

Universidad Aeronáutica en Querétaro



Innovación educativa para el desarrollo de México

TESIS

Trabajo Profesional para obtener el Titulo de Ingeniero en

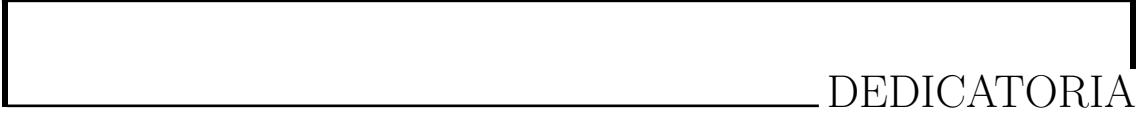
Electrónica y Control de Sistemas de Aeronaves.

Marco Antonio Aguilar Gallardo

Dirige: Antonio Flores

Municipio de Colón, Querétaro

9 de febrero de 2020



DEDICATORIA



AGRADECIMIENTOS

ÍNDICE DE FIGURAS

| | |
|--|----|
| 1.1. Objetivo | 2 |
| 1.2. Uso de una gimbal para una IMU [1] | 3 |
| 1.3. Uso de una gimbal en un motor de propulsión [2] | 3 |
| 1.4. Jerrie Cobb, uno de los Mercury 13, da un giro en la plataforma gimbal. Créditos: NASA | 4 |
| 1.5. Primer uso de la steadicam | 5 |
| 2.1. Similitudes entre humano y computadora | 6 |
| 2.2. El camino que sigue la señal óptica dentro de la cabeza humana. | 7 |
| 2.3. Componentes generales del ojo humano | 8 |
| 2.4. Formación de una imagen en el ojo | 9 |
| 2.5. Componentes de la camara | 10 |
| 2.6. Proceso general de captura de imagen por la cámara | 11 |
| 2.7. Captura de imagen | 11 |
| 2.8. Regla 1 | 12 |
| 2.9. Regla 2 | 12 |
| 2.10. Regla 3 | 12 |
| 2.11. Obtención geométrica de la imagen | 13 |
| 2.12. Modelo de lente fina | 13 |
| 2.13. Tarjetas procesadoras de datos | 14 |
| 2.14. Pruebas de performance | 15 |
| 2.15. Nodos conectados al Master | 17 |
| 2.16. Representación grafica de Topics | 17 |
| 2.17. Comunicación de mensajes | 18 |
| 2.18. Representación 2D de imágenes digitales mediante conjuntos de puntos discretos en una matriz rectangular | 19 |
| 2.19. Ilustración de cuantización [3].La misma imagen se muestra con diferentes niveles de cuantización: a 16, b 8, c 4, d 2. Muy pocos niveles de cuantificación producen bordes falsos y hacen que las características con bajo contraste desaparezcan parcial o totalmente. | 20 |
| 2.20. Símbolo característico RGB de OpenCV | 21 |
| 2.21. Representación del espacio de color RGB | 22 |

| | |
|--|----|
| 2.22. Representación del espacio de color HSI | 23 |
| 4.1. Camara 'Ocam' | 26 |
| 4.2. Nodos y topic | 27 |
| 4.3. Comunicación entre opencv y ROS | 28 |
| 4.4. Diagrama de flujo del programa publisher | 29 |
| 4.5. Diagrama de flujo para cambio de espacio de color | 30 |
| 4.6. Función Color [4] | 30 |
| 4.7. Función inRange | 31 |
| 4.8. Espectro de colores | 31 |
| 4.9. Separación de colores | 31 |
| 4.10. Separación de colores | 32 |
| 4.11. Odroid XU-4 | 36 |

ÍNDICE DE CUADROS

| | |
|--|----|
| 4.1. A table without vertical lines. | 32 |
|--|----|

CAPÍTULO 1

INTRODUCCIÓN

En el presente capítulo se expone el objetivo general, así como sus derivados. En la primera sección se aborda el tema de investigación donde especifica la justificación del presente trabajo, posteriormente se sintetiza algunas de las investigaciones que sirvieron como base para la elección del tema previamente descrito. Finalmente se dan las razones de la investigación y se exponen las aportaciones derivadas del tema de tesis.

1.1. Tema de investigación

En el campo de la aeronáutica hay una rama que en los últimos años ha sido objeto de estudio debido a su exponencial importancia para tareas críticas, se trata de los vehículos aéreos no tripulados UAV (del inglés unmanned aerial vehicle), donde dichas tareas críticas han podido alcanzar sus objetivos en parte gracias a la implementación reciente de visión artificial.

En la implementación de camaras para el sistema visual de los UAV's la estabilidad juega un rol importante, debido a que el sistema se encuentra expuesto a diversos factores que hacen que la captura de imagenes sea deficiente. Es por ello que la impletanción de un sistema estabilizador de camaras es necesario.

El tema principal de este trabajo se basa en la obtencion de imágenes y

1.2. Justificación

1.3. Objetivo

Diseñar, instrumentar y controlar un dispositivo gimbal que sea capaz de seguir un objeto a través de visión artificial para implementarse en un UAV de categoría pequeña a velocidad baja.

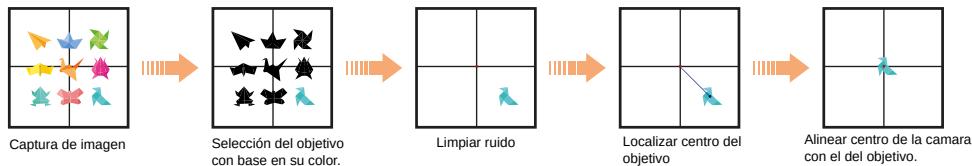


Figura 1.1: Objetivo

1.4. Objetivos específicos

- Obtener el modelo matemático de una gimbal.
- Diseñar e implementar el sistema embebido que dará el soporte electrónico a la gimbal.
- Capturar figuras geométricas definidas mediante el uso de una cámara digital y emplear algoritmos de visión artificial para la obtención de datos.
- Diseñar un controlador autónomo con base en el modelo matemático, previamente obtenido.

1.5. Estado de la cuestión

La aparición de la gimbal no es un término para nada nuevo, de hecho es viejo más de lo que muchos podemos creer. Fue en el 250 antes de nuestra era cuando el inventor Philo of Byzantium describió un bote de tinta de ocho lados con una abertura en cada lado, que se puede girar de modo que mientras cualquier cara está en la parte superior, se puede sumergir y entintar un bolígrafo, aunque la tinta nunca se agota a través de los agujeros de los otros lados. [5].

Desde entonces y hasta la fecha múltiples científicos han desarrollado investigaciones alrededor de dicho artefacto, algunos teniendo más éxito que otros; los cuales serán brevemente expuestos con la finalidad de obtener el estado actual en el que se encuentra la gimbal y su avance tecnológico.

■ Navegación inercial

En la navegación inercial, como se aplica a los barcos y submarinos, se necesita un mínimo de tres gimbals para permitir que un sistema de navegación inercial (masa estable) permanezca fijo en el espacio inercial, compensando los cambios en el guiñada, inclinación y balanceo del barco.

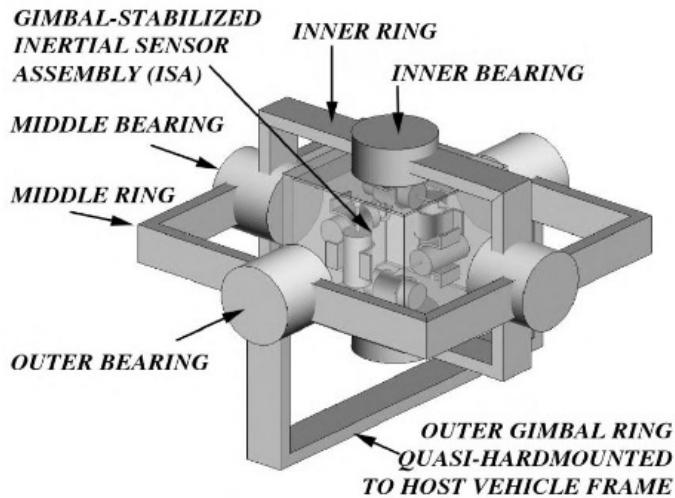


Figura 1.2: Uso de una gimbal para una IMU [1]

En esta aplicación, la Unidad de medición inercial (IMU) está equipada con tres giroscopios montados ortogonalmente para detectar la rotación alrededor de todos los ejes en el espacio tridimensional. Las salidas giroscópicas accionan motores que controlan la orientación de los tres gimbals según sea necesario para mantener la orientación de la IMU.

■ Motores de cohete

En la propulsión de naves espaciales, los motores de cohetes generalmente se montan en un par de gimbals para permitir que un solo motor logre el empuje sobre los ejes de inclinación y guiñada; o, a veces, solo se proporciona un eje por motor. Para controlar el giro, se utilizan motores gemelos con señales de control de inclinación diferencial o guiñada para proporcionar torque sobre el eje de balanceo del vehículo.

Uno de los motores más famosos es el J-2X. Es un motor de cohete avanzado altamente eficiente y versátil con las características ideales de empuje y rendimiento para impulsar la etapa superior del espacio de la NASA. [6]

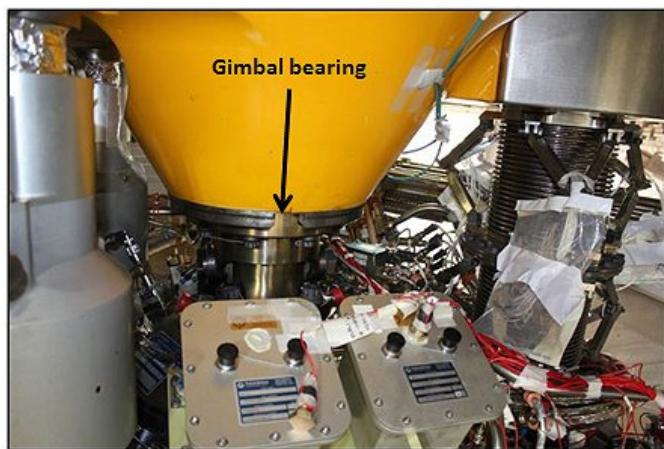


Figura 1.3: Uso de una gimbal en un motor de propulsión [2]

- **Entrenamiento para astronautas**

Sistema de simulación de maniobras de tipo caída que se pueden encontrar en el vuelo espacial fue creado por la NASA y era conocido como "the gimbal rig.". Tres jaulas tubulares de aluminio podrían girar por separado o en combinación para dar movimientos de balanceo, cabeceo y guiñada a velocidades de hasta 30 revoluciones por minuto, mayores que las esperadas en vuelos espaciales reales. Los chorros de gas nitrógeno, unidos a las tres jaulas, controlaron el movimiento. Desde el 15 de febrero hasta el 4 de marzo de 1960, la plataforma de cardán proporcionó una capacitación valiosa para los siete astronautas del Proyecto Mercurio. Cada uno experimentó unas cinco horas de tiempo de vuelo simulado. [7]



Figura 1.4: Jerrie Cobb, uno de los Mercury 13, da un giro en la plataforma gimbal. Créditos: NASA

- **Estabilizador de Camaras**

Los gimbals también se utilizan para montar todo, desde lentes de cámara pequeñas hasta telescopios fotográficos grandes.

En los equipos de fotografía portátiles, se utilizan gimbals de un solo eje para permitir un movimiento equilibrado de la cámara y las lentes. Esto resulta útil en la fotografía semi-profesional, así como en cualquier otro caso en el que se adopten teleobjetivos muy largos y pesados: un eje de la gimbal gira un lente alrededor de su centro de gravedad, lo que permite una manipulación fácil y suave mientras se rastrea a los sujetos en movimiento.

Los montajes de gimbal muy grandes en forma de montajes de altitud-altitud de 2 o 3 ejes se utilizan en la fotografía satelital con fines de seguimiento.

En la década de 1970, el director de fotografía estadounidense Garrett Brown tuvo una idea simple pero revolucionaria: hacer un dispositivo que pudiera suavizar las tomas de acción manuales.



Figura 1.5: Primer uso de la Steadicam

El resultado es el Steadicam (Que cumple con los principios físicos de la gimbal) Ganador de un Premio de la Academia, que hizo su debut cinematográfico en la película "Bound for Glory", y se destacó en las películas Rocky y "The Shining"

1.6. Contribuciones

1.7. Alcances

1.8. Estructura de la tesis

CAPÍTULO 2

MARCO TÉORICO

En este capítulo desarrolla la teoría que fundamenta el proyecto de investigación con base en el problema previamente descrito.

Antes de entrar a la teoría es necesario entender cuales son las fases del proyecto y el porque de ellas. La siguiente imagen muestra la similitud entre el sistema de adquisición de datos de un humano y la de un sistema digital.

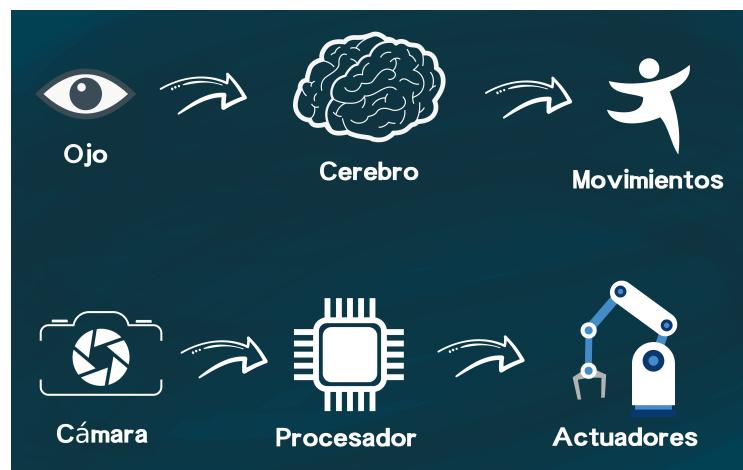


Figura 2.1: Similitudes entre humano y computadora

Donde se observa un diagrama de flujo que empieza con la adquisición de datos, en este caso la captura de una imagen, posterior se hace un procesamiento, es decir, se le da sentido a los datos, y finalmente se hace una acción con base en la tarea que el procesador ha generado.

2.1. Óptica

La visión artificial surge de un amplio estudio probabilístico y matemático del procesamiento de imágenes digitales, pero sobre todo de análisis humanos y de la intuición ya que de estas últimas el ingeniero hace selección de entre una u otra

técnica. Esta elección se basa usualmente en juicios visuales subjetivos.

Entender los conceptos básicos de la percepción humana es entonces pertinente, donde la Óptica nos ayudará a entender mejor como es que el ojo humano percibe y como lo hace una cámara.

La función de la óptica de una cámara es captar los rayos luminosos y concentrarlos sobre el sensor sensible de la cámara de video. Después de determinar el tipo de iluminación que mejor se adapta al problema, la elección de una óptica u otra influirá en la calidad de la imagen y el tamaño de los objetos.

2.1.1. Descripción general del sistema visual humano

La luz entra al ojo y estimula los sensores en la parte posterior. La señal que se crea luego viaja a través del nervio óptico, cruzando el nervio que proviene del otro ojo y llega a un órgano, en el interior del cerebro en el área del tálamo, llamado n úcleo geniculado lateral (LGN). Las salidas de la LGN se envían a la corteza visual en la parte posterior del cerebro. La corteza visual es quizás la parte más compleja del cuerpo humano. Es el lugar en el cerebro donde tiene lugar la mayor parte del procesamiento visual. [8]

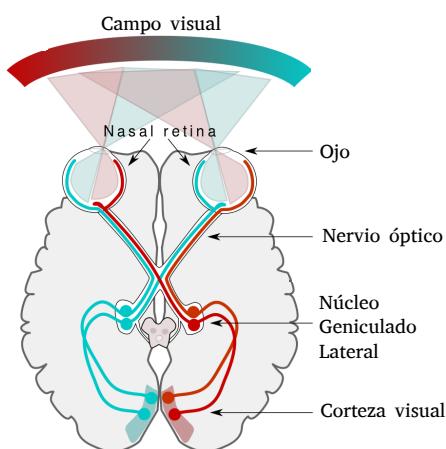


Figura 2.2: El camino que sigue la señal óptica dentro de la cabeza humana.

En términos generales, cuanto más nos alejamos del camino visual del ojo, menos entendemos lo que está sucediendo.

2.1.2. Estructura del ojo humano

Es necesario abordar algunos conceptos básicos para entendernos en un futuro, pero sobre todo comprender las similitudes del ojo humano y la cámara, además de analizar limitaciones físicas de la vista humana en los mismos términos que usaremos para nuestras imágenes digitales.

El ojo está formado de dos componentes principales:

- **Componentes ópticos:**

Permiten la formación de la imagen en la retina y son los siguientes: la córnea, el cristalino, la pupila, el humor acuoso y el humor vítreo que permiten la formación de una imagen en la retina.

- **Componentes neurológicos:**

son los que transforman la información óptica en eléctrica y transmiten la información al cuerpo geniculado lateral. Estos componentes son la retina y el nervio óptico.

- **Cornea:** La córnea es una estructura del ojo que permite el paso de la luz desde el exterior al interior del ojo y protege el iris y el cristalino. Posee propiedades ópticas de refracción y para garantizar su función debe ser transparente y es necesario que mantenga una curvatura adecuada.
- **Esclerótica:** Es el recubrimiento exterior blanco del ojo. La esclerótica le da su color blanco al globo ocular.
- **Coroides:** Es la capa de vasos sanguíneos y tejido conectivo entre la parte blanca del ojo y la retina (en la parte posterior del ojo). Es parte de la úvea y suministra los nutrientes a las partes internas del ojo.
- **Cuerpo ciliar:** Es una estructura circular que es una prolongación del iris, la parte de color del ojo. También contiene el músculo ciliar, el cual cambia la forma del cristalino cuando los ojos se enfocan en un objeto cercano. Este proceso se denomina acomodación.
- **Diafragma Iris:** que se expande o contrae para controlar la cantidad de luz que entra en el ojo. La apertura central del iris, llamada pupila, varía su diámetro de 2 a 8mm. El frente del iris contiene el pigmento visible del ojo, y la parte trasera contiene un pigmento negro.
- **Cristalino:** El cristalino es “la lente” del ojo y sirve para enfocar, ayudado por los músculos ciliares. El cristalino es una lente que actúa como una lente biconvexa, lenticular, flexible y avascular, cuya principal función es la de enfocar los objetos en las distintas distancias correctamente.
- **Retina:** Es la capa de tejido sensible a la luz que se encuentra en la parte posterior globo ocular. Las imágenes que pasan a través del cristalino del ojo se enfocan en la retina. La retina convierte entonces estas imágenes en señales eléctricas y las envía por el nervio óptico al cerebro.

[9]

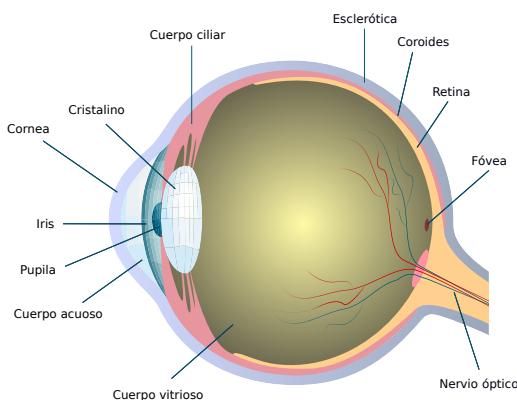


Figura 2.3: Componentes generales del ojo humano

2.1.3. Formación de imágenes en el ojo

El sentido de la vista en las personas tiene un funcionamiento complejo y necesita de dos elementos básicos: El ojo y el cerebro.

La luz es el tercer elemento más destacado en la visión. Sin ella somos incapaces de ver. Dentro del ojo se sigue una serie de pasos para poder capturar una imagen con base en la luz disponible.

- La luz pasa a través de la córnea y llega a la pupila que se contrae o expande según su intensidad. La pupila será más pequeña cuanta más luz haya para evitar deslumbramientos.
- El cristalino del ojo será quien proyecte las imágenes enfocadas en la retina. Puede aplanarse o abombarse según lo cerca o lejos que esté el objeto que veamos.
- La retina recibe la imagen invertida en sus paredes. La luz estimula los conos y los bastones quienes transforman esa información en impulsos nerviosos. Esta electricidad se trasladará al cerebro a través del nervio óptico.

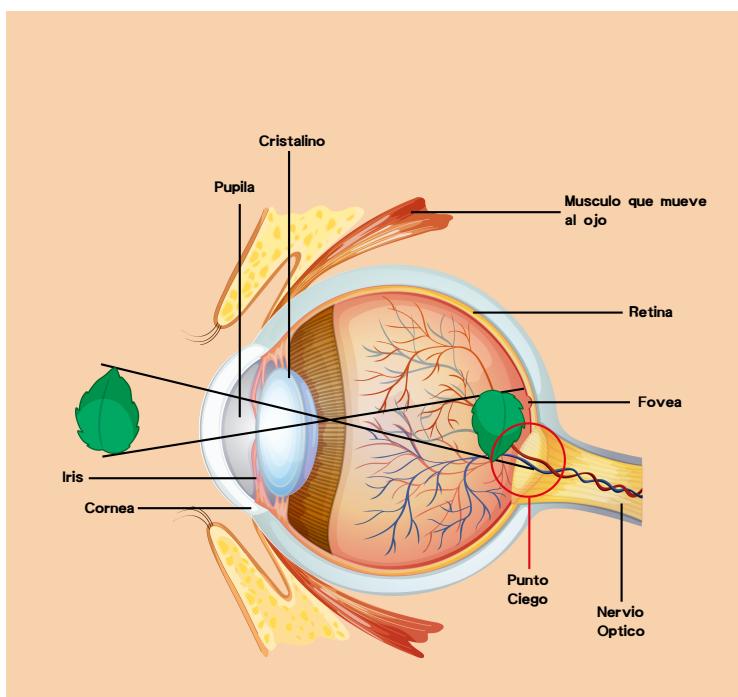


Figura 2.4: Formación de una imagen en el ojo

El cerebro es quien realmente ve las imágenes. Endereza la imagen invertida de la retina e interpreta la información de color, tamaño, posición, etc. La distancia entre el centro del cristalino y la retina (que llamaremos distancia focal), varía de aproximadamente 17mm a 14mm. [9]

2.1.3.1. Fotorreceptores

Los fotorreceptores son células especializadas de la retina del ojo responsables de convertir la luz en señales que son enviadas al cerebro. Los fotorreceptores nos dan la visión de color y la visión nocturna. [10]

2.1.4. Camara digital

La cámara digital es uno de los cambios más importantes en esta era de la digitalización de la información. Es un invento tan revolucionario y que dista tanto de su predecesor análogo.

Las cámaras digitales producen imágenes capturando o grabando las características de la luz de una escena o sujeto. Las partes principales de la cámara que participan en el proceso son el cuerpo de la cámara, el obturador de la cámara, la lente de la cámara, la apertura de la lente y el sensor de imagen de la cámara. [11]

2.1.5. Componetes de la camara

- **Lente:** El objetivo de la lente de la cámara es enfocar y dirigir la luz entrante. La lente de la cámara consta de una o más piezas de vidrio o plástico de forma precisa llamadas elementos. La luz que entra por los elementos se "dobla." se dirige al sensor de imagen donde se captura la información sobre la luz.
- **Obturador:** Sistema mecánico o electrónico que permite el paso de la luz a través del sistema óptico durante un tiempo determinado.
- **Diafragma:** Sistema mecánico o electrónico que gradúa la mayor o menor intensidad de luz que debe pasar durante el tiempo que está abierto el obturador. En nuestro ojo la pupila se encarga de hacer esa función.
- **Sistema de enfoque:** Gradúa la posición del objetivo, para que la imagen se forme totalmente donde está la placa sensible. Su función es similar a la que realiza el cristalino en el ojo.
- **Sensor:** En el ojo, la retina es la parte a la que llega la luz antes de transformarse en señales eléctricas. Si buscamos en la cámara fotográfica un elemento que se asemeje nos encontramos con los sensores CCD o CMOS. Se puede decir que estos dispositivos son los encargados de transformar la luz en carga eléctrica para crear cada pixel de la imagen.
El sensor de imagen tiene una cuadrícula con millones de elementos microscópicos de información de luz llamados "fotosites". Cada una de estas fotositas se conoce mejor como píxeles.

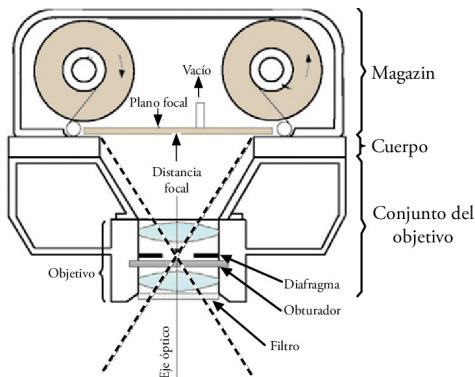


Figura 2.5: Componentes de la camara

2.1.6. Formación de la imagen en la cámara

En una cámara fotográfica se recibe la luz que traspasa el diafragma, pasa por los cristales de la cámara hasta llegar al CCD(Charge Coupled Device o, en español, Dispositivo de Carga Acoplada) o sensor, que es donde se forma la imagen correcta y se envía al procesador.

Algo similar pasa en el ojo, la pupila es el diagrama natural que filtra la luz que entra en el ojo, pasa por la lente (el cristalino) que converge los rayos hasta llegar a la retina, que es la estructura que tiene las células fotosensibles y dónde se produce la imagen, y a través del nervio óptico se transporta la información al cuerpo geniculado, que es la parte del cerebro donde se produce la visión.



Figura 2.6: Proceso general de captura de imágenes por la cámara

Cuando los rayos paralelos pasan a través de una lente convexa, convergen hacia un punto que se denomina punto focal.

Realmente toda lente tiene dos puntos focales según la luz pase en un sentido o en el opuesto. La distancia focal (mm, milímetros) es uno de los parámetros de los objetivos de las cámaras. Llamamos longitud focal de un objetivo a la distancia que existe entre el sensor (plano focal) y la lente.

La distancia focal está también relacionada con la cantidad de luz refractada por la lente. Es el denominado factor de potencia D cuyo valor es la inversa de la distancia focal y su unidad de medida la dioptría.

Otro valor que se encontrará en toda óptica es el número F. Este parámetro indica la relación entre la distancia focal y el diámetro del diafragma:

$$F = \frac{f}{D} \quad (2.1)$$

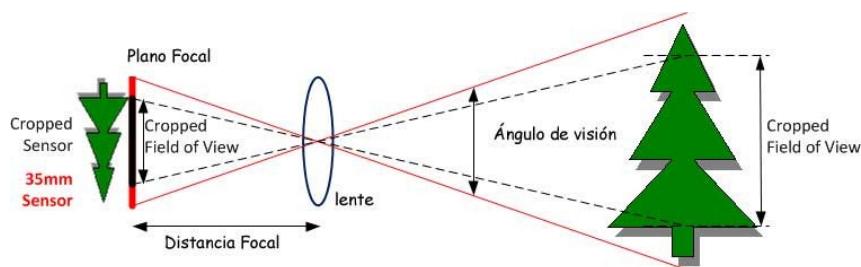


Figura 2.7: Captura de imágenes

Indica la cantidad de luz (brillantez) que se deja pasar por el objetivo y se puede regular mediante un anillo presente en la montura de la óptica.

En las ópticas habrá que tener por tanto en cuenta su F mínimo, que indicará la máxima cantidad de luz que puede atravesar la óptica y que tendrá que estar en concordancia con la sensibilidad de la cámara.

2.1.7. Cálculo de la imagen

La imagen de un objeto en una lente convergente se obtiene geométricamente aplicando las siguientes reglas:

- **Regla 1**

Cualquier rayo incidente paralelo al eje principal en la zona objeto sale pasando por el foco principal en la zona imagen

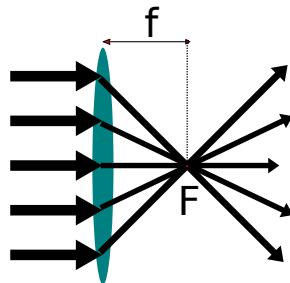


Figura 2.8: Regla 1

- **Regla 2**

Cualquier rayo incidente que pasa por el centro de la lente sale con la misma dirección, es decir, no sufre desviación

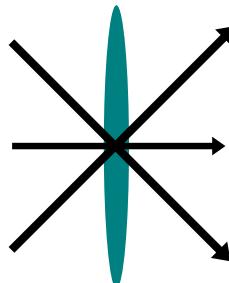


Figura 2.9: Regla 2

- **Regla 3**

Cualquier rayo incidente que corta al eje principal en la zona objeto a la misma distancia que la distancia focal, sale paralelo al eje principal en la zona imagen.

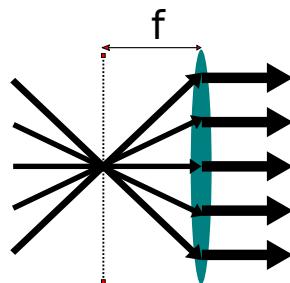


Figura 2.10: Regla 3

En la figura siguiente obtenemos la imagen P' del objeto P aplicando estas tres reglas. El rayo rojo es la primera regla; el rayo verdaderamente es la segunda regla y el rayo azul la tercera. El punto P' donde se cortan los tres rayos es donde se forma la imagen enfocada de P .

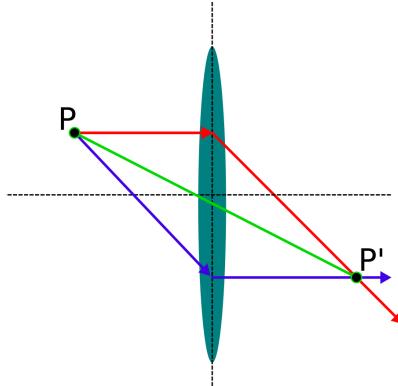


Figura 2.11: Obtención geométrica de la imagen

Estas tres reglas nos indican la trayectoria que seguirán tan sólo tres rayos de todos los que genera el objeto. La imagen vendrá dada por el punto donde se intersecten esos rayos en la zona imagen.

En las ópticas habrá que tener por tanto en cuenta su F mínimo, que indicará la máxima cantidad de luz que puede atravesar la óptica y que tendrá que estar en concordancia con la sensibilidad de la cámara. Por último sobre otro anillo similar se encuentra una escala graduada en metros, que sirve para regular el enfoque según la distancia del objeto encuadrado. Al moverlo el plano de elementos sensibles se aproxima a la lente para hacerlo coincidir con el de formación de la imagen.

Este es el modelo denominado de la **lente fina**. La lente fina es aquella en la que todo rayo que entra paralelo al eje óptico pasa por el foco posterior de la lente y todo rayo que pasa por el foco anterior sale de la lente paralelo al eje óptico.

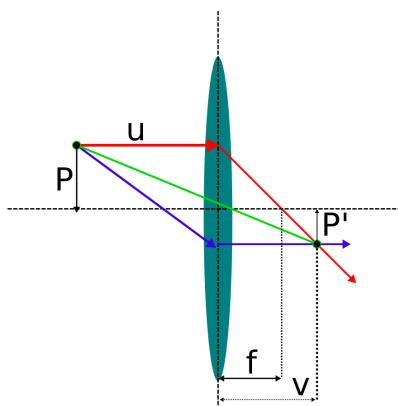


Figura 2.12: Modelo de lente fina

2.2. Procesamiento de datos

En los seres humanos, el sistema visual recopila hasta el 80 por ciento de todos los datos sensoriales recibidos del entorno. Para dar sentido a este diluvio de

información óptica, las entradas visuales que son captadas y convertidas en señales electroquímicas por los aproximadamente 130 millones de células sensibles a la luz en la retina se alimentan y procesan mediante una compleja red de células nerviosas en el cerebro. [12]

Es por lo anterior que la elección de un hardware que procese tanta información como lo hace el cerebro se vuelve una tarea prioritaria.

2.2.1. Tarjeta procesadora

La elección del hardware que será el cerebro de nuestro sistema es una tarea importante ya que de ello depende el éxito de nuestro proyecto, y en estos tiempos el mercado ofrece una gran variedad de tarjetas procesadoras.



Figura 2.13: Tarjetas procesadoras de datos

Pasando desde un FPGA hasta un sistema totalmente completo como las NUC de Intel, pero para fines de este proyecto nos enfocaremos solo en la hecha por Hard-Kernel, la ODROID. Esto porque nos ofrece una capacidad de procesamiento de datos apta para la visión artificial y a un bajo costo.

Las placas de la serie Odroid son similares a Raspberry Pi, pero tiene una mejor configuración y rendimiento. Se basa en la arquitectura ARM. [13]

ODROID significa Open + Android. Es una plataforma de desarrollo tanto de hardware como de software.

Especificaciones técnicas en apéndice B.

Con base en pruebas hechas por el proveedor, el modelo XU-4 resulta ser superior a otras tarjetas de desarrollo del mercado.

Ejecutaron varios puntos de referencia para medir la potencia informática en el XU4. Las mismas pruebas se realizaron en el Raspberry Pi 3 Modelo B, ODROID-C1 +, ODROID-C2 y ODROID-XU4. Los valores de los resultados de la prueba se escalaron uniformemente para fines de comparación. Se midió que la potencia informática del XU4 era 7 veces más rápida que la última Raspberry Pi 3 gracias a los núcleos 2Ghz Cortex-A15 y un ancho de banda de memoria de 64 bits mucho mayor.

Compilar código en el XU4 es super rápido. La memoria RAM DDR3 de 2 GB de alto rendimiento es una ventaja adicional que permite compilar la mayoría de los programas directamente en el XU4. [14]

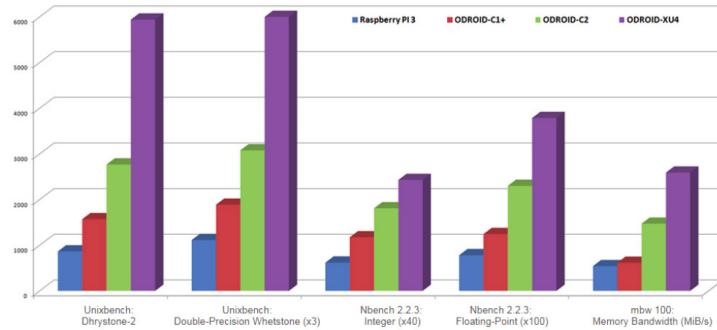


Figura 2.14: Pruebas de performance

2.2.2. Sistema operativo

Una vez elegida la tarjeta de desarrollo con la cual se estará trabajando para este proyecto, el siguiente paso es escoger el sistema operativo que dará soporte a nuestro sistema. Por defecto ODROID nos sugiere utilizar Ubuntu en su versión MATE.

2.2.2.1. Ubuntu MATE

Ubuntu MATE es una distribución de Linux gratuita y de código abierto y un derivado oficial de Ubuntu.

Ubuntu es uno, si no es que el más grande, empleador de Linux en el mundo. Linux está en el corazón de Ubuntu y hace posible crear sistemas operativos seguros, potentes y versátiles.

Ubuntu MATE toma el sistema operativo basado en Ubuntu y agrega el MATE Desktop.

Donde podemos definir MATE Desktop como una implementación de la metáfora del Desktop hecha de un conjunto de programas que se ejecutan en la parte superior de un sistema operativo de computadora, que comparten una interfaz gráfica de usuario (GUI) común. Las GUI de escritorio ayudan al usuario a acceder y editar archivos fácilmente. [15]

Ubuntu soporta arquitecturas armhf, tipo de arquitectura de la odroid, además de optimizar el sistema operativo sin sacrificar las ventajas que provee para una Computadora Portátil.

2.2.3. ROS

Un sistema de comunicación es a menudo una de las primeras necesidades que surgen al implementar una nueva aplicación de robot. El sistema de mensajería integrado y probado de ROS ahorra tiempo al administrar los detalles de la comunicación entre los nodos distribuidos a través del mecanismo anónimo de publicación / suscripción. Otro beneficio de usar un sistema de paso de mensajes es que te obliga a implementar interfaces claras entre los nodos en tu sistema, mejorando así la encapsulación y promoviendo la reutilización de código. La estructura de estas interfaces de mensajes se define en el mensaje IDL (Lenguaje de descripción de interfaz). Decidí usar ROS porque crear un software de robot verdaderamente robusto y de

uso general es difícil. Desde la perspectiva del robot, los problemas que parecen triviales para los humanos a menudo varían enormemente entre instancias de tareas y entornos. Hacer frente a estas variaciones es tan difícil que se necesita apoyo de un sistema de comunicación.

ROS tiene tres niveles de conceptos: el nivel del sistema de archivos, el nivel del gráfico de cómputo y el nivel de la comunidad. [16]

2.2.3.1. Nivel de sistemas de archivos de ROS

Los conceptos de nivel de sistema de archivos cubren principalmente los recursos de ROS que encuentra en el sistema, tales como:

- **Paquetes**

Los paquetes son la unidad principal para organizar el software en ROS. Un paquete puede contener procesos (nodos) de tiempo de ejecución de ROS, una biblioteca dependiente de ROS, conjuntos de datos, archivos de configuración o cualquier otra cosa que se organice conjuntamente de manera útil. Los paquetes son el elemento de construcción más atómico y el elemento de lanzamiento en ROS. Lo que significa que lo más granular que puede construir y lanzar es un paquete.

2.2.3.2. Nivel de gráfico de cómputo ROS

En términos generales, ROS sigue la filosofía de desarrollo de software de Unix en varios aspectos clave. Esto tiende a hacer que ROS se sienta "natural" para los desarrolladores que vienen de un entorno Unix, pero algo criptico.^{a1} El principio para aquellos que han usado principalmente entornos de desarrollo gráfico en Windows o Mac OS X. [17]

Los sistemas ROS consisten en numerosos programas pequeños informáticos que se conectan entre sí e intercambian mensajes continuamente. Estos mensajes viajan directamente de un programa a otro.

Los conceptos básicos del gráfico de cómputo de ROS son nodos, master, servidor de parámetros, mensajes, servicios, topics y bags, los cuales proporcionan datos al gráfico de diferentes maneras siguiendo la filosofía antes descrita.

- **Nodo**

los nodos son procesos que realizan cálculos. ROS está diseñado para ser modular a nivel de nodo; un sistema de control de robot generalmente comprende muchos nodos.

- **Master**

Un programa intermedio que conecta nodos.

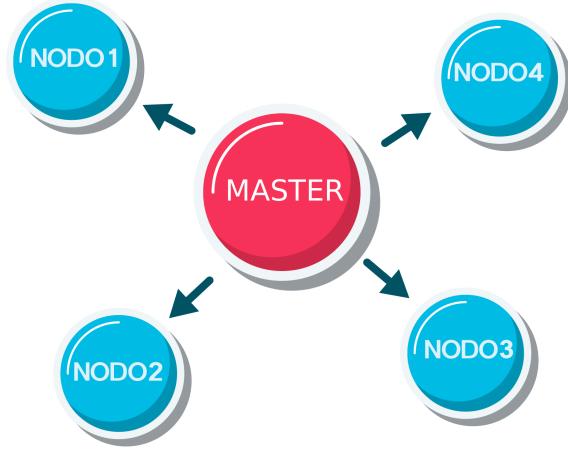


Figura 2.15: Nodos conectados al Master

- **Topics**

Los mensajes se enrutan a través de un sistema de transporte con semántica de publicación / suscripción. Un nodo envía un mensaje al publicarlo en un topic determinado. El topic es un nombre que se utiliza para identificar el contenido del mensaje.

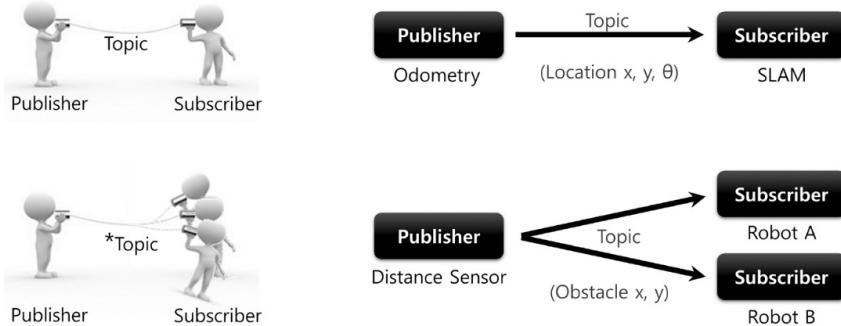


Figura 2.16: Representación grafica de Topics

- **Mensajes**

Los mensajes básicamente están pasando por el topic. Hay mensajes existentes basados en tipos de datos predefinidos, y los usuarios pueden escribir sus propios mensajes.

- **Parámetros**

El servidor de parámetros permite que los datos se almacenen por clave en una ubicación central. Actualmente es parte del Máster.

- **Servicios**

Ya hemos visto ROS Topics, que tiene un mecanismo de publicación y suscripción. El servicio ROS tiene un mecanismo de solicitud / respuesta. Una llamada de servicio es una función, que puede llamar siempre que un nodo cliente envíe una solicitud. El nodo que crea una llamada de servicio se llama nodo Servidor y el que llama al servicio se llama nodo cliente. [13]

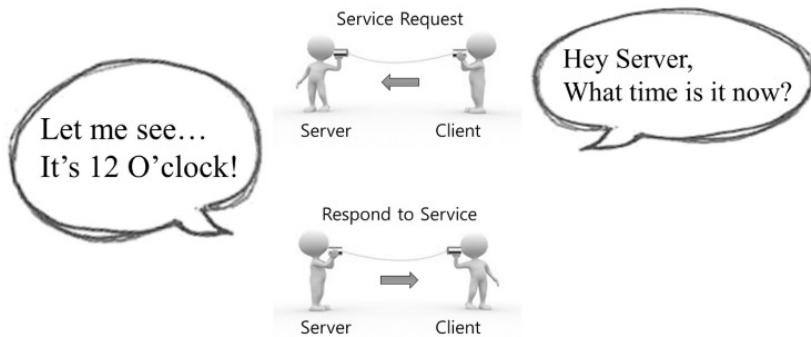


Figura 2.17: Comunicación de mensajes

2.2.4. Procesamiento de imágenes

Una vez definido el sistema que dará soporte a nuestro proyecto el siguiente paso es detallar el proceso por el cual una imagen es procesada en un ordenador.

Primero, la información contenida en las imágenes se puede representar de maneras completamente diferentes. Los más importantes son la representación espacial y la representación del número de onda. Estas representaciones solo miran datos espaciales desde diferentes puntos de vista. La conversión entre la representación espacial y el número de onda es la conocida transformada de Fourier. [3]

Empezaremos por definir que es una imagen digital; donde el concepto de imagen 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. [18]. El valor de esta función depende de la cantidad de luz que incide sobre la escena vista, así como de la parte que sea reflejada por los objetos que componen dicha escena.

Como consecuencia, $f(x,y)$ debe ser diferente de cero y finita. Esto es:

$$0 < f(x,y) < \infty \quad (2.2)$$

La función $f(x,y)$ se caracteriza por dos componentes: [9] Estos componentes son llamados iluminación y reflexión.

- **Iluminación:** la cantidad de luz incidente procedente de la fuente sobre la escena.
- **Reflexión:** La cantidad de luz reflejada por los objetos de la escena.

siendo descritos por $i(x,y)$ para la iluminación y $r(x,y)$ para la reflexión. El producto de ambas funciones proporciona la función $f(x,y)$

$$f(x,y) = i(x,y)r(x,y) \quad (2.3)$$

donde

$$0 < i(x,y) < \infty \quad (2.4)$$

y

$$0 < r(x,y) < 1 \quad (2.5)$$

La ecuación 2.5 indica que la reflectancia está acotada entre 0 (absorción total) y 1 (reflexión total). La naturaleza de $i(x,y)$ está determinada por la fuente de iluminación, y la de $r(x,y)$, por las características de los objetos.

Las funciones tienen un dominio y un rango. Si el dominio y rango son continuos, la señal es continua o analógica; si el dominio es discreto pero el rango no, la señal también será discreta; y si el dominio y el rango son discretos, como en el caso de las imágenes, la señal es digital.

Las computadoras no pueden manejar imágenes continuas sino solo matrices de números digitales. Por lo tanto, se requiere representar imágenes como conjuntos de puntos bidimensionales. Un punto en la cuadrícula 2D se llama píxel o pel.

Un píxel representa la irradiancia en la posición de cuadrícula correspondiente. En el caso más simple, los píxeles se encuentran en una cuadrícula rectangular. La posición del píxel se da en la notación común para matrices. El primer índice, m , denota la posición de la fila, el segundo, n , la posición de la columna.

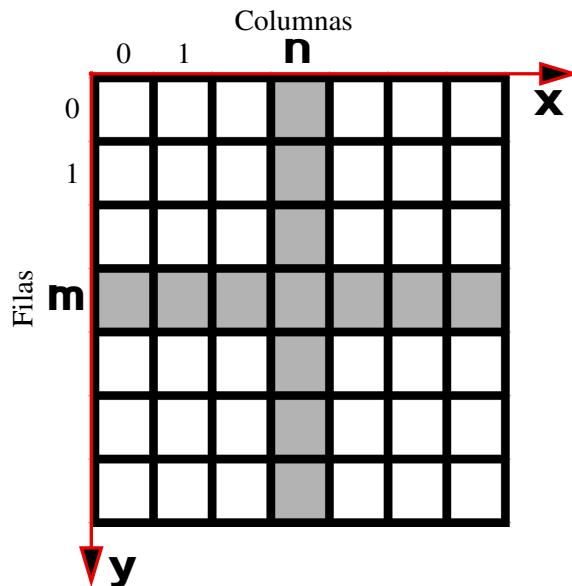


Figura 2.18: Representación 2D de imágenes digitales mediante conjuntos de puntos discretos en una matriz rectangular

La función de imagen $f(x,y)$ es digitalizada en la memoria del computador, tanto espacialmente como en amplitud. La digitalización de las coordenadas espaciales (x,y) está asociada al concepto de muestreo, mientras que la digitalización de la amplitud al de cuantificación de los niveles de gris. [9]

El muestreo es la conversión que sufren las dos dimensiones espaciales de la señal analógica, y que genera la noción de píxel.

Los puntos 2D (coordenadas de píxeles en una imagen) se pueden denotar usando un par de valores, $x = (x,y) \in \mathbb{R}^2$ [19]

$$x = \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.6)$$

Los puntos 2D también se pueden representar utilizando coordenadas homogéneas, $\tilde{x} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathbb{P}^2$ donde los vectores que difieren solo por escala se consideran equivalentes. $\mathbb{P}^2 = \mathbb{R}^3 - (0,0,0)$ se llama el espacio proyectivo 2D.

2.2.5. Cuantización

Para usar con una computadora, la energía medida en el plano de la imagen debe asignarse a un número limitado Q de valores discretos de gris. Este proceso se llama cuantización. El número de niveles de cuantificación requeridos en el procesamiento de imágenes se puede discutir con respecto a dos criterios. [19]

La cuantificación es la conversión que sufre la amplitud de la señal analógica; así se genera el concepto de nivel de gris o intensidad. En general, los datos de imagen se cuantifican en 256 valores de gris. Luego, cada píxel ocupa 8 bits o un byte. Para el caso de tener 256 niveles de gris (0-255), el 0 corresponde a un objeto no iluminado o que absorbe todos los rayos luminosos que inciden sobre él (negro), y el nivel 255 a un objeto muy iluminado o que refleja todos los rayos que inciden sobre él (blanco).

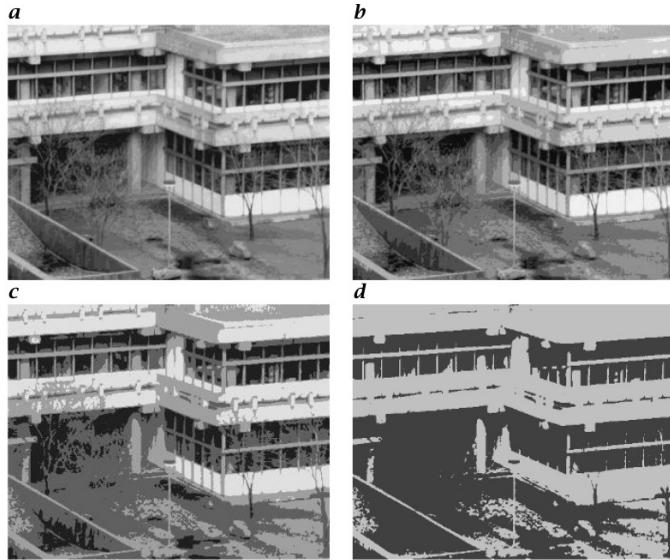


Figura 2.19: Ilustración de cuantización [3]. La misma imagen se muestra con diferentes niveles de cuantización: a 16, b 8, c 4, d 2. Muy pocos niveles de cuantificación producen bordes falsos y hacen que las características con bajo contraste desaparezcan parcial o totalmente.

Al efecto de disminuir la resolución y mantener las dimensiones físicas se le conoce como efecto tablero de ajedrez.

2.2.6. Librerías OPENCV

Dentro del mundo del procesamiento de imágenes digitales encontramos una amplia variedad de desarrolladores que han construido librería especializadas en el tratamiento de imágenes. Dentro de las que destacan nombres como OpenCV, HALCON, y point cloud library. Siendo la primera de estas de código abierto. Por esta razón he decidido utilizar OpenCV en este proyecto, aunado a que tiene soporte de contribuidores en todo el mundo.

OpenCV es una biblioteca open-source de Visión Artificial originalmente desarrollada por Intel en 1999. Desde entonces se ha utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Contiene más de 500

funciones que abarcan una gran gama de áreas en el proceso de Visión, como reconocimiento de objetos, reconocimiento facial, calibración de cámaras, visión estéreo y visión robótica y no dejan de añadirse nuevos algoritmos continuamente. Su principal ventaja es que se puede utilizar de manera gratuita, incluso para realizar aplicaciones comerciales. [20]



Figura 2.20: Símbolo característico RGB de OpenCV

Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS.

2.2.6.1. Espacio de color

El uso del color en el procesamiento de imágenes está motivado por dos factores principales. Primero, el color es un poderoso descriptor que a menudo simplifica la identificación y extracción de objetos de una escena. En segundo lugar, los humanos pueden discernir miles de tonos e intensidades de color, en comparación con solo los tonos de gris de una cámara. [21]

El espacio de color es un espacio geométrico tridimensional con ejes adecuadamente definidos para que los símbolos de todas las percepciones de color posibles de humanos u otros animales encajen en él en un orden correspondiente al orden psicológico. Los parámetros síquicos de la percepción del color son la luminosidad, el tono, y la saturación. El propósito de un modelo de color (también llamado espacio de color) es facilitar la especificación de colores de alguna manera estándar, generalmente aceptada. En esencia, un modelo de color es una especificación de un sistema de coordenadas y un subespacio dentro de ese sistema donde cada color está representado por un solo punto.

Hay multiples espacios de color en la actualida, siendo el espacio RGB el mas conocido, esto debido a su antigua creación hecha por Helmholtz y Schrödinger, asumiendo tres procesos fundamentales de visión del color R, G y B (Rojo, Verde y Azul).

$$\frac{dR}{R} = \frac{dG}{G} = \frac{dB}{B} = \text{constante} \quad (2.7)$$

donde dR es el incremento / decremento en magnitud del estímulo descrito por el proceso fundamental R, y comparable para los otros dos estímulos. [22]

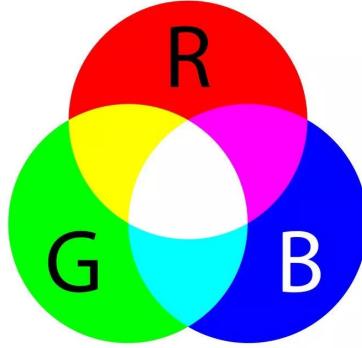


Figura 2.21: Representación del espacio de color RGB

Uno de los principales problemas que el modelo RGB presenta es la mezcla de la información del color (tono y saturación) y la intensidad. Aun así sigue siendo el modelo más utilizado y el que los productores de cámara utilizan para ser implementados en estas. Estos dos inconvenientes se resuelven con el siguiente espacio de color.

2.2.6.2. HSI

"HSI" se refiere al modelo de color de Tono, Saturación, Intensidad para presentar datos de color. El modelo de color de intensidad de saturación de tono (HSI) se parece mucho a las propiedades de detección de color de la visión humana. El componente de intensidad está relacionado con el componente de luminancia desacoplado del color. Los componentes de matiz y saturación están relacionados con la forma en que un humano percibe el color. Dicha relación con la visión humana hace que sea deseable utilizar el modelo de color HSI para las técnicas de procesamiento de imágenes en color, como la mejora de imágenes y la segmentación de imágenes. [23]

Las transformaciones matemáticas que permiten el paso del espacio RGB al HSI son realizadas normalmente mediante hardware específico. Las ecuaciones son:

$$I = \frac{R + G + B}{3} \quad (2.8)$$

$$H = \arctan \left(\frac{\sqrt{3}(G - B)}{(R - G) + (R - B)} \right) \quad (2.9)$$

$$S = 1 - \frac{\min(R, G, B)}{I} \quad (2.10)$$

Así como el espacio RGB se representa por un cubo, el HSI lo forman dos pirámides unidas por su base. Dependiendo de la intensidad se tiene un corte a los conos.

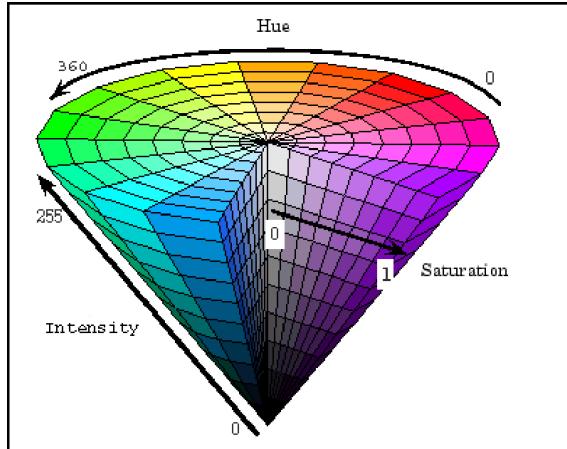


Figura 2.22: Representación del espacio de color HSI

Variaciones de este espacio de color son:

- **HSL**(Hue, Saturation Lightness)
- **HSV** (Hue Saturation Value)
- **HCI** (Hue Chroma / Colourfulness Lightness)
- **HVC** (Hue Value Chroma)
- **TSD** (Hue Saturation Darkness)

2.2.6.3. Características del modelo HSV

Esta variación del modelo HSI es la base en librerías dedicadas a la visión artificial. En resumen tenemos los siguientes valores para cada característica de este modelo.

Tono(H)

El tono es la porción de color del modelo, expresada como un número de 0 a 360 grados:

- El rojo cae entre 0 y 60 grados.
- El amarillo cae entre 61 y 120 grados.
- El verde cae entre 121-180 grados.
- El cian cae entre 181-240 grados.
- El azul cae entre 241-300 grados.
- Magenta cae entre 301-360 grados.

Saturación

La saturación describe la cantidad de gris en un color particular, del 0 al 100 por ciento. La reducción de este componente hacia cero introduce más gris y produce un efecto desvanecido. A veces, la saturación aparece como un rango de solo 0-1, donde

0 es gris y 1 es un color primario.

Valor(V)[brillo]

El valor funciona junto con la saturación y describe el brillo o la intensidad del color, de 0 a 100 por ciento, donde 0 es completamente negro y 100 es el más brillante y revela la mayor cantidad de color.

2.2.6.4. Espacio de color en OpenCV

En el caso de imágenes de 8 bits y 16 bits, R, G y B se convierten al formato de punto flotante y se escalan para ajustarse al rango de 0 a 1.

$$V \leftarrow \max(R, G, B) \quad (2.11)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R, G, B)}{V}, & \text{if } V = R \\ 0, & \text{otherwise} \end{cases} \quad (2.12)$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)), & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)), & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)), & \text{if } V = B \end{cases} \quad (2.13)$$

Si $H < 0$ entonces $H \leftarrow H + 360$. En la salida tenemos $0 \leq V \leq 1$, $0 \leq S \leq 1$, $0 \leq H \leq 360$. Los valores se convierten luego al tipo de datos de destino:

- Imagen de 8 bits: $V \leftarrow 255V$, $S \leftarrow 255S$, $H \leftarrow H/2$ (para ajustar de 0 a 255)
- Imágenes de 16 bits: (actualmente no compatible)
- Imágenes de 32 bits: H, S y V se dejan como están

[4]

CAPÍTULO 3

MODELO MATEMATICO

En esta sección se analizara el modelo matemático que rige al sistema, pasando por diferentes marcos de referencia, inercial, de la gimbal y de la camara.

CAPÍTULO 4

VISION ARTIFICIAL

4.1. Camara

Como se abordo al inicio de este proyecto, la cámara es la parte fundamental en la captura de datos, suple la función de un ojo y depende de diversos parámetros el que tengamos una captura de calidad. Para este trabajo profesional la cámara que se utilizo fue la del fabricante HardKernel y que lleva de nombre Ocam



Figura 4.1: Camara 'Ocam'

Que tiene las siguientes especificaciones.

- **Sensor:** CMOS image sensor.
- **Lente:** Lente estandar M12 distancia focal de 3.6mm.
- **Field of view:** 65 grados.
- **Tamaño del sensor:** 0.25inch (3673.6 μm x 2738.4 μm)
- **Tamaño del pixel:** 1.4 μm x 1.4 μm .
- **Interfaz:** USB 3.0 Super-Speed.

- **Frame rate:**

1920 x 1080 a 30fps, 1280 x 720 a45fps, 640 x 480 a30fps

El acceso directo a la memoria a través de USB 3.0 permite que los datos se escriban en la memoria principal sin pasar por la CPU. Reduce significativamente la carga de trabajo de la CPU.

4.2. Algoritmo general

Algorithm 1 Algoritmo general del sistema de vision

- 1: Calibrar camara
 - 2: Capturar frame a 60fps
 - 3: Publicar frame en ROS
 - 4: Suscribirse al nodo publicador
 - 5: Convertir RGB a HSV
 - 6: Acotar el modelo HSV al color de elección
 - 7: Agregar filtro morfológico
 - 8: Obtener centroide de la figura obtenida en 6
 - 9: Publicar coordenadas del centroide
-

4.3. Calibración de camara

4.4. Comunicación con ROS

En la sección de vision artificial se lanzan dos nodos, uno encargado de capturar y publicar frames a 60hz y el otro que se suscribe a dicho nodo y realiza un procesamiento con la información obtenida para posterior publicar coordenadas.

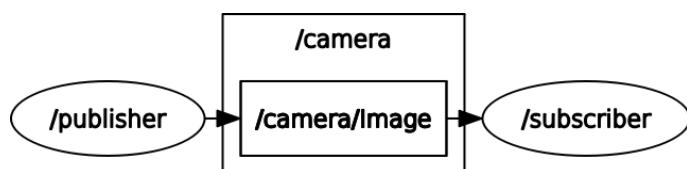


Figura 4.2: Nodos y topic

En la figura 4.1 se puede observar dos nodos conectados por un topic llamado /Image encargado de comunicar la imagen de un nodo a otro.

4.4.1. Publisher

Como vimos anteriormente opencv es una libreria open-source que se encarga del procesamiento de imagenes, también abordamos un poco acerca de como ROS comunica nodos y los tipos de datos que pueden ser publicados. Al hablar de que se va a publicar una imagen estamos refiriendonos a una matriz que en este caso sera

de 640 x 480.

ROS pasa las imágenes en su propio formato sensor_msgs/Image , pero en este caso usaremos ROS junto con las librerías de OpenCV. CvBridge es una biblioteca ROS que proporciona una interfaz entre ROS y OpenCV.

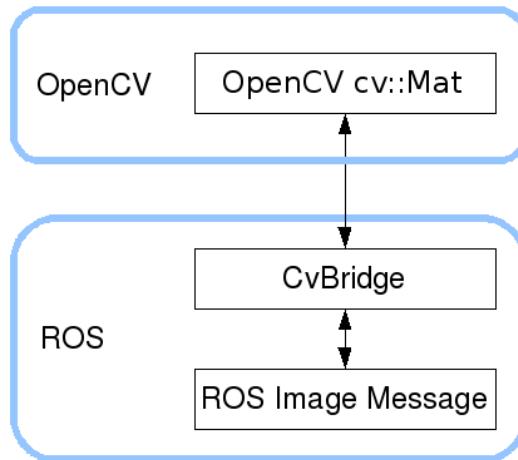


Figura 4.3: Comunicación entre opencv y ROS

Al convertir un mensaje sensor_msgs/Image en una imagen Cv, CvBridge reconoce dos casos de uso distintos:

- Queremos modificar los datos en el lugar.
Tenemos que hacer una copia de los datos del mensaje ROS.
- No modificaremos los datos.
Podemos compartir con seguridad los datos que posee el mensaje ROS en lugar de copiarlos.

La entrada es el puntero del mensaje de imagen, así como un argumento de codificación opcional. La codificación se refiere al destino CvImage.

Para codificaciones de imágenes populares, CvBridge opcionalmente realizará conversiones de color o profundidad de píxeles según sea necesario. Para este proyecto se utiliza bgr8: CV_8UC3, es decir, que el orden del color es Azul, Verde y Rojo.

Con el fin de entender este nodo se hizo el siguiente diagrama de flujo:

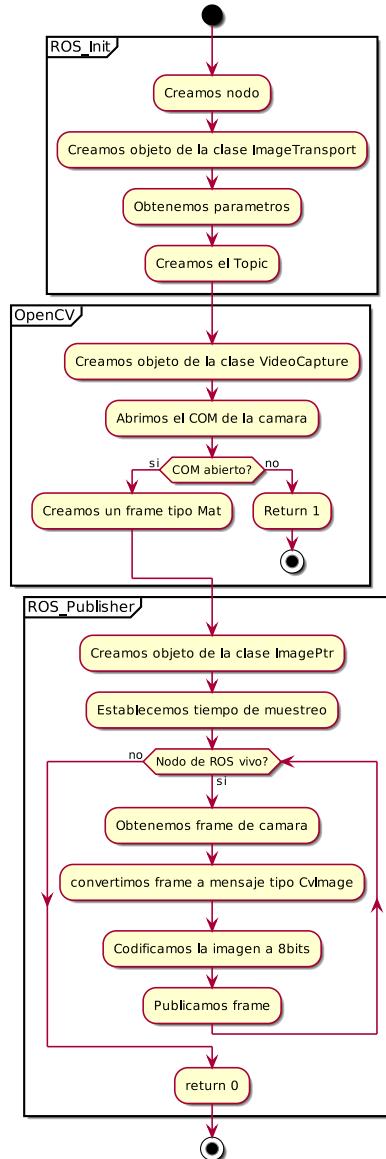


Figura 4.4: Diagrama de flujo del programa publisher

El código se encuentra en la parte del Apéndice A, 'codigo1'

4.4.2. Subscriber

Este nodo tiene dos funciones principales, Suscribirse al nodo publisher y publicar un array de tamaño 2 tipo entero que guardara las coordenadas en X y Y del centroide del target.

El proceso que se ejecuta se vera más a detalle en las siguientes secciones de este capítulo.

4.5. Espacio de color

Como se puede apreciar en el algoritmo general, el paso 5 es el cambio de espacio de color de RGB a HSV, además ya sabemos el porque es mejor generar colores

partiendo del modelo HSV. Así que lo que ahora sigue es la implementación en código y las pruebas que se hicieron para tener un rango de colores y así tener una base de datos a la cual recurriremos después.

El siguiente digrama muestra el flujo que el programa 2(ver apendice A) siguió para el cambio de espacio de color.

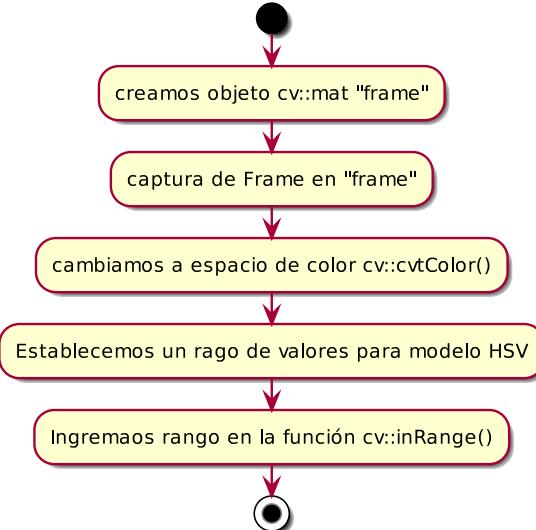


Figura 4.5: Diagrama de flujo para cambio de espacio de color

Empezamos con la función definida que nos dan las librerías de opencv para transformar de RGB a HSV.

```

◆ cvtColor()
void cv::cvtColor ( InputArray src,
                  OutputArray dst,
                  int code,
                  int dstCn = 0
)
  
```

Figura 4.6: Función Color [4]

Donde

- **src:** Imagen de entrada de 8 bits sin signo
- **dst:** Imagen de salida del mismo tamaño y profundidad que src.
- **code:** Código de conversión de espacio de color en este caso es COLOR_BGR2HSV.
- **dstCn:** Número de canales en la imagen de destino; si el parámetro es 0, el número de canales se deriva automáticamente de src y código.

El formato de color predeterminado en OpenCV a menudo se denomina RGB, pero en realidad es BGR (los bytes se invierten). Entonces, el primer byte en una imagen en color estándar (24 bits) será un componente azul de 8 bits, el segundo byte será verde y el tercer byte será rojo. El cuarto, quinto y sexto bytes serían el segundo píxel (Azul, luego Verde, luego Rojo), y así sucesivamente.

Ahora pasamos a la función que acota el color que deseemos, esta es la llamada `inRange()`

```
◆ inRange()
void cv::inRange ( InputArray  src,
                  InputArray  lowerb,
                  InputArray  upperb,
                  OutputArray dst
                )
```

Figura 4.7: Función `inRange`

Donde

- **src** primera matriz de entrada.
- **lowerb** matriz de límite inferior o un escalar.
- **upperb** matriz de límite superior o un escalar.
- **dst** Conjunto de salida dst del mismo tamaño que src y tipo CV_8U.

Debido a que no hay un solo color azul o rojo, etc, si no más bien un rango que cubre la gama de azules, amarillo, naranja, etc. Por esa razón se hicieron pruebas para determinar los valores HSV que corresponden a los siguientes colores: Amarillo, Azul, Rojo, Verde y Naranja y de esta manera a la hora de seguir un objetivo de dado color el sistema no tenga problemas en reconocer esa gama de color.

La siguiente figura ilustra una gama de colores que va desde el rojo al azul, donde la tarea es obtener un conjunto de imágenes que correspondan a un sector de la misma.

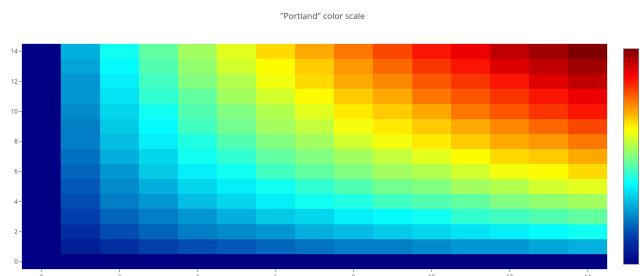


Figura 4.8: Espectro de colores

De donde apartir del código 2 se obtuvieron los siguientes resultados.

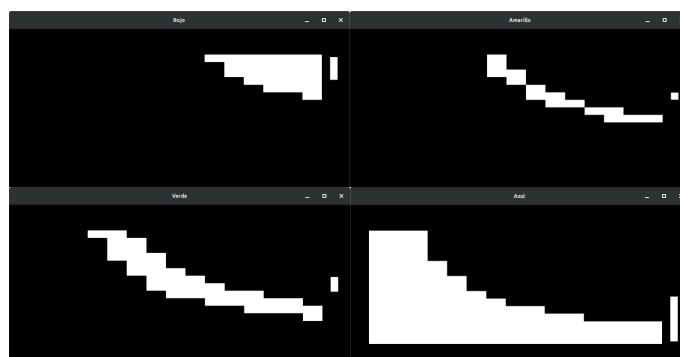


Figura 4.9: Separación de colores

En la figura 4.9 se aprecia la separación en 4 colores; Rojo, Amarillo, Verde y Azul. con los siguientes rangos de valores.

Cuadro 4.1: A table without vertical lines.

| | H_{min} | H_{max} | S_{min} | S_{max} | V_{min} | V_{max} |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Rojo | 0 | 10 | 100 | 255 | 100 | 255 |
| Amarillo | 25 | 35 | 50 | 255 | 50 | 255 |
| Verde | 35 | 75 | 100 | 255 | 100 | 255 |
| Azul | 75 | 130 | 55 | 255 | 55 | 255 |

A manera de tener una mejor visualización de los resultados, la siguiente figura concatena cada segmento de color y le asigna un color plano a cada uno.

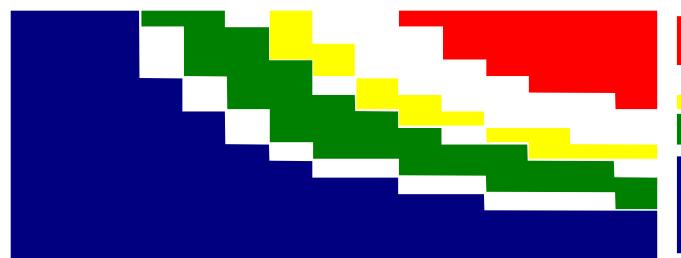


Figura 4.10: Separación de colores

APENDICE A

Código 1

```
1 #include <ros/ros.h>
2 #include <image_transport/image_transport.h>
3 #include <opencv2/highgui/highgui.hpp>
4 #include <cv_bridge/cv_bridge.h>
5 #include <sstream>
6 #include <iostream>
7
8 using namespace std;
9 using namespace cv;
10
11 image_transport::Publisher pub;
12 VideoCapture VC;
13 int CAMCOM;
14 int SAMPLETIME;
15 void publisher();
16
17
18 int main(int argc, char **argv)
19 {
20     ros::init(argc, argv, "image_publish");
21     publisher();
22
23     return 0;
24 }
25
26 void publisher()
27 {
28     ros::NodeHandle nh;
29     image_transport::ImageTransport imgt(nh);
30     nh.getParam("/com", CAMCOM);
31     nh.getParam("/sampletime", SAMPLETIME);
32     pub = imgt.advertise("camera/Image", 1);
33     VC.open(CAMCOM);
34     if (!VC.isOpened())
35     {
36         ROS_INFO("Camara no conectada");
```

```

37     }
38     else
39     {
40         ROS_INFO("Camara conectada en el COM %d", CAMCOM);
41         cv::Mat FRAME;
42         sensor_msgs::ImagePtr MSG;
43
44         ros::Rate loop_rate(SAMPLETIME); /*HZ*/
45
46         while (nh.ok())
47         {
48             VC >> FRAME;
49             if (!FRAME.empty())
50             {
51                 MSG = cv_bridge::CvImage(std_msgs::Header(),
52                                         "bgr8", FRAME).toImageMsg();
53                 pub.publish(MSG);
54                 cv::waitKey(30);
55             }
56             ros::spinOnce();
57             loop_rate.sleep();
58         }
59     }
}

```

4.6. Código 2

```

1 #include <opencv2/core.hpp>
2 #include <opencv2/imgcodecs.hpp>
3 #include <opencv2/highgui.hpp>
4 #include <opencv2/imgproc/imgproc.hpp>
5 #include <iostream>
6
7 using namespace cv;
8 using namespace std;
9
10 int low_H = 60;
11 int low_S = 60;
12 int low_V = 60;
13 int high_H = 85;
14 int high_S = 255;
15 int high_V = 255;
16
17 int main(int argc, char **argv)
18 {
19     Mat image;
20     Mat imgHSV;

```

```
21     image = imread(("IMAGE NAME"), IMREAD_COLOR);  
22  
23     if (image.empty()) // Check for invalid input  
24     {  
25         cout << "Could not open or find the image" << std::  
26             endl;  
27         return -1;  
28     }  
29  
30     cvtColor(image, imgHSV, COLOR_BGR2HSV);  
31     inRange(imgHSV, Scalar(low_H, low_S, low_V), Scalar(  
32         high_H, high_S, high_V), imgHSV);  
33  
34     namedWindow("Escala de colores", 0); // Create a window  
35         for display.  
36     resizeWindow("Escala de colores", 683, 316);  
37     namedWindow("HSV", 0); // Create a window for display.  
38     resizeWindow("HSV", 683, 316);  
39  
40     imshow("Escala de colores", image); // Show our image  
41         inside it.  
42     imshow("HSV", imgHSV); // Show our image  
43         inside it.  
44     waitKey(0); // Wait for a  
45         keystroke in the window  
46     return 0;  
47 }
```

APENDICE B

4.7. ODROID

El modelo XU-4 cuenta con las siguientes especificaciones: [24]

■ Procesador

Samsung Exynos5422 CortexTM-A15 2Ghz and CortexTM-A7 Octa core CPUs con Mali Mali-T628 MP6 GPU.

Es un procesador móvil que fue lanzado en 2014 por Samsung para su celular S5.

■ Almacenamiento

Hay dos tipos de almacenamiento de el sistema operativo. El primero es usando una microSD y el segundo es utilizando un modulo eMMC.

■ Alimentación

Es necesario alimentar con 5V y con un minimo de 4A para su optimo funcionamiento.

■ Perifericos

- USB hosts
- HDMI
- Ethernet RJ-45
- GPIO (Entradas y salidas de SPI,ADC e IRQ)
- Comunicación serial
- USB 3.0



Figura 4.11: Odroid XU-4

BIBLIOGRAFÍA

- [1] M. S. Grewal, *Global Positioning Systems, inertial navigation, and integration.* Wiley, 2007.
- [2] “gimbal bearing – liquid rocket engines (j-2x, rs-25, general).” <https://blogs.nasa.gov/J2X/tag/gimbal-bearing/>, 2013. Accessed: 2020-01-10.
- [3] B. Jähne, *Digital Image processing.* Springer, 6th revised and extended ed. ed., 1997.
- [4] OpenCV, “Color conversions.”
- [5] “Gimbal.” <https://playlists.net/artists/gimbal>. Accessed: 2020-01-10.
- [6] “Nasa’s j-2x rocket engine development.” <https://www.nasa.gov/exploration/systems/sls/j2x>. Accessed: 2020-01-10.
- [7] “The gimbal rig mercury astronaut trainer.” <https://www.nasa.gov/centers/glenn/about/history/mastif.html>, 2017. Accessed: 2020-01-10.
- [8] A. Bharath, *Next Generation Artificial Vision System.* Artech House, 2008.
- [9] J. R. M. Vilet, *Apuntes de Procesamiento Digital de Imágenes.* Facultad de Ingeniería UASLP, first ed., jan 2005.
- [10] A. A. of Ophthalmology, “Fotorreceptores,” nov 2017.
- [11] E. B. PHOTOGRAPHY, “How cameras work-the parts of a camera.”
- [12] MUNDIARIO, “Cómo el cerebro procesa las imágenes.”
- [13] L. Joseph, *Robot Operating System for Absolute Beginners: Robotics Programming Made Easy.* apress, 2018.
- [14] HardKernel, “Odroid xu-4,” jan 2020.

- [15] Ubuntu, “What is ubuntu mate?,” aug 2014.
- [16] “Ros/concepts - ros wiki.” <http://wiki.ros.org/ROS/Concepts>. Accessed: 2020-01-10.
- [17] M. Quigley, *Programming Robots with ROS*. O'Reilly, 2015.
- [18] A. de la Escalera, *Visión por Computador. Fundamentos y métodos*. Prentice Hall, 2011.
- [19] R. Szeliski, *Computer Vision: Algorithms and Applications*. Texts in Computer Science, Springer, 2011.
- [20] OpenCV, “About it.”
- [21] R. E. W. Rafael C. Gonzalez, *Digital image processing*. Prentice Hall, 2nd ed ed., 2002.
- [22] R. G. Kuehni, *Color Space and Its Divisions Color Order from Antiquity to the Present*. Wiley-Interscience, 2003.
- [23] T. C. W. P. Kim, “Image filtering in hs color space,” *University of Washington*, 2000.
- [24] HardKernel, *ODROID-XU4*. HardKernel, 2017.