan el intérprete EiC; Ch - intérprete ANSI C/C++ creado y soportado por la compañía SoftIntegration; Matlab® - gran entorno para el cálculo numérico y simbólico creado por Mathworks; y muchos más.

3.7.1. Estructura y características

La librería OpenCV esta dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus muchas áreas de aplicación destacarían: interacción hombre-máquina (HCI); segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura del movimiento (SFM); y robots móviles.

La librería OpenCV proporciona numerosos elementos de alto nivel, como veremos en secciones posteriores, que facilitan sobre manera el trabajo al usuario (tanto al docente como al investigador). Por ejemplo, proporciona filtros Microsoft DirectShow para realizar tareas tales como: calibración de la cámara, seguidores de objetos (Kalman tracker y ConDensation tracker), etc. Todos ellos se pueden utilizar en Microsoft GraphEdit para ilustrar de forma bastante sencilla numerosas aplicaciones de visión.

De igual forma, la librería OpenCV proporciona numerosas aplicaciones de ejemplo que ilustran como emplear las distintas funciones de la librería.

También los entornos de scripting hacen uso de estas funciones para implementar su funcionalidad. La librería OpenCV proporciona una gran diversidad de entornos. Todas estas herramientas de alto nivel hacen uso de un paquete de clases C++ y funciones C de alto nivel que utilizan a su vez funciones muy eficientes escritas en C. Concretamente, el conjunto de funciones suministradas por la librería OpenCV se agrupan en los siguientes bloques:

- Estructuras y operaciones básicas: matrices, grafos, árboles, etc.
- Procesamiento y análisis de imágenes: filtros, momentos, histogramas, etc.
- Análisis estructural: geometría, procesamiento del contorno, etc.
- Análisis del movimiento y seguimiento de objetos: plantillas de movimiento, seguidores (i.e. Lucas-Kanade), flujo óptico, etc.
- Reconocimiento de objetos: objetos propios (eigen objects), modelos HMM, etc.
- Calibración de la cámara: morphing, geometría epipolar, estimación de la pose (i.e. POSIT), etc.
- Reconstrucción tridimensional (funcionalidad experimental): detección de objetos, seguimiento de objetos tridimensionales, etc.
- Interfaces gráficos de usuarios y adquisición de vídeo.

3.7.2. Interfaces gráficos y herramientas

La librería OpenCV proporciona varios paquetes de alto nivel para el desarrollo de aplicaciones de visión. Todos ellos se pueden agrupar en librerías de C/C++ dirigidas a usuarios avanzados y en herramientas de scripting dirigidas, en este caso, a usuarios de nivel medio (ideal para practicar con las distintas técnicas de procesamiento de imágenes y visión). Al primer grupo pertenecen HighGUI y CvCam, mientras que al segundo pertenecen Hawk y OpenCV Toolbox para Matlab.

HighGUI permite la escritura/lectura de imágenes en numerosos formatos (BMP, JPEG, TIFF, PxM, SunRaster, etc.) y la captura de stream de video de capturadoras Matrox® y cámaras/capturadoras con drivers VFW/WDM (la mayoría del mercado); la creación de ventanas para visualizar imágenes en ellas, las ventanas HighGUI recuerdan su contenido (no es necesario implementar callbacks de repintado); y además, nos proporciona mecanismos muy fáciles de interaccionar con ellas: trackbars, capturando la entrada del teclado y el ratón.

CvCam nos proporciona un único interfaz de captura y reproducción bajo Linux y Win32; callbacks para la gestión de stream de vídeo o ficheros AVI y un mecanismo fácil para implementar visión estéreo con dos cámaras USB o una estéreo-cámara.

Hawk es un entorno visual con el intérprete ANSI C EiC como núcleo; soporta plugins; proporciona soporte para OpenCV,IPL y HighGUI vía plugin; y soporte de video.

Por último, la librería OpenCV proporciona un toolbox para Matlab que se caracteriza por lo siguiente:

- Utiliza tipos nativos de Matlab (matrices, estructuras).
- Compatibilidad con la Image Processing Toolbox.

3.7.3. Aplicaciones

Como hemos detallado en secciones previas, la librería OpenCV nos proporciona el marco de trabajo ideal, tanto por el desarrollo de pequeñas aplicaciones prácticas, ideales como apoyo a la docencia en asignaturas relacionadas con la visión por computador, como para el desarrollo de la labor investigadora de cualquier grupo de visión.

Una de las herramientas más utilizadas a nivel académico es el paquete matemático Matlab. Este paquete se utiliza en numerosas asignaturas como apoyo a la docencia y es ampliamente conocido por el mundo universitario. Matlab dispone de un toolbox denominado Image Processing Toolbox que permite cargar imágenes en distintos formatos, visualizarlas y manejarlas como matrices numéricas. Sin embargo, carece de muchas de las características que debería cumplir cualquier paquete de visión por computador que se precie. La librería OpenCV proporciona un toolbox que aprovecha muchos de los tipos de datos, estructuras y en general, la potencia que este paquete ofrece, para facilitar al usuario de nivel medio-bajo el acercamiento a muchas de las tareas habituales de la visión por computador.

Otro de los lenguajes más utilizados en el ámbito universitario es el lenguaje C. La librería OpenCV también proporciona, como ya hemos visto en secciones anteriores, numerosas librerías de apoyo que nos permiten implementar de forma sencilla y rápida aplicaciones de visión, obviando aspectos tales como las características de la cámara, el formato de las imágenes, etc.

Capítulo 4

Introducción a la cirugía laparoscópica

La cirugía laparoscópica o "mínimamente invasiva" es una técnica especializada para la realización de cirugías. Anteriormente, esta técnica se usaba por lo general para cirugía ginecológica y de vesícula biliar. Durante los últimos 10 años, el uso de esta técnica se ha ampliado e incluye la cirugía intestinal. En la cirugía tradicional "abierta", el cirujano usa una sola incisión para entrar al abdomen. La cirugía laparoscópica usa varias incisiones de 0.5 a 1 cm. Cada incisión se denomina "puerto". En cada puerto se inserta un instrumento tubular conocido como trocar. Durante el procedimiento, a través de los trocares se pasan instrumentos especializados y una cámara especial llamada laparoscopio. Al iniciar el procedimiento, el abdomen se infla con el gas llamado dióxido de carbono para proporcionar al cirujano un espacio de trabajo y visibilidad. El laparoscopio transmite imágenes de la cavidad abdominal a los monitores de vídeo de alta resolución del quirófano. Durante la operación, el cirujano observa las imágenes detalladas del abdomen en el monitor. El sistema permite que el cirujano realice las mismas operaciones que la cirugía tradicional pero con incisiones más pequeñas.

En ciertos casos, el cirujano puede elegir usar un tipo de puerto especial que es lo suficientemente amplio como para insertar una mano. Cuando se usa un puerto para mano, la técnica quirúrgica se llama laparoscopia "asistida con la mano". La incisión necesaria para un puerto para la mano es más grande que las demás incisiones de laparoscopia, pero es normalmente más pequeña que la incisión para una cirugía tradicional.

Las publicaciones y estudios realizados en las dos últimas décadas demuestran que la cirugía laparoscópica constituye uno de los mayores avances tecnológicos de finales del siglo XX y principios de siglo XXI. La aplicación de la cirugía de puerto único, la cirugía por orificios naturales y la cirugía robótica, es la expresión del desarrollo ilimitado de la mínima invasión.

El término laparoscopia proviene del griego laparo- (flanco) y –skopia (instrumento de observación) y se utiliza para describir el procedimiento mediante el cual se examina el peritoneo con un endoscopio. La Real Academia de la Lengua Española define laparoscopia (o laparoscopía) como "una técnica de exploración visual que permite observar la cavidad pélvica-abdominal con un instrumento conocido como laparoscopio".

4.1. Evolución

Algunos historiadores atribuyen al cirujano árabe Albukasim (936-1013 d.C) la primera revisión de una cavidad interna, al emplear el reflejo de la luz mediante espejos para examinar el cuello uterino [25].

Abul I Qasim Jalaf ibn al-Abbas al-Zahrawi, conocido como Abulcasim, nace en el año 936 en Madinat-al-Zahra, Córdoba, y muere en el año 1013. Es considerado por muchos el primer cirujano de la historia. Se saben muy pocas cosas de su vida mas allá de las que explica en sus propias obras. Su nombre aparece mencionado por primera vez en una obra de Abu Muhammad ibn Hazm, citándolo como uno de los médicos más famosos de Al-Ándalus. La primera biografía detallada se escribió sesenta años después de su muerte por Al-Humaydi, en su obra Jadhwat al-Muqtabis.

Escribió la enciclopedia Kitab Al-Tasrif, que posee gran importancia y fama en el mundo de la medicina, donde recopila la mayoría de los procedimientos médicos y farmacéuticos de la época en 30 volúmenes, y donde se ilustran los principales elementos quirúrgicos medievales [17].

En la Edad Media, esta corriente de medicina islámica fue seguida por muchos cirujanos europeos. Describe multitud de técnicas quirúrgicas, como por ejemplo, sobre la cauterización de las heridas, extracción de cálculos de la vejiga, o la posibilidad de utilizar el fórceps en la extracción del feto. Fue el primero en emplear el hilo de seda en las suturas [6].

No obstante, hasta que no transcurrieron otros siete siglos no comenzaron a desarrollarse avances significativos en la técnica de la laparoscopia. En el año 1804, col la introducción del conductor de luz a cargo de Philip Bozzini (1773-1809) se sentaron las bases para el posterior desarrollo de lo que actualmente conocemos como endoscopio. Dicha fuente de luz estaba formada por una vela que reflejaba el rayo luminoso en un espejo para visualizar los órganos de las distintas cavidades de cuerpo humano.

Fue Desormeaux en el año 1863 quien desarrolló el primertubo de endoscopio que funcionaba de una manera aceptable, y posteriormente Maxomiolian Nitze y Thomas Edison, que introdujeron respectivamente lentes de visualización y una pequeña bombilla, lograron mejorar en gran medida la capacidad iluminativa de dicho sistema. No obstante, la utilización de bombillas incandescentes por parte de Edison provocaba ciertos problemas de abrasión por sobrecalentamiento de las mismas [35].

Los dos pioneros de la laparoscopia fueron, Georg Kelling (1866-1945), cirujano alemán de Dresden y Hans Christian Jacobaeus (1879-1937), un médico internista de Estocolmo, Suecia.

Aunqie Kelling se consideraba a sí mismo como cirujano, él invirtió mucha energía y tiempo para desarrolla métodos de tratamiento no qurúrgicos o "de invasión mínima". Realizó cientos de experimentos en cadáveres y animales, midiendo la capacidad gástrica. En 1890, en contra del escepticismo de sus compañeros de profesión, Kelling diseñó un esofagoscopio, combinando un tubo rígido proximal con un sistema óptico flexible y distal, diseñado previamente por Nitze.

Georg Kelling pasa el verano de 1898 bajo la tutoría del famoso cirujano e inventor del esofagoscopio y gastroscopio clínicos, Mikulicz. Allí Kelling mejora sus técnivas endoscópicass y realiza varias pruebas usando su sistema de tubo endoscopio semiflexible. Kelling también trabaja junto con el cirujano Vitezlav Chlumsky (1867-1943). Ambos practicaron insuflaciones gastrointestinales de alta presión para examinar la eficacia de las gastroenteroanastomosis, procedimiento después perfeccionado por Mikulicz. Estos estudios demostraron que las enteroanastomosis entre el estomago y el intestino delgado podía resistir elevadas presiones intraluminales.

Años después G.Kelling enfocó su atención a las hemorragias gastrointestinales dentro de la cavidad abdominal, las cuales eran fatales para la mayoría de los pacientes en aquellos días. Las gastroenterorragias eran particularmente muy peligrosas, y no siempre manifestaban por hematemesis o melena. El único método disponible para establecer el diagnóstico de estas hemorragias internas era por medio de la laparotomía exploradora; sin embargo, como Kelling observó, el abrir el abdomen de los paciente podría empeorar su estado. Para detener este sangrado intra-abdomina, Kelling propuso un tratamiento no quirúrgico: la insuflación de gas intraperitoneal a alta presión, técnica que él llamó "lufttamponade".

En 1901, Kelling basándose en sus estudios de hiperinflación peritoneal calculó que una presión intraabdominal de 50 mmHg podía detener el sangrado. Kelling realizón numerosos experimentos en perros vivos, produciendo insuflaciones de hasta 100 mmHg; 2 de sus 20 perros murieron directamente por esta alta presión intraabdominal.

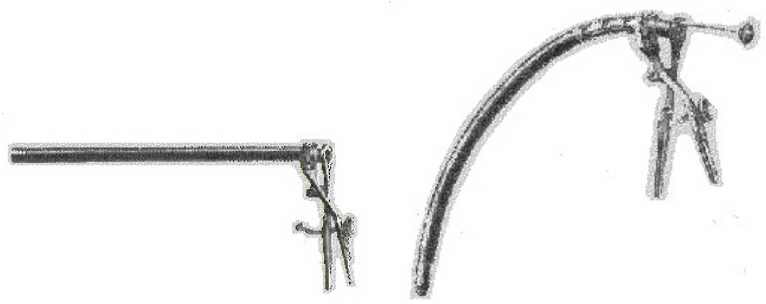
Por aquellos años la patofisiología de neumoperitoneo era desconocida y Kelling proclamó a su procedimiento como totalmente inocuo. Kelling también quiso observar el efecto de la insuflación de la cavidad abdominal en los órganos abdominales, para observar el comportamiento de estos cuando se introducía aire. Para encontrar la respuesta ideo un método introduciendo un endoscopio en la cavidad abdominal cerrada (celioscopía). Para visualizar los efectos del "lufttamponade" a altas presiones en los órganos intraabdominales, Kelling introdujo un cistoscopio de Nitze directamente a través de la pared abdominal. Observó que los órganos disminuían su tamaño y tomaban un color más pálido.

Georg Kelling continuó con su trabajo en la insuflación gástica, pero dejando de lado las celioscopías y no publicó nada en relación al "lufttamponade" o la endoscopía de una cavidad abdominal cerrada. Se infiere que, la celioscopía de Kelling fue creada como un método adicional para poder ver los efectos del "luftmaponade.[en] los órganos intraabdominales, y no como una técnica endoscópica per se.

La primera referencia sobre el uso de la técnica laparoscópica aparece en una publicación de Bernheim, de 1911, quien con una cabeza de lámpara eléctrica y un proctoscopio fue capaz de ver el estómago, la vesícula biliar y el hñigado de su paciente [7].

Desconociendo el trabajo previo de G.Kelling, Hans Christian Jacobaeus (1879-1937), un médico internista sueco, recopiló información a cerca de sus experiencias con la técnica de laparotorascopía en humanos en 1910. Desafortunadamente, hay muy poco detalle informativo de sus experiencias iniciales con la endoscopía abdominal. Antes de iniciar su trabajo con las torascopías, él ya conocía el neumotórax artificial y el tratamiento de las

peritonitis tuberculosa con neumoperitoneo.



(a) Antes de su inserción. (b) Después de su inserción.

Figura 4.1: Esofagoscopio creado por G.Kelling.

Jacobaeus tituló su primer trabajo sobre endoscopía abdominal En relación a la posibilidad de aplicar el citoscopio en el examen de las cavidad serosas. Este internistasueco evacuó liquido ascitis y creó el neumoperitoneo usando un trócar con una válvula unidireccional. La primera experiencia de Jaavobaeus con este procedimiento fue básicamente restringida a pacientes con ascitis; él sólo examinó por laparoscopía a 2 pacientes sin ascitis.

La siguiente contribución de gran importancia fue realizada por el ginecólogo e ingeniero Kurt Semm, que en 1960 consiguió dar solución al problema de la presión abdominal, al diseñar un insuflador que registra la presión del gas intraabdominal y mide el flujo de inyección, y en 1964 introdujo la luz fría, que no solo permitía una mejor visión, sino que conseguía eliminar el riesgo de quemaduras por sobrecalentamiento anteriormente mencionados. Varios años después también introdujo el gancho de disección y coagulación y el cable de fibra óptica, que aún es usado en la actualidad. Por todos estos avances, Semm es considerado el padre de la cirugía laparoscópica moderna [28].

En la década de los ochenta se incorporó el vídeo a las endoscopias. El dispositivo de carga acoplada (charged couple device, CCD) se introduce en un endoscopio clínico. El CCD es un sensor con diminutas células fotoeléctricas que registran la imagen, que es posteriormente procesada por una cámara. Esta técnica fue presentada en 1989 en el Congreso del American College en Atlanta. En pocos meses, la técnica se popularizó a nivel mundial y se comenzó a investigar con mayor exigencia el ámbito de la cirugía laparoscópica, que se vio amparada por el desarrollo de nuevas formas de sutura, la aparición de un instrumental mucho más variado que el de los años anteriores, y una gran imaginación quirúrgica por parte de grandes nombres de la medicina como Cuschieri, Dubois, Katkhouda, Mouiel, o Zucker.

A partir de los años noventa, con el enorme desarrollo en la tecnología del video y la transmisión de imágenes, la cirugía laparoscópica se comienza a desarrollar entra las diferentes especialidades quirúrgicas que comparten el tratamiento del abdomen(cirugía, ginecología y urología).

Las ventajas convencen a los detractores iniciales de la cirugía laparoscópica ya que proporciona una mejor visión y abordaje de la cavidad abdominal, disminuye el dolor postoperatorio, redice la tasa de infección de la herida quirúrgica y las complicaciones

infecciosas, acorta el tiempo de estancia hospitalaria, disminuye el desarrollo de adherencias y de complicaciones obstructivas, permite la rápida reincorporación a la vida social y laboral, brinda un mejor resultado estético, reduce el coste total de la enfermedad y previene la hernia incisional [3].

Las imágenes electrónicas obtenidas de los videolaparoscopios actuales son de una calidad óptima, en alta definición e incluso visión tridimensional, lo que favorece considerablemente las maniobras terapéuticas. Además, facilitan la enseñanza y la obtención de información gráfica.

Aunque la técnica de la laparoscopia siga presentando cierta posibilidad de que aparezcan complicaciones, no se puede negar que es una técnica segura y fiable. Después de varios años puesta en práctica, se ha demostrado que los resultados obtenidos son, en muchos casos, mejores que los proporcionados por la cirugía abierta.

A partir del año 2000 surge la aplicación de la tecnología robótica a la cirugía laparoscópica, aportando ventajas a las limitaciones de esta, como la pérdida de la sensación táctil, limitaciones en la maniobrabilidad, imagen bidimensional, particularidades del instrumental, complejidad de los procedimientos y necesidad de neumoperitoneo [38].

La calidad de la imagen tridimensional intraoperatoria con sensación de profundidad, la perfecta sincronización manos-ojos, la precisión de los instrumentos, la exactitud de sus suturas y la exéresis que se realiza con mayor destreza y confort, hará probablemente que pronto la cirugía robótica sea ampliamente difundida a pesar de su elevado coste [5].

En cirugía laparoscópica serán necesarios los robots quirúrgicos para las tareas que requieran una gran precisión. Indudablemente, a medida que avanza la tecnología van desapareciendo las limitaciones de los sistemas actuales, de manera que se producirá una aceptación y mejora cada vez mayores, con disminución de los costes económicos de inversión y mantenimiento.

En la actualidad aparecen nuevos intentos de acceder a la zona quirúrgica con la mínima invasión y han aparecido nuevos conceptos, como la cirugía endoscópica transluminal por orificios naturales (NOTES, Natural Orifice Translumenal Endoscopic Surgery), la cirugía laparoscópica a través de incisión única (SILS, single Incision Laparoscopic Surgery) y la cirugía transanal a través de puerto único (TAMIS, cirugía transanal mínimamente invasiva) [11].

La tendencia es avanzar cada vez más hacia técnicas menos invasivas, intentando reproducir una cirugía más segura para el paciente y más cómoda para el cirujano. La cirugía laparoscópica se consideró el principal paradigma de la cirugía en las décadas de los ochenta y de los noventa, pero no es el final de la escalada para ofrecer un método menos invasivo al paciente quirúrgico. Es posiblemente un peldaño más de los ingentes esfuerzos que hace la comunidad médica para alcanzar este objetivo.

4.2. Cirugía laparoscópica asistida con la mano (CLAM)

Las técnicas laparoscópicas presentan ciertas limitaciones que han jugado un papel importante en el desarrollo de estas, algunas de las limitaciones son:

- Capacidad restringida de manipulación del espécimen enfermo.
- La reducida retroalimentación táctil asociada con un abordaje totalmente laparoscópico.
- La visualización completa del campo quirúrgico puede no ser posible utilizando la videolaparoscopia.
- La larga duración de la cirugía laparoscópica.
- La preocupación por la seguridad de los pacientes.

Estas restricciones de la cirugía laparoscópica estándar ayudaron a instigar el desarrollo de la cirugía asistida laparoscópicamente (LAS) para el tracto gastrointestinal. Después del examen laparoscópico inicial y la preparación, la cirugía laparoscópica asistida utiliza una pequeña incisión hecha sobre la posición de la patología intestinal. El intestino involucrado se administra a través de esta incisión y se realiza la escisión del segmento enfermo. La unión del intestino restante se realiza de forma extracorpórea con técnicas estándar y visión estereoscópica normal. Una vez que el intestino es unido vuelve a la cavidad abdominal y se cierra la incisión. Una desventaja de l método LAS es que el neumoperitoneo se pierde cuando se abre la incisión para llevar a cabo la resección. Después de la reanastomosis, se debe cerrar la incisión y restablecer el neumoperitoneo para completar la operación.

La cirugía laparoscópica asistida manualmente (CLAM), en ingles hand-assisted laparoscopy surgery (HALS) es una técnica algo diferente que la cirugía laparoscópica asistida. Con HALS, se utiliza un aparato para mantener el neumoperitoneo mientra se inserta la mano del cirujano a través de una pequeña incisión en el abdomen. Como en la cirugía laparoscópica estándar, el cirujano visualiza la zona mediante un monitor, pero además tiene la ventaja que implica el uso de su mano. La mano posee 7 grados de libertad que proporcionan un posicionamiento completo en el espacio. Es importante destacar que el pulgar y el dedo indice de la mano intracavitaria pueden ser utilizados para asegurar la hemostasia en el caso de una hemorragia intraoperatoria. Debido a que la CLAM permite el mantenimiento de la sensación táctil y promueve un grado de coordinación mano-ojo, esta variación en la cirugía laparoscópica ha sido más fácil de dominar para los cirujanos entrenados exclusivamente en la cirugía abierta.

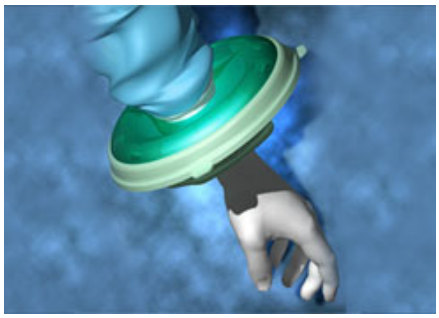


Figura 4.2: Dispositivo para eliminar posibles perdidas de neumoperitoneo, conocido con el nombre de GelPort.

Debido a lo comentado anteriormente, las técnicas quirúrgicas laparoscópicas asistidas manualmente tienen el potencial de:

- Facilitar la cirugía laparoscópica.
- Reducir el tiempo de la operación.
- Acortar la curva de aprendizaje asociada con los procedimientos quirúrgicos laparoscópicos.
- Mejorar la seguridad.

- Permiten una disección precisa de los especimenes sobre los que se trabaja.

La experiencia inicial con HALS se ha limitado a unos pocos centros importantes, y las investigaciones acerca de la correcta implementación de esta técnica ha sido bastante reducida. Existe la necesidad de investigar las cuestiones básicas de los procedimientos asistidos manualmente y establecer métodos de capacitación par evitar el desarrollo errático que caracterizó la cirugía laparoscópica en sus comienzos. Se requiere nuevos concepos de diseño para la instrumentación de estas técnicas y es probable que muchos de los dispositivos necesitan extensas investigaciones y gastos de capital. Es necesario elaborar estrategias para optimizar las técnicas de laparoscopia asistida manualmente.

Por ejemplo, los problemas aparentemente simples, como la altura apropiada de la mesa de operaciones y la orientación de esta, necesitan ser estudiados porque la configuración de la sala de operaciones es diferente para HALS que para la cirugía laparoscópica estándar o para cirugía abierta clásica. Además, el hecho de que el cirujano tenga una mano fuera del abdomen y otra dentro puede ser una posición incómoda y contribuir a la fatiga del operador durante prodecimientos operativos largos y complejos. También necesita ser aclarado si la mano de ayuda es la mano del cirujano operante o la mano del cirujano asistente.

De acuerdo con los primeros informes, el lugar de la mano auxiliar debe ser considerado como un puerto de la operación y triangulado con el otro puerto de la operación laparoscópica de manera que abos puertos formen ángulos de azimut iguales con el puerto de visualización laparoscópica. El posicionamiento de los puertos de la operación en ángulos iguales permite al cirujano dirigirse ergonómicamente al órgano objetivo. Sin embargo, si la mano auxiliar está demasiado cerca del órgano objetivo, puede dificultar la visión de este órgano y dificultar los movimientos operativos. S la mano auxiliar está demasiado lejos del órgano objetivo, la fatiga de la mano puede convertirse en un factor significativo en la realización segura del procedimiento. Además, es necesario examinar el riesgo potencial de causar daños a otros órganos durante la operación.

Una revisión preliminar de la cirugía laparoscópica asistida por la mano sugiere la necesidad de investigar los siguientes aspectos:

- Dispositivos o técnicas para facilitar la retirada de la mano operativa o la inserción de instrumentos en el abdomen.
- Desarrollo de instrumentos manuales o activados por los dedos.
- Dispositivos para permitir el cambio de la mano intracavitaria de derecha a izquierda, o viceversa, expedita y segura.
- Creación de maniaobras o isntrumentos efectivos para el control de la hemorragia intraoperatoria mayor.
- Diseño de un sellado cómodo alrededor de la mano intracavitaria para minimizar la fatiga muscular, entumecimiento e hinchazón.
- Optimización del alcance efectivo de la mano intracavitaria.

Debido a que la tecnología laparoscópica asistida a mano está en sus comienzos, existe la necesidad urgente de desarrollar estudios adecuados para explorar su potencia. Además, sólo unos pocos instrumentos operativos estándar son adecuados para la CLAM, y se

necesita desarrollar una nueva instrumentación para esta técnica. Debe desarrollarse un ambiente ergonómico para la práctica de la cirugía laparoscópica manual. Por último, se necesitan nuevos métodos para promover la educación quirúrgica y difundir el conocimiento de las capacidades de la cirugía laparoscópica asistida manualmente.

Capítulo 5

Desarrollo del sistema de visión estereoscópico

En el presente trabajo de fin de grado se ha realizado un sistema de visión artificial, para la implementación en operaciones de cirugía laparoscópica asistida mediante mano. El sistema se puede dividir en dos partes. La primera parte consta de una serie de programas cuya función es calcular los valores de las variables con los que se va a trabajar. La segunda parte se corresponde con el programa principal donde se utilizan las variables calculadas en la primera parte. Los programas pertenecientes a la primera parte en su mayoría son independientes, de forma que no necesitarían la ejecución de resto para funcionar correctamente, permitiendo que el usuario pueda utilizar alguno de ellos de forma individual y fuera de este sistema. La fase del desarrollo en la que se encuentra el sistema de visión artificial ha motivado la forma de realizar el sistema.

El objetivo de este capítulo es explicar las diferentes opciones disponibles a la hora de realizar cada uno de los pasos necesarios para el desarrollo del sistema y los motivos para la selección de la opción implementada. Una de las bases sobre las que se ha desarrollado el sistema es la visión estereoscópica, esta ha sido explicada en un capítulo previo donde también se han explicado una serie de pasos para el desarrollo de sistemas de visión estereoscópica. Estos pasos y alguno más, añadido de forma complementaria pero necesario para conseguir alcanzar el objetivo del sistema, han sido los seguidos en el desarrollo del sistema.

5.1. Materiales

Antes de explicar los pasos seguidos para el desarrollo de la aplicación, es necesario explicar los materias disponibles y sus características debido a la posible influencia de estos tanto en el desarrollo del sistema como en el resultado final. Para la realización de este trabajo de fin de grado, ha sido necesario la utilización un ordenador con las librerías de openCV instaladas y la utilización de los siguientes materiales.

5.1.1. Cámaras Logitech C310

Se han utilizado dos cámaras iguales, modelo C310, de la marca Logitech. El posicionamiento de estas cámaras para la toma de imágenes se puede apreciar en la figura 5.1. Ambas están colocadas en un soporte metálico, al que están unidas mediante bridas. Esto

hace que estén alineadas en el eje z y eje y. La distancia entre los centros de las lentes de ambas cámaras es de 86 mm.

Las especificaciones de las cámaras son:

- Sistemas operativos compatibles
  - Windows Vista.
  - Windows 7 (32bits o 64bits).
  - Windows 8.
  - Windows 10.
- Requisitos mínimos
  - 1 GHz.
  - 512 MB de memoria RAM.
  - 200 MB de espacio en el disco duro.
  - Conexión a internet.
  - Puerto USB 1.1.
- Especificaciones técnicas
  - Videoconferencias HD (1280 x 720 pixeles) con el sistema recomendado.
  - Captura de vídeo HD: Hasta 1280 x 720 píxeles.
  - Tecnología Logitech Fluid Crystal^{TM}.
  - Fotos: Hasta 5 megapíxeles (mejora por software).
  - Micrófono integrado con reducción de ruido.
  - Certificación USB 2.0 de alta velocidad (recomendable).
  - Clip universal para monitores LCD, CRT o portátiles.
- Software de cámara Web Logitech
  - Controles panorámicos, inclinación y zoom.
  - Captura de vídeo y fotos.
  - Seguimiento facial.
  - Detección de movimiento.

Figura 5.1: Cámaras C310.

5.1.2. Guantes

Para conocer la situación espacial de la mano dentro de la cavidad torácica es necesario poder discriminar la mano del cirujano del resto de elementos que se encuentren a su alrededor. Los guantes utilizados por los cirujanos normalmente son de látex, véase figura 5.2(a), o de nitrilo, véase figura 5.2(b). Estos dos tipos de guantes han sido utilizados para el desarrollo del sistema. También se ha probado a utilizar un tercer guante de color negro, véase figura 5.2(c), al cual se le han recortado los dedos pulgar, indice y corazón, quedando visibles.

(a) Guante de latex. (b) Guante de nitrilo. (c) Guante recortado.

Figura 5.2: Guantes utilizados en el desarrollo del sistema de visión artificial.

Durante el desarrollo del sistema se planteó la posibilidad de añadir un patrón que permitiese la identificación de las diferentes partes de la mano, permitiendo una rápida y fácil identificación de la posición o gesto realizado por la mano. Esta posibilidad se tuvo que descartar por la baja probabilidad de éxito, ya que los guantes durante las cirugías se manchan de sangre ocultando parcial o totalmente el patrón, haciendo imposible al identificación de estos.

5.1.3. Patrón de calibración

Como ya se ha explicado en capítulos anteriores las lentes pueden tener imperfecciones. Cuando se habla de imperfecciones se hace referencia a la forma de las lentes, siendo esféricas en vez de parabólicas. Las imperfecciones provocan distorsiones en la imagen obtenida, siendo aconsejable la eliminación de las distorsiones antes de trabajar con la imagen. Otro aspecto importante es el posicionamiento de las cámaras como ya se comentó anteriormente, para hacer que coincidan las lineas epipolares de forma que los objetos de la escena tengan la misma coordenada en el eje vertical.

La corrección de distorsiones y rectificación se realiza mediante un programa, el cual mediante una imagen patrón realiza la rectificación y corrección de distorsiones. La imagen patrón utilizada es una cuadrícula de color blanco y negro, igual a un tablero de ajedrez, véase figura 5.3.

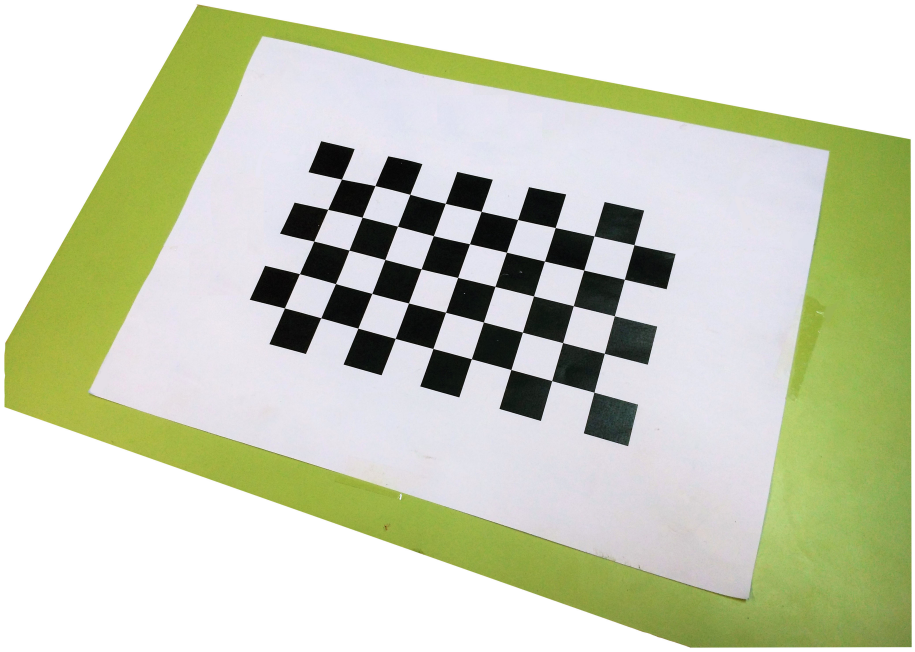


Figura 5.3: Imagen patrón utilizada por el programa.

Se lleva a cabo la toma de diferentes imágenes desde posiciones y ángulos distintos, siendo el programa capaz de identificar las esquinas de los cuadros interiores y generar los valores de los parámetros necesarios para corregir las imágenes.

5.2. Toma de imágenes

Las posibles soluciones para realizar la toma de imágenes son dos: la utilización de una sola cámara o la utilización de dos o más cámaras.

Para tomar imágenes con una sola cámara, esta se ha de desplazar en torno a la escena, y el desplazamiento ha de ser controlado ya que es necesario conocer la posición desde la que se ha realizado la fotografía. El sistema de visión artificial en desarrollo se va a fijar a un brazo robótico, permitiendo el movimiento de este por el interior de la cavidad abdominal del paciente. Por lo que la realización de movimientos y el posicionamiento durante la toma de imágenes no supondría ningún problema.

La toma de imágenes mediante dos o más cámaras situadas en posiciones conocidas permite no tener que realizar ningún movimiento, reduciendo el tiempo de toma de imágenes y la complejidad. Esta opción se emplea en aquellas aplicaciones en las que los resultados del procesamiento de las imágenes han de obtenerse en el menor tiempo posible. Uno de los principales inconvenientes de este método es el tamaño debido al uso de dos o más cámaras. Con el avance de la tecnología las dimensiones de las cámaras se han reducido considerablemente por lo que las dimensiones de estos sistemas han dejado de ser un inconveniente.

Esta aplicación realiza la captura de imágenes mediante dos cámaras situadas en distintas coordenadas, pero con los ejes x y z alineados. Los motivos que han llevado a implementar esta solución son los siguientes:

- Simplicidad. La implementación de este sistema no requiere del movimiento de cámaras. Tan solo se necesita conocer la posición de una respecto de la otra.

- Facilidad del cálculo de distancias. El posicionamiento de las cámaras con los ejes x y z alineados permite el cálculo de las distancias muy fácilmente. Esto se debe a que el modelo geométrico del sistema, en el que las disparidades entre las imágenes solamente se encuentran en los ejes x, es muy sencillo.

- Rapidez. El hecho de que sea un sistema de visión artificial cuyo objetivo es la cooperación con el ser humano en cirugías laparoscópicas asistidas mediante mano, proporcionando una fuente de luz que se ha de mover de forma coordinada con los movimientos de la mano del cirujano hace que sea de gran importancia la rapidez del sistema. Cuanto más rápida sea la respuesta del sistema ante un movimiento del cirujano mejor será el sistema. La toma de imágenes mediante dos cámaras es menor, favoreciendo la velocidad del sistema.

5.2.1. Estructura del código de la toma de imágenes

El código utilizado para la toma de imágenes ha sido necesario implementarlo en todos los programas realizados y siempre es el mismo. La explicación del código correspondiente a la toma de imágenes es irrelevante para el usuario pero se ha considerado necesario ya que cabe la posibilidad de utilizar este código como base para el desarrollo de otros sistemas de visión artificial o la mejora del mismo. Este código se puede dividir en los siguientes pasos:

1. Declaración de las cámaras. Cuando se conectan varias cámaras a un ordenador, estas quedan determinadas mediante un número dado por el propio sistema operativo, en este paso se establece que número representa a cada cámara declarando dos variables, una para cada cámara. Es importante que la cámara que se encuentra a la derecha corresponda con la cámara derecha del sistema y de igual forma para la cámara izquierda. Si esto no ocurre los resultados obtenidos en la ejecución de los siguiente programas no serán correctos. Para que las cámaras se configuren de forma correcta, el usuario tiene que conectar primero la cámara izquierda y posteriormente la cámara derecha, de esta forma quedaran asignadas directamente con los valores asociados por defecto. Los valores asignados por defecto se encuentran definidos en un archivo, *config.h*, mediante las variables: *CAM_IZDA* y *CAM_DCHA*.

2. Configuración del tamaño de las imágenes. Se determina el tamaño de las imágenes que se van a obtener mediante las cámaras con el objetivo de eliminar la posible aparición de problemas cuando se trabaje sobre las dos imágenes. El tamaño de las imágenes esta establecido en un archivo de configuración, *config.h*, mediante las variables: *FRAME_WIDTH* y *FRAME_HEIGHT*.

3. Obtención de las imágenes. Se realiza la captura de un frame de cada cámara, la captura de ambos frames se tiene que realizar de la forma más rápida y con el menor tiempo posible entre las capturas. Si las imágenes distan mucho en el tiempo el resultado final del sistema no sería correcto.

4. Comprobación de la toma de imágenes. En este paso se verifica que ambas imágenes han sido tomadas. En el caso de que la variable correspondiente a una de ellas se encuentre vacía, automáticamente se detendría la ejecución evitando la aparición de posibles errores en pasos posteriores.

No se ha realizado una explicación de las funciones utilizadas debido a su bajo grado de complejidad.

5.3. Calibración de las cámaras

Aunque ya se ha realizado una explicación de los siguientes conceptos en el capitulo de visión estereoscópica se ha considerado oportuno realizar un breve recordatorio.

El proceso de calibración tiene como objetivo la determinación de una serie de parámetros propios de una cámara que permiten conocer cómo proyecta de forma bidimensional la información de la escena observada, con el fin de poder obtener información tridimensional a partir de las imágenes bidimensionales. Los parámetros pueden ser intrínsecos o extrínsecos. Los parámetros intrínsecos se corresponden con las características internas, geométricas u ópticas de la cámara, mientras que los parámetros extrínsecos se corresponden con el posicionamiento de la cámara en el sistema de coordenadas. En todo sistema de visión artificial debe realizarse la calibración de las cámaras que se van a utilizar. En aquellos sistemas que utilizan dos o más cámaras es recomendable realizar la calibración para todas las cámaras aunque estas sean el mismo modelo ya que pueden tener ligeras diferencias.

5.3.1. Parámetros intrínsecos

Son aquellas características propias de cada cámara. Estas pueden cambiar si se modifica la óptica o el sensor de la cámara y determinan como la cámara representa la información tridimensional obtenida de la escena en dos dimensiones, mediante una imagen bidimensional.

Los parámetros intrínsecos obtenidos mediante el proceso de calibración son:

- Punto principal. Se corresponde con el punto de intersección entre el plano de formación de imagen y la recta perpendicular a dicho plano que pasa por el centro de la cámara.
- Distancia focal. Distancia entre el centro óptico y el punto principal.
- Vector de distorsión. Debido a el uso de lentes convergentes las imágenes presentan una distorsión, principalmente en las esquinas. La distorsión se cuantifica y se almacena en este vector.
- Error de píxeles. Indica la precisión de la calibración.

5.3.2. Parámetros extrínsecos

Son aquellos que aportan la información necesaria para poder llevar a cabo una reconstrucción tridimensional a partir de imágenes, relacionando la información bidimensional obtenida mediante una cámara con la referencia tridimensional del mundo.

Los parámetros extrínsecos son:

- Vector de transalación, T. Se refleja la ubicación del centro de la cámara respecto del centro del sistema de coordenadas del mundos real. Este vector tiene tres coordenadas, las cuales corresponden a cada uno de los ejes: X, Y y Z.

- Matriz de rotación, R. Determina la rotación de la posición del centro óptico de la cámara respecto a los ejes del sistema de coordenadas del mundo real.

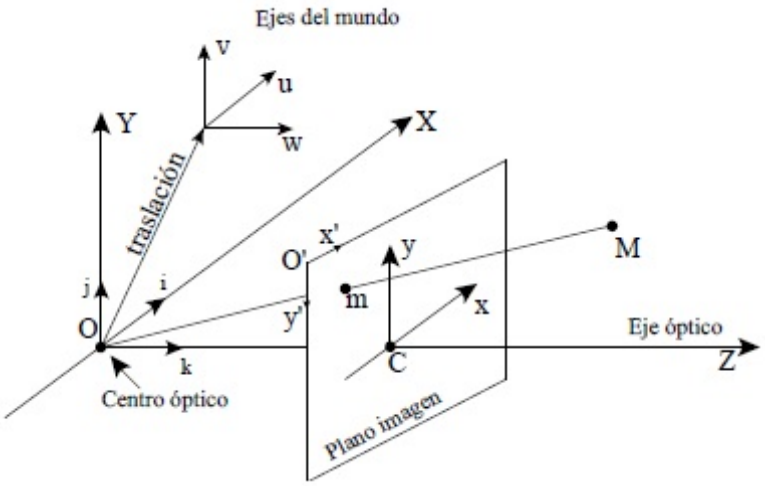


Figura 5.4: Ejemplo de proyección de los puntos del mundo 3D al plano de la imagen en 2D.

5.3.3. Método de calibración

Para la obtención de los parámetros tanto intrínsecos como extrínsecos se ha utilizado el método de Zhang, basado en el uso de un patrón como referente. El cual se utiliza para reconstruir el proceso de formación de las imágenes ya que se conoce el posicionamiento de los puntos en el patrón respecto al sistema de coordenadas exterior.

Tras obtener una aproximación inicial mediante un método lineal, se realiza un refinamiento iterativo empleando un criterio de máxima probabilidad. La calibración se realiza sobre un plano de puntos, y aunque no es necesario conocer el desplazamiento que se realiza entre la toma de distintas imágenes, para que el método funcione correctamente se necesita al menos tres imágenes con el patrón tomado en distintas orientaciones. Este número de imágenes puede ser inferior si se fijan los valores de algunos parámetros intrínsecos. Por ejemplo, si no se calcula la ortogonalidad del plano imagen, sólo son necesarias dos imágenes. Los parámetros se obtienen implícitamente, es decir, lo que se obtiene es una matriz cuyos elementos son función de los parámetros intrínsecos. La distorsión que se incluye es de tipo radial y se modela con dos coeficientes. [21]

Se ha seleccionado este método por dos motivos. El primero de ellos es la facilidad en cuanto a la implementación, necesitando una casi nula preparación de la escena para ser llevado a cabo. El segundo es la popularidad de este, teniendo gran cantidad de documentación.

5.3.4. Estructura del programa de calibración de las cámaras

La calibración de las cámaras es realizada mediante la ejecución del programa *CalibracionCamaras*. El código ha sido desarrollado de forma modular, lo que ha permitido

realizar una división en partes de este. Las partes en las que se ha descompuesto el código de calibración de las cámaras, citadas en orden de ejecución, son:

1. Toma de imágenes del patrón.
2. Selección valores del patrón.
3. Búsqueda de esquinas.
4. Distorsión y rectificación.
5. Visualización y guardado.

Toda interacción del programa con el usuario se realiza mediante la consola, como la toma de decisiones que ha de realizar el usuario ante las diferentes opciones que le da el programa.

Toma de imágenes

Al iniciar la ejecución del programa CalibracionCamaras, este comprueba que no haya imágenes en la carpeta donde van a ser almacenadas las del actual proceso de calibración. Evitando así un posible error en la calibración por la utilización de imágenes correspondientes a dos escenas distintas. Si detecta que hay imágenes antiguas pide al usuario que las borre y finaliza su ejecución. Si no hay imágenes antiguas comienza el proceso de toma de imágenes del patrón.

El patrón tiene que ser una cuadrícula, formada por cuadros negros y blancos. Es importante que cuando se tomen las imágenes, este se encuentre totalmente plano. A la hora de capturar las imágenes se aconseja que todos los cuadros de la cuadrícula se encuentren en ambas imágenes, esto no supondría un grave problema ya que de no ser así estas imágenes serían eliminadas por el propio programa. Para poder visualizar las imágenes captadas por ambas cámaras se abrirá una ventana, con el nombre *entrada*, donde aparecerán las imágenes capturadas por las cámaras en tiempo real. Además aparecerán unas instrucciones en la consola, véase figura 5.5. La toma de imágenes se realizará mediante la pulsación de la tecla *ESPACIO*, y para finalizar la toma de imágenes se pulsara la tecla *ESCAPE*. Se recomienda tomar al menos 7 capturas con orientaciones distintas.

Una vez finalizada la toma de imágenes se le dará la opción al usuario de visualizar las imágenes tomadas, con el fin de comprobar que estas han sido tomadas correctamente.

Selección de valores del patrón

El método de calibración utilizado necesita una serie de valores de entrada, los cuales se corresponden con características del patrón utilizado. Estos valores son tres:

- Tamaño del lado del cuadrado expresado en milímetros
- Número de esquinas a buscar en el ancho de la cuadrícula.
- Número de esquinas a buscar en el alto de la cuadrícula.

```
Pulse SPACIO para guardar las imagenes.
Pulse ESC para finalizar el programa.

RECUERDE: pulse la tecla siempre que tenga seleccionada
cualquiera de las ventanas Izquierda o Derecha, si
intenta escribir en la ventana de comandos no funcionara.
Imagenes 1      guardadas
Imagenes 2      guardadas
Imagenes 3      guardadas
Imagenes 4      guardadas
Imagenes 5      guardadas
Imagenes 6      guardadas
Imagenes 7      guardadas

Quiere visualizar imagenes guardadas (y/n):
```

Figura 5.5: Visualización de la consola en el proceso de la toma de imágenes.

```
Tamanio lado del cuadrado en mm por defecto = 18
Numero de esquinas ancho por defecto = 9
Numero de esquinas alto por defecto = 5
Desea cambiar los valores(y/n): y
Introduzca tamanio lado en mm: 18
Introduzca numero de esquinas ancho: 9
Introduzca numero de esquinas alto: 5
```

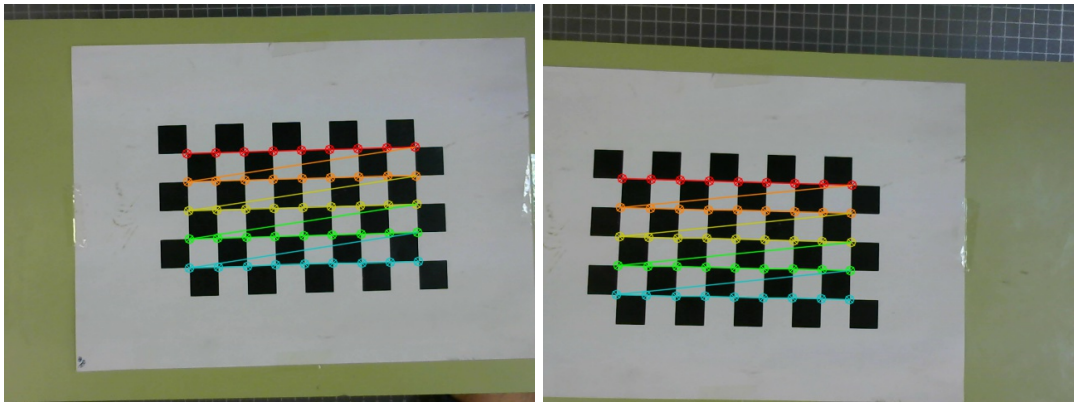
Figura 5.6: Visualización de la consola en el proceso de cambio de parámetros.

El número de esquinas en ambos casos se refiere a las esquinas interiores. Llamamos esquinas interiores a los puntos donde se encuentran cuatro esquinas de cuatro cuadrados distintos. El número de esquinas en ambos casos tiene que ser el máximo, en caso contrario la función encargada de realizar esta búsqueda podrá confundir unas esquinas con otras. Provocando un error en el proceso de calibración y rectificado. Se han fijado una serie de valores por defecto, los cuales se corresponden con el patrón utilizado durante el desarrollo del sistema. En caso de duda ante la determinación de los valores del patrón tan solo tiene que tomar como ejemplo el patrón utilizado, véase 5.3, también puede observar la figura 5.7.

Búsqueda de esquinas

El proceso de búsqueda de esquinas se va a realizar mediante la función *findChessboardCorners*. Esta función se encuentra en la biblioteca OpenCV. Los parámetros de entrada son: la imagen del patrón, el número de esquinas interiores, filas y columnas. Las salidas de la función son: una variable booleana, que toma el valor true si se han encontrado las salidas y false en caso contrario. La otra salida es un array de las posiciones de cada una de las esquinas detectadas. En caso de que la salida de la función sea true, se almacena el array de las posiciones.

Para finalizar se da la opción al usuario de visualizar los resultados de la búsqueda de esquinas. Si durante la visualización se observa que no coincide el patrón de localización de esquinas dibujado en ambas imágenes, el proceso de calibración es erróneo y se tendría que volver a realizar.



(a) Imagen izquierda. (b) Imagen derecha.

Figura 5.7: Visualización de los resultado de la búsqueda de esquinas.

Distorsión y rectificación

La corrección de las distorsiones y la rectificación de las imágenes es el objetivo del proceso de calibración. Debido al gran uso de OpenCV en aplicaciones de visión artificial y estereoscópica, el proceso para la corrección de las distorsiones y el rectificado de las imágenes ha sido normalizado. Este proceso se puede encontrar en uno de los ejemplos que trae la propia biblioteca en la carpeta samples, debido a que este ejemplo es propio de los desarrolladores de OpenCV se ha considerado innecesaria la realización de modificaciones. El proceso de calibración requiere de la ejecución de varias funciones, todas ellas pertenecientes a la biblioteca OpenCV, para alcanzar el resultado buscado.

En primer lugar se ejecuta la función calibrateCamera, obteniendo los parámetros intrínsecos y extrínsecos a partir del conjunto de imágenes tomadas al patrón de calibración. Al utilizar cámaras iguales no es necesario realizar el cálculo de los parámetros para ambas cámaras. En este caso se han calculado los parámetros para una sola de las cámaras y se han considerado iguales a los de la otra cámara.

La siguiente función que se ha utilizado ha sido *stereoCalibrate*. Esta función tiene como objetivo la obtención de los parámetros que establecen la relación entre ambas cámaras, permitiendo conocer la posición de la segunda cámara respecto de la primera. Algunos de los parámetros obtenidos como resultado de la ejecución de esta función son: la matriz de rotación entre los sistemas de coordenadas de ambas cámaras, el vector de traslación entre ambos sistemas de coordenadas, etc.

A continuación se utiliza la función *stereoRectify*. Con esta función se consigue hacer que todas las lineas epipolares sean paralelas, consiguiendo que los objetos de la escena se encuentren a la misma altura en ambas imágenes. Para conseguirlo se trabaja con las matrices de rotación obtenidas a partir de la ejecución de la función *stereoCalibrate*.

Por último, se utiliza la función *initUndistortRectifyMap*. Después de obtener todos los parámetros correspondientes a las distorsiones y la rectificación, se han de juntar para formar los mapas de transformación. Mediante la aplicación de estos mapas sobre las imágenes se consigue corregir los defectos presentes en las imágenes. Para la creación de estos mapas se utiliza la función citada anteriormente, *initUndistortRectifyMap*.

El orden en la ejecución de las diferentes funciones es muy importante ya que algunos de los valores obtenidos mediante la ejecución de la primera función son utilizados en la ejecución de la segunda, ocurriendo lo mismo con todas las funciones utilizadas. Para la aplicación de los mapas de transformación sobre las imágenes se utiliza la función *remap*, pero esto ya se corresponde con el siguiente apartado.

Visualización y guardado

Después de realizar la corrección de distorsiones y el rectificado, es aconsejable observar los resultados ya que puede surgir algún error, y el desconocimiento de este implicaría la obtención de resultados no válidos en los resultados finales del sistema. Si se opta por visualizar los resultados, aparecerá una ventana donde se mostraran las imágenes capturadas por ambas cámaras en vivo, esta ventana tendrá unas lineas rojas para poder comprobar que los objetos se encuentran a la misma altura en ambas imágenes.

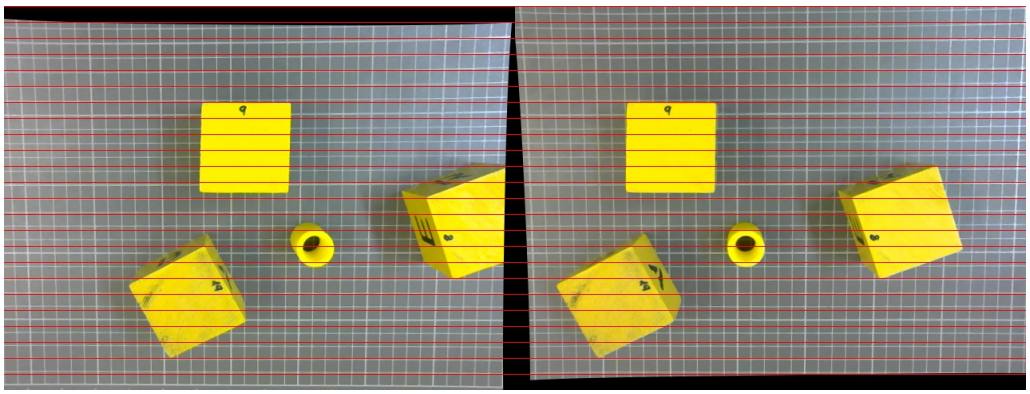


Figura 5.8: Visualización de los resultados de la corrección de distorsiones y la rectificación.

5.4. Color

El objetivo principal de este trabajo de fin de grado es la realización de un sistema de visión artificial que permita iluminar la mano del cirujano durante una operación laparoscópica asistida mediante mano. El sistema tiene que ser capaz de identificar la mano para posteriormente calcular la distancia a la que se encuentra y permitir que el sistema se mueva de forma coordinada con la mano del cirujano.

La identificación de la mano del cirujano dentro de la cavidad abdominal del paciente puede resultar algo complejo, ya que pueden darse situaciones en las que esta quede oculta de forma parcial. Algunas de estas situaciones pueden producirse por sangrados internos en los que la mano queda recubierta de sangre de forma que el sistema de visión no la identifique correctamente, también puede ocurrir que algún órgano se sitúe entre el sistema y la mano del cirujano.

Algunas de las soluciones barajadas para la identificación de la mano del cirujano han sido:

- Identificación de patrones. Se basa en la utilización de un guante con una serie de patrones en diferentes puntos del guante como falanges de los dedos, torso y palma.

De esta forma el sistema de visión artificial podría identificar cada uno de los patrones permitiendo obtener el posicionamiento de la mano y la distancia al sistema de visión. El posicionamiento de la mano sería de gran ayuda ya que permitiría conocer los gestos realizados por el cirujano, como un corte o una sutura. El problema surge cuando los patrones puedan quedar ocultos, siendo muy probable en esta aplicación, impidiendo el cálculo de la distancia y posicionamiento de la mano del cirujano.

- Identificación de colores. Debido a que el rango de colores presentes en el interior de un ser humano no es muy amplio, esta solución propone discriminar el color del guante de las imágenes capturadas por el sistema. Como ya se comentó en el capítulo del color, este puede variar dependiendo de algunas características de la escena como la iluminación. Además la mano del cirujano pueda estar cubierta por sangre u otras sustancias que pueden afectar al color. Una ventaja que presenta la identificación mediante colores es que al trabajar con toda la mano del cirujano sería muy improbable que esta quedase totalmente oculta o que el color de toda ella no se correspondiese con el color a identificar.

Debido a que el gesto de la mano no es una condición necesaria, se ha optado por implementar la identificación de colores. Como se comentó en el capítulo del color, existen varios modelos del color. En cada uno de estos el color esta formado por una serie de parámetros distintos. La siguiente decisión a realizar es el modelo de color que se va a utilizar en las imágenes procesadas mediante el sistema de visión artificial. Los dos modelos del color principales son el RGB y HSV.

Se ha tomado la decisión de poder trabajar con ambos modelos del color. Aunque el modelo HSV en un principio presenta mejores características para implementar en el sistema de visión artificial que se esta realizando, el modelo RGB es mucho más popular. El modelo HSV se comporta mucho mejor ante cambios de iluminación, es decir, el cambio producido en los parámetros que forman el color en el modelo HSV es muy pequeño. Permitiendo poder filtrar una escena a partir de un color y manteniendo una alta calidad en los resultados aunque la iluminación cambie. Sin embargo, en el modelo RGB un cambio en la iluminación puede suponer un mayor cambio en los parámetros del color obteniendo resultado erróneos en el filtrado de imágenes cuando se producen cambios de iluminación.

El sistema de visión artificial va a ser utilizado por cirujanos. Ellos no tienen porque conocer el proceso de formación del color o los valores correspondientes para la formación de un color determinado en un modelo del color. Para facilitar el proceso de determinación del color a filtrar se ha decidido implementar un selector de color. Un selector de color es una herramienta utilizada en programas de diseño gráfico como Adobe Photoshop, Paint, etc, para la obtención del color. El funcionamiento de un selector de color consiste en la selección de un pixel de la imagen haciendo clic sobre él y el selector del color le proporciona el color correspondiente a dicho pixel. El problema de solo trabajar con un pixel es que a día de hoy las imágenes digitales presentan una densidad de píxeles tan elevada que el ser humano no es capaz de visualizar cada pixel de forma individual y por lo tanto tampoco su color. Por ello los selectores de color tienen la posibilidad de establecer el tamaño de la muestra a seleccionar cuando se hace clic sobre un pixel. De esta forma el color proporcionado por el selector de color se corresponde con la media del color de una zona determinada por el usuario.

Como ya se ha comentado anteriormente el color se define mediante unos parámetros

que cambian dependiendo del modelo de color utilizado. Debido a las posibles variaciones del color, los parámetros que definen este también se ven alterados. Haciendo un filtro especifico para un color determinado los resultados del filtro no serian los deseados por el usuario. Para corregir estos problemas en el filtrado de colores en imágenes digitales se trabaja con rangos de colores. Un ejemplo de este problema se puede observar en la figura 5.9 (a), donde podemos ver un guante con las puntas de tres dedos recortadas y sobre la que se quiere aplicar un filtro de color para la eliminación de todos los colores que no se correspondan con el color de los dedos. El color de los dedos se representa en el modelo RGB mediante unos valores. Aplicando el filtro de color para el color de los dedos se obtiene el resultado de la figura 5.9 (b), donde podemos ver una imagen completamente en negro, es decir, la imagen sobre la que se realiza el filtrado no presenta ningún pixel con el color considerado para los dedos. Sin embargo la figura 5.9 (c) se corresponde con el resultado del filtrado de la misma imagen pero esta vez se ha filtrado un rango de valores. Aún proporcionando un rango de valores hay zonas correspondientes con los dedos que han sido eliminadas, esto se debe a la iluminación de la escena.

(a) Entrada (b) Único (c) Rango

Figura 5.9: Resultado del filtrado de una imagen.

La selección del color a filtrar se va a realizar mediante dos métodos diferentes, cada uno de estos utiliza un modelo de color diferente. El usuario elegirá el método a implementar entre los dos siguientes:

- Deslizadores. Este método se basa en la utilización de deslizadores para la selección del color, el cual esta representado mediante el modelo HSV. El color en el modelo HSV se optiene mediante tres parámetros: matiz, saturación y valor. A cada uno de estos parámetros se le ha asignado dos deslizadores, uno para el valor máximo del parámetro y otro para el mínimo. Para facilitar el proceso de ajuste de los parámetros para la selección de un color se ha implementado un selector de color, de esta forma si el usuario hace clic sobre la imagen obtendrá el color correspondiente a la zona de la imagen donde ha cliqueado para posteriormente mover los deslizadores con el fin realizar un ajuste más preciso del rango de colores a filtrar. El tamaño de la muestra del selector de color tiene un valor fijo. Cuando finalice la ejecución del programa los valores para los diferentes parámetros serán escritos en un fichero para su posterior utilización.

- Selector de color. Basado en la utilización de un selector de color, mediante el cual se obtendrán los parámetros del color correspondientes al modelo RGB. El funcionamiento del selector de colores es igual que en el método anterior salvo que en este caso

el tamaño de la muestra puede ser modificado por el usuario mediante un deslizador. También se podrá configurar el rango de los valores mediante otro deslizador.

5.4.1. Estructura del programa de configuración del filtro

En el código perteneciente al filtrado de colores se ha realizado la implementación de ambos métodos, como ya se ha dicho anteriormente. El filtro de colores se realiza mediante la ejecución de programa *FiltroColor*. Lo primero que tendrá que hacer el usuario una vez haya ejecutado el programa será la selección del método a utilizar, esto se realizara desde la consola donde aparecer un pequeño menú indicándole el número a introducir dependiendo de la opción que quiera elegir.

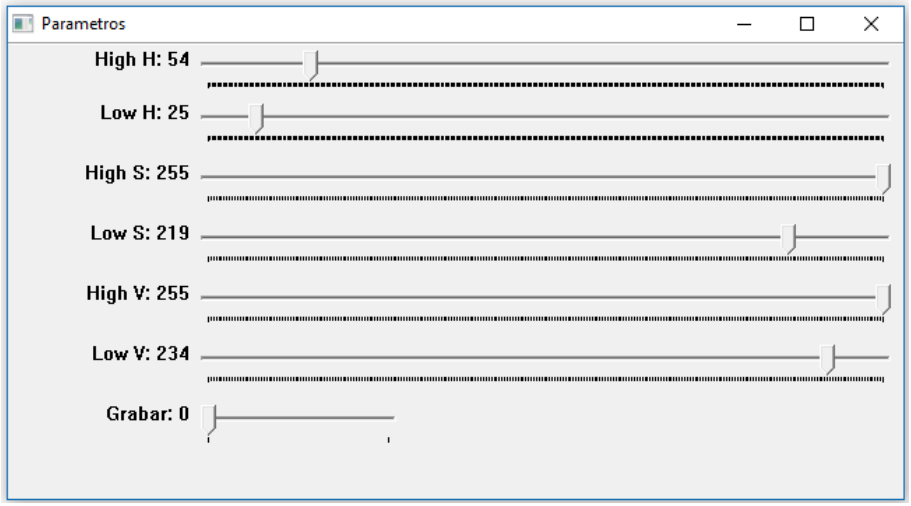
Una vez seleccionado el método aparecerán tres ventanas: entrada, salida y parámetros. La ventana entrada y salida serán iguales para cualquiera de los dos métodos mientras que la ventana parámetros será diferente. En la ventana entrada se podrá visualizar las imágenes captadas por una de las dos cámaras del sistema, en esta ventana se implementa el selector de color para ambos métodos, es decir, es la ventana donde el usuario tiene que cliquear para seleccionar el color. En la ventana salida se podrá observar el resultado del filtro de color sobre las imágenes captadas por la cámara del sistema. Por último, la ventana de parámetros esta compuesta por deslizadores, la cantidad de deslizadores dependerá del método seleccionado por el usuario, y mediante las cuales se podrán configurar una serie de parámetros necesarios para el filtrado de las imágenes.

En el método Deslizadores, el cual se corresponde con un filtrado mediante el modelo de color HSV, la ventana parámetros tendrá siete deslizadores. Los seis primeros se corresponderán con el valor máximo y mínimo para cada uno de los parámetros que forman el color en el modelo HSV. El séptimo deslizador habilitará el grabado de los valores en un fichero, este deslizador toma el valor cero o uno. Si el usuario termina la ejecución del programa y el valor del séptimo deslizador es uno, los valores se grabaran en un fichero. En caso de que fuese cero, no.

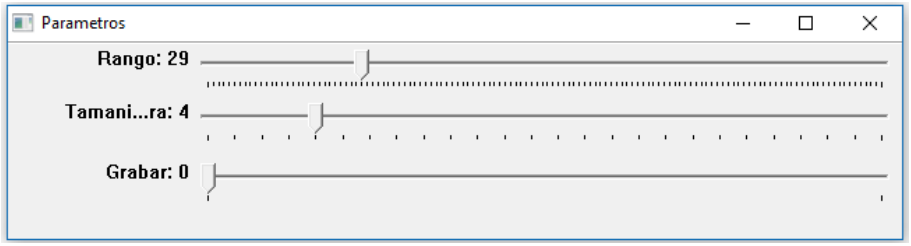
En el método Selector de color, correspondiente con el filtrado de las imágenes mediante el modelo RGB del color, la ventana parámetros tendrá tres deslizadores. El primero de estos deslizadores se corresponde con el rango de colores que se van a filtrar. Cuanto mayor sea este valor mayor cantidad de colores pasaran el filtro. El segundo deslizador se corresponde con el tamaño de la muestra. Como ya se comentó anteriormente la determinación de los valores de un color se realizaba mediante la media de los valores del color de la vecindad del pixel seleccionado por el usuario, el tamaño de la vecindad se determina mediante el valor de este segundo deslizador. El tercer y último deslizador tiene la misma función que para el método anterior, habilitación del guardado de las componentes del color en un fichero.

Para terminar la ejecución del programa FiltroColor, el usuario tiene que pulsar la tecla ESC.

En la figura 5.11 y 5.12 se pueden observar los resultados obtenidos mediante el filtrado de color en una imagen. Para los dos métodos se ha utilizado la misma imagen. En ambos casos se ha intentado obtener el mejor resultado con cada uno de los métodos, obteniendo unos resultados muy similares. Aún así se pueden apreciar algunas diferencias. El resultado obtenido por el método Selector de color tiene una menor precisión en la



(a) Deslizadores



(b) Selector de color

Figura 5.10: Ventanas de parámetros asociadas a cada uno de los métodos. Los valores asignados a cada parámetros son los utilizados para la obtención de los resultados de las figuras 5.11 y 5.12.

selección del color haciendo que algunas zonas con una ligera variación del color también esten incluidas. Sin embargo en el resultado obtendio mediante el método Deslizadores, estas mismas zonas han sido excluidas. Es necesario recordar que los párametros de cada uno de los métodos pueden ser configurables, pudiendo bajar la precisión del método Selector de color para que tuvise una menor tolerancia a los cambios de color en la imagen.

5.5. Correspondencia estereocópica

La correspondencia estereoscópica se realiza mediante la obtención de las diferentes proyecciones de un punto en la escena en las imágenes del par estereoscópico. La realización de una buena correspondencia es la parte más compleja dentro de los sistemas de visión artificial. Como ya se explicó, las técnicas para realizar una correspondencia estereoscópica se pueden dividir en dos grandes grupos: las técnicas basadas en el área y las técnicas basadas en las características.

En el sistema de visión artificial se han utilizado las técnicas basadas en el área. Las técnicas basadas en las características necesitan de puntos de borde o regiones que sirvan como elementos identificativos a la hora de realizar la correspondencia estereoscópica. Debido a que solo se filtra el color y no se realiza ningún tipo de segmentación de la imagen

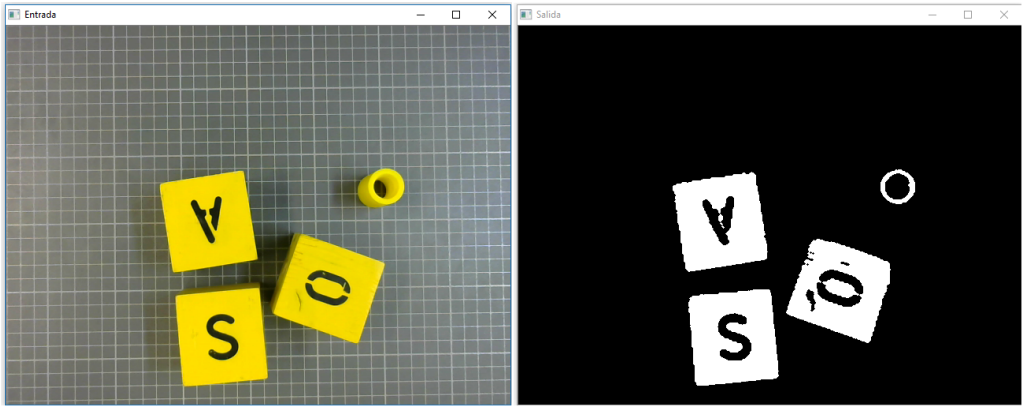


Figura 5.11: Resultado de la implementación del método Deslizadores.

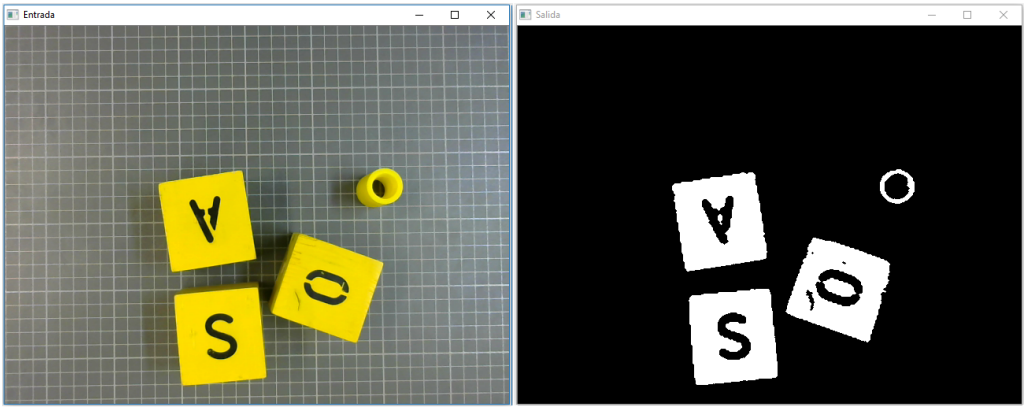


Figura 5.12: Resultado de la implementación del método Selector de color.

se tienen que utilizar las técnicas basadas en el área. La principal ventaja del uso de esta técnica es la obtención de mapas de disparidad mucho más densos en comparación con las técnicas basadas en las características. También presenta desventajas, como la posibilidad de obtener falsas correspondencias.

La biblioteca de funciones utilizada, OpenCV, tiene una serie de funciones que permiten la obtención de estas correspondencias mediante el Block Matching. Debido a la dificultad que implica este proceso, las funciones realizan una serie de operaciones sobre las imágenes dadas para mejorar la calidad de la salida proporcionada.

La implementación de algoritmos Block Matching en OpenCV se realiza mediante los siguientes pasos:

1. Llevar a cabo un prefiltrado de la imágenes para normalizar o intensificar la intensidad de las imágenes.
2. Búsqueda de correspondencias entre ambas imágenes de forma paralela a las líneas epipolares.
3. Filtrado de los resultados para eliminar posibles ruidos presentes en el mapa de disparidades.

Todos los algoritmos basados en Block Matching trabajan con ventanas de píxeles, es decir, para obtener las correspondencias estereoscópicas comparan el valor tanto de los

píxeles evaluados como el de sus vecindades, el conjunto formado por el pixel y su vecindad es conocido como ventana. La gran mayoría de los algoritmos permiten la elección del tamaño de la ventana.

En el prefiltrado de las imágenes de entrada se puede llevar a cabo una normalización de las imágenes o la intensificación de los bordes. El normalizado de la imagen se realiza mediante la resta del valor medio de intensidad de los píxeles de su ventana, estableciendo un máximo y un mínimo. Con el normalizado se consigue reducir el ruido presente en las imágenes. La intensificación de los bordes se realiza asignadoles pesos a los píxeles de una ventana, siendo mayor el peso de los píxeles centrales que el de los extremos, consiguiendo resaltar los bordes de los objetos.

La búsqueda de correspondencias en las funciones de OpenCV utilizan la restricción epipolar. Los objetos en ambas imágenes se encuentran a la misma altura, debido a esta restricción las correspondencias tienen que estar presentes en la misma linea epipolar y deja de ser necesario recorrer la imagen al completo. El algoritmo selecciona una ventana en una de las imágenes y realiza el sumatorio de las intensidades de los píxeles en dicha ventana. A continuación recorre los píxeles de la linea epipolar de la otra imagen, estableciendo ventanas del mismo tamaño y calculando el sumatorio de las intensidades. La correspondencia se establece entre el pixel de la primera imagen y el pixel de la segunda cuyas ventanas tengan la menor diferencia entre los sumatorios de intensidad. El valor almacenado para esa posición se corresponde con el desplazamiento entre ambos píxeles.

En el postprocesado se realiza una filtrado de los resultados obtenidos mediante la búsqueda de correspondencias. Este filtrado es necesario para la eliminación de falsos positivos. Estos son debidos a que el algoritmo selecciona el mejor de los resultados, pero se puede dar el caso que todos los resultados obtenidos son malos, y el algoritmo selecciona el menos malo. Otro aspecto que trata el postprocesado son las zonas correspondientes a bordes de objetos y el nivel de textura en las ventanas.

Los diferentes tipos de algoritmos para la obtención de las correspondencias estereoscópicas implementados en una función presente en la biblioteca de OpenCV son:

- Graph-Cuts, correspondencia densa global. Los algoritmos de correspondencia globales intentan encontrar similaridades entre intensidades de píxeles a nivel de imagen completa, creando áreas de píxeles con el mismo nivel de similaridad. Afectando la correspondencia de un píxel a sus vecinos. [37]

  Este algoritmo agrupa los píxeles en función de la profundidad y la similaridad, tratando de minimizar la función de energía. Permitiendo identificar formas y bordes, y reduciendo los problemas causados por las oclusiones, iluminación y textura. Esto supone un coste de procesamiento muy alto, inhabilitando el uso de este algoritmo en sistemas en tiempo real.

- Block Matching, correspondencia densa local. Este algoritmo busca las correspondencias para cada píxel, estableciendo las disparidades mediante la comparación de las intensidades de los píxeles en las ventanas.

  Las principal ventaja de este algoritmo es su bajo coste computacional, soportando la carga de trabajo de sistemas de visión en tiempo real. Las principales desventajas son dos. La primera de ellas es el resultado obtenido en los bordes de las imágenes. El hecho de que las imágenes estén tomadas desde diferentes posiciones hace que

los bordes de las imágenes no coincidan. Cuando se implementan algoritmos de Block Matching se obtienen resultados muy malos debido a las grandes diferencias de intensidad entre las ventanas situadas en los bordes de las imágenes. La segunda desventaja es la posibilidad de encontrarse con texturas muy similares, esto impide que el algoritmo encuentre correspondencias para esas zonas.

- Semi-Global Block Matching, correspondencia densa híbrida. Este tipo de algoritmos se encuentran situados entre los algoritmos locales y globales. Es decir, intenta obtener unos resultados tan buenos como los obtenidos mediante los algoritmos globales, y un coste computacional tan bajo como el de los algoritmos locales.

  Este tipo de algoritmo lleva a cabo la correspondencia estereoscópica mediante la minimización de una función de energía de las ventanas, para ello divide las imágenes en ventanas. Aumentando el coste computacional del algoritmo pero mejorando la calidad de los resultados.

Comentadas las características principales de cada uno de los algoritmos se ha de elegir uno para implementarse en el sistema de visión estereoscópica en desarrollo. El sistema va a trabajar en tiempo real, esto obliga a desechar la utilización del algoritmo Graph-Cuts.

El algoritmo de correspondencia densa local, Block Matching, esta implementado en la función *stereoBM*. Esta función presenta una serie de parámetros de entrada que permiten modificar el proceso de obtención de correspondencias estereoscópicas, alterando el mapa de disparidad que muestra la función como resultado de su ejecución. Los parámetros son:

- **numDisparities**, se corresponde con el valor máximo de disparidades. Estableciendo el rango de disparidades, desde 0 hasta numDisparities. Cuanto mayor sea el valor de este parámetro mayor será el rango de distancias que se aprecien en el mapa de disparidad.
- **blockSize**, permite definir el tamaño de la ventana de píxeles con la que se va a trabajar. En esta función la ventana esta definida como un cuadrado de como mínimo 5 píxeles de lado. Si el valor de blockSize es muy elevado, se reducirá el ruido en el mapa de disparidades pero será menos preciso. En caso contrario, el mapa de disparidades será más preciso pero aumentar la cantidad de falsas correspondencias, es decir, el ruido.

El algoritmo de correspondencia híbrida local, Semi-Global Block Matching, se encuentra implementado en la función *stereoSGBM*. Al igual que la función *stereoBM*, presenta una serie de parámetros que permiten modificar algunas variables del algoritmo. Estos parámetros son:

- **numDisparity**, es el valor mínimo de disparidad. Normalmente esta valor es cero pero puede ocurrir que debido a la rectificación de las imágenes estas se encuentren desplazadas teniendo que modificar el valor de este parámetro.
- **numDisparities**, se corresponde con el rango de disparidades, definido como la resta entre el valor máximo de disparidades y numDisparity. Este parámetro siempre toma valores superiores a cero y estos tienen que ser divisibles entre 16.
- **blockSize**, permite determinar el tamaño de la ventana a comparar en ambas imágenes. Este valor tiene que ser positivo. Normalmente tendría que tomar valores entre 3 y 11.

- **P1**, permite controlar la suavidad en los cambios de disparidad. Esto se consigue mediante una penalización, fijada por el valor asignado a P1, sobre aquellos píxeles cuya disparidad difiera en más o menos uno el valor de disparidad de los píxeles vecinos. Se recomienda el cálculo del valor de P1 mediante la siguiente ecuación:

$$P1 = 8 * numero\,caneles\,imagen * blockSize * blockSize \tag{5.1}$$

- **P2**, permite controlar la suavidad en los cambios de disparidad al igual que P1. La diferencia se encuentra en que P2 es el valor de penalización para aquellos píxeles cuya diferencia de disparidad con los píxeles vecinos cambie en más de uno. Para la obtención de un mapa de disparidades correcto el valor de P1 tiene que ser siempre menor que el valor de P2. Al igual que para P1, se recomienda el cálculo del valor de P2 mediante la siguiente ecuación:

$$P2 = 32 * numero\,caneles\,imagen * blockSize * blockSize \tag{5.2}$$

- **disp12MaxDiff**, se corresponde con el valor máximo permitido en la comprobación de disparidad izquierda-derecha. Esta comprobación consiste en la obtención del mapa de disparidades dos veces, la primera de ellas calculando las disparidades de los píxeles de la imagen izquierda y la segunda calculando las disparidades de los píxeles de la imagen derecha. Si la disparidad entre un mismo pixel en las dos imagen es mayor qu el valor asignado a disp12MaxDiff se considera que ese pixel representa el borde de un objeto. Si se le asigna un valor negativo a disp12MaxDiff se cancela la comprobación.

- **preFilterCap**, representa el valor de truncación para los píxeles de las imágenes. Mediante este parámetro se establecen los valores máximos y mínimos, preFilterCap y -preFilterCap, para las intensidades de los píxeles.

- **uniquenessRatio**, determina el margen porcentual con él que el mejor de los resultados tiene que superar al segundo mejor para no ser considerado falso positivo. Permitiendo la eliminación de falsos positivos. Se recomienda que el valor asignado se encuentre entre 5 y 15.

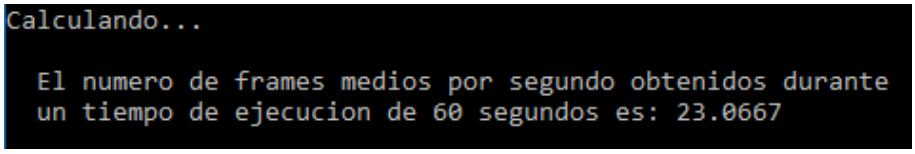
- **speckleWindowSize**, es el tamaño de la ventana sobre la que se aplica el filtro de ruido, speckle noise. Los valores para este parámetro están comprendidos entre 50 y 200. En el caso de querer deshabilitar el filtro se asigna el valor cero.

- **speckleRange**, se corresponde con la máxima variación de disparidad entre las regiones. Si se esta realizando el filtrado, un valor de 1 o 2 es suficiente.

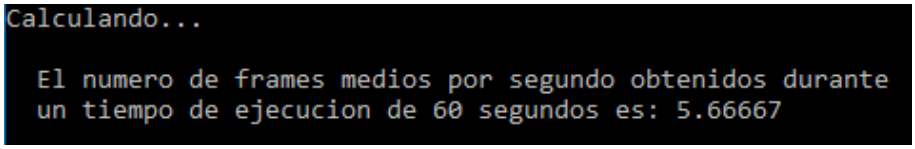
- **Mode**, determina el tipo de algoritmo utilizado para la realización de la correspondencia estereoscópica. Se pueden utilizar tres: SGBM, HH y SGBM 3WAY, seleccionados mediante los valores, 0, 1 y 2, respectivamente.

Tanto la función *stereoBM* como la función *stereoSGBM* son validas para el sistema de visión artificial en desarrollo. Para seleccionar una de las dos tenemos que establecer unos criterios. El primero de ellos es la rapidez, esta se va a medir en fotogramas por segundo(FPS). El segundo criterio es la calidad del mapa de disparidad. El peso asignado al criterio de calidad ha sido mayor que el de rapidez. Aunque en caso de que la rapidez no sea suficiente el algoritmo sería descartado.

Para poder medir el número de fotogramas por segundo que son capaces de obtener cada una de las funciones se ha realizado unos programas auxiliares, *pruebaBM* y *pruebaSGMB*, presentes en el anexo. Debido a que la cantidad de fotogramas por segundo puede variar se calcula el valor medio de fotogramas por segundo en 60 segundos. La calidad de los resultados puede variar dependiendo de los parámetros de entrada para cada una de las funciones. Estos parámetros han sido modificados con el fin de obtener los mejores resultados con cada una de las funciones.



(a) StereoBM



(b) StereoSGBM

Figura 5.13: Obtención de la cantidad de imágenes por segundo capaz de procesar cada método.

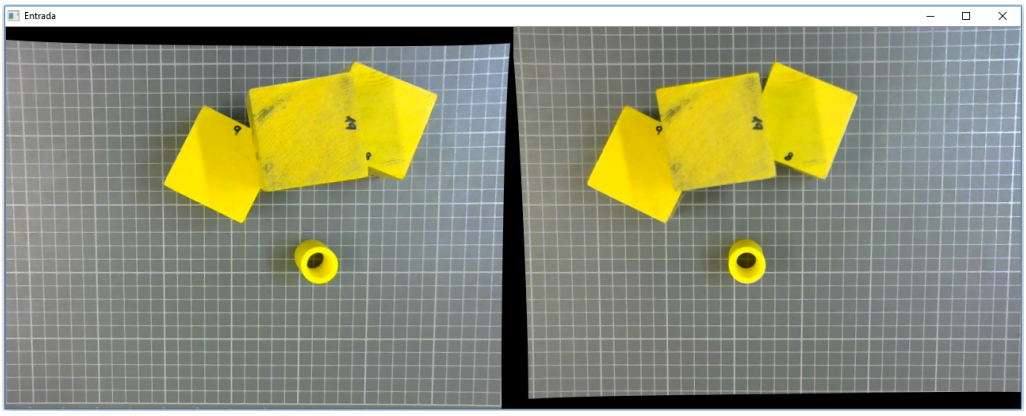
Como se puede observar en las figuras 5.13 y 5.14, la cantidad de fotogramas por segundo que se obtienen mediante la función *stereoBM* es muy superior al número de fotogramas por segundo obtenidos con la función *stereoSGBM*. Sin embargo, la calidad obtenida mediante la función *stereoSGBM* es mejor que la obtenida mediante la función *stereoBM*, presentando una menor cantidad de ruido. Debido a que la cantidad de fotogramas por segundo que se obtiene mediante la función *stereoSGBM* se considera suficiente y la calidad del resultado de esta es mejor. La función *stereoSGBM* será la utilizada en el sistema.

5.5.1. Estructura del programa de estereovisión

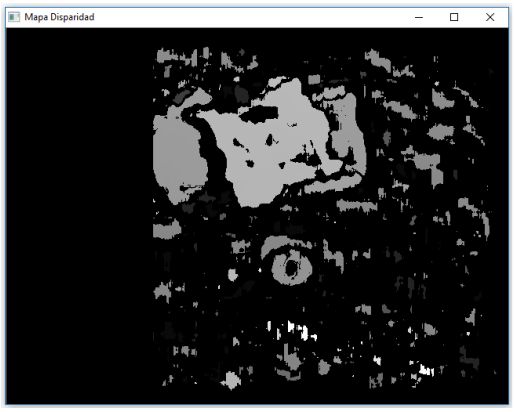
En este apartado se realizará una explicación a cerca como se ha implementado la correspondencia estereoscópica en el código del sistema de visión.

La realización de la correspondencia estereoscópica va a ser llevada a cabo por la función *stereoSGBM*, como ya se ha explicado anteriormente esta función presenta una serie de parámetros de entrada. Una mal ajuste de estos parámetros provoca la obtención de mapas de disparidad de muy baja calidad. Debido al público al que va destinado este sistema en desarrollo se ha visto la necesidad de facilitar el proceso de calibración de estos parámetros. El ajuste de estos parámetros se debería de realizar cada vez que trabajase con una escena diferente, en el caso de las cirugías laparoscópicas asistidas mediante mano no tendría porque ser necesario un ajuste de estos parámetros ya que aunque el paciente no sea el mismo el espacio de trabajo va a tener las mismas caracterísicas o muy parecidas.

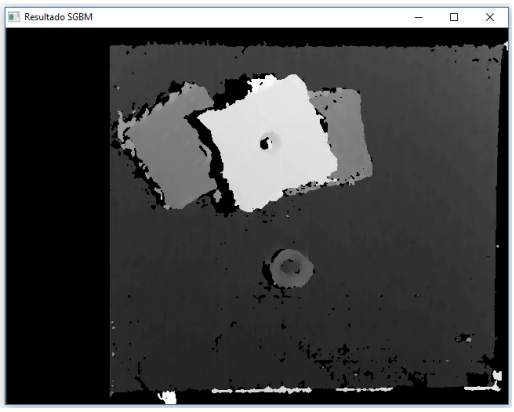
El ajuste de los parámetros de entrada se realiza desde un programa que tiene esa finalidad, con una interfaz gráfica para que el usuario pueda modificar los valores y observar el



(a) Entrada



(b) StereoBM



(c) StereoSGBM

Figura 5.14: Resultados de los diferentes métodos para llevar a cabo la correspondencia esteroscópica, ante las mismas imágenes de entrada.

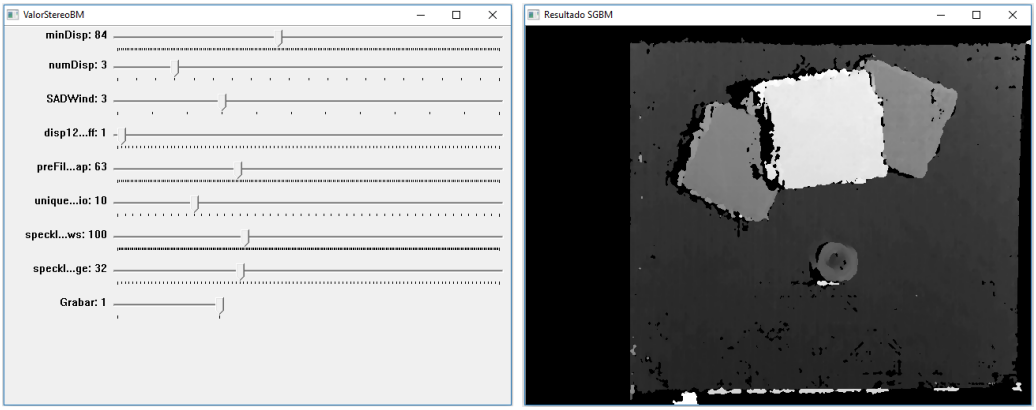
resultado de esas modificaciones de forma instantánea, permitiendo guardar dichos valores en un fichero para su posterior uso en el programa principal. El programa que realiza la calibración de la función *stereoSGBM* se le ha llamado *CalibracionSGBM*, y le código de este se puede observar en el anexo.

Antes de ejecutar el programa *CalibracionSGBM*, se ha de disponer de los valores de una serie de parámetros correspondientes con la calibración de las cámaras y la rectificación de las imágenes. Estos valores se obtienen mediante la ejecución de programa *CalibracionCamaras*. El programa *CalibracionSGBM* esta programado para que pueda detectar la ausencia de los parámetros necesarios. En caso de que no se disponga de ellos, *CalibracionSGBM* interrumpirá su ejecución y le pedirá al usuario que ejecute el programa *CalibraciónCamaras*.

Una vez ejecutado el programa, se preguntara al usuario mediante la consola si desea leer los valores de los parámetros del fichero. Si el usuario decide leer estos, los valores de los parámetros pasaran a tener el mismo valor que en el fichero. Sin embargo, si el usuario decide no leerlos, los valores de los parámetros se corresponderán por unos valores estandar.

Seguidamente, aparecerán tres ventanas: entrada, parámetros y salida. La ventana en-

trada permite la visualización de las imágenes captadas por las cámaras. En la ventana de salida se puede observar el mapa de disparidad, obtenido como resultado de la ejecución de la función *stereoSGBM*. La ventana parámetros se utiliza para calibrar los parámetros de configuración de la función *stereoSGBM*. Se puede observar que en la ventana parámetros hay nueve deslizadores, los ocho primeros se corresponden con los ocho parámetros de configuración de la función *stereoSGBM*. El último deslizador tiene como función la habilitación del grabado de los parámetros de configuración en un fichero. Los valores que puede tomar el último deslizador son cero o uno. Si el usuario finaliza la ejecución del programa mientras el valor para el noveno deslizador es 1, los parámetros de los ocho deslizadores anteriores se grabaran en un fichero.



(a) Parámetros StereoSGBM (b) Resultado StereoSGBM

Figura 5.15: Resultado de la ejecución del programa CalibraSGBM.

En la figura 5.15(a) se puede observar el resultado, donde la entrada se corresponde con la figura 5.14(a) y los parámetros utilizados son los mostrados en la figura 5.15(b). La realización del ajuste de los valores de los parámetros se realiza principalmente con los dos primeros, numDisparity y numDisparities, ya que son los que tienen un mayor grado de influencia en el resultado. El resto de valores sirven para realizar un ajuste más fino del resultado.

5.6. Cálculo de distancias

En este capítulo se explicara detalladamente el programa principal, cuyo objetivo es la determinación de la distancia entre el brazo robótico y la mano del cirujano, permitiendo el posicionamiento del brazo robótico respecto a la mano del cirujano en el interior del abdomen del paciente. Para poder realizar el cálculo de las distancias ha sido necesario la utilización de todos los programas explicados anteriormente.

Este programa realiza las siguientes funciones:

- Corrección de las distorsiones de las lentes de las cámaras. Para ello se utilizan algunos de los parámetros obtenidos en el proceso de calibración de las cámaras.

- Remapeo de las imágenes, igual que en la corrección de distorsiones se utilizan parámetros obtenidos en la calibración de las cámaras. Esto permite igualar la altura

de los objetos en las imágenes de entrada, de esta forma desaparecen las disparidades verticales.

- Filtrado de la imágenes mediante un color seleccionado anteriormente por el usuario. El color seleccionado por el usuario se le de un fichero creado por el programa *FiltroColor*.

- Obtención del mapa de disparidad mediante la función StereoSGBM. Los parámetros utilizados en la ejecución de StereoSGBM son leídos del fichero en el que anteriormente han sido escritos por el programa *CalibraSGBM*.

- Conversión de las distorsiones almacenadas en el mapa de distorsiones en distancias.

Todas las funciones realizadas por le programa, menos la última, han sido descritas y comentadas con anterioridad por lo que no se van a comentar en este apartado.

Para calcular la distancia de un punto de la escena al sistema de visión, es necesario conocer la disparidad de dicho punto, la longitud focal efectiva del sistema y la linea base. En la sección Visión estereoscópica, se ha explicado en profundidad el proceso a llevar a cabo para obtener la distancia a la que se encuentra un objeto de la escena. También se ha explicado el significado de cada uno de los parámetros necesarios para llevar a cabo el cálculo de distancias.

Para realizar el cálculo de las distancias de cada punto de la escena se dispone de dos métodos. El primero de ellos sería utilizando la ecuación que podemos encontrar en el apartado de Cálculo de la distancia, en la sección de Visión estereoscópica. El segundo método consiste en la utilización de una de las funciones disponibles en la biblioteca OpenCV. La función *reprojectImageTo3D* necesita la imagen o matriz de disparidades, y una matriz 4x4 obtenida mediante el proceso de calibración de las cámaras. Su salida es un vector de tres matrices, y cada matriz representa a una coordenada (x, y, z). De forma que el punto (x, y), tiene las coordenadas correspondientes a los valores en esa posición en cada una de las matrices.

Debido a la facilidad de implementación de un método frente al otro y a la obtención de una salida más completa, se ha decidido implementar la función *reprojectImageTo3D*. La salida de esta función facilita la conversión de si misma en una nube de puntos para su representación.

Conocido el método a implementar, se va a profundizar en su función. Esto nos permitirá tener un mayor conocimiento de la función y poder obtener unos mejores resultados. La función *reprojectImageTo3D* tiene los siguientes parámetros:

- **disparity**, este parámetro es de entrada y se corresponde con el mapa de disparidades. El mapa de disparidades tiene que tener alguno de los siguientes formatos: 8-bit unsigned, 16-bit signed, 32-bit signed o 32-bit floating-point.

- **_3dImage**, parámetro de salida, del mismo tamaño que el mapa de disparidad. Este es un vector de matrices donde cada matriz se corresponde a una coordenada del espacio. La posición de un punto con las coordenadas, x e y, en la imagen de entrada viene determinada por el valor de los elementos que ocupan la posición (x,y) en cada una de las matrices.

- **Q**, es un parámetro de entrada, el cual se corresponde con una matriz 4x4 obtenida mediante la función *stereoRectify* e indispensable para realizar el paso de disparidad a distancia.

- **handleMissingValues**, parámetro de entrada que determina el comportamiento de la función ante posiciones donde no se han obtenido valores de disparidad. Los valores que toma son true o false. Si el valor es true, aquellos valores en los que la disparidad es desconocida se les fijará una distancia muy elevada pasando a pertenecer al fondo de la escena.

- **ddepth**, este parámetro es de entrada y sirve para determinar la profundidad de los valores resultantes y almacenados en _3dImage. Estos pueden ser CV_32F, CV_16S o CV_32S.

La salida _3dImage de la función *reprojectImageTo3D* contiene el valor de las tres coordenadas del espacio. En este caso solamente nos interesa la coordenada z, que determina la distancia entre el sistema de visión artificial y la mano del cirujano. Se realiza un separación de las diferentes matrices para trabajar tan solo con la matriz de la coordenada z.

5.6.1. Estructura del programa principal

En este programa se utilizan partes del código empleadas en los programas anteriores, como el filtrado de una imagen y la realización de la correspondencia estereoscópica entre dos imágenes. En la realización de estos procesos se han introducido una serie de modificaciones para conseguir el mejor resultado posible. Además se ha añadido el código correspondiente al cálculo de la distancia entre el sistema de visión y los objetos de la escena.

La primera parte del programa se basa en la lectura de variables de distintos ficheros. Estos ficheros son los siguientes:

- Fichero de calibración. En este fichero se leen las variables, *map_r1, map_r2, map_l1 y map_l2*, que permiten realizar el remapeo de las imágenes eliminando las distorsiones producidas por la lente y rectifica la proyección de los objetos de forma que se proyecten sobre la misma linea epipolar. También se lee la variable Q que permite realizar la conversión de disparidades a distancia.

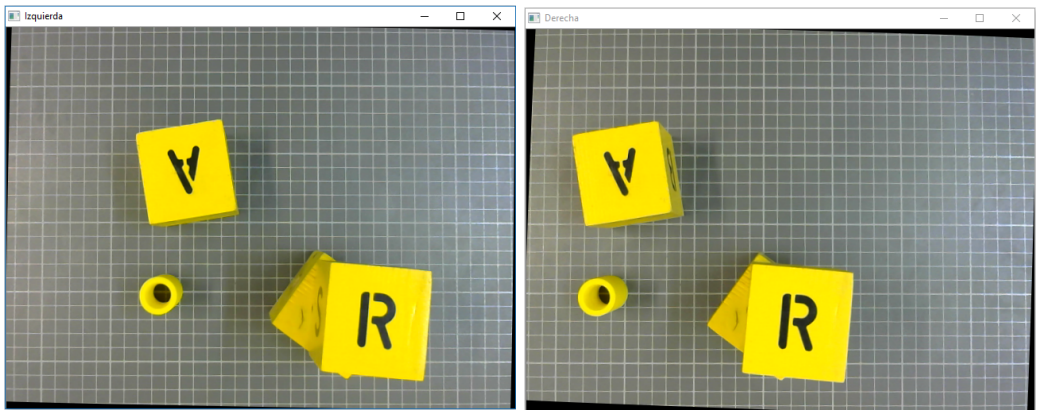
- Fichero de color. En este fichero se encuentra almacenado el color que se va a utilizar de filtro en las imágenes. Este color puede encontrarse definido en dos modelos del color diferentes, RGB o HSV. Siendo capaz de saber en cuál de los dos modelo se ha representado el color.

- Fichero de calibración *stereoSGBM*. En este fichero se almacenan los valores de los parámetros a utilizar en la función encargada de realizar la correspondencia estereoscópica. Estos parámetros han sido comentados en la sección correspondiente con la calibración de la función *stereoSGBM*.

Una vez finalizada la lectura de las variables se configura la resolución de las imágenes con las que se va a trabajar. Para ello, se ha de definir el número de píxeles para el ancho y alto de las imágenes se han establecido, ambos valores han sido asignados a unas variables *FRAME_HEIGHT y FRAME_WIDTH*, definidas en el fichero config.h. Por último

se crear un elemento, en este caso una elipse, para la realización de las transformaciónes morfologicas.

La segunda parte del programa es el bucle principal. Al comienzo de este se realiza la toma de imágenes de ambas cámaras y el remapeo de estas. Es importante hacer la toma de imágenes en el menor tiempo posible, intentando reducir al máximo el tiempo entre ambas imágenes. Si el tiempo entre la toma de imágenes es muy elevado aumenta la posibilidad de encontrarse con variaciones en la escena y por lo tanto errores en el resultado final. Una vez tomadas se remapean utilizando la función *remap*.



(a) Imagen izquierda (b) Imagen derecha

Figura 5.16: Imágenes de entrada al programa, obtenidas mediante las cámaras después del proceso de remapeo.

El siguiente paso en el bucle principal es el proceso de filtrado. Al poder trabajar con dos modelos de color, el proceso de filtrado difiere levemente. Las imágenes tomadas mediante las fuciones de OpenCV están representadas mediante el modelo RGB, aunque OpenCV altera el orden de las componentes del modelo RGB, siendo azul-verde-rojo en vez de rojo-verde-azul. El color a filtrar esta definido en el modelo RGB tan solo se realiza un binarización de la imagen estableciendo a uno todos los píxeles con un color dentro del rango del color a filtrar y a cero aquellos que no. Esto se realiza mediante la función *inRange*. Si el color esta definido en el modelo HSV se cambia el modelo del color en el cual esta representada la imagen, pasando en este caso de RGB a HSV. Para posteriormente realizar la binarización en función de los valores del color.

A las imágenes resultantes del filtrado se las ha llamado máscaras. Sobre las máscaras se ha realizado una apertura y un cierre, con el fin de reducir las diferencias entre imágenes contiguas en el tiempo eliminando aquellas zonas que pueden provocarlas. Las operaciones morfológicas, apertura y cierre, son realizadas con el elemento definido en la primera parte del programa.

A continuación se realiza la correspondencia estereoscópica mediante la función StereoSGBM. En OpenCV, StereoSGBM es una clase. Hasta ahora se ha clasificado como una función con el único fin de evitar confusiones a aquellas personas sin conocimientos en el lenguaje de programación C++.

Una clase es un bloque de construcción fundamental de los programas orientados a objetos, la cual contiene la especificación de los datos que describen un objeto junto con la descripción de las acciones que un objeto puede ejecutar.[27]

La clase StereoSGBM se crea mediante la función miembro *create*, pasando los parámetros leídos del fichero de calibración stereoSGBM. Para obtener el mapa de disparidad se ejecuta la función miembro *compute*, la cual tiene como parámetros de entrada la imagen izquierda y la derecha y una variable donde se almacenará el mapa de disparidades. La profundidad del mapa de disparidades esta predeterminada por la propia función. El mapa de disparidades no se puede representar directamente, para poder representarlo se tiene que normalizar y convertirlo en una imagen en escala de grises, provocando una perdida de información debida a la reducción de la resolución en amplitud. La normalización del mapa de disparidades se realiza mediante la función *normalize*.

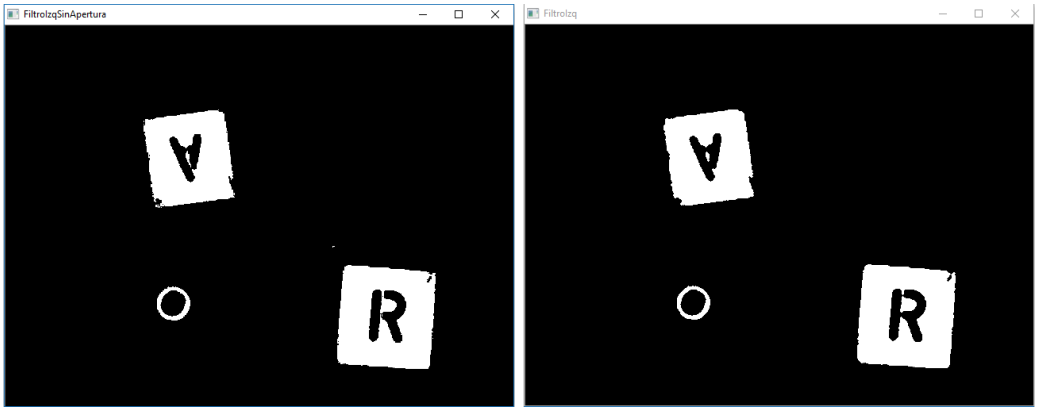
El proceso de combinación del mapa de disparidades y la máscara de color se ha podido desarrollar de diferentes formas, al final se ha optado por la explicada anteriormente ya que con ella se obtienen los mejores resultados. Las diferentes opciones que se han barajado son:

- La primera de las opciones es la obtención del mapa de disparidades a partir de las imágenes ya filtradas. Es decir, se obtendrían tanto la máscara de la imagen izquierda como derecha y se aplicarían sobre las imágenes de entrada, obteniendo dos imágenes en las que todos los colores diferentes al color elegido habrían sido eliminados. El problema de esta opción es la obtención del mapa de disparidades mediante la función *compute* de la clase *StereoSGBM*, ya que la cantidad de datos proporcionada es insuficiente y las correspondencias estereoscópicas obtenidas no son correctas provocando la obtención de un mapa de disparidad de pésima calidad.

- La segunda opción es la obtención del mapa de disparidades antes para posteriormente aplicar sobre este la máscara obtenida de combinar la máscara correspondiente a la imagen derecha y a la izquierda. El mapa de disparidad obtenido mediante esta opción tiene una mayor calidad que el obtenido mediante la primera opción. Aunque el hecho de combinar dos máscaras hace que al ser aplicadas sobre el mapa de disparidad, algunas zonas de este no quedan ocultas al usuario, obteniendo un ligero error en la visualización.

- La tercera opción consiste en utilizar la máscara correspondiente a la imagen izquierda. Durante el estudio realizado sobre la clase *StereoSGBM* y como realizaba el cálculo del mapa de disparidades mediante la función *compute*, se llegó a la conclusión de que el mapa de disparidades se crea a partir de la imagen izquierda proporcionada a la función *compute*. Por lo que aplicando solamente la máscara izquierda se obtienen unos resultados con un error menor que en las opciones anteriores.

Después de observar los resultados obtenidos mediante las diferentes opciones se decidió implementar la tercera opción. Ya que el error cometido al representar el mapa de disparidades en combinación con la máscara es muy pequeño.



(a) Antes. (b) Después.

Figura 5.17: Máscaras correspondientes a la imagen izquierda, antes y después de aplicar sobre esta una apertura y un cierre.

Como se puede observar en la figura 5.17 con la aplicación de las operaciones morfológicas de apertura y cierre se consigue eliminar algunos errores en la máscara obtenida en primer lugar, donde el algoritmo obtiene falsos positivos para dicho color.

(a) Antes. (b) Después.

Figura 5.18: Mapas de disparidad antes y después de la combinación con la máscara.

Observando la figura 5.18(b), muchos de los ruidos presentes en el mapa de disparidad se han eliminado gracias a la combinación de este con la máscara. Se puede observar un error en la zona superior del mapa de disparidad de la figura 5.18(a), este se debe al fondo utilizado. Siendo este un patrón uniforme en el que el algoritmo encargado de realizar la correspondencia estereoscópica, no es capaz de realizarla correctamente en dicha zona.

El último paso del programa es la conversión de disparidades a distancias. Para ello se ha utilizado la función *reprojectImageTo3D* pero antes se ha tenido que realizar algunos cambios en el mapa de disparidades. Estos cambios son realizados para una mejora en la precisión en el cálculo de las distancias. Esta mejora se consigue aumentado la resolución en amplitud del mapa, para ello se pasa de trabajar con una profundidad de 16 bites a una de 32 bites. De esta forma se pasa de trabajar con 65536 niveles de gris a 4294967296. Para

realizar este aumento en el rango de amplitud de la imagen se utiliza la función *copyTo*, la cual es una función miembro de la clase *Mat*.

Una vez ejecutada la función *reprojectImageTo3D* obtenemos el vector de tres matrices donde cada una de ella almacena los datos de una coordenada. Para que el usuario pueda conocer estos se ha implementado una función que permite mediante un clic sobre el mapa de disparidades conocer la distancia de la zona cliqueada al sistema de visión artificial, mostrándose en la consola. Al igual que ocurría con el selector de color, puede que el usuario haga clic sobre un pixel cuya disparidad no haya podido ser calculado o presente un error. Para evitar este problema cada vez que el usuario haga clic sobre un pixel se trabajará con un cuadrado de 20x20 píxeles cuyo centro es el pixel seleccionado. Donde el valor de la distancia será calculado mediante la media de las distancias para cada uno de los píxeles presentes en dicho cuadrado. Para aquellos puntos que se encuentren en el borde del objeto, el cálculo de la distancia será incorrecto ya que se utilizarán valores que no pertenezcan a dicho objeto. Un ejemplo de ejecución de los resultados obtenidos en la consola se puede observar en la figura 5.19.

```
Distancia: [358.09, 0, 0, 0] mm
Distancia: [358.098, 0, 0, 0] mm
Distancia: [358.627, 0, 0, 0] mm
Distancia: [358.194, 0, 0, 0] mm
Distancia: [358.603, 0, 0, 0] mm
Distancia: [402.039, 0, 0, 0] mm
Distancia: [403.528, 0, 0, 0] mm
Distancia: [402.398, 0, 0, 0] mm
Distancia: [404.309, 0, 0, 0] mm
Distancia: [403.622, 0, 0, 0] mm
Distancia: [424.595, 0, 0, 0] mm
Distancia: [424.242, 0, 0, 0] mm
Distancia: [424.779, 0, 0, 0] mm
Distancia: [425.003, 0, 0, 0] mm
Distancia: [612.933, 0, 0, 0] mm
```

Figura 5.19: Distancias obtenidas a partir del mapa de disparidades de la figura 5.18(b).

5.7. Resultado

En esta sección se van a analizar los resultados obtenidos en la ejecución del programa principal, es decir, se va a realizar una comprobación del error cometido por el sistema de visión artificial a la hora de obtener la distancia entre los objetos presentes en la escena y él mismo. Como ya se ha dicho repetidas veces, para la ejecución del programa principal se han de ejecutar todos los programas anteriores correctamente. Los resultados que se van a mostrar a continuación se corresponden con una ejecución exitosa de todos los programas, cuyos resultados no van a ser ni mostrados ni analizados.

Para el cálculo de la distancia a la que se encuentran los objetos de la escena del sistema se dispone de un soporte, véase figura 5.20. Mediante este soporte se ha conseguido una mejora de precisión en la realización de las mediciones de distancia. El soporte tiene una regla que permite conocer la altura de su plataforma de sujeción. Las mediciones son

realizadas observando el valor en él que se sitúa la plataforma en la que se ha fijado el sistema de visión. Los errores en las mediciones realizadas pueden ser causados por dos motivos diferentes:

- Tamaño del sistema de visión. Debido a la anchura del sistema, el valor observado en la regla del soporte no se corresponde con la distancia del sistema.
- Posicionamiento del sistema de visión. La sujeción del sistema no garantiza que este se encuentra perpendicular al plano del suelo provocando unas mediciones mayores a las reales.

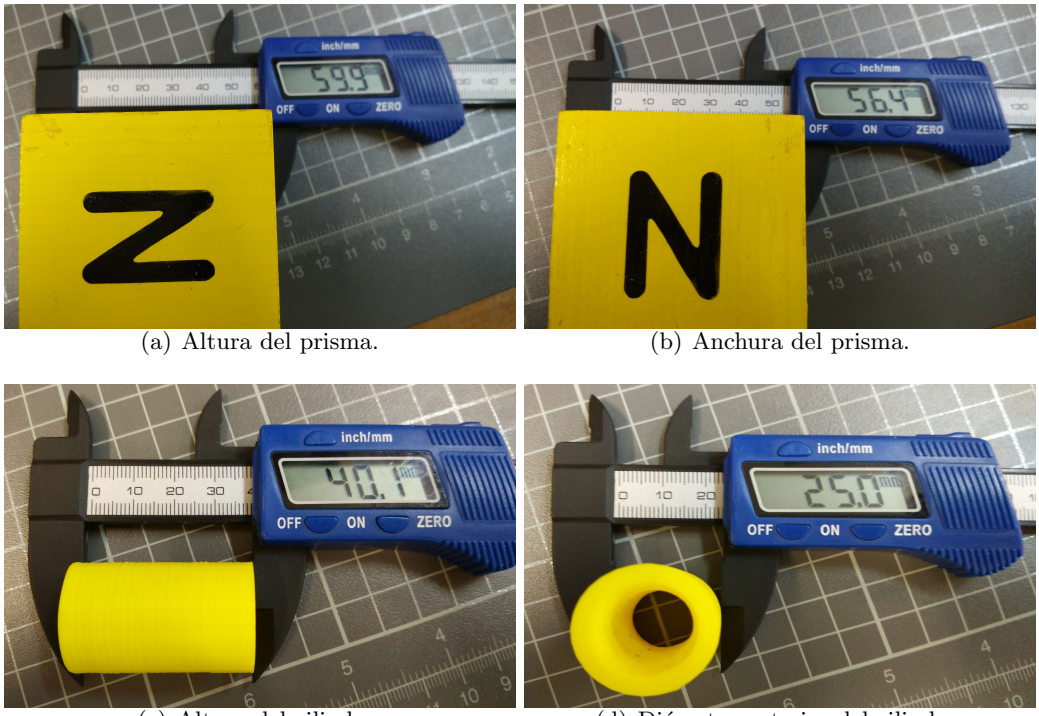
(a) (b)

Figura 5.20: Soporte de ayuda para la estimación de la distancia de los objetos respecto del sistema.

En un primer lugar, se quiso trabajar con la mano pero las dificultades que presentaba para la obtención de la distancia de forma manual entre la mano y el sistema, aumentando la posibilidad de obtener un error en la distancia, hizo que se buscasen alternativas. La alternativa es seguir trabajando con los prismas y el cilindro hueco, los cuales se pueden ver en figuras anteriores. Ambos componentes pueden ser medidos de forma precisa, reduciendo la posibilidad de cometer errores en el cálculo de la distancia.

En el proceso de medición se ha utilizado un pie de rey. Las medidas se pueden observar en la figura 5.21 y son:

- Prisma. Altura: 59.9 mm. Anchura: 56.4 mm.
- Cilindro hueco. Altura: 40.1 mm. Diámetro: 25 mm.



(a) Altura del prisma. (b) Anchura del prisma.

(c) Altura del cilindro. (d) Diámetro exterior del cilindro.

Figura 5.21: Medidas realizadas sobre el primas y el cilindro hueco.

Una vez conocidas las medidas de los objetos a utilizar en la escena, los colocamos en esta, véase figura 5.22. Como se puede observar la escena presenta tres prismas y el cilindro, estableciendo tres alturas diferentes. Todos los prismas se encuentran en horizontal, por lo que la altura de los prismas es la valor de la anchura medido. Las distancias obtenidas de forma manual son:

- Dos prismas apilados.

$$Altura_{DosPrismas} = Anchura_{Prisma} \cdot 2 = 56{,}4 \cdot 2 = 112{,}8mm \tag{5.3}$$

- Único prisma.

$$Altura_{UnPrisma} = Anchura_{Cilindro} = 56{,}4mm \tag{5.4}$$

- Cilindro.

$$Altura_{Cilindro} = Altura_{Prisma} = 40{,}1mm \tag{5.5}$$

Antes de ejecutar el programa principal para la obtención de las distancia, se calcula la distancia de forma manual. Para poder realizar los cálculos se tiene que conocer la altura a la que se encuentra situado el sistema de visión no se corresponde con la altura de la plataforma medida mediante el soporte. A la altura de la plataforma se tiene que restar la del sistema de visión. Ambas mediciones pueden observarse en las figuras 5.23(a) y 5.23(b). Las distancias entre los objetos y el sistema son:

- Dos prismas apilados.

$$\begin{aligned} Distancia_{DosPrismas} &= Altura_{Plataforma} - Altura_{Sistema} - \\ Altura_{DosPrismas} &= 490 - 24{,}2 - 112{,}8 = 353mm \end{aligned} \tag{5.6}$$

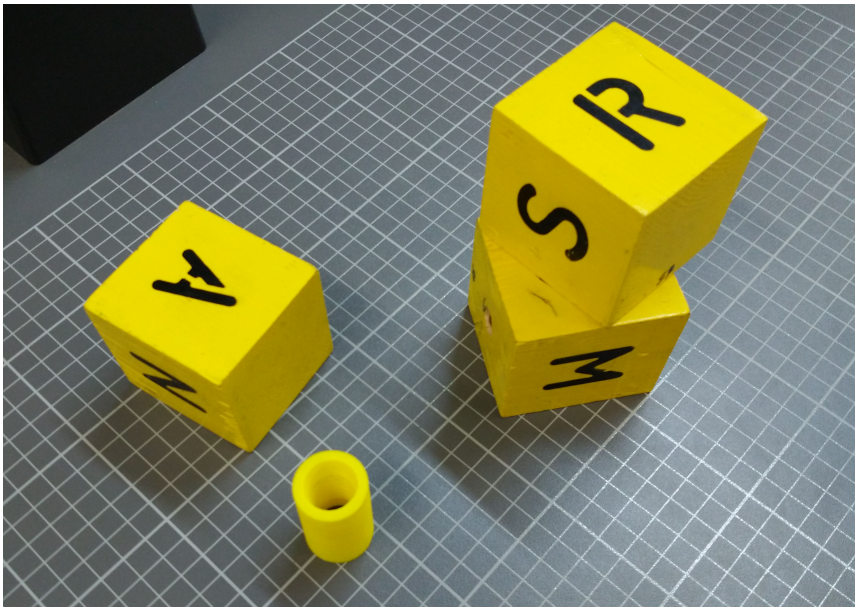
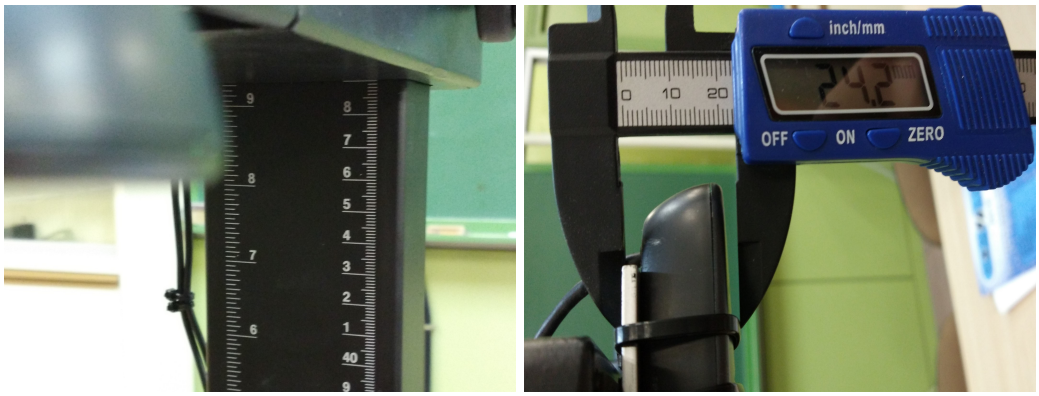


Figura 5.22: Posicionamiento de los objetos en la escena.



(a) Plataforma. (b) Sistema.

Figura 5.23: Medición de las alturas.

- Único prisma.

$$\begin{aligned} Distancia_{UnPrisma} &= Altura_{Plataforma} - Altura_{Sistema} - \\ Altura_{UnPrisma} &= 490 - 24{,}2 - 56{,}4 = 409{,}4mm \end{aligned} \tag{5.7}$$

- Cilindro.

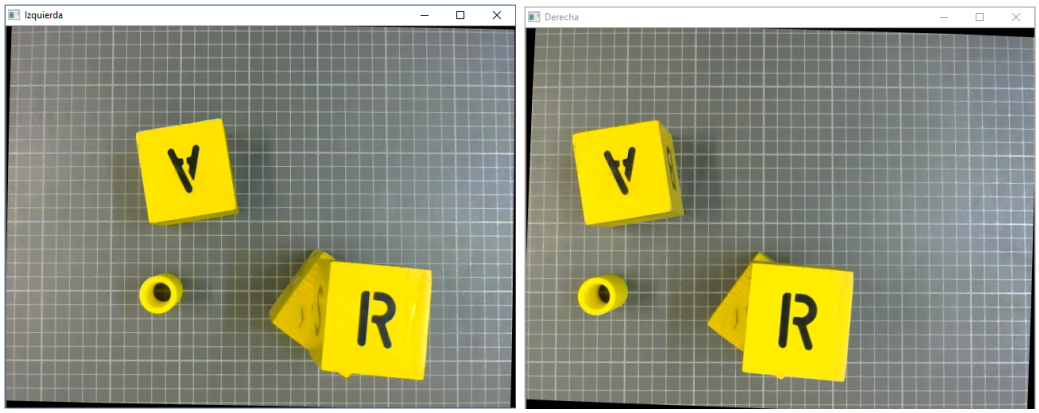
$$\begin{aligned} Distancia_{Cilindro} &= Altura_{Plataforma} - Altura_{Sistema} - \\ Altura_{Cilindro} &= 490 - 24{,}2 - 40{,}1 = 425{,}7mm \end{aligned} \tag{5.8}$$

A continuación ejecutamos el programa principal. Las diferentes partes en las que se va a dividir el programa principal y sobre las que se va a realizar un análisis de los resultados son:

- Rectificación y corrección de distorsiones.

- Máscara individual.
- Mapa de disparidad antes y después de la combinación con la máscara.
- Cálculo de la distancia.

Los valores que permiten llevar a cabo la rectificación de las imágenes y corrección de las distorsiones son calculados de la misma forma e independientemente de los objetos de la escena. Debido a esto, el grado de satisfacción obtenido con los resultados tendría que ser el mismo que en el capítulo anterior. En figura 5.24, se pueden observar los resultados del proceso de rectificación y corrección.



(a) Imagen izquierda (b) Imagen derecha

Figura 5.24: Resultado de la rectificación de las imágenes y corrección de las distorsiones de las lentes.

Como se puede observar, los resultados son los esperados. Los objetos de la escena se encuentran alineados horizontalmente de forma que se eliminan las disparidades en el eje vertical, y solamente se encuentran disparidades en el eje horizontal.

La figura 5.25 se corresponde con el resultado obtenido de realizar el filtrado de color tanto a la imagen derecha como izquierda para posteriormente juntarles. Como se ha mencionado en capítulos anteriores la iluminación juega un papel muy importante a la hora de filtrar un color. El hecho de que la iluminación pueda variar puede llegar a ser problemático, al utilizar un rango para los diferentes valores del color reduce los problemas que puedan surgir. Además el usuario dispone de un programa para la determinación del color a filtrar por lo que siempre que quisiera lo podría cambiar.

El mapa de disparidad obtenido a partir de las imágenes de entrada, véase figura 5.26, presenta unos muy buenos resultados para la calidad de las cámaras utilizadas. Un mapa de disparidad de baja calidad es aquel en el que no se distinguen claramente los objetos presentes en la escena, normalmente esto se debe a que no se han seleccionado los valores adecuados de las variables de la clase *StereoSGBM*. Otro aspecto que determina la calidad de un mapa de disparidad es la cantidad de ruido, a mayor cantidad de ruido peor calidad. En el mapa de disparidad obtenido se puede observar una ausencia casi completa de ruido y una distinción clara entre los objetos con algo de ruido en las áreas correspondientes con los bordes de los objetos.

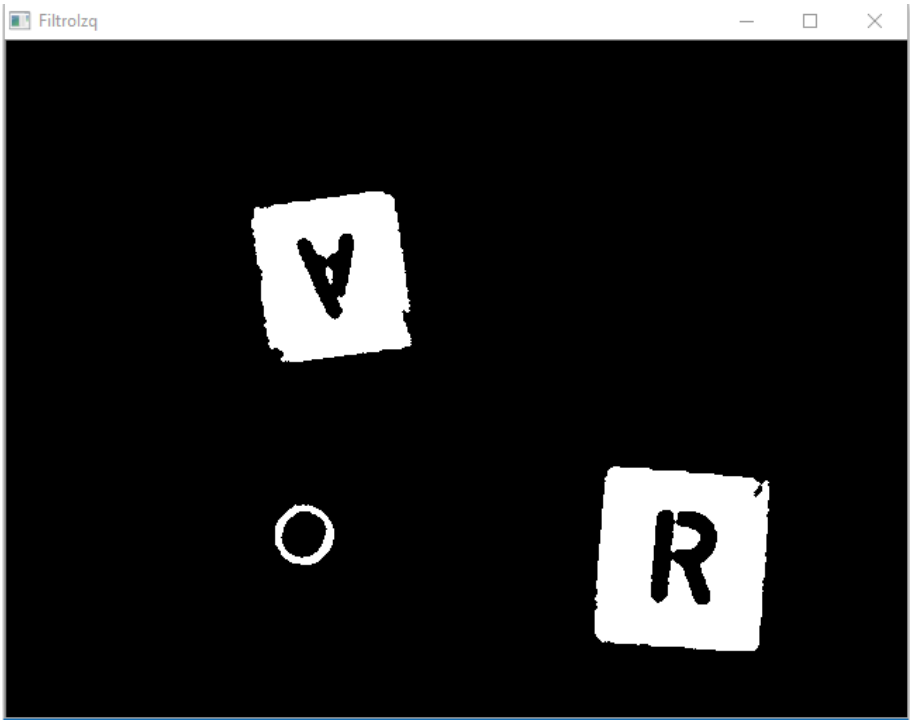


Figura 5.25: Máscara correspondiente a la imagen izquierda.

El mapa de disparidad filtrado se obtiene juntando la máscara y el mapa de disparidades. Sabiendo que la calidad de ambos es elevada cabe esperar que el resultado de la combinación de ambos mantenga el nivel de calidad. El resultado se puede observar en la figura 5.27. La calidad de este ha mejorado, el ruido presente en el mapa de disparidad ha sido eliminado gracias a la combinación con la máscara, véase figura 5.25, ya que esta no tiene nada de ruido.

Para la obtención de la distancias se tiene que hacer clic sobre el mapa de disparidades combinado con la máscara. Cada vez que se haga clic sobre un pixel, se mostrara por consola un valor medio de la distancia de la vecindad de píxeles. Aunque en el mapa la zona se encuentre en negro, lo que supondría una distancia muy elevada puede que no sea así. Ya que al realizar la combinación con la máscara tan solo se ponen en negro aquellas zonas que no se corresponden con el color seleccionado.

Para la obtención de las distancias para cada uno de los elementos de la escena se ha decidido realizar 5 mediciones, con el fin de eliminar algún posible error puntual, como ha ocurrido en la última medición realizada al cilindro. En este caso el valor obtenido no es muy diferente del resto de valores, permitiendo el descarte de un error del algoritmo para la realización de correspondencias estereoscópicas. Este error se debe a que el pixel seleccionado se encuentra muy cerca del borde del objeto y al realizarse una media de la distancia de una cuadrado de 20 píxeles de lado, los píxeles no correspondientes con el objeto han provocado la modificación del valor obtenido. En este caso, para evitar trabajar con estos datos se eliminan los valores máximos y mínimos para cada objeto y se hace la media de los tres valores restantes. Por lo tanto los valores son:

Figura 5.26: Mapa de disparidades.

- Dos prismas apilados.

$$DistanciaFinal_{DosPrismas} = 358{,}3mm \tag{5.9}$$

- Único prisma.

$$DistanciaFinal_{UnPrisma} = 409{,}2mm \tag{5.10}$$

- Cilindro.

$$DistanciaFinal_{Cilindro} = 424{,}8mm \tag{5.11}$$

Para comprobar la calidad de los resultados obtenidos es necesario calcular el error relativo y absoluto de las mediciones. Cuanto menor sean ambos errores mejor sera la calidad de los resultados obtenidos. El error absoluto es la diferencia entre el valor calculado y el valor obtenido. El error relativo es el error relativo entre el valor calculado. Las distancias obtenidas para cada elemento de la escena pueden verse en la figura 5.28.

El error absoluto es:

- Dos prismas apilados.

$$ErrorAbs_{DosPrismas} = |Distancia_{Manual} - Distancia_{Programa}| = |353 - 358{,}3| = 5{,}3mm \tag{5.12}$$

- Único prisma.

$$ErrorAbs_{UnPrisma} = |Distancia_{Manual} - Distancia_{Programa}| = |409{,}4 - 409{,}2| = 0{,}2mm \tag{5.13}$$

Figura 5.27: Mapa de disparidades combinado con la máscara.

- Cilindro.

$$ErrorAbs_{Cilindro} = |Distancia_{Manual} - Distancia_{Programa}| = |425{,}7 - 424{,}8| = 0{,}9mm \tag{5.14}$$

El error relativo es:

- Dos prismas apilados.

$$ErrorRel_{DosPrismas} = \frac{ErrorRel_{DosPrismas}}{Distancia_{Manual}} \cdot 100 = \frac{5{,}3}{353} \cdot 100 = 1{,}5\,\% \tag{5.15}$$

- Único prisma.

$$ErrorRel_{UnPrisma} = \frac{ErrorRel_{UnPrisma}}{Distancia_{Manual}} \cdot 100 = \frac{0{,}2}{409{,}4} \cdot 100 = 0{,}04\,\% \tag{5.16}$$

- Cilindro.

$$ErrorRel_{Cilindro} = \frac{ErrorRel_{Cilindro}}{Distancia_{Manual}} \cdot 100 = \frac{0{,}9}{425{,}7} \cdot 100 = 0{,}21\,\% \tag{5.17}$$

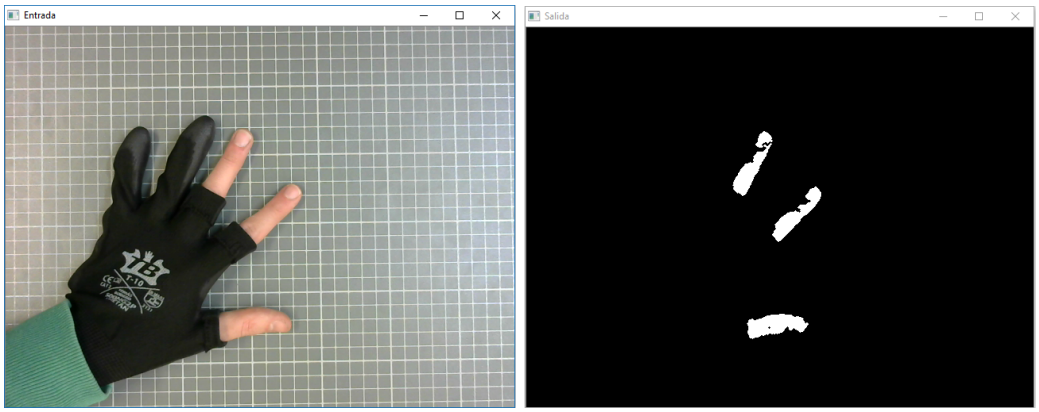
El hecho de que los valores obtenidos para los diferentes errores relativos sean tan pequeños indica la obtención de unos resultados muy precisos.

Como se dijo anteriormente, realizar los cálculos de la distancia de forma manual cuando el objeto de la escena es una mano puede resultar complicado y poco preciso. Al querer comprobar la calidad de los resultados obtenidos, trabajar con una mano puede suponer un aumento del error obteniendo en el análisis de la calidad. El objetivo del sistema de visión es el cálculo de la distancia a la que se encuentran los dedos de la

```
Distancia: [358.09, 0, 0, 0] mm
Distancia: [358.098, 0, 0, 0] mm
Distancia: [358.627, 0, 0, 0] mm
Distancia: [358.194, 0, 0, 0] mm
Distancia: [358.603, 0, 0, 0] mm
Distancia: [409.597, 0, 0, 0] mm
Distancia: [410.121, 0, 0, 0] mm
Distancia: [409.011, 0, 0, 0] mm
Distancia: [408.951, 0, 0, 0] mm
Distancia: [408.654, 0, 0, 0] mm
Distancia: [424.595, 0, 0, 0] mm
Distancia: [424.242, 0, 0, 0] mm
Distancia: [424.779, 0, 0, 0] mm
Distancia: [425.003, 0, 0, 0] mm
Distancia: [612.933, 0, 0, 0] mm
```

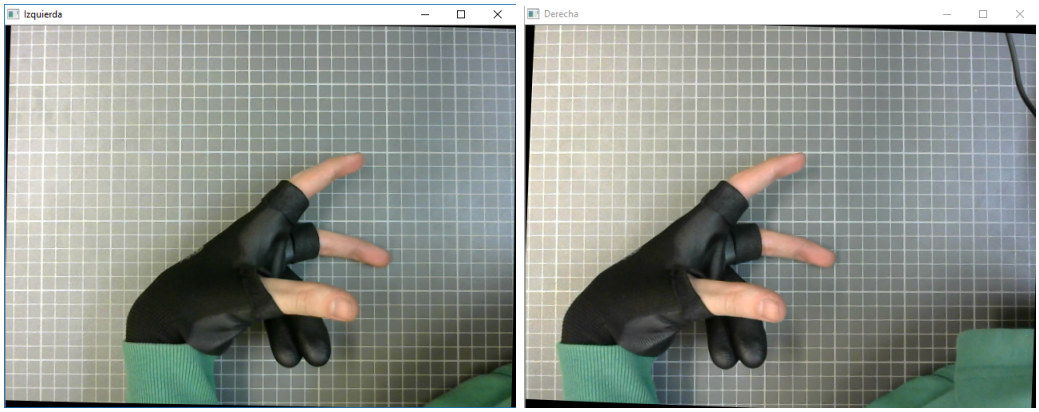
Figura 5.28: Resultado mostrados por consola.

mano del cirujano. Para comprobar que el sistema cumple con su objetivo se va a probar utilizando una escena donde haya una mano presente. Para identificar los dedos de la mano ha sido necesario cambiar el color del filtro.



(a) Entrada. (b) Salida.

Figura 5.29: Calibrado del filtro de color.



(a) Imagen izquierda. (b) Imagen derecha.

Figura 5.30: Imágenes de entrada obtenidas después del remapeado.

(a) Antes. (b) Después.

Figura 5.31: Mapas de disparidad antes y después de combinarlo con la máscara.

```
Distancia: [435.992, 0, 0, 0] mm
Distancia: [437.053, 0, 0, 0] mm
Distancia: [435.702, 0, 0, 0] mm
Distancia: [437.127, 0, 0, 0] mm
Distancia: [437.87, 0, 0, 0] mm
Distancia: [437.901, 0, 0, 0] mm
Distancia: [436.091, 0, 0, 0] mm
Distancia: [436.876, 0, 0, 0] mm
Distancia: [436.086, 0, 0, 0] mm
Distancia: [439.955, 0, 0, 0] mm
Distancia: [394.851, 0, 0, 0] mm
Distancia: [395.194, 0, 0, 0] mm
Distancia: [395.966, 0, 0, 0] mm
Distancia: [396.004, 0, 0, 0] mm
Distancia: [395.315, 0, 0, 0] mm
Distancia: [395.572, 0, 0, 0] mm
Distancia: [396.571, 0, 0, 0] mm
Distancia: [396.028, 0, 0, 0] mm
Distancia: [374.031, 0, 0, 0] mm
Distancia: [374.092, 0, 0, 0] mm
Distancia: [374.049, 0, 0, 0] mm
Distancia: [375.075, 0, 0, 0] mm
Distancia: [375.102, 0, 0, 0] mm
Distancia: [374.271, 0, 0, 0] mm
Distancia: [375.254, 0, 0, 0] mm
Distancia: [375.486, 0, 0, 0] mm
Distancia: [375.354, 0, 0, 0] mm
Distancia: [375.777, 0, 0, 0] mm
```

Figura 5.32: Distancia obtenida para cada uno de los dedos.

Como se puede observar en las figuras 5.30, 5.31 y 5.32, el resultado de la ejecución del programa es totalmente positivo. Correspondiéndose los resultados obtenidos para las distancias con las distancias reales.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

En el desarrollo del presente trabajo de fin de grado se ha desarrollado un sistema de visión artificial para un entorno robotizado de cirugía laparoscópica asistida. El objetivo final es la dotación de un sistema de guiado a un robot quirúrgico para asistir al cirujano en operaciones de cirugía laparoscópica. El sistema es capaz de obtener la tercera dimensión de la escena en tiempo real a partir de las imágenes obtenidas mediante dos cámaras.

En la realización del sistema se ha empleado la biblioteca de funciones, OpenCV, lo que ha supuesto una gran ayuda permitiendo utilizar algoritmos optimizados aumentando la velocidad de ejecución del programa y facilitando la realización de este. Gracias a la gran cantidad de documentación que se puede encontrar en la red acerca de los diferentes algoritmos ha sido posible la realización de una aplicación robusta y que opera en tiempo real.

Aparte de la realización de una aplicación que cumpliese con los requisitos como la reconstrucción 3D de la escena y el tiempo de ejecución se ha prestado especial atención a la realización de un código fácil de entender y utilizar, y esto se ha conseguido mediante la modularidad. El código es modular ya que consta de una serie de programas que proporcionan los valores de las variables utilizadas en el programa principal pero la funcionalidad de estos es independiente de la ejecución del programa principal, pudiendo ser utilizados de forma independiente.

Los programas creados son tres y permiten realizar las siguientes acciones:

- El primer programas *calibracionCamaras* permite obtener los valores para realizar el rectificado de las distorsiones producidas por las lentes de las cámaras y alinea las lineas epipolares.
- El segundo programas es *filtroColor*. Este programa permite la selección de un color a partir de una imagen mediante diferentes métodos.
- El tercer programas *calibracionSGBM* permite la calibración de los parámetros del algoritmo encargado de realizar la calibración estereoscópica de una forma fácil y visual.

Todos los valores obtenidos mediante estos se almacenan en diferentes ficheros. Permitiendo la utilización de estos en un futuro desarrollo del sistema o de otro con características

similares.

El proceso de calibración realizado por el programas *calibracionCamaras* permite la obtención de resultados de muy alta calidad. Aunque para la obtención de estos resultados se necesita realizar el proceso de toma de imágenes perfecto, donde la iluminación es correcta, la escena esta inmóvil, etc.

Los resultados obtenidos en el filtrado del color son muy precisos, permitiendo la identificación y selección del color elegido por el usuario. En algún caso podrían aparecer errores en el filtrado, estos errores son debidos a la iluminación de la escena o a la incorrecta selección del color por parte del usuario.

El proceso de calibración del algoritmo encargado de obtener el mapa de disparidades influye de forma directa con la calidad del mapa de disparidades obtenido. Realizando el proceso de calibración correctamente y seleccionando los valores óptimos para los distintos parámetros, el mapa de disparidades obtenido tiene una gran resolución. Permitiendo identificación de variaciones de la profundidad muy pequeñas.

La precisión de los resultados obtenidos en los procesos anteriores es inversamente proporcional al error cometido en el cálculo de la tercera dimensión. Si los resultados obtenidos en los procesos anteriores son erróneos o la precisión de estos es muy baja los resultados obtenidos en el cálculo de la tercera dimensión presentaran errores que impedirían el uso del sistema. Los resultados obtenidos en el cálculo de distancias, cuando los resultados de los procesos anteriores son de gran precisión, son inferiores al 1.5 %. Teniendo en cuenta las cámaras utilizadas y el algoritmo de correspondencia estereoscópica, el grado de precisión alcanzado es propio de sistemas profesionales.

Para concluir, se puede decir que se han alcanzado los objetivos inicialmente planteados en el proyecto pues se ha logrado un sistema de visión con la precisión y velocidad que requiere un robot colaborativo en un entorno de cirugía laparoscópica.

6.2. Líneas futuras

Debido a los avances que se están produciendo dentro del campo de la visión artificial, de la inteligencia artificial y de la informática a nivel de software, cualquier sistema desarrollado a día de hoy en este ámbito puede encontrarse casi obsoleto en un periodo de tiempo relativamente corto. Esto es producido por una serie de factores como el aumento de la capacidad de procesamiento de datos, la mejora de las cámaras utilizadas, etc. Por lo que este proyecto se podrá mejorar con el paso del tiempo.

A día de hoy las vías a desarrollar del sistema de visión y desde mi punto de vista son dos:

- La primera de ellas consiste en la representación de los puntos mediante una nube de puntos, permitiendo la creación de escenas en tres dimensiones. El modelado mediante nubes de puntos es algo relativamente novedoso pero ya existen algunas bibliotecas para el tratamiento de nubes de puntos como por ejemplo, Point Cloud Library(PCL). Esta biblioteca de funciones, PCL, tiene algoritmos que permiten: obtener las características de la escena, la reconstrucción de las superficies, creación de modelos matemáticos que se ajusten de forma óptima a los puntos, segmentar

una imagen obteniendo las diferentes partes de esta, etc. Esta se puede implementar mediante el lenguaje de programación C++.

- La segunda vía a desarrollar es el reconocimiento de las acciones realizadas por el cirujano. Esto se podría conseguir haciendo uso de la biblioteca de funciones para el tratamiento de nubes de puntos y de la inteligencia artificial. Permitiría conseguir una iluminación de la zona de trabajo mucho mejor ya que la orientación del robot dependería de la acción a realizar.

- La tercera vía sería la obtención de un algoritmo de calibración más robusto. Los resultados obtenidos con el sistema de visión están muy condicionados por los resultados obtenidos durante el proceso de calibración y al ser realizado mediante una toma de imágenes de forma manual la posibilidad de cometer errores aumenta.

En este trabajo de fin de grado se ha conseguido la distinción de tres dedos de la mano del cirujano, junto con las coordenadas espaciales de cada punto que forma parte de alguno de los dedos. El tratamiento de puntos en el espacio se hace mucho más sencillo al utilizar bibliotecas orientadas al trabajo con nubes de puntos como la Point Cloud Library. Mediante el uso de las funciones de la PCL se puede realizar la reconstrucción de la superficie correspondiente a cada uno de los dedos y trabajar con el conjunto de puntos que forman un dedo como si de un solo punto se tratase. Permitiendo obtener una serie de características propias de cada dedo.

Sabiendo la posición y los movimientos de cada uno de los dedos del cirujano, y junto con el uso de la inteligencia artificial se podrían identificar las diferentes acciones realizadas por el cirujano como una incisión o una sutura. Para ello se tendría que realizar un proceso de entrenamiento del sistema donde el asociaría movimientos y posiciones de los dedos con acciones. Este aprendizaje sería el principal inconveniente del sistema ya que se necesitaría una gran cantidad ejemplos de realización para cada una de las acciones a identificar. Una vez que se hubiese realizado el entrenamiento del sistema, este debería ser capaz de identificar la acción que esta realizando el cirujano y para realizar la orientación del brazo robótico de forma optima obteniendo una mejora en la iluminación del área de trabajo.

La identificación de las acciones realizadas por el cirujano permitiría el desarrollo de sistemas más potentes en cuanto a funcionalidad. Si un sistema es capaz de identificar las diferentes acciones que puede realizar un cirujano y almacenar estas en un orden distinto dependiendo del tipo de cirugía a realizar sobre el paciente. En caso de error en el orden de las acciones el sistema sería capaz de detectarlo y avisar al cirujano del error que esta cometiendo, también podría detectar unos movimientos o una posición inadecuada cuando se esta realizando una de las acciones. Esto sería de gran ayuda en las operaciones de larga duración donde el cansancio del cirujano puede provocar una perdida de precisión en las acciones o la realización de estas de forma incorrecta.

Otra mejora se podría llevar a cabo en el proceso de calibración utilizado el usuario tiene que tomar una serie de imágenes de forma manual de un patrón desde diferentes puntos de vista. Esta toma de imágenes no esta normalizada o estandarizada ya que en un principio no es necesario pero una estandarización del proceso permitiría la obtención de mejores resultados y la eliminación de posibles errores en el proceso de calibración.

Bibliografía

[1] Adrian Kaebler, Gary Bradski. *Learning OpenCV. Computer Vision in C++ with the OpenCV Library*, 2008.

[2] American Society of Colon and Rectal Surgeons, ASCRS. *Cirugía Laparoscópica*. Recuperado de `https://www.fascrs.org/cirugia-laparoscopica`.

[3] Armando Ramon Iturralde, Tania González, Mariano Castillo. *Cirugía Urológica de mínimo acceso*, 2010.

[4] Arturo de la Escalera. *Visión por computador, fundamentos y métodos.*

[5] A. Amodeo, A. Linares. *Robotic laparoscopic surgery: cost and training*, 2009.

[6] A.I. Makki. *Needles & Pins*, 2006.

[7] A. Serrano. *Historia de la cirugía laparoscópica*, 2011.

[8] Bernd Jähne, Horst Haubecker, Peter Geibler. *Handbook of computer vision and applications. Volume 1, Sensors and Imaging*, 1999.

[9] Bernd Jähne, Horst Haubecker. *Computer Vision and Applications. A guide for stidents and practitioners*, 2000.

[10] Cristian Daniel Ortega, Fernando Ernesto Moyano. *Técnicas de Implementación de Visión Estereoscópica en Robótica*, 2013

[11] E. Flora, T.G. Wilson, I.J. Martin. *Annals of surgery. A review of natural orifice transluminal endoscopy surgery for intraabdominal surgery: experimental models, techniques and applicability to the clinical setting*, 2008.

[12] Gonzalo Pajaras Martinsanz, Jesús M. de la Cruz García. *Visión por computador. Imágenes digitales y aplicaciones.*

[13] Grzegorz S. Litynski. *Highlights in the History of Laparoscopy.*

[14] Heiko Hirschmuller. *Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information*

[15] Heiko Hirschmuller. *Stereo processing by semiglobal matching and mutual information. Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2008.

[16] Humenberger, C.Zinnern, M.Webber, W.Kubinger, M.Vincze. *A fast stereo matching algorithm suitable for embeddesd real-time systems. Computer Vision and Imagen Understanding.*

[17] Ingrid Hehmeyer, Aliya Khan. *Islam's forgotten contributions to medical science*, 2007.

[18] Instituto de Ingeniería Eléctrica. *Segmentación y posicionamiento 3D en la interpretación del candombe.* Recuperado de `http://iie.fing.edu.uy/investigacion/grupos/gti/timag/trabajos/2014/candombe/calibracion.html`.

[19] Intel Corporation. *Información acerca de la biblioteca OpenCV 3.1, para C++.* Recuperado de `http://docs.opencv.org/3.1.0/`.

[20] José Miguel Guerrero Hernández, Gonzalo Pajares Martinsanz, María Guijarro Mata-García. *Técnicas de procesamiento de imágenes estereoscópicas.*

[21] Josep Isern González. *Estudio experimental de métodos de calibración y autocalibración de cámaras*, 2003.

[22] Journal of the Society of Laparoendoscopic Surgeons, JSLS. *The Evolution of Laparoscopy and the Revolution in Surgery in the Decade of the 1990s.* Recuperado de `https://www.fascrs.org/cirugia-laparoscopica`.

[23] Journal of the Society of Laparoendoscopic Surgeons, JSLS. *Hand-Assisted Laparoscopic Surgery - HALS.* Recuperado de `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3015432/`.

[24] J.Y. Bouguet. *MATLAB calibration tool.*. Recuperado de `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[25] Kurt Semm. *Endoscopic intraabdominal surgery*, 1984.

[26] Linda Shapiro, Gerore Stockman. *Computer Vision.*

[27] Luis Joyanes Aguilar. *Programación en algortimos, estructuras de datos y objetos*, 1999.

[28] Mariano Perez. *Historia de la cirugía laparoscópica y de la terapia mínimamente invasiva*, 2005.

[29] Martín Montalvo Martínez. *Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena*, 2010.

[30] Rafael Aracil, Luis M. Jiménez, Oscar Reinoso, César Fernández. *Estudio de algoritmos de cálculo de disparidad para el control de convergencia en sistemas estereoscópicos.*

[31] Reinhard Klette. *Concise Computer Vision. An introduction into theory and algorithms*, 2014.

[32] Robert Laganière. *OpenCV 2. Computer Vision Application Programming Cookbook*, 2011.

[33] Roberto Rodriguez Morales, Juan Humbarte Sossa Azuela. *Procesamiento y análisis digital de imágenes.*

[34] Samarth Brahmbhatt. *Practical OpenCV.*

[35] S.J. Spaner, G.L. Warnock. *A Brief History of Endoscopy, Laparoscopy, and Laparoscopic Surgery*, 2009.

[36] Tzay Y. Young, King Sun Fu. *Handbook of pattern recognition and image processing*, 1986.

[37] Unai Mujika Torrontegi. *Reconstrucción densa de modelos tridimensionales utilizando Visión Artificial*, 2010.

[38] I. Yoshino, M. Hashizume, M. Shimada. *Journal of Thoracic and Cardiovascular Surgery*, 2001.

[39] Z.Zhang. *A flexible new technique for camera calibration*, 1998.

Anexo I: Código

Calibración de las cámaras

```
1 /// Programa: Calibracion
2 /// Descripcion: Realiza la calibracion de dos camaras para su utilizacion
3 /// en un sistema de vision estereoscopica. La calibracion se realiza
4 /// mediante un patron, un tablero de ajedrez, el cual es imprescindible
5 /// para poder calibrar las camaras.
6 /// Autor: Carlos Castedo Hernandez
7 /// Fecha: 12/03/2017
8
9 #include <iostream>
10 #include <iomanip>
11 #include <stdio.h>
12 #include <opencv2/opencv.hpp>
13 #include <opencv2/highgui/highgui.hpp>
14 #include <opencv2/calib3d/calib3d.hpp>
15 #include "../include/config.h"
16
17 using namespace cv;
18 using namespace std;
19
20 int main()
21 {
22     /// Calibracion
23     vector<Mat> izq_imagenes, dcha_imagenes;              // Vectores de Mats
24     Mat combo(FRAME_HEIGHT, 2 * FRAME_WIDTH, CV_8UC3);   // Imagen para
    visualizacin
25
26     // Comprobacion imagenes residuales
27     string filepath_izda = "../Data/Imagenes/Izquierda1.jpg";
28     string filepath_dcha = "../Data/Imagenes/Derecha1.jpg";
29
30     Mat dch = imread(filepath_dcha.c_str(),IMREAD_COLOR);
31     Mat izq = imread(filepath_izda.c_str(),IMREAD_COLOR);
32     if(!izq.empty() || !dch.empty()){
33         cout << "Borre las imagenes de la carpeta Data/Imagenes" << endl;
34         return 0;
35     }
36
37     VideoCapture capIzda(1), capDcha(0);     // Declaracin  c maras
38
39     // Set tamao imagenes capturadas
40     capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
41     capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
42     capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
43     capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
44
45     // Comprobacin imagen de  c maras
```

```
46      if(!capIzda.open(1) || !capDcha.open(0)){
47          cout << "Error abrir camaras" << endl;
48          return 0;
49      }
50
51      namedWindow("Entrada"); // Ventana imagenes entrada
52
53      int i = 0;       // Nmero de imagen
54      char key = 'z'; // Key
55
56      // Instrucciones usuario
57      cout << "Pulse SPACIO para guardar las imagenes." << endl
58           << "Pulse ESC para finalizar el programa." << endl << endl
59           << "RECUERDE: pulse la tecla siempre que tenga seleccionada" <<
        endl
60           << "cualquiera de las ventanas Izquierda o Derecha, si " << endl
61           << "intenta escribir en la ventana de comandos no funcionara." <<
        endl;
62
63      // Bucle de trabajo
64      while(key != 27){
65          // Adquisicin fotogramas
66          capIzda.grab();
67          capDcha.grab();
68
69          // Asignacin fotogramas a Mats
70          Mat izqFrame, dchaFrame;                    // Imagen izquierda y
        derecha
71          capIzda.retrieve(izqFrame);
72          capDcha.retrieve(dchaFrame);
73
74          if(dchaFrame.empty() || izqFrame.empty()){
75          cout << "izqFrame o dchaFrame vacia" << endl;
76          return 0;
77          }
78
79          flip(dchaFrame,dchaFrame,-1);
80
81          // Combinacion imagen izquierda + derecha
82          izqFrame.copyTo(combo(Range::all(), Range(0, FRAME_WIDTH)));
83          dchaFrame.copyTo(combo(Range::all(), Range(FRAME_WIDTH, 2*
        FRAME_WIDTH)));
84
85          imshow("Entrada", combo);    // Visualizacion imagenes
86
87          // Guardado de imagenes
88          if(key == ' '){                          // Aumenta nmero de imagen
89              i += 1;
90              stringstream izq_name, dcha_name;    // Declaracin nombres
        imagenes
91              izq_name << "Izquierda" << i << ".jpg";
92              dcha_name << "Derecha" << i << ".jpg";
93
94              // Escritura de imagenes
95              imwrite(string("../Data/Imagenes/")+izq_name.str(),izqFrame);
96              imwrite(string("../Data/Imagenes/")+dcha_name.str(),dchaFrame);
97
98              izq_imagenes.push_back(izqFrame);      // A adir imagen vector
        imagenes
99              dcha_imagenes.push_back(dchaFrame);
100
```

```
101             cout << "Imagenes " << i << "\tguardadas" << endl;  // Feedback
    usunario
102         }
103         key = (char)waitKey(30);     // Lectura de pulsacion teclado
104     }
105
106     destroyAllWindows();
107
108
109     char visIma;
110     cout << "\nQuiere visualizar imagenes guardadas (y/n): ";
111     cin >> visIma;
112     if(visIma =='y'){
113         namedWindow("Imagenes capturadas");
114         for(unsigned i = 0; i < izq_imagenes.size(); i++)
115         {
116             izq_imagenes[i].copyTo(combo(Range::all(), Range(0, FRAME_WIDTH)
    ));
117             dcha_imagenes[i].copyTo(combo(Range::all(), Range(FRAME_WIDTH,
    2*FRAME_WIDTH)));
118
119             imshow("Imagenes capturadas",combo);
120             waitKey(0);
121         }
122         destroyAllWindows();
123     }
124
125
126     cout << "Imagenes leidas: " << izq_imagenes.size() << endl;
127
128     float tam_lado = 18.f;  // En milimetros
129     int esq_int_ancho = 9;  // Numero de esquinas ancho
130     int esq_int_alto = 5;   // Numero de esquinas alto
131     char opcion;
132
133     // Posible modificacion de valores por usuario
134     cout << "\nTamanio lado del cuadrado en mm por defecto = " << tam_lado
    << endl
135          << "Numero de esquinas ancho por defecto = " << esq_int_ancho <<
    endl
136          << "Numero de esquinas alto por defecto = " << esq_int_alto << endl
137          << "Desea cambiar los valores(y/n): ";
138     cin >> opcion;
139
140     if(opcion == 'y'){
141         cout << "Introduzca tamanio lado en mm: ";
142         cin >> tam_lado;
143
144         cout << "Introduzca numero de esquinas ancho: ";
145         cin >> esq_int_ancho;
146
147         cout << "Introduzca numero de esquinas alto: ";
148         cin >> esq_int_alto;
149     }
150
151     vector<Point3f> ob_p;
152     for(int i = 0; i < esq_int_alto; i++) {
153         for(int j = 0; j < esq_int_ancho; j++) {
154             ob_p.push_back(Point3f(j * tam_lado, i * tam_lado, 0.f));
155         }
156     }
```

```
157
158     cout << "\nDesea observar el resultado de la busqueda de esquinas(y/n): ";
159     cin >> opcion;
160
161     Mat izq_ima, dcha_ima;                          // Imagen izquierda y derecha
162     vector<vector<Point2f> > izq_ima_puntos, dcha_ima_puntos;
163     vector<vector<Point3f> > puntos_objetivo;
164     for(unsigned i = 0; i < izq_imagenes.size(); i++) {
165         // Asignacion de imagenes
166         izq_ima = izq_imagenes[i];
167         dcha_ima = dcha_imagenes[i];
168
169         vector<Point2f> izq_ima_p, dcha_ima_p;  // Vectores(x,y), almacenar posiciones esquinas
170
171         // Busqueda esquinas
172         bool izq_patron_found = findChessboardCorners(izq_ima, Size(esq_int_ancho, esq_int_alto), izq_ima_p, CALIB_CB_ADAPTIVE_THRESH + CALIB_CB_NORMALIZE_IMAGE+ CALIB_CB_FAST_CHECK);
173         bool dcha_patron_found = findChessboardCorners(dcha_ima, Size(esq_int_ancho, esq_int_alto), dcha_ima_p, CALIB_CB_ADAPTIVE_THRESH + CALIB_CB_NORMALIZE_IMAGE+ CALIB_CB_FAST_CHECK);
174
175         if(izq_patron_found && dcha_patron_found) {
176             puntos_objetivo.push_back(ob_p);
177             Mat gray;                                   // Imagen gris
178             cvtColor(izq_ima, gray, CV_BGR2GRAY);   // Convertir imagen izquierda de RGB a GRIS
179             cornerSubPix(gray, izq_ima_p, Size(5, 5), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS + CV_TERMCRIT_ITER, 30, 0.1));
180             cvtColor(dcha_ima, gray, CV_BGR2GRAY);  // Convertir imagen derecha de RGB a GRIS
181             cornerSubPix(gray, dcha_ima_p, Size(5, 5), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS + CV_TERMCRIT_ITER, 30, 0.1));
182             izq_ima_puntos.push_back(izq_ima_p);    // A adir vector puntos(x,y)
183             dcha_ima_puntos.push_back(dcha_ima_p);  // A adir vector puntos(x,y)
184
185             if(opcion == 'y'){
186                 stringstream izq_corner, dcha_corner;   // Declaracin nombres imagenes
187                 izq_corner << "izq_corner" << i << ".jpg";
188                 dcha_corner << "dcha_corner" << i << ".jpg";
189
190                 Mat im_show = izq_ima.clone();
                                    // Creacin copia imagen izquierda
191                 drawChessboardCorners(im_show, Size(esq_int_ancho, esq_int_alto), izq_ima_p, true);     // Dibujar en imagen las esquinas
192                 im_show.copyTo(combo(Range::all(), Range(0, FRAME_WIDTH)));
193                 imwrite(string("../Data/Imagenes/")+izq_corner.str(),im_show);
194
195                 im_show = dcha_ima.clone();
                                    // Creacin copia imagen derecha
196                 drawChessboardCorners(im_show, Size(esq_int_ancho, esq_int_alto), dcha_ima_p, true);     // Dibujar en imagen las esquinas
197                 im_show.copyTo(combo(Range::all(), Range(FRAME_WIDTH, 2*FRAME_WIDTH)));
```

```
198                 imwrite(string("../Data/Imagenes/")+dcha_corner.str(),
    im_show);
199
200                 namedWindow("Combo Imagenes Leidas");    // Declaracion
    ventana
201                 imshow("Combo Imagenes Leidas",combo);   // Mostrar imagen
202
203                 cout << "Pulse cualquier tecla para continuar..." << endl;
204                 waitKey(0);
205             }
206         } else {
207             izq_imagenes.erase(izq_imagenes.begin() + i);
208             dcha_imagenes.erase(dcha_imagenes.begin() + i);
209         }
210     }
211     destroyAllWindows();
212
213     /// Rectificacion
214     Mat cameraMatrix, distCoeffs;
215     vector<Mat> rvecs, tvecs;
216     cout << "\nError cometido en calibracion" << endl;
217
218     float rms_error_1 = calibrateCamera(puntos_objetivo, izq_ima_puntos,
    izq_imagenes[0].size(), cameraMatrix, distCoeffs, rvecs, tvecs);
219     cout << "RMS error 1" << rms_error_1 << endl;
220
221     Mat R, T, E, F;
222     double rms_error_2 = stereoCalibrate(puntos_objetivo,izq_ima_puntos,
    dcha_ima_puntos,cameraMatrix,distCoeffs,cameraMatrix,distCoeffs,
    izq_imagenes[0].size(),R,T,E,F);
223     cout << "RMS error 2" << rms_error_2 << endl;
224
225     Mat Rl, Rr, Pl, Pr, Q;
226     Size imagen_size(FRAME_WIDTH, FRAME_HEIGHT);
227     stereoRectify(cameraMatrix,distCoeffs,cameraMatrix,distCoeffs,
    imagen_size,R,T,Rl,Rr,Pl,Pr,Q);
228
229     Mat map_l1, map_l2, map_r1, map_r2;
230     initUndistortRectifyMap(cameraMatrix,distCoeffs, Rl,Pl,imagen_size,
    CV_16SC2, map_l1,map_l2);
231     initUndistortRectifyMap(cameraMatrix,distCoeffs, Rr,Pr,imagen_size,
    CV_16SC2, map_r1,map_r2);
232
233     cout << "\nDesea visualizar la calibracion y rectificacion(y/n): ";
234     cin >> opcion;
235
236     if(opcion == 'y'){
237         destroyAllWindows();     // Destruir todas las ventanas
238
239         VideoCapture capIzda(1), capDcha(0);     // Declaracion camaras
240
241         // Set tamao imagenes capturadas
242         capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
243         capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
244         capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
245         capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
246
247         namedWindow("Imagenes rectificadas");    // Nueva ventana
248
249         // Bucle visualizacion imagenes calibradas y rectificadas
250         cout << "\nPulse tecla ESC para salir de la visualizacion" << endl;
```

```
251         int j = 0;
252         while(char(waitKey(1)) != 27) {
253             capIzda.grab();      // Captura de frame camara izquierda
254             capDcha.grab();      // Captura de frame camara derecha
255 
256             Mat framel, framel_rect, framer, framer_rect;
257             capIzda.retrieve(framel);    // Asignacion frame a Mat framel
258             capDcha.retrieve(framer);    // Asignacion frame a Mat framer
259 
260             if(framel.empty() || framer.empty())     break;  // Comprobacion
    Mats son imagenes
261 
262             flip(framer,framer,-1);
263 
264             // Remapeo de las imagenes
265             remap(framel, framel_rect, map_l1, map_l2, INTER_LINEAR);
266             remap(framer, framer_rect, map_r1, map_r2, INTER_LINEAR);
267 
268             // Creacion imagen mayor, combinacion imagen izquierda + derecha
269             Mat combo(FRAME_HEIGHT, 2 * FRAME_WIDTH, CV_8UC3);
270             framel_rect.copyTo(combo(Range::all(), Range(0, FRAME_WIDTH)));
271             framer_rect.copyTo(combo(Range::all(), Range(FRAME_WIDTH, 2*
    FRAME_WIDTH)));
272 
273             // Dibujar lineas horizontales observar rectificado
274             for(int y = 0; y < combo.rows; y += 20)
275                 line(combo, Point(0, y), Point(combo.cols, y), Scalar(0, 0,
    255));
276 
277             imshow("Imagenes rectificadas", combo); // Mostrar imagen
278 
279             stringstream comborec;    // Declaracin nombres imagenes
280             comborec << "comborec" << j << ".jpg";
281 
282             imwrite(string("../Data/Imagenes/PruebaCorreccion")+comborec.str
    (),combo);
283 
284             j++;
285         }
286     }
287 
288     /// Escritura fichero
289     string filecalibracion = "../Data/Calibracion.yml";    // Nombre fichero
290     FileStorage fs(filecalibracion,FileStorage::WRITE);
291     time_t rawtime;
292     time(&rawtime);
293 
294     // Campos escritos en fichero
295     fs << "Date" << asctime(localtime(&rawtime));
296     fs << "cameraMatrix" << cameraMatrix;
297     fs << "distCoeffs" << distCoeffs;
298     fs << "R" << R;
299     fs << "T" << T;
300     fs << "E" << E;
301     fs << "F" << F;
302     fs << "Rl" << Rl;
303     fs << "Rr" << Rr;
304     fs << "Pl" << Pl;
305     fs << "Pr" << Pr;
306     fs << "Q" << Q;
307     fs << "map_l1" << map_l1;
```

```
308     fs << "map_l2" << map_l2;
309     fs << "map_r1" << map_r1;
310     fs << "map_r2" << map_r2;
311
312     fs.release();
313
314     return 0;
315 }
```

Filtro de color

```
1 /// Programa: FiltroColor
2 /// Descripcion: Realiza la seleccion del color a filtrar en la imagen,
3 /// esta se puede realizar mediante sliders o cliqueando en la imagen sobre
4 /// el color que se quiere seleccionar.
5 /// Autor: Carlos Castedo Hernandez
6 /// Fecha: 12/03/2017
7
8 #include "opencv2/core/core.hpp"
9 #include "opencv2/calib3d/calib3d.hpp"
10 #include <opencv2/highgui/highgui.hpp>
11 #include <opencv2/imgproc/imgproc.hpp>
12 #include <stdio.h>
13 #include <string.h>
14 #include <iostream>
15 #include <time.h>
16 #include "../include/config.h"
17
18 using namespace cv;
19 using namespace std;
20
21 /// Variables globales para selector color
22 int pos_x = 0,  // Coordenada x en imagen
23     pos_y = 0;  // Cordenada y en imagen
24 bool make = false;
25
26 void CallBackFunc(int event, int x, int y, int flags, void* userdata);
27 void RGBtoHSV(int r, int g, int b, int &h, int &s, int &v);
28 void HSVtoRGB(int h, int s, int v, int &r, int &g, int &b);
29 void setToZero(int &ha, int &hb, int &hc, int &la, int &lb, int &lc);
30
31 int main()
32 {
33     /// Configuracion camaras
34     VideoCapture cam(0);
35     cam.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
36     cam.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
37
38     /// Seleccion metodo de toma de color
39     cout << "Metodos seleccion del color \n" <<
40             "    Sliders(HSV) -> 1 \n" <<
41             "    Selector color(RGB) -> 2 \n" <<
42             "Elija una opcion: ";
43     int opcion;
44     cin >> opcion;
45
46     Mat element = getStructuringElement( MORPH_ELLIPSE, Size(TAM_STREL,
    TAM_STREL));
47
48     /// Variables del color
49     int H = 0, S = 0, V = 0;
50
51     int hH = 0, hS = 0, hV = 0,
52         lH = 0, lS = 0, lV = 0;
53
54     int R = 0, B = 0, G = 0;
55
56     int hR = 0, hG = 0, hB = 0,
57         lR = 0, lG = 0, lB = 0;
```

```
58
59
60      /// Variables
61      int rangeValue = 10,
62          change_rangeValue = 0,
63          tamDiag = 4,
64          change_tamDiag = 0,
65          grabar = 0;
66
67      namedWindow("Salida");
68      namedWindow("Parametros");
69      namedWindow("Entrada");
70
71      setMouseCallback("Entrada", CallBackFunc, NULL);
72
73      /// Sliders
74      if(opcion == 2){
75          resizeWindow("Parametros", FRAME_WIDTH, 140);
76          createTrackbar("Rango","Parametros", &rangeValue, 127);
77          createTrackbar("Tamanio Muestra","Parametros", &tamDiag, 25);
78          createTrackbar("Grabar","Parametros", &grabar, 1);
79
80      }else if(opcion == 1){
81          resizeWindow("Parametros", FRAME_WIDTH, 325);
82          createTrackbar("High H","Parametros", &hH, 359);
83          createTrackbar("Low H","Parametros", &lH, 359);
84          createTrackbar("High S","Parametros", &hS, 255);
85          createTrackbar("Low S","Parametros", &lS, 255);
86          createTrackbar("High V","Parametros", &hV, 255);
87          createTrackbar("Low V","Parametros", &lV, 255);
88          createTrackbar("Grabar","Parametros", &grabar, 1);
89      }
90
91      setMouseCallback("Entrada", CallBackFunc, NULL);
92
93      while(1){
94          /// Captura de imagen y comprobacion
95          Mat entrada, resultado;
96          cam.retrieve(entrada);
97          if(entrada.empty()) {
98              cout << "Error conexion camara" << endl;
99              break;
100         }
101
102         if(opcion == 1){
103             /// Metodo sliders
104             if(make){
105                 Vec3b colorHSV;
106                 int n_pos_x = pos_x - (tamDiag/2);
107                 int n_pos_y = pos_y - (tamDiag/2);
108
109                 for(int i = 0; i < tamDiag; i++){
110                     for(int j = 0; j< tamDiag; j++){
111                         colorHSV = entrada.at<Vec3b>(n_pos_y,n_pos_x);
112                         H += colorHSV.val[0];
113                         S += colorHSV.val[1];
114                         V += colorHSV.val[2];
115                         n_pos_x += 1;
116                     }
117                     n_pos_y += 1;
118                 }
```

```
119                 H /= tamDiag*tamDiag;
120                 S /= tamDiag*tamDiag;
121                 V /= tamDiag*tamDiag;
122 
123                 hH = H;
124                 lH = H;
125                 hS = S;
126                 lS = S;
127                 hV = V;
128                 lV = V;
129 
130                 // Posicionamiento en valor cliqueado
131                 setTrackbarPos("High H", "Parametros", hH);
132                 setTrackbarPos("Low H", "Parametros", lH);
133                 setTrackbarPos("High S", "Parametros", hS);
134                 setTrackbarPos("Low S", "Parametros", lS);
135                 setTrackbarPos("High V", "Parametros", hV);
136                 setTrackbarPos("Low V", "Parametros", lV);
137 
138                 make = false;
139             }
140 
141             Mat imgHSV;
142             cvtColor(entrada,imgHSV, COLOR_BGR2HSV);
143 
144             if(lH < 180){
145                 Mat  imgBin_179, imgBin_0;              //Img con binariz
    hasta 179 y a partir de 0 resp.
146                 Scalar HSV_179 = Scalar(179,hS,hV);     //Hasta 179
147                 Scalar HSV_0   = Scalar(0,lS,lV);       //De 0 en adelante
148                 inRange(imgHSV, Scalar(lH, lS, lV),  HSV_179, imgBin_179);
149                 //En inRange quitamos 180 para que trabaje en la escala
    0-180 de openCV
150                 inRange(imgHSV, HSV_0, Scalar(hH-180, hS, hV), imgBin_0);
151                 add(imgBin_179, imgBin_0, resultado);
152             }
153             else  //si no nos hemos metido en zona la negativa con Hmin
154             {
155                 //En inRange quitamos 180 para que trabaje en la escala
    0-180 de openCV
156                 inRange(imgHSV, Scalar(lH-180, lS, lV), Scalar(hH-180, hS,
    hV), resultado);
157             }
158         }
159         else if(opcion == 2){
160             /// Metodo selector de color
161             if(make||rangeValue!=change_rangeValue||tamDiag!=change_tamDiag)
    {
162                 Vec3b colorRGB;
163                 int n_pos_x = pos_x - (tamDiag/2);
164                 int n_pos_y = pos_y - (tamDiag/2);
165 
166                 // Calculo valor medio de pixeles
167                 for(int i = 0; i < tamDiag; i++){
168                     for(int j = 0; j< tamDiag; j++){
169                         colorRGB = entrada.at<Vec3b>(n_pos_y,n_pos_x);
170                         B += colorRGB.val[0];
171                         G += colorRGB.val[1];
172                         R += colorRGB.val[2];
173                         n_pos_x++;
174                     }
```

```
175                     n_pos_y++;
176                 }
177
178                 B /= tamDiag*tamDiag;
179                 G /= tamDiag*tamDiag;
180                 R /= tamDiag*tamDiag;
181
182                 // Rango valores
183                 hR = R + rangeValue,
184                 hG = G + rangeValue,
185                 hB = B + rangeValue,
186                 lR = R - rangeValue,
187                 lG = G - rangeValue,
188                 lB = B - rangeValue;
189
190                 if(hR > 255){ hR = 255; }
191                 if(hG > 255){ hG = 255; }
192                 if(hB > 255){ hB = 255; }
193                 if(lR < 0){ lR = 0; }
194                 if(lG < 0){ lG = 0; }
195                 if(lB < 0){ lB = 0; }
196
197                 make = false; // Condicion para que solo cambie con clic
198                 change_rangeValue = rangeValue;
199                 change_tamDiag = tamDiag;
200             }
201             inRange(entrada,Scalar(lB,lG,lR),Scalar(hB,hG,hR),resultado);
202         }else{
203             cout << "Eleccion incorrecta" << endl;
204             return -1;
205         }
206
207         /// Apertura y cierre
208         morphologyEx(resultado,resultado, MORPH_OPEN,  element);
209         morphologyEx(resultado,resultado, MORPH_CLOSE, element);
210
211         /// Visualizacion entrada y salida
212         imshow("Entrada",entrada);
213         imshow("Salida",resultado);
214
215         if(waitKey(30) == 27){
216             if(grabar == 1){
217                 /// Configurar valores del modelo del color
218                 if(opcion == 1){ setToZero(hB,hG,hR,lB,lG,lR); }
219                 else{ setToZero(hH,hS,hV,lH,lS,lV); }
220                 /// Guardado de variables en fichero
221                 string fileColor = DATA_FOLDER + string("ColorDetecta.xml");
222                 FileStorage fs(fileColor,FileStorage::WRITE);
223                 time_t rawtime;
224                 time(&rawtime);
225
226                 fs << "Date" << asctime(localtime(&rawtime));
227                 fs << "hH" << hH;
228                 fs << "hS" << hS;
229                 fs << "hV" << hV;
230                 fs << "lH" << lH;
231                 fs << "lS" << lS;
232                 fs << "lV" << lV;
233                 fs << "hR" << hR;
234                 fs << "hG" << hG;
235                 fs << "hB" << hB;
```

```
236                 fs << "lR" << lR;
237                 fs << "lG" << lG;
238                 fs << "lB" << lB;
239 
240                 fs.release();
241                 cout << "Guardado OK" << endl;
242             }
243             break;
244         }
245     }
246     cam.release();
247     return 0;
248 }
249 
250 void CallBackFunc(int event, int x, int y, int flags, void* userdata)
251 {
252     // Permite obtener las coordenadas del cursor en la imagen cuando
253     // el usuario hace clic izquierdo
254     if( event == EVENT_LBUTTONDOWN ){
255         pos_x = x;
256         pos_y = y;
257         make = true;
258     }
259 }
260 
261 void RGBtoHSV(int r, int g, int b, int &h, int &s, int &v){
262 
263     Mat paletaRGB(20,20,CV_8UC3);
264     Mat paletaHSV(20,20,CV_8UC3);
265 
266     for(int y=0;y<paletaRGB.rows;y++){
267         for(int x=0;x<paletaRGB.cols;x++){
268             Vec3b colorRGB = paletaRGB.at<Vec3b>(Point(x,y));
269             colorRGB.val[0] = b;
270             colorRGB.val[1] = g;
271             colorRGB.val[2] = r;
272             paletaRGB.at<Vec3b>(Point(x,y)) = colorRGB;
273         }
274     }
275 
276     cvtColor(paletaRGB,paletaHSV,CV_BGR2HSV);
277 
278     Vec3b colorHSV = paletaHSV.at<Vec3b>(Point(0,0));
279     h = colorHSV.val[0];
280     s = colorHSV.val[1];
281     v = colorHSV.val[2];
282 }
283 
284 void HSVtoRGB(int h, int s, int v, int &r, int &g, int &b){
285 
286     Mat paletaRGB(20,20,CV_8UC3);
287     Mat paletaHSV(20,20,CV_8UC3);
288 
289     for(int y=0;y<paletaHSV.rows;y++){
290         for(int x=0;x<paletaHSV.cols;x++){
291             Vec3b colorHSV = paletaHSV.at<Vec3b>(Point(x,y));
292             colorHSV.val[0] = h;
293             colorHSV.val[1] = s;
294             colorHSV.val[2] = v;
295             paletaHSV.at<Vec3b>(Point(x,y)) = colorHSV;
296         }
```

```
297     }
298
299     cvtColor(paletaHSV,paletaRGB,CV_HSV2BGR);
300
301     Vec3b colorRGB = paletaRGB.at<Vec3b>(Point(0,0));
302     b = colorRGB.val[0];
303     g = colorRGB.val[1];
304     r = colorRGB.val[2];
305 }
306
307 void setToZero(int &ha, int &hb, int &hc, int &la, int &lb, int &lc){
308     ha = 0;
309     hb = 0;
310     hc = 0;
311     la = 0;
312     lb = 0;
313     lc = 0;
314 }
```

Calibración de stereoSGBM

```
#include "opencv2/core/core.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <iostream>
#include "../include/config.h"

using namespace cv;
using namespace std;

int main()
{
    /// Lectura fichero calibracion
    Mat map_r1, map_r2; //pixel maps para x e y para rectificar imagen Dcha
    Mat map_l1, map_l2; //pixel maps para x e y para rectificar imagen Dcha
    Mat Q;              //necesaria para remapeo 3D

    string fileCalib = "../Data/Calibracion.yml";   // Directorio fichero
    FileStorage fs_cal(fileCalib, FileStorage::READ);
    fs_cal["Q"] >> Q;
    fs_cal["map_l1"] >> map_l1;
    fs_cal["map_l2"] >> map_l2;
    fs_cal["map_r1"] >> map_r1;
    fs_cal["map_r2"] >> map_r2;

    fs_cal.release();

    // Comprobacion lectura valores
    if(map_r1.empty() || map_r2.empty() || map_l1.empty() || map_l2.empty()
    || Q.empty())
    {
        cout << "ERROR al cargar fichero de calibracion" << endl;
        return -1;
    }

    /// Configuracion camaras
    VideoCapture capIzda(1), capDcha(0);
    capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
    capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
    capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
    capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);

    if(!capIzda.open(1) || !capDcha.open(0)){
        cout << "Error abrir camaras" << endl;
        return 0;
    }

    /// Inicializacion StereoSGBM
    int opcion = 0;
    string fileSGBM = "../Data/ParametrosStereo.xml";

    cout << "Seleccione una de las opciones:\n" <<
            "Leer valores de fichero -> 1\n" <<
            "Utilizar valores estandar -> 2\n" <<
            "Opcion: ";
```

```
58
59      cin >> opcion;
60
61
62
63      int minDisparity, numDisparity, SADWindowSize, P1, P2, disp12MaxDiff,
        preFilterCap, uniquenessRatio, speckleWindow, speckleRange;
64      int grabar = 0;
65
66      if(opcion == 2){
67          minDisparity     = 30;
68          numDisparity     = 2;
69          SADWindowSize    = 3;
70          P1               = 8 * 3 * SADWindowSize * SADWindowSize;
71          P2               = 32 * 3 * SADWindowSize * SADWindowSize;
72          disp12MaxDiff    = 1;
73          preFilterCap     = 63;
74          uniquenessRatio  = 10;
75          speckleWindow    = 100;
76          speckleRange     = 32;
77
78      }else if(opcion == 1){
79          FileStorage fs_BM(fileSGBM, FileStorage::READ);
80
81          fs_BM["minDisparity"] >> minDisparity;
82          fs_BM["numDisparity"] >> numDisparity;
83          fs_BM["SADWindowSize"] >> SADWindowSize;
84          fs_BM["P1"] >> P1;
85          fs_BM["P2"] >> P2;
86          fs_BM["disp12MaxDiff"] >> disp12MaxDiff;
87          fs_BM["preFilterCap"] >> preFilterCap;
88          fs_BM["uniquenessRatio"] >> uniquenessRatio;
89          fs_BM["speckleWindow"] >> speckleWindow;
90          fs_BM["speckleRange"] >> speckleRange;
91          fs_BM.release();
92      }else{ return -1;}
93
94      //Inicializacion sliders
95      namedWindow("ValorStereoBM");
96      resizeWindow("ValorStereoBM",FRAME_WIDTH,FRAME_HEIGHT);
97      createTrackbar("minDisp","ValorStereoBM", &minDisparity, 200);
98      createTrackbar("numDisp","ValorStereoBM", &numDisparity, 20);
99      createTrackbar("SADWind","ValorStereoBM", &SADWindowSize, 11);
100     createTrackbar("disp12MaxDiff","ValorStereoBM", &disp12MaxDiff, 100);
101     createTrackbar("preFilterCap","ValorStereoBM", &preFilterCap, 200);
102     createTrackbar("uniquenessRatio","ValorStereoBM", &uniquenessRatio, 50);
103     createTrackbar("speckleWindows","ValorStereoBM", &speckleWindow, 300);
104     createTrackbar("speckleRange","ValorStereoBM", &speckleRange, 100);
105     createTrackbar("Grabar","ValorStereoBM", &grabar, 1);
106
107
108     /// Bucle principal
109     while(1){
110         /// Toma de imagenes
111         // Captura de frames
112         capIzda.grab();
113         capDcha.grab();
114
115         // Asignacion de frames a imagenes
116         Mat izqFrame, dchaFrame;
117         capIzda.retrieve(izqFrame);
```

```
118         capDcha.retrieve(dchaFrame);
119
120         // Comprobacion imagenes obtenidas
121         if(izqFrame.empty() || dchaFrame.empty()){
122             cout << "ERROR. Imagenes de camaras." << endl;
123             return 0;
124         }
125
126         flip(dchaFrame,dchaFrame,-1);
127
128         /// Rectificado
129         Mat frameIzdaRect, frameDchaRect;
130         remap(izqFrame, frameIzdaRect, map_l1, map_l2, INTER_LINEAR);
131         remap(dchaFrame, frameDchaRect, map_r1, map_r2, INTER_LINEAR);
132
133         //Combinacion de imagenes de entrada
134         Mat entrada(FRAME_HEIGHT, 2 * FRAME_WIDTH, CV_8UC3);
135         frameIzdaRect.copyTo(entrada(Range::all(), Range(0, FRAME_WIDTH)));
136         frameDchaRect.copyTo(entrada(Range::all(), Range(FRAME_WIDTH, 2*
    FRAME_WIDTH)));
137
138         // Visualizacion entrada
139         namedWindow("Entrada");
140         imshow("Entrada", entrada);    // Visualizacion imagenes
141
142         /// Correspondencia estereoscopica
143         Mat disp, disp8;     // Declaracion imagenes disparidad
144
145         // Correccion de valores stereoSGBM
146         if(numDisparity < 1) numDisparity = 1;
147         if(SADWindowSize < 3) SADWindowSize = 3;
148
149         // Declaracion y computacion de stereoSGBM
150         Ptr<StereoSGBM> sgbm = StereoSGBM::create(minDisparity,numDisparity
    *16,SADWindowSize,P1,P2,disp12MaxDiff,preFilterCap,uniquenessRatio,
    speckleWindow,speckleRange,true);
151         sgbm->compute(frameIzdaRect,frameDchaRect, disp);
152         normalize(disp, disp8, 0, 255, CV_MINMAX, CV_8U);    // Mapa de
    disparidad
153
154         // Visualizacion mapa disparidad
155         namedWindow("Resultado SGBM");
156         imshow("Resultado SGBM",disp8);
157
158         if(waitKey(30) == 27) {
159             if(grabar == 1){
160                 //string fileSGBM = "../Data/ParametrosStereo.xml";
161                 FileStorage fs(fileSGBM,FileStorage::WRITE);
162                 time_t rawtime;
163                 time(&rawtime);
164                 fs << "Date" << asctime(localtime(&rawtime));
165                 fs << "minDisparity" << minDisparity;
166                 fs << "numDisparity" << numDisparity;
167                 fs << "SADWindowSize" << SADWindowSize;
168                 fs << "P1" << P1;
169                 fs << "P2" << P2;
170                 fs << "disp12MaxDiff" << disp12MaxDiff;
171                 fs << "preFilterCap" << preFilterCap;
172                 fs << "uniquenessRatio" << uniquenessRatio;
173                 fs << "speckleWindow" << speckleWindow;
174                 fs << "speckleRange" << speckleRange;
```

```
175
176                     fs.release();
177                     cout << "Guardado OK" << endl;
178                 }
179                 break;
180             }
181         }
182         return(0);
183 }
```

Prueba Block Matching

```
1  #include "opencv2/core/core.hpp"
2  #include "opencv2/calib3d/calib3d.hpp"
3  #include <opencv2/highgui/highgui.hpp>
4  #include <opencv2/imgproc/imgproc.hpp>
5  #include <stdio.h>
6  #include <time.h>
7  #include <string.h>
8  #include <iostream>
9  #include "../include/config.h"
10
11 using namespace std;
12 using namespace cv;
13
14 int main()
15 {
16     /// Lectura fichero calibracion
17     Mat map_r1, map_r2; //pixel maps para x e y para rectificar imagen Dcha
18     Mat map_l1, map_l2; //pixel maps para x e y para rectificar imagen Dcha
19     Mat Q;              //necesaria para remapeo 3D
20
21     string fileCalib = "../Data/Calibracion.yml";
22     FileStorage fs_cal(fileCalib, FileStorage::READ);
23     fs_cal["Q"] >> Q;
24     fs_cal["map_l1"] >> map_l1;
25     fs_cal["map_l2"] >> map_l2;
26     fs_cal["map_r1"] >> map_r1;
27     fs_cal["map_r2"] >> map_r2;
28
29     fs_cal.release();
30
31     // Comprobacion lectura valores
32     if(map_r1.empty() || map_r2.empty() || map_l1.empty() || map_l2.empty())
33     {
34         cout << "ERROR al cargar fichero de calibracion" << endl;
35         return -1;
36     }
37
38     ///Setup adquisicion imagen
39     VideoCapture capIzda(0), capDcha(1);
40     capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
41     capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
42     capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
43     capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
44
45     Mat element = getStructuringElement( MORPH_ELLIPSE, Size(TAM_STREL,
       TAM_STREL));
46
47     float i = 2.0;
48     int it = 0;
49     float media = 0;
50
51     int numDisparity = 0,
52         blockSize = 25;
53
54     namedWindow("Mapa Disparidad",WINDOW_AUTOSIZE);
55     namedWindow("Parametros");
56     resizeWindow("Parametros",FRAME_WIDTH,80);
57     createTrackbar("numDisp","Parametros", &numDisparity, 80);
```

```
58      createTrackbar("blockSize","Parametros", &blockSize, 50);
59      cout << "Calculando..." << endl;
60
61      const clock_t begin_time = clock();
62
63       while(1){
64          /// Captura de imagenes
65          capIzda.grab();
66          capDcha.grab();
67
68          Mat frameIzda, frameIzdaRect, frameDcha, frameDchaRect;
69          capIzda.retrieve(frameIzda);
70          capDcha.retrieve(frameDcha);
71
72          if(frameIzda.empty() || frameDcha.empty()){
73              cout << "Error en camaras" << endl;
74              return 0;
75          }
76
77          flip(frameDcha,frameDcha,-1);
78
79          /// Rectificado
80          remap(frameIzda, frameIzdaRect, map_l1, map_l2, INTER_LINEAR);
81          remap(frameDcha, frameDchaRect, map_r1, map_r2, INTER_LINEAR);
82
83          cvtColor(frameDchaRect, frameDchaRect,COLOR_BGR2GRAY);
84          cvtColor(frameIzdaRect, frameIzdaRect,COLOR_BGR2GRAY);
85
86          /// StereoBM
87          Mat disp, disp8, disp_compute;     // Declaracion imagenes StereoSGBM
88
89          // Declaracion y computacion de stereoSGBM
90          if(blockSize%2 == 0){ blockSize += 1; }
91          Ptr<StereoBM> bm = StereoBM::create(numDisparity*16,blockSize);
92          bm->compute(frameIzdaRect, frameDchaRect, disp);
93          normalize(disp, disp8, 0, 255, CV_MINMAX, CV_8U);
94
95          // Visualizacion mapa disparidad
96          imshow("Mapa Disparidad",disp8);
97
98          it++;
99
100         /// Calculo FPS
101         float tiempo = float( clock () - begin_time ) / CLOCKS_PER_SEC;
102         if(tiempo > i){
103              media += it;
104              it = 0;
105              i = i + 1.0;
106             if(i == 60.0){
107                 cout << "\n  El numero de frames medios por segundo
    obtenidos durante   \n" <<
108                         "  un tiempo de ejecucion de 60 segundos es: " <<
    media/i << endl;
109                 break;
110             }
111         }
112
113         /// Guardado imagenes
114         if(waitKey(30) == 27){
115             imwrite("./Entrada.png",frameDcha);
116             imwrite("./Resultado.png",disp8);
```

```
117                 break;
118             }
119         }
120         capDcha.release();
121         capIzda.release();
122         return 0;
123 }
```

Prueba Stereo Global Block Matching

```
1  #include "opencv2/core/core.hpp"
2  #include "opencv2/calib3d/calib3d.hpp"
3  #include <opencv2/highgui/highgui.hpp>
4  #include <opencv2/imgproc/imgproc.hpp>
5  #include <stdio.h>
6  #include <time.h>
7  #include <string.h>
8  #include <iostream>
9  #include "../include/config.h"
10
11 using namespace std;
12 using namespace cv;
13
14
15 int main()
16 {
17     /// Lectura fichero calibracion
18     Mat map_r1, map_r2; //pixel maps para x e y para rectificar imagen Dcha
19     Mat map_l1, map_l2; //pixel maps para x e y para rectificar imagen Dcha
20     Mat Q;              //necesaria para remapeo 3D
21     string fileCalib = "../Data/Calibracion.yml";
22     FileStorage fs_cal(fileCalib, FileStorage::READ);
23     fs_cal["Q"] >> Q;
24     fs_cal["map_l1"] >> map_l1;
25     fs_cal["map_l2"] >> map_l2;
26     fs_cal["map_r1"] >> map_r1;
27     fs_cal["map_r2"] >> map_r2;
28
29     fs_cal.release();
30
31     // Comprobacion lectura valores
32     if(map_r1.empty() || map_r2.empty() || map_l1.empty() || map_l2.empty())
33     {
34         cout << "ERROR al cargar fichero de calibracion" << endl;
35         return -1;
36     }
37
38         /// Lectura fichero parametros stereoSGBM
39     int minDisparity, numDisparity, SADWindowSize, P1, P2, disp12MaxDiff,
       preFilterCap, uniquenessRatio, speckleWindow, speckleRange;
40
41     string fileSGBM = "../Data/ParametrosStereo.xml";
42     FileStorage fs_BM(fileSGBM, FileStorage::READ);
43
44     fs_BM["minDisparity"] >> minDisparity;
45     fs_BM["numDisparity"] >> numDisparity;
46     fs_BM["SADWindowSize"] >> SADWindowSize;
47     fs_BM["P1"] >> P1;
48     fs_BM["P2"] >> P2;
49     fs_BM["disp12MaxDiff"] >> disp12MaxDiff;
50     fs_BM["preFilterCap"] >> preFilterCap;
51     fs_BM["uniquenessRatio"] >> uniquenessRatio;
52     fs_BM["speckleWindow"] >> speckleWindow;
53     fs_BM["speckleRange"] >> speckleRange;
54     fs_BM.release();
55
56     ///Setup adquisicion imagen
57     VideoCapture capIzda(0), capDcha(1);
```

```
58      capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
59      capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
60      capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
61      capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
62
63      Mat element = getStructuringElement( MORPH_ELLIPSE, Size(TAM_STREL,
        TAM_STREL));
64
65      float i = 2.0;
66      int it = 0;
67      float media = 0;
68
69      namedWindow("Mapa disparidad",WINDOW_AUTOSIZE);
70      cout << "Calculando..." << endl;
71
72      const clock_t begin_time = clock();
73
74       while(1){
75          /// Captura de imagenes
76          capIzda.grab();
77          capDcha.grab();
78
79          Mat frameIzda, frameIzdaRect, frameDcha, frameDchaRect;
80          capIzda.retrieve(frameIzda);
81          capDcha.retrieve(frameDcha);
82
83          if(frameIzda.empty() || frameDcha.empty()){
84              cout << "Error en camaras" << endl;
85              return 0;
86          }
87
88          flip(frameDcha,frameDcha,-1);
89
90          /// Rectificado
91          remap(frameIzda, frameIzdaRect, map_l1, map_l2, INTER_LINEAR);
92          remap(frameDcha, frameDchaRect, map_r1, map_r2, INTER_LINEAR);
93
94          /// StereoSGBM
95          Mat disp, disp8;     // Declaracion imagenes StereoSGBM
96
97          // Declaracion y computacion de stereoSGBM
98          Ptr<StereoSGBM> sgbm = StereoSGBM::create(minDisparity,numDisparity
        *16,SADWindowSize,P1,P2,disp12MaxDiff,preFilterCap,uniquenessRatio,
        speckleWindow,speckleRange,true);
99          sgbm->compute(frameIzdaRect, frameDchaRect, disp);
100         normalize(disp, disp8, 0, 255, CV_MINMAX, CV_8U);    // Mapa
        disparidad
101
102         // Visualizacion Mapa disparidad
103         imshow("Mapa disparidad",disp8);
104
105         it++;
106
107         /// Calculo FPS
108         float tiempo = float( clock () - begin_time ) / CLOCKS_PER_SEC;
109         if(tiempo > i){
110              media += it;
111              it = 0;
112              i = i + 1.0;
113             if(i == 60.0){
114                 cout << "\n  El numero de frames medios por segundo
```

```
    obtenidos durante   \n" <<
115                         "  un tiempo de ejecucion de 60 segundos es: " <<
    media/i << endl;
116                 break;
117             }
118         }
119
120         /// Guardado de imagenes
121         if(waitKey(30) == 27){
122             imwrite("./Entrada.png",frameDcha);
123             imwrite("./Resultado.png",disp8);
124             break;
125         }
126     }
127     capDcha.release();
128     capIzda.release();
129     return 0;
130 }
```

Principal con filtrado de color anterior

```
#include "opencv2/core/core.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <iostream>
#include "../include/config.h"

using namespace std;
using namespace cv;

void CallBackFunc(int event, int x, int y, int flags, void* userdata);

int pos_x = 0,  // Coordenada x en imagen
    pos_y = 0;  // Cordenada y en imagen
bool make = false;

int main()
{
    /// Lectura fichero calibracion
    Mat map_r1, map_r2; //pixel maps para x e y para rectificar imagen Dcha
    Mat map_l1, map_l2; //pixel maps para x e y para rectificar imagen Dcha
    Mat Q;              //necesaria para remapeo 3D
    string fileCalib = "../Data/Calibracion.yml";
    FileStorage fs_cal(fileCalib, FileStorage::READ);
    fs_cal["Q"] >> Q;
    fs_cal["map_l1"] >> map_l1;
    fs_cal["map_l2"] >> map_l2;
    fs_cal["map_r1"] >> map_r1;
    fs_cal["map_r2"] >> map_r2;

    fs_cal.release();

    // Comprobacion lectura valores
    if(map_r1.empty() || map_r2.empty() || map_l1.empty() || map_l2.empty())
    {
        cout << "ERROR al cargar fichero de calibracion" << endl;
        return -1;
    }

    /// Lectura fichero rango colores
    int hH,hS,hV,lH,lS,lV,hR,hG,hB,lR,lG,lB;
    bool modeHSV = true;
    string fileFiltro = "../Data/ColorDetecta.xml";
    FileStorage fs_fil(fileFiltro, FileStorage::READ);
    fs_fil["hH"] >> hH;
    fs_fil["hS"] >> hS;
    fs_fil["hV"] >> hV;
    fs_fil["lH"] >> lH;
    fs_fil["lS"] >> lS;
    fs_fil["lV"] >> lV;
    fs_fil["hR"] >> hR;
    fs_fil["hG"] >> hG;
    fs_fil["hB"] >> hB;
    fs_fil["lR"] >> lR;
    fs_fil["lG"] >> lG;
```

```
59      fs_fil["lB"] >> lB;
60      fs_fil.release();
61
62      if(hH == 0 && hS == 0 && hV == 0 && lH == 0 && lS == 0 && lV == 0){
63          modeHSV = false;
64      }
65
66      /// Lectura fichero parametros stereoSGBM
67      int minDisparity, numDisparity, SADWindowSize, P1, P2, disp12MaxDiff,
        preFilterCap, uniquenessRatio, speckleWindow, speckleRange;
68
69      string fileSGBM = "../Data/ParametrosStereo.xml";
70      FileStorage fs_BM(fileSGBM, FileStorage::READ);
71
72      fs_BM["minDisparity"] >> minDisparity;
73      fs_BM["numDisparity"] >> numDisparity;
74      fs_BM["SADWindowSize"] >> SADWindowSize;
75      fs_BM["P1"] >> P1;
76      fs_BM["P2"] >> P2;
77      fs_BM["disp12MaxDiff"] >> disp12MaxDiff;
78      fs_BM["preFilterCap"] >> preFilterCap;
79      fs_BM["uniquenessRatio"] >> uniquenessRatio;
80      fs_BM["speckleWindow"] >> speckleWindow;
81      fs_BM["speckleRange"] >> speckleRange;
82      fs_BM.release();
83
84      ///Setup adquisicion imagen
85      VideoCapture capIzda(1), capDcha(0);
86      capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
87      capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
88      capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
89      capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
90
91      Mat element = getStructuringElement( MORPH_ELLIPSE, Size(TAM_STREL,
        TAM_STREL));
92
93      namedWindow("Mapa disparidad, prefiltrado");
94      namedWindow("Mapa disparidad, postfiltrado");
95      setMouseCallback("Mapa disparidad, postfiltrado", CallBackFunc, NULL);
96      namedWindow("Izquierda");
97      namedWindow("Derecha");
98
99      while(1){
100         /// Captura de imagenes
101         capIzda.grab();
102         capDcha.grab();
103
104         Mat frameIzda, frameIzdaRect, frameDcha, frameDchaRect;
105         capIzda.retrieve(frameIzda);
106         capDcha.retrieve(frameDcha);
107
108         if(frameIzda.empty() || frameDcha.empty()){
109             cout << "Error en camaras" << endl;
110             return 0;
111         }
112
113         flip(frameDcha,frameDcha,-1);
114
115         /// Rectificado
116         remap(frameIzda, frameIzdaRect, map_l1, map_l2, INTER_LINEAR);
117         remap(frameDcha, frameDchaRect, map_r1, map_r2, INTER_LINEAR);
```

```
118
119         /// Color
120         Mat izqmascara;
121         if(modeHSV){
122             /// Conversion de RGB a HSV
123             Mat modelColor_izda;
124             cvtColor(frameIzdaRect,modelColor_izda,COLOR_BGR2HSV);   //Pasa
    izda a HSV
125
126             /// Distincion colores
127             inRange(modelColor_izda,Scalar(lH,lS,lV),Scalar(hH,hS,hV),
    izqmascara);
128
129         }else{
130             /// Distincion colores
131             inRange(frameIzdaRect,Scalar(lB,lG,lR),Scalar(hB,hG,hR),
    izqmascara);
132         }
133
134         /// Apertura imagenes resultado
135         morphologyEx(izqmascara,izqmascara, MORPH_OPEN, element);
136         morphologyEx(izqmascara,izqmascara, MORPH_CLOSE, element);
137
138         /// StereoSGBM
139         Mat disp, disp8, disp_compute;     // Declaracion imagenes StereoSGBM
140
141         // Declaracion y computacion de stereoSGBM
142         Ptr<StereoSGBM> sgbm = StereoSGBM::create(minDisparity,numDisparity
    *16,SADWindowSize,P1,P2,disp12MaxDiff,preFilterCap,uniquenessRatio,
    speckleWindow,speckleRange,true);
143         sgbm->compute(frameIzdaRect, frameDchaRect, disp);
144         normalize(disp, disp8, 0, 255, CV_MINMAX, CV_8U);
145         disp.convertTo(disp_compute,CV_32F, 1.f/16.f);
146
147         // Visualizacion StereoSGBM
148         Mat mapaDispFil;
149         disp8.copyTo(mapaDispFil,izqmascara);
150         imshow("Mapa disparidad, postfiltrado",mapaDispFil);
151
152         /// Imagen 3D
153         Mat ima3D;
154         reprojectImageTo3D(disp_compute,ima3D,Q,true);
155
156         if(make == true){
157             ima3D = ima3D(Range(pos_y-5,pos_y+5),Range(pos_x-5,pos_x+5));
158             Mat z_roi(ima3D.size(), CV_32FC1);
159             int from_to[] = {2,0};
160             mixChannels(&ima3D, 1, &z_roi, 1, from_to, 1);
161             cout << "Distancia: " << mean(z_roi) << " mm" << endl;
162
163             make = false;
164         }
165
166         if(waitKey(30) == 27){ break; }
167     }
168     capDcha.release();
169     capIzda.release();
170     return 0;
171 }
172
173 void CallBackFunc(int event, int x, int y, int flags, void* userdata)
```

```
174 {
175     // Permite obtener las coordenadas del cursor en la imagen cuando
176     // el usuario hace clic izquierdo
177     if( event == EVENT_LBUTTONDOWN ){
178         pos_x = x;
179         pos_y = y;
180         make = true;
181     }
182 }
```

Principal con filtrado de color posterior

```
#include "opencv2/core/core.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <iostream>
#include "../include/config.h"

using namespace std;
using namespace cv;

void CallBackFunc(int event, int x, int y, int flags, void* userdata);

int pos_x = 0,  // Coordenada x en imagen
    pos_y = 0;  // Cordenada y en imagen
bool make = false;

int main()
{
    /// Lectura fichero calibracion
    Mat map_r1, map_r2; //pixel maps para x e y para rectificar imagen Dcha
    Mat map_l1, map_l2; //pixel maps para x e y para rectificar imagen Dcha
    Mat Q;              //necesaria para remapeo 3D
    string fileCalib = "../Data/Calibracion.yml";
    FileStorage fs_cal(fileCalib, FileStorage::READ);
    fs_cal["Q"] >> Q;
    fs_cal["map_l1"] >> map_l1;
    fs_cal["map_l2"] >> map_l2;
    fs_cal["map_r1"] >> map_r1;
    fs_cal["map_r2"] >> map_r2;

    fs_cal.release();

    // Comprobacion lectura valores
    if(map_r1.empty() || map_r2.empty() || map_l1.empty() || map_l2.empty())
    {
        cout << "ERROR al cargar fichero de calibracion" << endl;
        return -1;
    }

    /// Lectura fichero rango colores
    int hH,hS,hV,lH,lS,lV,hR,hG,hB,lR,lG,lB;
    bool modeHSV = true;
    string fileFiltro = "../Data/ColorDetecta.xml";
    FileStorage fs_fil(fileFiltro, FileStorage::READ);
    fs_fil["hH"] >> hH;
    fs_fil["hS"] >> hS;
    fs_fil["hV"] >> hV;
    fs_fil["lH"] >> lH;
    fs_fil["lS"] >> lS;
    fs_fil["lV"] >> lV;
    fs_fil["hR"] >> hR;
    fs_fil["hG"] >> hG;
    fs_fil["hB"] >> hB;
    fs_fil["lR"] >> lR;
    fs_fil["lG"] >> lG;
```

```
59      fs_fil["lB"] >> lB;
60      fs_fil.release();
61
62      if(hH == 0 && hS == 0 && hV == 0 && lH == 0 && lS == 0 && lV == 0){
63          modeHSV = false;
64      }
65
66      /// Lectura fichero parametros stereoSGBM
67      int minDisparity, numDisparity, SADWindowSize, P1, P2, disp12MaxDiff,
        preFilterCap, uniquenessRatio, speckleWindow, speckleRange;
68
69      string fileSGBM = "../Data/ParametrosStereo.xml";
70      FileStorage fs_BM(fileSGBM, FileStorage::READ);
71
72      fs_BM["minDisparity"] >> minDisparity;
73      fs_BM["numDisparity"] >> numDisparity;
74      fs_BM["SADWindowSize"] >> SADWindowSize;
75      fs_BM["P1"] >> P1;
76      fs_BM["P2"] >> P2;
77      fs_BM["disp12MaxDiff"] >> disp12MaxDiff;
78      fs_BM["preFilterCap"] >> preFilterCap;
79      fs_BM["uniquenessRatio"] >> uniquenessRatio;
80      fs_BM["speckleWindow"] >> speckleWindow;
81      fs_BM["speckleRange"] >> speckleRange;
82      fs_BM.release();
83
84      ///Setup adquisicion imagen
85      VideoCapture capIzda(0), capDcha(1);
86      capIzda.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
87      capIzda.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
88      capDcha.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
89      capDcha.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
90
91      Mat element = getStructuringElement( MORPH_ELLIPSE, Size(TAM_STREL,
        TAM_STREL));
92
93      namedWindow("Final");
94      setMouseCallback("Final", CallBackFunc, NULL);
95
96      while(1){
97          /// Captura de imagenes
98          capIzda.grab();
99          capDcha.grab();
100
101         Mat frameIzda, frameIzdaRect, frameDcha, frameDchaRect;
102         capIzda.retrieve(frameIzda);
103         capDcha.retrieve(frameDcha);
104
105         if(frameIzda.empty() || frameDcha.empty()){
106             cout << "Error en camaras" << endl;
107             return 0;
108         }
109
110         flip(frameDcha,frameDcha,-1);
111
112         /// Rectificado
113         remap(frameIzda, frameIzdaRect, map_l1, map_l2, INTER_LINEAR);
114         remap(frameDcha, frameDchaRect, map_r1, map_r2, INTER_LINEAR);
115
116         /// Color
117         Mat izqmascara, dchamascara, izqsalida, dchasalida;
```

```
118         if(modeHSV){
119             /// Conversion de RGB a HSV
120             Mat modelColor_izda, modelColor_dcha;
121             cvtColor(frameIzdaRect,modelColor_izda,COLOR_BGR2HSV);   //Pasa
    izda a HSV
122             cvtColor(frameDchaRect,modelColor_dcha,COLOR_BGR2HSV);   //Pasa
    dcha a HSV
123
124             /// Distincion colores
125             inRange(modelColor_izda,Scalar(lH,lS,lV),Scalar(hH,hS,hV),
    izqmascara);
126             inRange(modelColor_dcha,Scalar(lH,lS,lV),Scalar(hH,hS,hV),
    dchamascara);
127         }else{
128             /// Distincion colores
129             inRange(frameIzdaRect,Scalar(lB,lG,lR),Scalar(hB,hG,hR),
    izqmascara);
130             inRange(frameDchaRect,Scalar(lB,lG,lR),Scalar(hB,hG,hR),
    dchamascara);
131         }
132
133         /// Combinacion imagenes
134         Mat combo_mascara(FRAME_HEIGHT, 2 * FRAME_WIDTH, CV_8UC1);
135         dchamascara.copyTo(combo_mascara(Range::all(), Range(0, FRAME_WIDTH)
    ));
136         izqmascara.copyTo(combo_mascara(Range::all(), Range(FRAME_WIDTH, 2*
    FRAME_WIDTH)));
137
138         namedWindow("MascaraInicio");
139         imshow("MascaraInicio",combo_mascara);
140
141         /// Apertura imagenes resultado
142         morphologyEx(izqmascara,izqmascara, MORPH_OPEN, element);
143         morphologyEx(dchamascara,dchamascara, MORPH_OPEN, element);
144         morphologyEx(izqmascara,izqmascara, MORPH_CLOSE, element);
145         morphologyEx(dchamascara,dchamascara, MORPH_CLOSE, element);
146
147
148         /// Combinacion mascara y imagen entrada
149         Mat mascaraTotal;
150         bitwise_or(izqmascara,dchamascara, mascaraTotal);
151
152         namedWindow("Mascara");
153         imshow("Mascara", mascaraTotal);
154
155         /// StereoSGBM
156         Mat disp, disp8, disp_compute;      // Declaracion imagenes StereoSGBM
157
158         // Declaracion y computacion de stereoSGBM
159         Ptr<StereoSGBM> sgbm = StereoSGBM::create(minDisparity,numDisparity
    *16,SADWindowSize,P1,P2,disp12MaxDiff,preFilterCap,uniquenessRatio,
    speckleWindow,speckleRange,true);
160         sgbm->compute(frameIzdaRect, frameDchaRect, disp);
161         normalize(disp, disp8, 0, 255, CV_MINMAX, CV_8U);
162         disp.convertTo(disp_compute,CV_32F, 1.f/16.f);
163
164         // Visualizacion StereoBM
165 //          imshow("Resultado",disp8);
166
167         Mat mapaDispFil;
168         //disp8.copyTo(mapaDispFil,mascaraTotal);
```

```
169
170             //namedWindow("FINAL");
171             imshow("Final",disp8);
172
173             /// Imagen 3D
174             Mat ima3D;
175             reprojectImageTo3D(disp_compute,ima3D,Q,true);
176
177             if(make == true){
178                 ima3D = ima3D(Range(pos_y-2,pos_y+2),Range(pos_x-2,pos_x+2));
179                 Mat z_roi(ima3D.size(), CV_32FC1);
180                 int from_to[] = {2,0};
181                 mixChannels(&ima3D, 1, &z_roi, 1, from_to, 1);
182                 cout << "Depth: " << mean(z_roi) << " mm" << endl;
183
184                 make = false;
185             }
186
187             if(waitKey(30) == 27){ break; }
188         }
189         capDcha.release();
190         capIzda.release();
191         return 0;
192 }
193
194 void CallBackFunc(int event, int x, int y, int flags, void* userdata)
195 {
196     // Permite obtener las coordenadas del cursor en la imagen cuando
197     // el usuario hace clic izquierdo
198     if( event == EVENT_LBUTTONDOWN ){
199         pos_x = x;
200         pos_y = y;
201         make = true;
202     }
203 }
```