

Universidad Aeronáutica en Querétaro



Innovación educativa para el desarrollo de México

---

**Desarrollo y control de una gimbal  
de dos grados de libertad mediante  
visión artificial para el seguimiento  
de objetivos**

---

Trabajo Profesional para obtener el Titulo de Ingeniero en  
Electrónica y Control de Sistemas de Aeronaves.

Marco Antonio Aguilar Gallardo

Dirige: Antonio Flores Moreno

Municipio de Colón, Querétaro 26 de mayo de 2020



# Proyecto de titulación



Departamento de Ingeniería  
Universidad Aeronáutica en Querétaro  
Ingeniería en Electrónica y Control de Sistemas de Aeronaves

Innovación educativa para el desarrollo de México

## **Desarrollo y control de una gimbal de dos grados de libertad mediante visión artificial para el seguimiento de objetivos**

Marco Antonio Aguilar Gallardo

1. Asesor **Antonio Flores Moreno**

Maestro en ciencias  
Asesor Técnico

Supervisores Antonio Flores Moreno and María del Carmen

26 de mayo de 2020

**Marco Antonio Aguilar Gallardo**

*Desarrollo y control de una gimbal de dos grados de libertad mediante visión artificial para el seguimiento de objetivos*

Innovación educativa para el desarrollo de México, 26 de mayo de 2020

Asesores: Antonio Flores Moreno

Supervisores: Antonio Flores Moreno y María del Carmen

**Proyecto de titulación**

*Ingeniería en Electrónica y Control de Sistemas de Aeronaves*

Universidad Aeronáutica en Querétaro

Departamento de Ingeniería

Carretera Estatal 200 Querétaro – Tequisquiapan No. 22154

76278

# Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

# Abstract (different language)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.



# Acknowledgement

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mau-

ris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Tema de investigación . . . . .	1
1.2	Justificación . . . . .	1
1.3	Objetivo . . . . .	2
1.4	Objetivos específicos . . . . .	2
1.5	Estado de la cuestión . . . . .	2
1.6	Contribuciones . . . . .	5
1.7	Alcances . . . . .	6
<b>2</b>	<b>Marco Tórico</b>	<b>7</b>
2.1	Óptica . . . . .	7
2.1.1	Descripción general del sistema visual humano . . . . .	8
2.1.2	Estructura del ojo humano . . . . .	8
2.1.3	Formación de imágenes en el ojo . . . . .	10
2.1.4	Camara digital . . . . .	11
2.1.5	Componetes de la camara . . . . .	11
2.1.6	Formación de la imagen en la camara . . . . .	12
2.1.7	Cálculo de la imagen . . . . .	13
2.1.8	Modelo de la cámara . . . . .	15
2.2	Procesamiento de datos . . . . .	17
2.2.1	Tarjeta procesadora . . . . .	17
2.2.2	Sistema operativo . . . . .	18
2.2.3	ROS . . . . .	19
2.2.4	Procesamiento de imagenes . . . . .	22
2.2.5	Cuantización . . . . .	24
2.2.6	Librerias OPENCV . . . . .	25
2.2.7	Calibración de camara . . . . .	25
2.2.8	Espacio de color . . . . .	27
2.2.9	Transformaciones morfológicas . . . . .	30
2.2.10	Contraste y brillo . . . . .	35
2.2.11	Teorema de Green . . . . .	36

<b>3 Modelo matemático</b>	<b>39</b>
3.1 Marco de referencia . . . . .	39
3.1.1 Marco de referencia Inercial [I] . . . . .	40
3.1.2 Marco de referencia del cuerpo [B] . . . . .	41
3.1.3 Marco de referencia de la gimbal [G] . . . . .	42
3.2 Modelo matematico gimbal . . . . .	42
<b>4 Visión Artificial</b>	<b>45</b>
4.1 Cámara . . . . .	45
4.2 Algoritmo general . . . . .	46
4.3 Comunicación con ROS . . . . .	46
4.3.1 Publisher . . . . .	47
4.3.2 Subscriber . . . . .	48
4.4 Calibración de camara . . . . .	49
4.5 Espacio de color . . . . .	50
4.5.1 Pruebas de campo . . . . .	54
4.6 Corrección de brillo . . . . .	56
4.6.1 Corrección de Gamma . . . . .	59
4.7 Filtro morfológico . . . . .	61
4.8 Centroide . . . . .	62
<b>5 Sistema Embebido</b>	<b>67</b>
5.1 Arquitectura . . . . .	67
5.2 Mecanismo móvil . . . . .	68
5.3 Prototipo físico . . . . .	70
5.4 ROS-Microcontrolador . . . . .	71
5.5 Control . . . . .	72
<b>Bibliografía</b>	<b>77</b>
<b>Índice de figuras</b>	<b>81</b>
<b>Índice de cuadros</b>	<b>85</b>
<b>Lista de códigos</b>	<b>87</b>
<b>A Apendice A</b>	<b>89</b>
<b>B Apendice B</b>	<b>93</b>
B.1 ODROID . . . . .	93

# Introducción

*“ You can’t do better design with a computer,  
but you can speed up your work enormously.*

— Wim Crouwel  
(Graphic designer and typographer)

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 cámaras 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 imágenes sea deficiente. Es por ello que la implementación de un sistema estabilizador de cámaras es necesario.

Tareas tales como la geo-localización requieren como entrada ángulos de azimuth y de elevación que son entregadas por el sistema gimbal embebido en el UAV. El algoritmo de geo-localización utiliza la salida del GPS, el vector de línea de visión normalizado del algoritmo de visión y la actitud para estimar la posición del objeto en el marco de inercia y la distancia al objeto.

## 1.2 Justificación

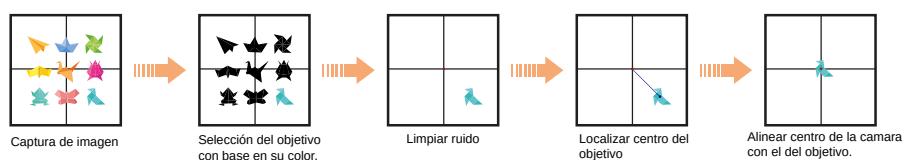
La presente investigación se enfocara en la implementación de una gimbal de dos grados de libertad utilizando un framework de código abierto y bajo

la licencia de Open-Source siendo este el motivo principal de contribuir a la comunidad y el de la creación de un repositorio de libre acceso para futuras investigaciones y mejoramiento de la calidad.

Para la universidad aeronáutica en Querétaro representara solo el inicio de una potencial investigación en MAV'S<sup>1</sup>.

## 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.:** Representación grafica del objetivo del proyecto

## 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

---

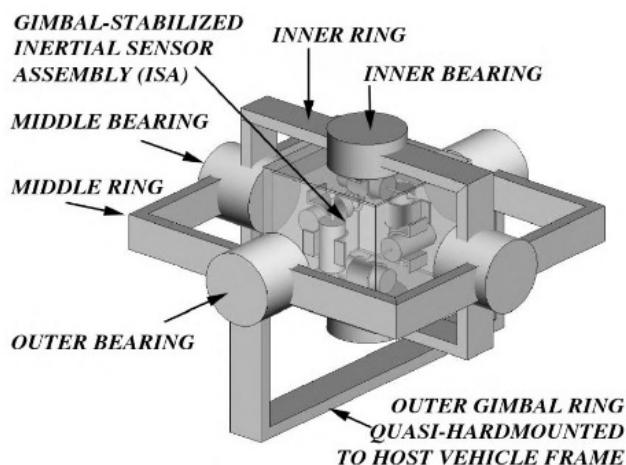
<sup>1</sup>Micro air vehicle

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. [@Pla]

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 un sensor inercial [Moh07]

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.[@NASB]



**Figura. 1.3.:** Uso de una gimbal en un motor de propulsión [@NASa]

#### ■ 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.[@NAS17]



**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 Rockyz "The Shining"

## 1.6 Contribuciones

Actualmente la forma en que se desarrollan los sistemas gimbal de los vehículos aéreos no tripulados están basados en el procesamiento de datos con Arduino conectado a ROS, la idea de migrar a una tarjeta micro controladora mas fiable como la TMS570LC43 hace que esta sea una de las principales aportaciones, ya que queda bajo licencia de software libre, además de la

investigación en mejorar el algoritmo de búsqueda de centroide de la visión artificial, mediante el uso de corrección de factor gamma. Lo que nos puede permitir ajustar los filtros con un sensor de luminancia

A nivel escolar, es la primera tesis que incluye visión artificial en la carrera de Ingeniería Electrónica y Control de Sistemas de Aeronaves, siendo esta investigación el inicio de un proyecto de instrumentación de MAV con la incorporación de una tarjeta monoprocesador con ROS.

## 1.7 Alcances

La investigación actual tiene los siguientes alcances:

- Hacer el control para dos grados de libertad.
- Utilizar filtros morfológicos (opening y closing).
- Que la variación de gamma sea por software y no por hardware
- No será acoplado a un vehículo aéreo, queda en simulación en tierra.

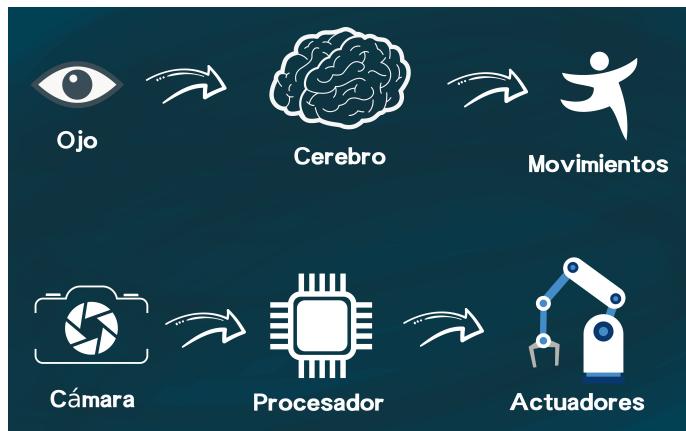
# Marco Téorico

“Never memorize something that you can look up.

— Albert Einstein  
(Theoretical physicist)

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

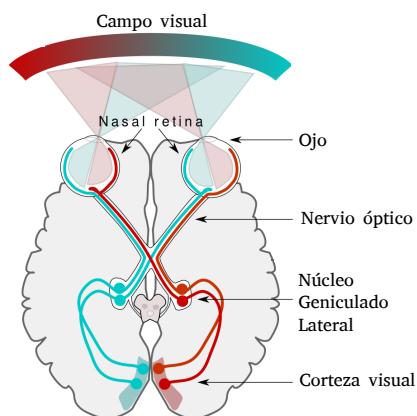
La visión artificial surge de un amplio estudio probabilístico y matemático del procesamiento de imágenes digitales, pero sobre todo de análisis humanos y de la intuición ya que de estas últimas el ingeniero hace selección de entre una u otra técnica. Esta elección se basa usualmente en juicios visuales 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. [Ani08]



**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 esta 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.

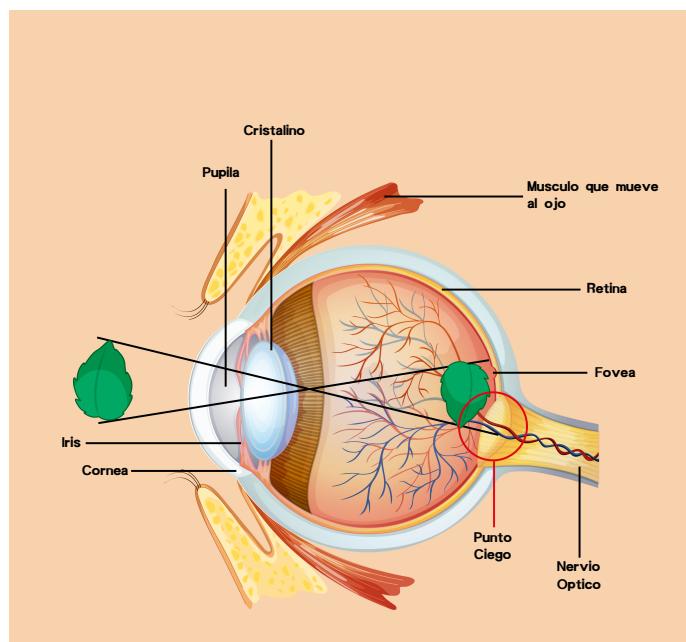
[Jos05]

### 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. 2.3.:** 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.[Jos05]

### **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.[@Ame17]

#### **2.1.4 Camara digital**

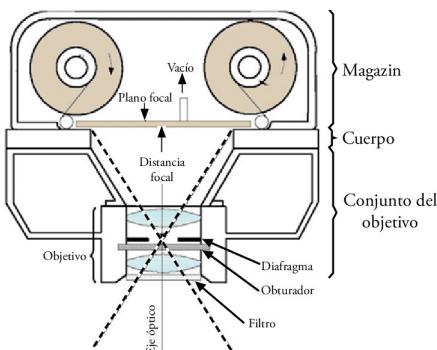
La cámara digital es uno de los cambios más importantes en esta era de la digitalización de la información. Es un invento tan revolucionario y que dista tanto de su predecesor análogo.

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. [@PHO]

#### **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 "dobra." 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. 2.4.: Componentes de la camara**

## 2.1.6 Formación de la imagen en la camara

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. 2.5.:** Proceso general de captura de imagen por la cámara

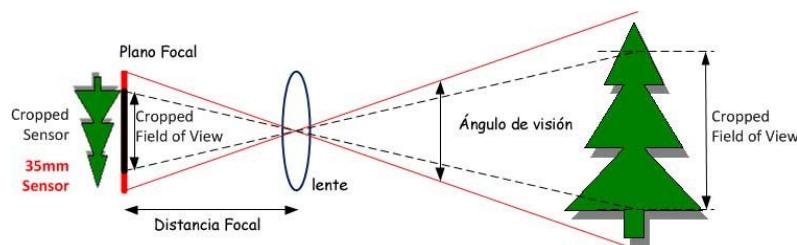
Cuando los rayos paralelos pasan a través de una lente convexa, convergen hacia un punto que se denomina punto focal.

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. 2.6.:** Captura de imagen

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.

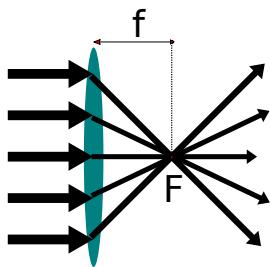
En las ópticas habrá que tener por tanto en cuenta su F mínimo, que indicará la máxima cantidad de luz que puede atravesar la óptica y que tendrá que estar en concordancia con la sensibilidad de la cámara.

### 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**

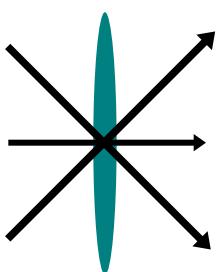
Cualquier rayo incidente paralelo al eje principal en la zona objeto sale pasando por el foco principal en la zona imagen



**Figura. 2.7.:** Regla 1

■ **Regla 2**

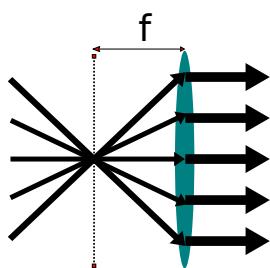
Cualquier rayo incidente que pasa por el centro de la lente sale con la misma dirección, es decir, no sufre desviación



**Figura. 2.8.:** Regla 2

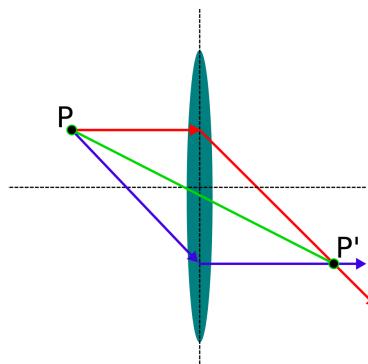
■ **Regla 3**

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. 2.9.:** 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  $P$ .

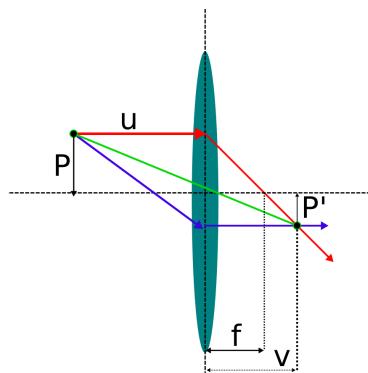


**Figura. 2.10.:** Obtención geométrica de la imagen

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.



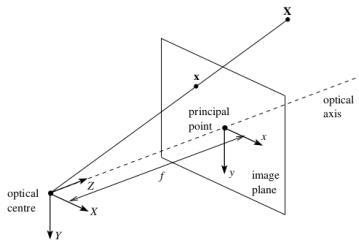
**Figura. 2.11.:** Modelo de lente fina

## 2.1.8 Modelo de la cámara

El modelo básico de la cámara también llamado ‘pin-hole’ representa la transformación de las coordenadas de los puntos de la escena en las coordenadas

de la imagen. [Ric08]

La siguiente imagen ilustra la proyección del punto  $X$  de  $\mathbb{R}^3$  al punto  $x$  de  $\mathbb{R}^2$



**Figura. 2.12.:** Modelo pinhole tomado de [Tay06]

Por triángulo semejantes podemos obtener las ecuaciones que relacionan un punto en el espacio con su proyección en la imagen.

$$\mathbf{x} = \frac{f}{Z} \mathbf{X} \quad (2.2)$$

$$\mathbf{y} = \frac{f}{Z} \mathbf{Y} \quad (2.3)$$

Queda aquí clara la perdida de una dimensión que se explicó en la introducción, ya que todos los puntos cuya relación entre las coordenadas  $X$  y  $Z$  sea constante darán la misma coordenada  $x$  en la imagen (igual con  $Y$ ,  $Z$  e  $y$ ).[Jos05]

Despejando de las ecuaciones anteriores obtenemos la distancia focal como

$$f = Z \frac{\mathbf{X}}{\mathbf{X}} = Z M \quad (2.4)$$

De donde  $M$  es el factor de magnificación.

En la visión artificial es común el uso del modelo de pin-hole mostrado en la figura 2.12. Los rayos convergen en el origen del plano de la cámara y una imagen invertida es proyectada dentro del plano como se describió en las ecuaciones 2.2 y 2.3.

Podemos escribir el punto del plano de la imagen en forma homogénea  $\tilde{\mathbf{p}} = (x', y', z')$  donde

$$x' = \frac{f X}{z'}, y' = \frac{f Y}{z'}, z' = Z \quad (2.5)$$

La ecuación 2.5 puede ser vista también como una matriz

$$\tilde{\mathbf{p}} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (2.6)$$

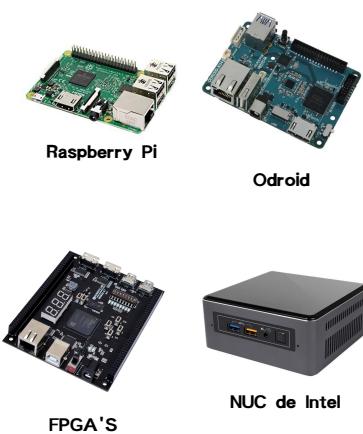
Dicha matriz es de utilidad cuando se la lente presenta un tipo de distorsión y hay que corregirlo, en los proximos capítulos se hablara acerca del metodo de calibracion de camara.

## 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.[@MUN] 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 HardKernel, 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. [Jos18]

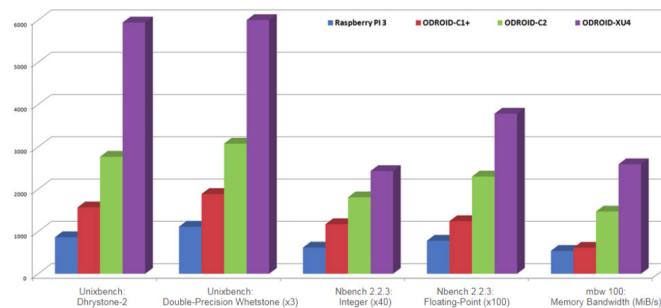
ODROID significa Open + Android. Es una plataforma de desarrollo tanto de hardware como de software.

### 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 súper 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.[@Har20]



**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.

### 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.[@Ubu14]

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. [@ROS]

#### 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.

## 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.<sup>a1</sup> El principio para aquellos que han usado principalmente entornos de desarrollo gráfico en Windows o Mac OS X.[Qui15]

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.

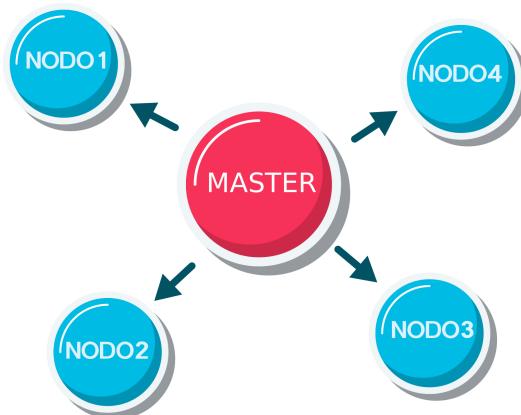
Los conceptos básicos del gráfico de cómputo de ROS son nodos, master, servidor de parámetros, mensajes, servicios, topics y bags, los cuales proporcionan datos al gráfico de diferentes maneras siguiendo la filosofía antes descrita.

### ■ 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

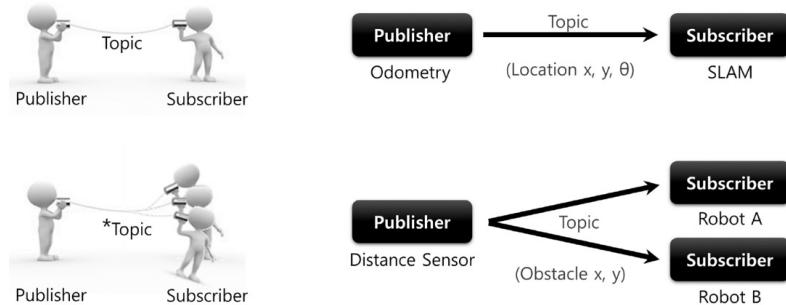
Un programa intermedio que conecta nodos.



**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

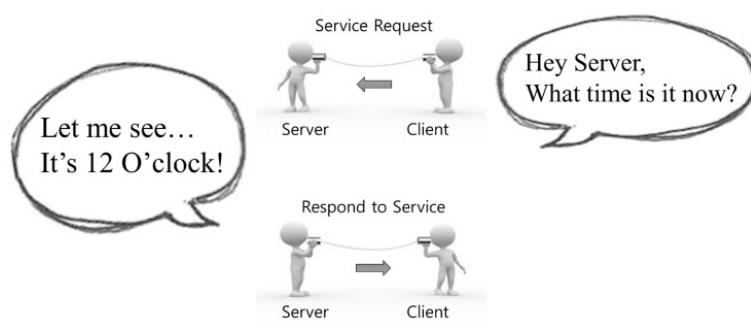
Los mensajes básicamente están pasando por el topic. Hay mensajes existentes basados en tipos de datos predefinidos, y los usuarios pueden escribir sus propios mensajes.

## ■ 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.[Jos18]



**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.[Jäh97]

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.[Esc11] 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:

$$0 < f(x, y) < \infty \quad (2.7)$$

La función  $f(x,y)$  se caracteriza por dos componentes: Estos componentes son llamados iluminación y reflexión.[Jos05]

- **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  $f(x, y)$

$$f(x, y) = i(x, y)r(x, y) \quad (2.8)$$

donde

$$0 < i(x, y) < \infty \quad (2.9)$$

y

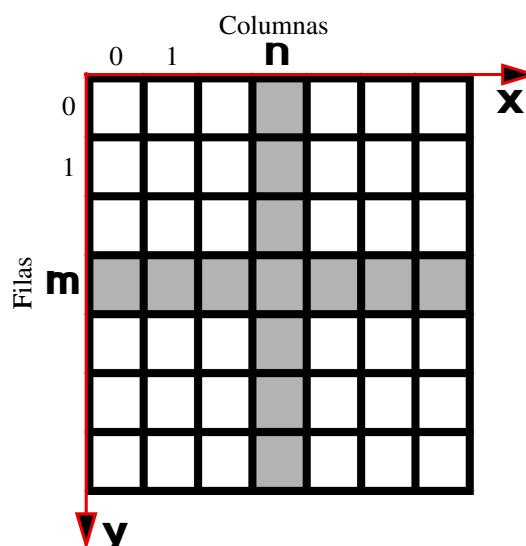
$$0 < r(x, y) < 1 \quad (2.10)$$

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.[Jos05] 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,  $x = (x, y) \in \mathbb{R}^2$ [Sze11]

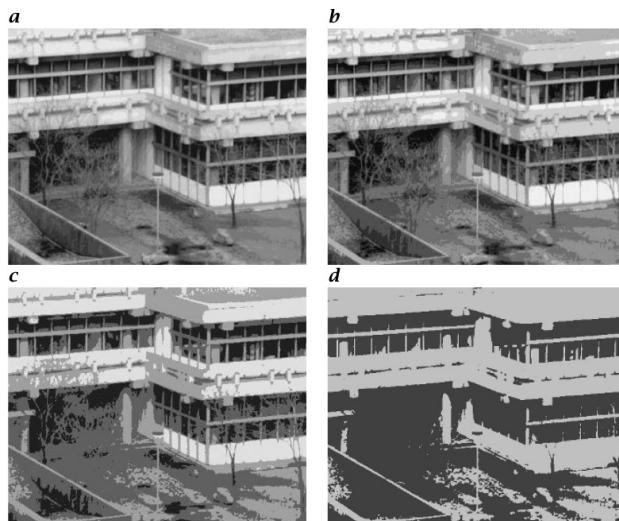
$$x = \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.11)$$

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.[Sze11]

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[Jäh97]. 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.[@Opea] Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS.

## 2.2.7 Calibración de cámara

La calibración de cámaras es un paso esencial en la obtención de buenos resultados en los algoritmos de visión artificial, dado su importancia es que hay múltiples algoritmos que satisfacen la necesidad de tener una imagen clara y calibrada.

Las cámaras han existido por mucho, mucho tiempo. Sin embargo, con la introducción de las cámaras baratas de modelo pinhole a fines del siglo XX, se convirtieron en una ocurrencia común en nuestra vida cotidiana. Desafortunadamente, este ahorro económico viene con su precio: distorsión significativa. Afortunadamente, estas son constantes y con una calibración y algunas reasignaciones podemos corregir esto. Además, con la calibración también puede

determinar la relación entre las unidades naturales de la cámara (píxeles) y las unidades del mundo real (por ejemplo, milímetros).[@Opec]

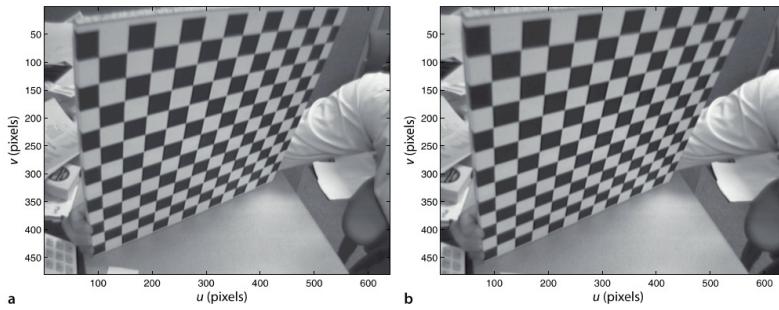
En este proyecto abordaremos el método de calibración hecho por Zhang (Zhang, 1998, 2000) que básicamente consiste en utilizar una plantilla 2D y que toma ventaja el hecho de tomar calibración basadas en las medidas de las coordenadas de la plantilla con las ventajas de la auto calibración en la cual no es necesaria utilizar plantilla.

La distorsión geométrica es generalmente el efecto más problemático que encontramos para las aplicaciones robóticas, y comprende dos componentes: radial y tangencial. La distorsión radial hace que los puntos de la imagen se traduzcan a lo largo de líneas radiales desde el punto principal. [Cor11]. El error radial puede aproximarse a un polinomio de la siguiente manera.

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \dots \quad (2.12)$$

Donde definimos a  $r$  como la distancia del punto de imagen desde el punto principal.

La distorsión ocurre cuando la ampliación disminuye con la distancia desde el punto principal, lo que hace que las líneas rectas cerca del borde de la imagen se curven hacia afuera.



**Figura. 2.20.:** Tomada de [Cor11]; a) Imagen con distorsión; b) Imagen despues de calibrar

Las coordenadas del punto  $(u,v)$  despues de la distorsion es dado por

$$u^d = u + \delta_u, v^d = v + \delta_v \quad (2.13)$$

Donde tenemos que

$$\begin{pmatrix} \delta_u \\ \delta_v \end{pmatrix} = \begin{pmatrix} u(k_1r^2 + k_2r^4 + k_3r^6 + \dots) \\ v(k_1r^2 + k_2r^4 + k_3r^6 + \dots) \end{pmatrix} + \begin{pmatrix} 2p_1uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2v^2) + 2p_1uv \end{pmatrix} \quad (2.14)$$

Donde el primer termino corresponde a la distorsión radial, mientras que el segundo corresponde a la distorsión tangencial.

## 2.2.8 Espacio de color

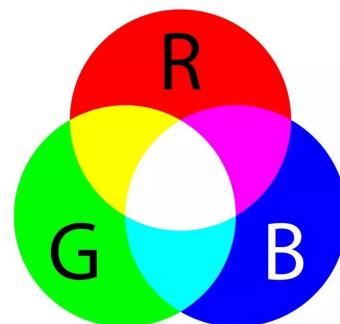
El uso del color en el procesamiento de imágenes está motivado por dos factores principales. Primero, el color es un poderoso descriptor que a menudo simplifica la identificación y extracción de objetos de una escena. En segundo lugar, los humanos pueden discernir miles de tonos e intensidades de color, en comparación con solo los tonos de gris de una cámara.[Raf02]

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).

$$\frac{dR}{R} = \frac{dG}{G} = \frac{dB}{B} = \text{constante} \quad (2.15)$$

donde dR es el incremento / decremento en magnitud del estímulo descrito por el proceso fundamental R, y comparable para los otros dos estímulos.[Kue03]



**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 implementadas en estas. Estos dos inconvenientes se resuelven con el siguiente espacio de color.

## 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.[Kim00]

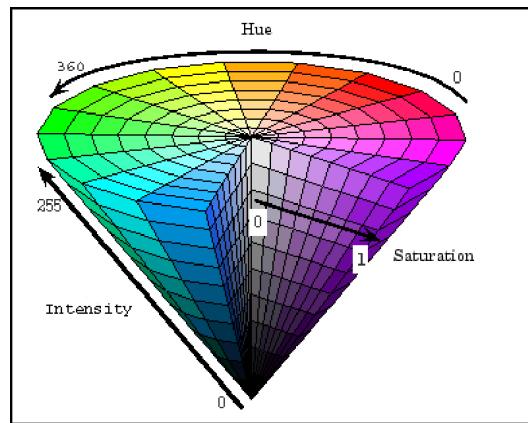
Las transformaciones matemáticas que permiten el paso del espacio RGB al HSI son realizadas normalmente mediante hardware específico. Las ecuaciones son:

$$I = \frac{R + G + B}{3} \quad (2.16)$$

$$H = \arctan \left( \frac{\sqrt{3}(G - B)}{(R - G) + (R - B)} \right) \quad (2.17)$$

$$S = 1 - \frac{\min(R, G, B)}{I} \quad (2.18)$$

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)

### 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.

## 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.

$$V \leftarrow \max(R, G, B) \quad (2.19)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R, G, B)}{V}, & \text{if } V = R \\ 0, & \text{otherwise} \end{cases} \quad (2.20)$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)), & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)), & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)), & \text{if } V = B \end{cases} \quad (2.21)$$

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

[@Opeb]

## 2.2.9 Transformaciones morfológicas

Lo que conocemos como realidad proviene de las percepciones obtenidas por los sentidos, por ejemplo el sentido de la vista nos puede entregar un objeto

casi que borroso pero nuestra memoria y capacidad de asociamiento nos permite relacionarlo con algún objeto guardado en nuestra base de datos. Hacer lo anterior en un sistema embebido involucra tener una base de datos amplio, lo que genera un costo razonable de recursos de memoria y procesamiento, debido a esto es que existen diversas maneras de "limpiar" lo que nuestros sentidos captan, que en este caso es lo que capta la cámara, tengan el menor ruido posible.

Es aquí donde las transformaciones morfológicas toman lugar, debido a su uso en imágenes binarias que permiten simplificar el uso de imágenes digitales. Estas basan su funcionamiento en una rama de las matemáticas conocidas como "Teoría de conjuntos". Se puede consultar más información acerca de la matemática detrás de las operaciones morfológicas y teoría de conjuntos en la obra de [Ser84]

## Conceptos elementales de teoría de conjuntos

Algunas definiciones básicas son descritas, tomando en cuenta que la notación para estos son mayúsculas ( $X, Y, Z \dots$ ), mientras que para los elementos son las minúsculas ( $q, r, t \dots$ ).

Las siguientes definiciones fueron tomadas de [Ort02]

**Definición 2.2.1. Inclusión**  $X$  es subconjunto de  $Y$  si todos los elementos de  $X$  pertenecen a  $Y$ :

$$X \subseteq Y \iff \{p \in X \rightarrow p \in Y\} \quad (2.22)$$

La inclusión es reflexiva, antisimétrica y transitiva.

**Definición 2.2.2. Complemento** El complemento de  $X$ ,  $X^c$ , son todos los elementos que no pertenecen a  $X$ .

**Definición 2.2.3. Unión** La unión de dos conjuntos se constituye por los elementos que pertenecen a uno o al otro:

$$X \cup Y = \{p | p \in X \text{ o } p \in Y\} \quad (2.23)$$

Al igual que la intersección, la unión de conjuntos es commutativa, asociativa e idempotente

**Definición 2.2.4. Intersección** La intersección de dos conjuntos  $X$  e  $Y$  es el conjunto de los elementos que pertenecen a ambos conjuntos:

$$X \cap Y = \{p | p \in X \text{ y } p \in Y\} \quad (2.24)$$

La intersección es conmutativa, asociativa e idempotente. Esta última propiedad es importante en morfología y significa que  $X \cap X = X$ .

**Definición 2.2.5. Translación** Un conjunto  $X$  es trasladado por un vector  $v$ , cuando cada uno de los elementos de ese conjunto sufre esa translación. Se denominará al nuevo conjunto  $X_v$ .

### Transformaciones morfológicas elementales

Las transformaciones morfológicas son algunas operaciones simples basadas en la forma de la imagen. Normalmente se realiza en imágenes binarias. Necesita dos entradas, una es nuestra imagen original, la segunda se llama elemento de estructuración o kernel que decide la naturaleza de la operación. Dos operadores morfológicos básicos son la erosión y la dilatación. Luego, sus variantes de formas como Apertura, Cierre, Gradiente, etc. también entran en juego. [@UCS19]

Para el presente trabajo, todas las entradas son imágenes binarias, esto es, blanco y negro provenientes de la salida del proceso del cambio de espacio de color a HSV, donde el color blanco representa el segmento del objeto a seguir. Además de que las operaciones morfológicas serán aplicadas como filtros de ruido en la imagen, debido a eso solo serán necesarias dos operaciones: apertura y cierre, que están basadas en erosión y dilatación.

La esencia de las transformaciones morfológicas es la obtención de estructuras geométricas en los conjuntos donde se está procesando, mediante el empleo de otro conjunto de forma conocida, a este último se le denomina como elemento estructurante. El tamaño y la forma del elemento estructurante se selecciona de acuerdo a las formas que el usuario requiera obtener.



**Figura. 2.23.:** Ejemplo de formas básicas de elementos estructurantes planos

### Erosión

La transformación de erosión es el resultado de comprobar si el elemento estructurante  $Y$  está totalmente incluido dentro del conjunto  $X$ . Cuando esto no ocurre, el resultado de la erosión es el conjunto vacío. [Ort02].

La erosión de un conjunto  $X$  por un elemento estructurante  $Y$  se define

como el conjunto de puntos o elementos  $x$ , pertenecientes a  $X$ , de forma que cuando el elemento estructurante  $Y$  se traslada a ese punto, el elemento queda incluido en  $X$ . [Ort02].

Entonces podemos definir matemáticamente a la erosión de la siguiente manera:

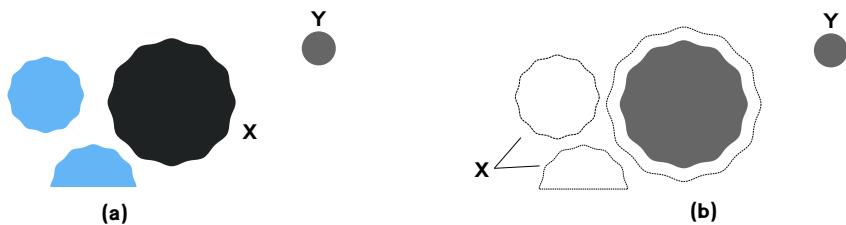
$$X \Theta \tilde{B} = \{x | B_x \subset X\} \quad (2.25)$$

de donde

- $\Theta$  = Resta de Minkowski
- $\tilde{B}$  = Elemento estructural simétrico

Que en palabras del autor de la obra [Esc11] la erosión es la degradación progresiva de uno de los campos (0 ó 1). Un elemento del campo a degradar seguirá perteneciendo al mismo si está rodeado de elementos iguales a él. En caso contrario, pasará al otro campo. En un proceso iterativo terminaría por destruir la imagen.

De la ecuación 2.17 podemos decir entonces que la erosión es más pronunciada con base en el tamaño del elemento estructural.



**Figura. 2.24.:** (a) Erosión de  $X$  por el elemento estructurante  $Y$ . (b) Los elementos conectados del conjunto  $X$  más pequeños que  $Y$  son eliminados.

De la imagen anterior se puede observar como en (b) el proceso de erosión disminuye la dimensión de las figuras, y en algunos casos elimina la totalidad de estos.

## Dilatación

La transformación de dilatación es el proceso inverso.<sup>a1</sup> de la erosión, en otras palabras, las operaciones de erosión y dilatación son duales una de otra.

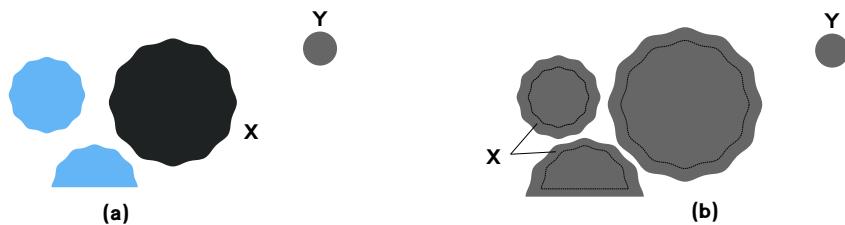
Es el crecimiento progresivo de uno de los campos (0 ó 1). Un elemento del campo contrario a crecer será convertido si posee algún vecino perteneciente al campo que se expande. En caso contrario, permanecerá igual. Los elementos pertenecientes al campo a expandir evidentemente no se

modifican.[Jos05]

Tomando en cuenta la dualidad antes mencionada, podemos obtener matemáticamente la dilatación de la siguiente manera:

$$X \otimes \tilde{B} = (X^c \otimes \tilde{B})^c \quad (2.26)$$

La siguiente figura ilustra el resultado de aplicar la operación de dilatación de donde el elemnto estructurante Y tiene forma de disco.



**Figura. 2.25.:** (a) Dilatación de X por el elemnto estructurante Y. (b) El conjunto X aumenta su definición.

Como se puede observar en (b) el elemnto estructurante Y aumenta la definición del objeto X.

## Apertura y Cierre

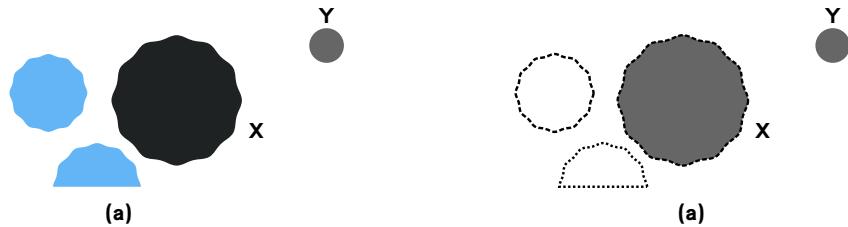
En términos de morfología básica, hay dos operaciones fundamentales que funcionan como filtros en imágenes digitales, estos son conocidos como apertura y cierre. Una vez definido la dilatación como  $X \rightarrow \delta(X)$  y la erosión como  $X \rightarrow \epsilon(X)$  y conociendo el principio básico de su funcionamiento podemos darnos cuenta que no hay manera de determinar el origen de X desde las imágenes  $\delta(x)$  o  $\epsilon(X)$ , es decir, que no admiten inversa, pero es posible aproximarse a la forma genuina de X mediante la aplicación de la operación de erosión seguida de la dilatación, al proceso anterior se le conoce como apertura. Mientras que si efectuamos una dilatación seguida de una erosión se conoce como cierre.

### Apertura

La apertura (opening en la literatura anglosajona) se definirá entonces como una combinación de erosiones y dilataciones en las que lo primero es un

erosión y la última una dilatación, siempre con el mismo elemento estructural:[Jos05]

$$X \circ B = (X \Theta \tilde{B}) \quad (2.27)$$

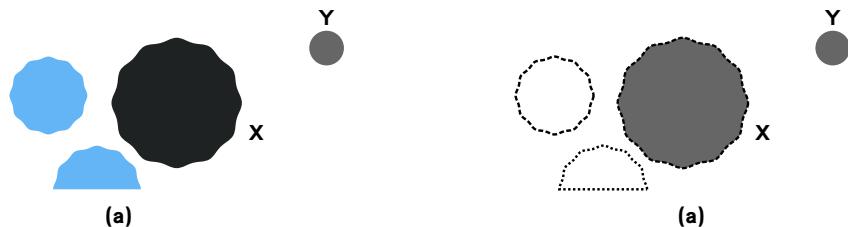


**Figura. 2.26.: (a)** Apertura morfológica del conjunto X por el elemento estructurante Y. **(b)** Eliminación de objetos menores en tamaño al elemento estructurante. La apertura redondea las convexidades importantes.

### Cierre

La dualidad de la operación de apertura es la del cierre(closing en literatura anglosajona), así que podemos definirlo matemáticamente así:

$$X \bullet B = (X \odot \tilde{B}) \Theta B \quad (2.28)$$



**Figura. 2.27.: (a)** Apertura morfológica del conjunto X por el elemento estructurante Y. **(b)** El cierre redondea las concavidades importantes.

## 2.2.10 Contraste y brillo

Hablar de contraste y brillo es hablar de transformación de pixeles que a su vez hace referencia a otro termino que aún no se ha abordado en este proyecto, que es el de operador de procesamiento de imagen. En la obra [Sze11] dicho operador es definido como:

$$g(x) = h(f(x)) \quad o \quad g(x) = h(f_0(x) \dots f_n(x)) \quad (2.29)$$

Donde  $g(x)$  y  $h(x)$  son definidas como imagen.

Es decir que un operador es una imagen de salida tomando una o varias

imágenes de entrada, que para un sistema discreto la ecuación anterior se obtiene:

$$g(i, j) = h(f(i, j)) \quad (2.30)$$

De donde  $i, j$  se acotan en las especificaciones del sensor de la cámara.

Dos procesos puntuales de uso común son la multiplicación y la suma con una constante,

$$g(x) = \alpha f(x) + \beta \quad (2.31)$$

- El parámetro  $\alpha > 0$  es conocido como 'Gain' y es el encargado de controlar el nivel de contraste.
- El parámetro  $\beta$  es conocido como 'Bias' y se encarga de controlar el brillo.

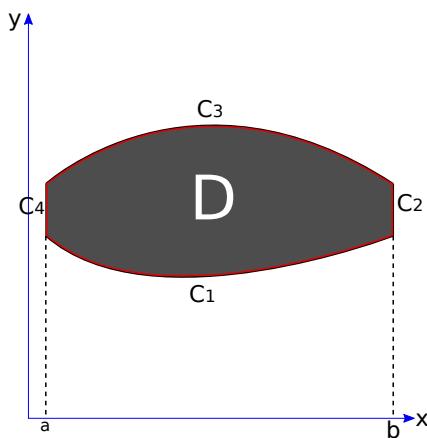
Y de igual manera que en la ecuación 2.22 podemos acotar la ecuación 2.23 de la siguiente manera:

$$g(i, j) = \alpha * f(i, j) + \beta \quad (2.32)$$

Donde  $i, j$  nos indican la ubicación de los pixeles.

## 2.2.11 Teorema de Green

En el campo de las matemáticas hay un teorema que nos da la relación entre una integral de línea, es decir donde la función a integrar se evalúa a lo largo de una curva,  $C$  y una integral doble sobre una región plana  $D$  con límites  $C$ , tal como se ilustra en la siguiente figura.



**Figura. 2.28.:** Área de  $D$  limitada por  $C$

Dicho teorema es ampliamente utilizado para la detección de bordes. Si lo aterrizamos en el campo de la visión artificial, y del trabajo de [YAN94] podemos obtener que el momento de orden ( $p + q$ ) de una imagen bidimensional  $g(x, y)$  se define como

$$m_{pq} = \int_y \int_x g(x, y) x^p y^q dx dy \quad (2.33)$$

Que para imágenes digitales tenemos entonces

$$m_{pq} = \sum_y \sum_x g(x, y) x^p y^q \quad (2.34)$$

Que de lo anterior y suponiendo que tenemos imágenes binarias, osea que 1 para objetos y 0 para el fondo la ecuación queda así

$$m_{pq} = \sum_{(x,y) \in R} x^p y^q \quad (2.35)$$

De donde  $R$  es la región del objeto.

**Definición 2.2.6. Momento** El momento  $M_{pq}^{(f)}$  de una imagen  $f(x, y)$ , donde  $p, q$  son enteros positivos y  $r = p+q$  es el orden del momento.

$$M_{pq}^{(f)} = \int_D \int p_{pq}(x, y) f(x, y) dx dy \quad (2.36)$$

donde  $p_{00}(x, y), p_{10}(x, y), \dots, p_{kj}(x, y)$  son funciones de base polinomiales definidas en  $D$ .

## Momentos

La opción más común es una base de potencia estándar [Flu09]  $p_{kj}(x, y) = x^k y^j$  eso lleva a momentos geométricos. Que para imágenes binarias obtenemos lo siguiente

- $m_{00}$  = Es la 'masa' de la imagen, para imágenes binarias es el área del objeto.
- $m_{10}/m_{00}$  y  $m_{01}/m_{00}$  = define el centro de gravedad, o en otras palabras el centroide de la imagen.
- $m_{20}$  y  $m_{02}$  = Describen la 'distribución de masas' de la imagen con respecto a los ejes x, y



# Modelo matemático

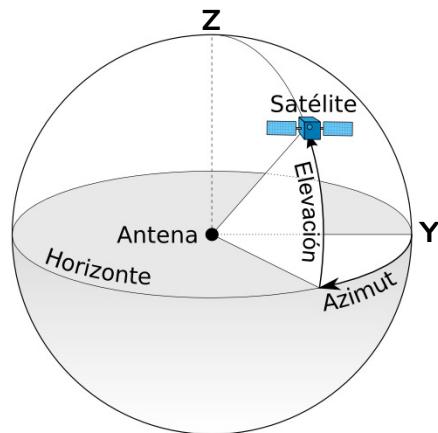
“ In mathematics, you don’t understand things. You just get used to them.

— John von Neumann  
(mathematician, physicist, computer scientist, and polymath.)

En el presente capítulo se aborda el modelo matemático para el seguimiento de objetivos de una gimbal embebida en un UAV.

## 3.1 Marco de referencia

Los movimientos de la gimbal se basan en las coordenadas horizontales de azimuth y elevación, para gimbal de 3 grados de libertad se agrega un tercer eje cuyo movimiento se conoce como roll. El eje de azimuth y la elevación se visualizan más fácilmente al pensar en la posición de un objeto en relación con el horizonte.



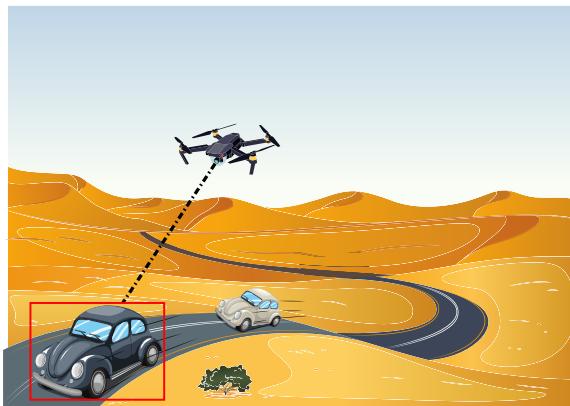
**Figura. 3.1.:** Angulo de azimuth

El angulo de azimuth es la posición alrededor del horizonte, medida desde un punto de referencia como el norte verdadero o el sur verdadero. Los movimientos de azimuth ocurren alrededor del eje Z (vertical).

La elevación es la distancia del objeto por encima o por debajo del horizonte (también conocida como altitud en aplicaciones de astronomía y aeroespaciales). Los movimientos de elevación ocurren alrededor del eje Y.

Los movimientos de balanceo ocurren alrededor del eje X a medida que gira con los ejes Y y Z.

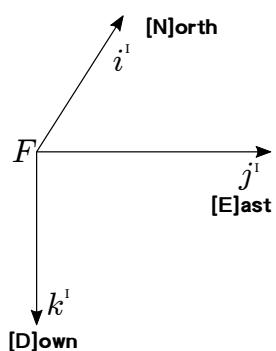
El control que se hace en la gimbal está dado por un motor con control de posición y una IMU (sensor inercial). Dicha gimbal está montada en la parte delantera del vehículo aéreo como se muestra en la siguiente figura.



**Figura. 3.2.:** Seguimiento de un objetivo utilizando una cámara sujetada a una gimbal embebida en un vehículo aéreo no tripulado.

### 3.1.1 Marco de referencia Inercial [I]

Antes de abordar el control de la gimbal, es necesario y obvio conocer el comportamiento de dicho sistema, empezaremos por definir el marco de referencia inercial, es decir el de la tierra. La siguiente imagen ilustra los ejes de este marco.

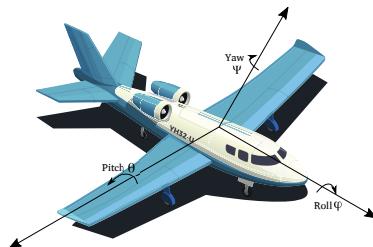


**Figura. 3.3.:** Marco de referencia inercial

Donde definimos a k apuntando hacia el centro de la tierra, este sistema de coordenadas refieren a (Norte, Este y Abajo) de ahí su nombre NED(siglas en inglés).

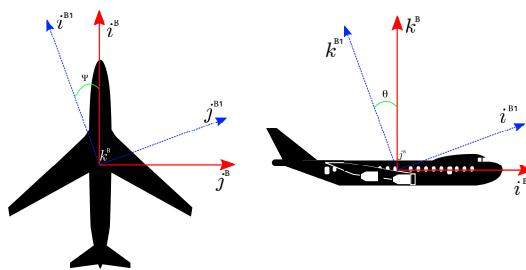
### 3.1.2 Marco de referencia del cuerpo [B]

Una aeronave tiene la libertad de rotar en 3 ejes, en la aeronáutica son conocidos como Yaw, Pitch y Roll. Dichas rotaciones son importantes en el control de aeronaves ya que sirven para conocer la dinámica del sistema, dicha rotaciones pueden ser mejor detalladas en una imagen, por lo que en la siguiente ilustración se grafica un plano y se asigna nombre a cada rotación de los ejes.



**Figura. 3.4.:** Rotación en 3 ejes.

Vamos a llamar marco de referencia del cuerpo a las coordenadas del UAV, es necesario establecer el centro del plano en el centro de masas de la aeronave, como se ilustra en la siguiente figura.



**Figura. 3.5.:** Marco de referencia del cuerpo.

De donde definimos las dos rotaciones que serán controladas en este trabajo pitch y yaw( $\theta, \psi$ ). El subíndice B hace referencia a Body y nos indica que estamos hablando de las coordenadas del UAV.

### 3.1.3 Marco de referencia de la gimbal [G]

Un último paso es definir un marco de referencia para la cámara, considerando que la cámara se encuentra en la parte delantera de la aeronave y que el centro de la gimbal es el origen.

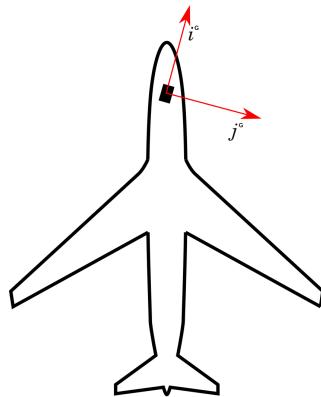


Figura. 3.6.: Marco de referencia del la cámara.

De donde el subíndice G hace referencia a que estamos en el plano de la gimbal. El vector unitario K apunta hacia afuera, es decir hacia nosotros.

## 3.2 Modelo matemático gimbal

Poniendo en el mismo plano dos referencias, el de la gimbal y el del body, tal como se ilustra en la siguiente figura

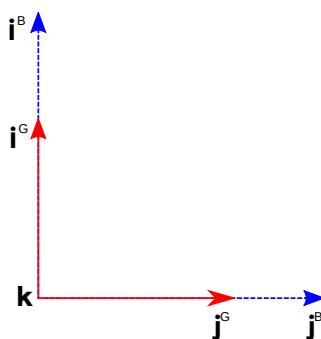


Figura. 3.7.: Marco de referencia Body y gimbal

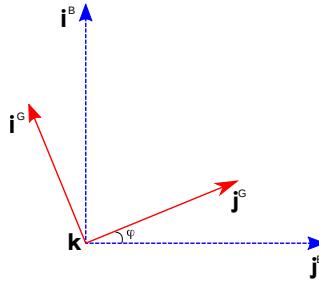
Cualquier punto en el plano puede ser expresado como:

$$p_{i^B, j^B} = [p_i^B, p_j^B]^T = p_i i^B + p_j j^B \quad (3.1)$$

$$p_{i^G, j^G} = [p_i^G, p_j^G]^T = p_i \cdot i^G + p_j \cdot j^G \quad (3.2)$$

$$\mathbf{P}_{i,j}^B = \mathbf{P}_{i,j}^G \quad (3.3)$$

Si hacemos rotar el frame de la gimbal respecto al eje K obtenemos que:



**Figura. 3.8.:** Rotación del marco de referencia Gimbal respecto al del cuerpo

$$p_{i^B, j^B} = [p_i^B, p_j^B]^T = p_i i^B + p_j j^B \quad (3.4)$$

$$p_{i^G, j^G} = [p_i^G, p_j^G]^T = p_i i^G + p_j j^G \quad (3.5)$$

Como

$$p_i = p_{i,j}^G \cdot i^B = p_i^G i^B \cdot i^G + p_j^G i^B \cdot j^G \quad (3.6)$$

$$p_j = p_{i,j}^G \cdot j^B = p_i^G j^B \cdot i^G + p_j^G j^B \cdot j^G \quad (3.7)$$

Entonces

$$\begin{bmatrix} p_i^B \\ p_j^B \end{bmatrix} = R \begin{bmatrix} p_i^G \\ p_j^G \end{bmatrix}$$

Donde la matriz de rotación etsa dada por:

$$R = \begin{bmatrix} i^B \cdot i^G & i^B \cdot j^G \\ j^B \cdot i^G & j^B \cdot j^G \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}$$



# Visión Artificial

“ Let's go invent tomorrow instead of worrying about what happened yesterday.

— Steve Jobs

(American business magnate, industrial designer, investor, and media proprietor.)

En este capítulo se abordarán temas relacionados con la obtención del centroide de una figura, pasando por la calibración de la cámara, el cambio del espacio de color y aplicación de filtros morfológicos. Además de presentar el algoritmo de visión y dando pruebas hechas con una cámara corriendo en el framework ROS.

## 4.1 Cámara

Como se abordó 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 utilizó fue la del fabricante HardKernel y que lleva de nombre Ocam.



**Figura. 4.1.:** Cámara 'Ocam' obtenida de [@Har]

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  $\mu\text{m}$  x 2738.4  $\mu\text{m}$ )
- **Tamaño del pixel:** 1.4  $\mu\text{m}$  x 1.4  $\mu\text{m}$ .
- **Interfaz:** USB 3.0 Super-Speed.  
1920 x 1080 a 30fps, 1280 x 720 a45fps, 640 x 480 a30fps
- **Frame rate:**

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

El proceso de la obtención del centroide de la figura requiere seguir una serie de pasos relacionados con el procesamiento de imágenes, podemos entonces definir esos pasos como un algoritmo, descrito en la siguiente lista.

---

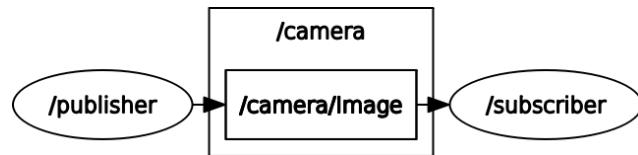
### **Algorithm 1** Obtener centroide de figura

---

- 1: Calibrar cámara
  - 2: Capturar frames a 60fps
  - 3: Publicar frames en ROS
  - 4: Corrección de brillo y saturación
  - 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 7 con función moments de openCV
  - 9: Publicar coordenadas del centroide
- 

## 4.3 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.



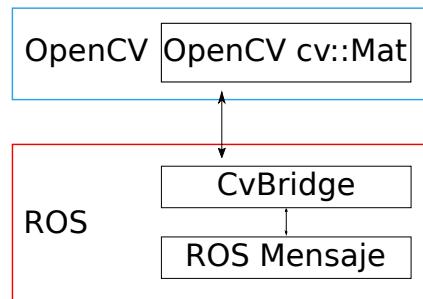
**Figura. 4.2.:** Nodos y topic

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.

### 4.3.1 Publisher

Como vimos anteriormente opencv es una librería open-source que se encarga del procesamiento de imágenes, 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 refiriéndonos 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

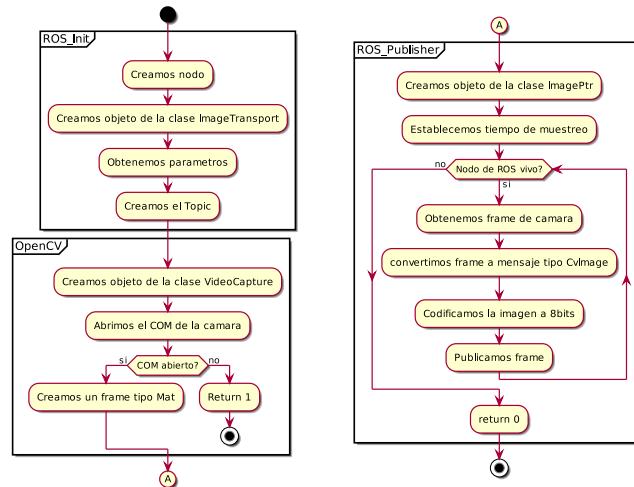
Al convertir un mensaje sensor\_msgs/Image en una imagen Cv, CvBridge reconoce dos casos de uso distintos:

- 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, 'código 1'

### 4.3.2 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 objetivo.

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

Dichos procesos están incluidos en una clase llamada 'objectTracking' que se ilustra en el siguiente diagrama.

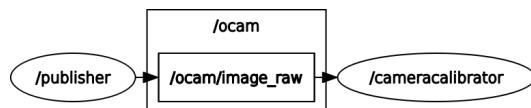


**Figura. 4.5.:** Diagrama de clases del subscriber

## 4.4 Calibración de camara

Siguiendo la calibración basada en el algoritmo de Zhangs, previamente descrito en el capítulo 2, comenzamos por posicionar la cámara en una mesa fija y colocando un tablero de ajedrez de dimensión 2.65 x 2.6cm cada cuadrito.

Lanzando los nodos de ROS y graficando en RQT



**Figura. 4.6.:** Comunicación ROS con calibrador

Para obtener una buena calibración, se movió el tablero de ajedrez en el marco de la cámara de la siguiente manera:

- Tablero de ajedrez a la izquierda, derecha, arriba y abajo del campo de visión de la cámara
  - Barra X - izquierda / derecha en el campo de visión
  - Barra Y - arriba / abajo en el campo de visión
  - Barra de tamaño: hacia / lejos e inclinación de la cámara
- Tablero de ajedrez que llena todo el campo de visión
- Tablero de ajedrez inclinado hacia la izquierda, derecha, arriba y abajo (sesgado)

Tal como se ilustra en la siguiente figura



**Figura. 4.7.:** Proceso de calibración utilizando un tablero de ajedrez

Despues del proceso de calibración los resultados son los siguientes: Matriz de distorsión  $D[]$

$$D = \begin{bmatrix} -0,439076 & 0,215870 & 0,001905 & 0,005790 & 0,000000 \end{bmatrix} \quad (4.1)$$

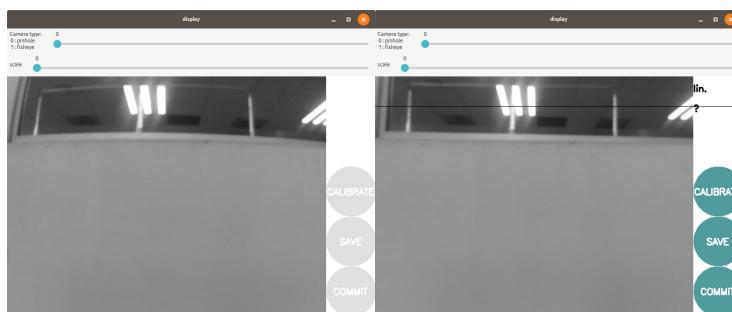
Matriz de parámetros intrínsecos de la cámara

$$K = \begin{bmatrix} 656,41089 & 0 & 320,06053 \\ 0. & 654,93593 & 222,93267 \\ 0. & 0 & 1 \end{bmatrix} \quad (4.2)$$

Matriz de proyección de cámara  $P$

$$P = \begin{bmatrix} 581,95032 & 0. & 324,43609 & 0. \\ 0. & 613,90649 & 221,23437 & 0. \\ 0. & 0. & 1. & 0. \end{bmatrix} \quad (4.3)$$

Una vez cambiando los parametros anteriores al publicador da como resultado la siguiente imagen

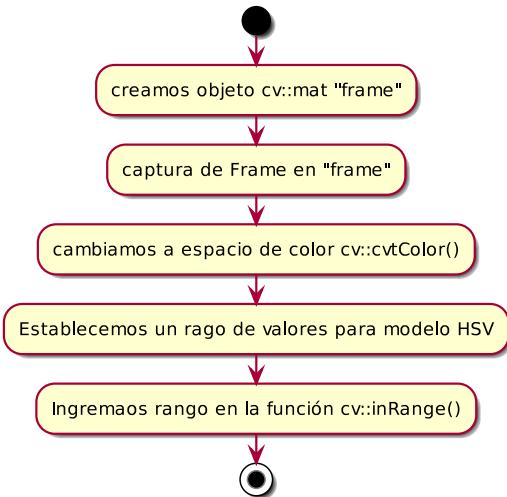


**Figura. 4.8.:** Proceso de calibración utilizando un tablero de ajedrez

## 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 diagrama muestra el flujo que el programa 2(ver apéndice A) siguió para el cambio de espacio de color.



**Figura. 4.9.:** 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.

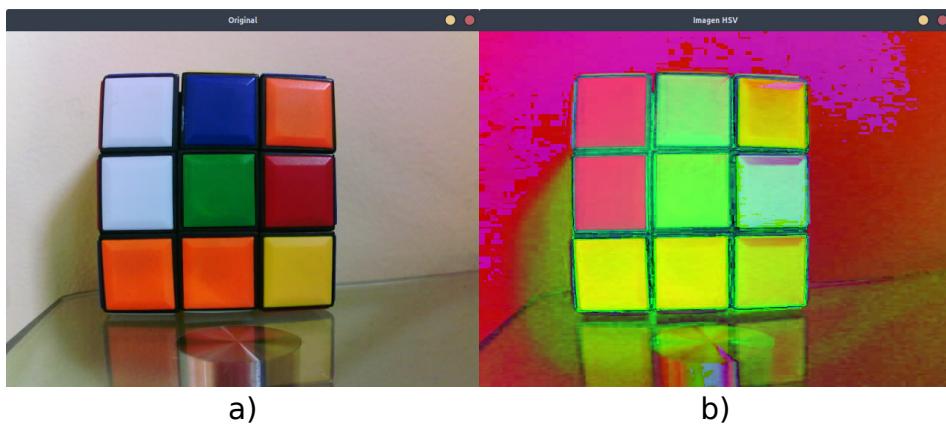
#### Función 4.5.1: cvtColor()

```
void cv::cvtColor(Mat &_src, Mat &dst ,int code, int dstCn )
```

Donde

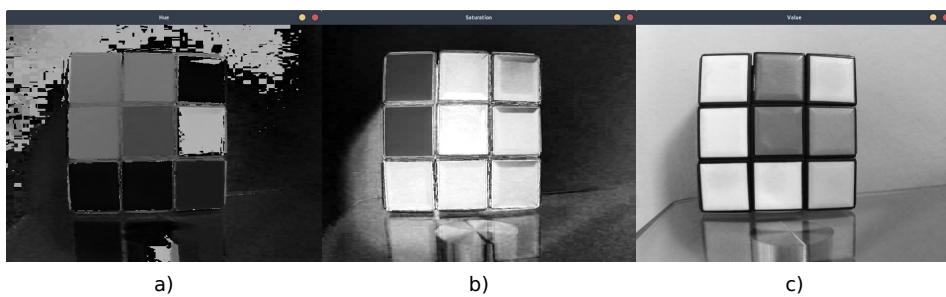
- **src:** Imagen de entrada de 8 bits sin signo
- **dst:** Imagen de salida del mismo tamaño y profundidad que src.
- **code:** Código de conversión de espacio de color en este caso es COLOR\_BGR2HSV.
- **dstCn:** Número de canales en la imagen de destino; si el parámetro es 0, el número de canales 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.



**Figura. 4.10.:** a)Imagen RGB; b) Imagen HSV

En la figura anterior podemos apreciar el cambio del espacio de color, en b) tenemos el resultado de aplicar la función cvtColor con el código COLOR\_BGR2HSV, que podemos dividir a su vez en 3 canales, H|Hue, S|Saturation y V|Value.



**Figura. 4.11.:** Espacio de color HSV, a)Hue; b)Saturation; c)Value

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

#### Función 4.5.2: inRange()

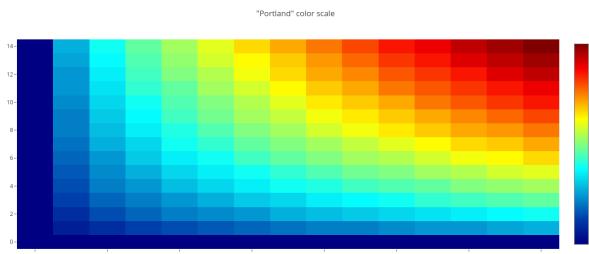
```
void cv::cvtColor(Mat &src, Mat &lowerb, Mat &upperb, Mat &dst )
```

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.

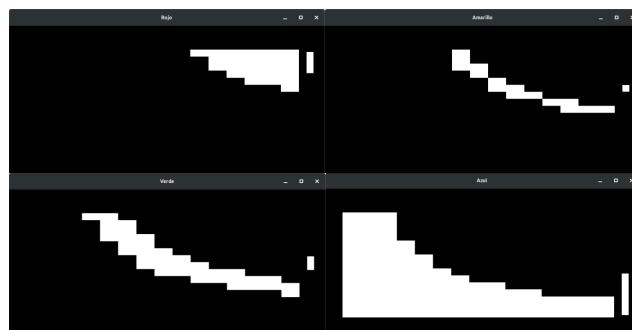
Debido a que no hay un solo color azul o rojo, etc, si no más bien un rango que cubre la gama de azules, amarillo, naranja, etc. Por esa razón se hicieron pruebas para determinar los valores HSV que corresponden a los siguientes colores: Amarillo, Azul, Rojo, Verde y Naranja y de esta manera a la hora 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. 4.12.:** Espectro de colores

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



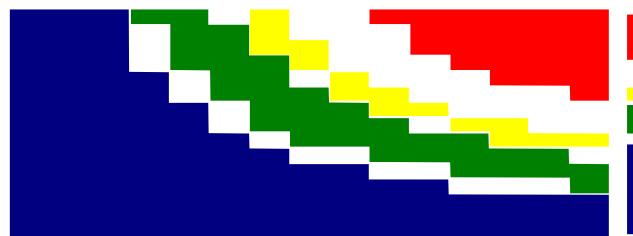
**Figura. 4.13.:** Separación de colores

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

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.

**Tab. 4.1.:** Valores para cada rango de color

	H <sub>min</sub>	H <sub>max</sub>	S <sub>min</sub>	S <sub>max</sub>	V <sub>min</sub>	V <sub>max</sub>
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

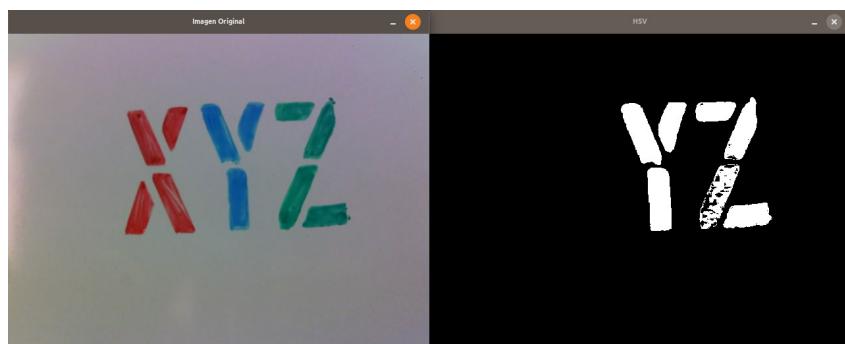


**Figura. 4.14.:** Separación de colores

#### 4.5.1 Pruebas de campo

Una vez obtenido los valores aproximados de 4 rango de color, lo siguiente fue probar dichos valores en una imagen captada por la cámara.

Las pruebas se hicieron con luz emitida por un foco, y con poca interacción con la energía solar, probando primero el rango de color azul.



**Figura. 4.15.:** Separación de color azul

Como se puede observar en la figura anterior gran parte del color verde esta encapsulada en lo que deberia ser únicamente color azul, esto debido aunque en las pruebas iniciales no se consideró la variación de iluminación en el entorno, por lo que reajustar los valores se hace una tarea necesaria. Aunque los primeros valores de la tabla 4.1 no son exactos, nos sirven para tener una referencia de donde empezar a trabajar.

Una vez corregido el resultado del color azul se muestra en la siguiente figura.



**Figura. 4.16.:** Separación de colore azul

Que al final tenemos la separación de los colores primarios como se muestra en la siguiente figura, donde se debe mencionar uno de los principales problemas que presenta este algoritmo.



**Figura. 4.17.:** Separación de colores RGB utilizando el modelo HSV

El problema es que debido a que no tenemos un sensor que mida la intensidad de luminosidad nuestro rango de los valores S y V van a cambiar dependiendo del entorno en el que este la cámara es decir, que si hay alguna sombra que haga pensar al algoritmo que el color es verde, cuando en realidad es azul. Para las pruebas anteriores se obtuvieron de resultado los siguiente valores.

**Tab. 4.2.:** Valores para cada rango de color en un ambiente iluminado por energía eléctrica

	H <sub>min</sub>	H <sub>max</sub>	S <sub>min</sub>	S <sub>max</sub>	V <sub>min</sub>	V <sub>max</sub>
Rojo	0	10	70	255	70	255
Amarillo	25	35	50	255	50	255
Verde	35	90	55	255	55	255
Azul	95	130	55	255	55	255

Otro problema que es palpable de ver es la captación de ruido, en las dos

figuras anteriores hay presencia de pequeño ruido tanto fuera como dentro de las letras. Esto se puede solucionar agregando un par de filtros básicos, los llamadas opening y closing.

## 4.6 Corrección de brillo

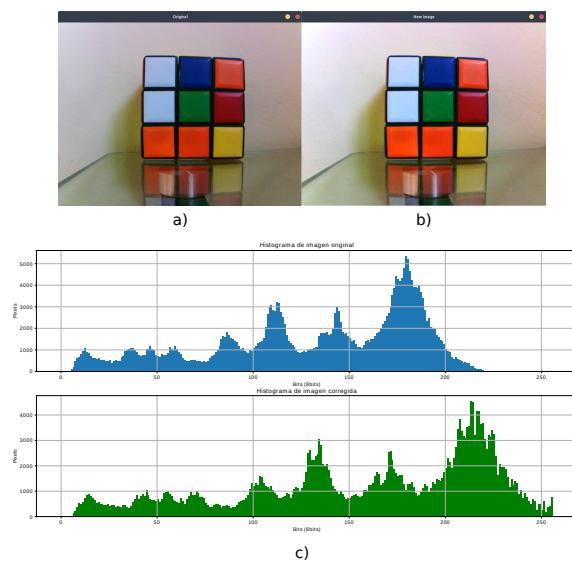
Como se pudo observar en pruebas anteriores el rango que se asigna a cada color en el modelo HSV está completamente ligado con la cantidad de iluminación de nuestra imagen, es por eso que predefinir un conjunto de valores se hace una tarea complicada, una solución sería poner 3 barras de interfaz para acomodar los valores de H S y V, lo cual sería una solución sencilla pero un poco tardada, ya que el sistema tendrá que responder en tiempo real a los cambios de los valores antes descritos.

Otra posible solución es corregir el brillo por medio de la librería de OpenCV, y de la misma manera poner una barra en la interfaz que ajuste el brillo de modo manual. Dicha solución parece ser la mejor opción para este proyecto debido a que solo necesitamos mover un rango de valor y dejar fijos los parámetros del espacio de color, esto, claro se tiene que hacer antes de transformar de RGB a HSV.

El ajuste de brillo y saturación se da a partir de la siguiente expresión

$$g(i, j) = \alpha * f(i, j) + \beta \quad (4.4)$$

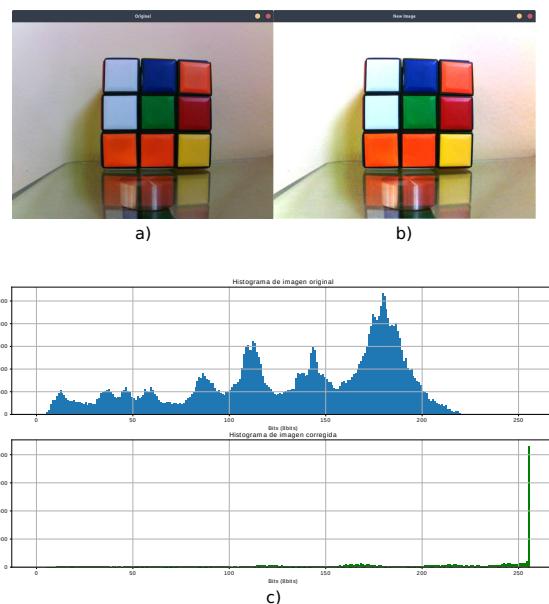
Donde  $\alpha$  es el encargado de ajustar el contraste, mientras que  $\beta$  se encarga del brillo. Los valores de  $\alpha$  van en el rango de 1 a 3, de tipo float y los de  $\beta$  van de 0 a 100 de tipo entero.



**Figura. 4.18.:** a) Imagen captada por la cámara; b) Corrección de contraste; c) Histograma de a y b

Al aplicar un valor de  $\alpha$  de 1.2 podemos observar que la imagen b) contrasta mejor los colores, esto puede ser observado mediante un histograma que muestra en Y las incidencias de píxeles y en el eje X muestran los valores de 0 a 256 bits correspondientes a una imagen en escala de grises, es decir, que son histogramas que muestran la cantidad de negro de un píxel, siendo el 0 el negro y el 256 el blanco.

El histograma de color verde está notablemente desplazado hacia la derecha, esto se debe a que la imagen ahora tiene más tonos grises que negros, por lo que tenemos una imagen más clara.

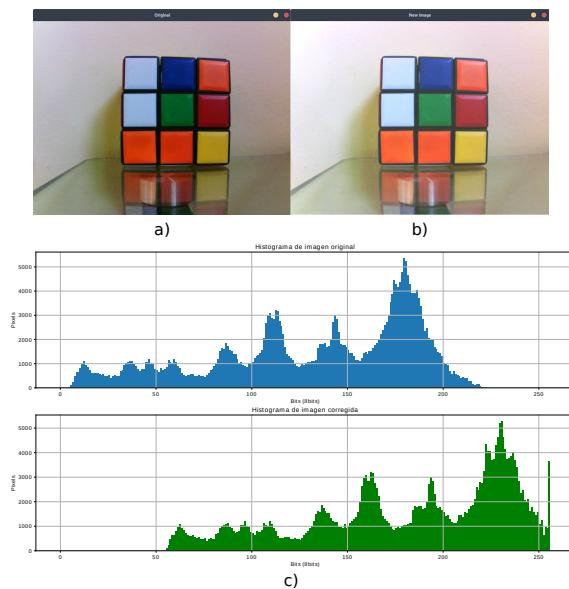


**Figura. 4.19.:** a) Imagen captada por la cámara; b) Corrección de contraste en 1.5; c) Histograma de a y b

Ahora vemos en b) una imagen más clara y a su vez vemos como la mayor parte de píxeles están llevados hacia la derecha del histograma, esto quiere decir que la intensidad ahora tiende a ser blanca.

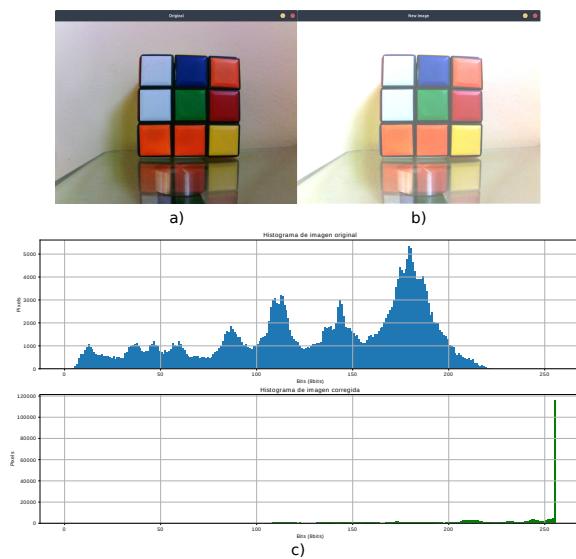
Esto es con respecto a  $\alpha$ , pero ahora veamos que pasa si modificamos el brillo. En la siguiente imagen mantenemos el valor de contraste en 0 pero el brillo lo modificamos a 50.

Como se puede observar la imagen esta más clara pero a su vez esta más opaca. Esto hace que los colores se vean un poco matizados, Y en el histograma vemos que la mayoría de los píxeles están sobre la izquierda.



**Figura. 4.20.:** a) Imagen captada por la cámara; b) Corrección de brillo en 50; c) Histograma de a y b

Finalmente probamos el valor del brillo al máximo, es decir, al 100 dando como resultado la siguiente imagen.



**Figura. 4.21.:** a) Imagen captada por la cámara; b) Corrección de brillo en 100; c) Histograma de a y b

Al igual que el histograma al modificar el contraste en 1.5 la gráfica c) muestra la mayor parte de los píxeles en blanco, debido a que ahora la imagen ya está muy clara que a su vez se ve con poco brillo.

Puede ocurrir que jugar con el valor de  $\beta$  mejore el brillo, pero al mismo tiempo la imagen aparecerá con un ligero velo a medida que se reducirá

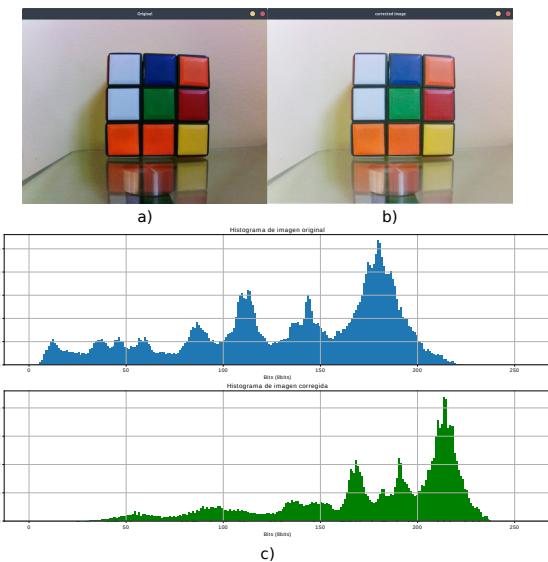
el contraste. La ganancia  $\alpha$  se puede usar para disminuir este efecto, pero debido a la saturación, perderemos algunos detalles en las regiones brillantes originales.

#### 4.6.1 Corrección de Gamma

Corrección de  $\gamma$  puede ser utilizado para modificar el brillo de una imagen utilizando una transformación no lineal entre valores de entrada y su salida mapeada.

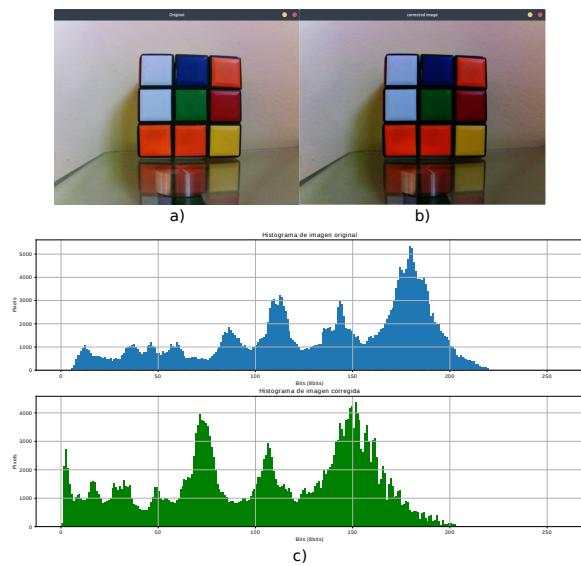
$$O = \left( \frac{I}{255} \right)^\gamma * 255 \quad (4.5)$$

Cuando  $\gamma < 1$ , las regiones oscuras originales serán más brillantes y el histograma se desplazará hacia la derecha, mientras que será lo opuesto con  $\gamma > 1$ .



**Figura. 4.22.:** a) Imagen captada por la cámara; b) Corrección de gamma  $< 1$ ; c) Histograma de a y b

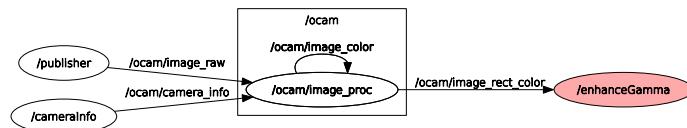
La figura anterior ilustra una modificación de gamma a 0.5, es decir, que hace que la imagen se vea con más brillo de lo que originalmente es, esto nos permite obtener mejores resultados al momento de aplicar técnicas de identificación de color cuando el entorno de la imagen es un tanto oscuro. Por otro lado tenemos cuando gamma es mayor a 1, que en la figura 4.23 el valor de  $\gamma$  es de 1.5



**Figura. 4.23.:** a) Imagen captada por la cámara; b) Corrección de gamma  $> 1$ ; c) Histograma de a y b

De esta manera el valor de gamma va a variar dependiendo del entorno en el que se encuentre la cámara, una manera de no hacerlo manual sería con la implementación de un sensor de luminosidad pero eso no entra en los límites de este proyecto.

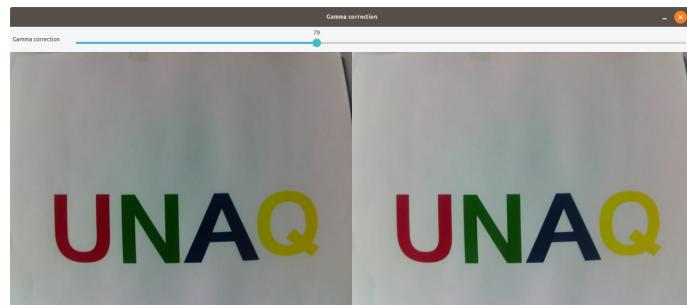
Que visto desde los nodos de ROS quedaría así:



**Figura. 4.24.:** Comunicación de nodos

El nodo en rojo es el correspondiente a la etapa de corrección, que se agrega a los nodos publicador y calibrador.

Como prueba se utilizó un conjunto de letras de diferentes colores, impresas en papel blanco



**Figura. 4.25.:** corrección de gamma, la imagen de la derecha es la corregida

De la imagen anterior podemos observar dos cosas importantes, la primera es la barra de valores que interactúan con el usuario, que va de 0 a 200 y es la correspondiente al valor de gamma. Y en segundo claramente se percibe un realce de colores y como es que el fondo se ve más claro, que esto ayuda de alguna manera a limpiar la imagen al no agregar ruido innecesario.

## 4.7 Filtro morfológico

Como vimos en la sección de HSV el entorno puede llegar a introducir a nuestra imagen un poco de ruido por lo que es necesario aplicar técnicas de filtrado.

Las librerías de opencv proveen operadores morfológicos que nos ayudan a limpiar nuestra imagen. Dicho filtros son los que abordamos en el capítulo 1, opening y closing que nos ayudaran a limpiar el objetivo por dentro y por fuera.

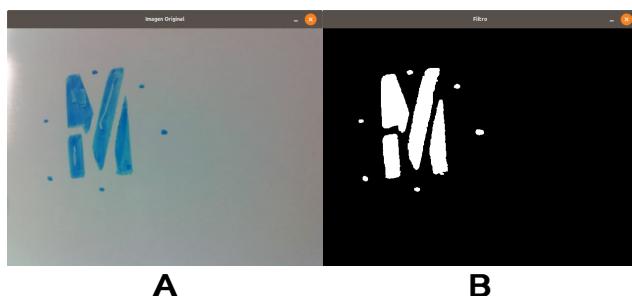
### Función 4.7.1: Operación Morfológica

```
void morphologyOperation(Mat &_Image, Mat &Image_, int oper, int size, int elemen)
```

De la función anterior tenemos que:

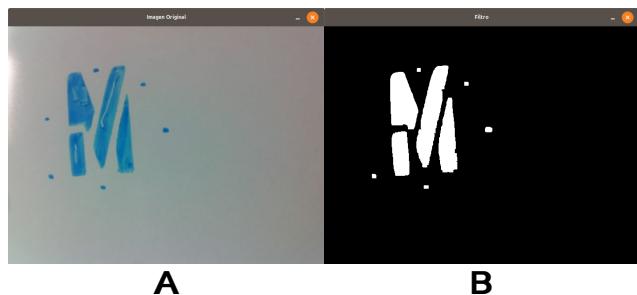
- **\_Image** es la imagen de entrada.
- **Image\_** es la imagen de salida
- **Oper** indica la operación Morfológica, 0 para opening y 1 para closing.
- **size** El tamaño del kernel.
- **elemn** La forma del kernel; 0 → Rectangular; 1 → Elipse y 2 → Cruz.

Los últimos dos parámetros son lo que se modificaron en las siguientes pruebas.



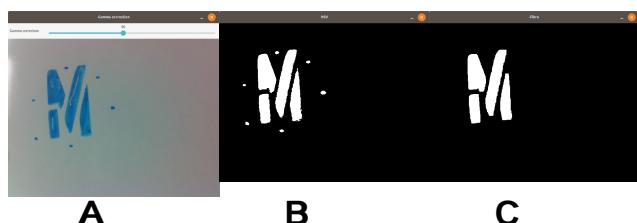
**Figura. 4.26.: A) Imagen original, B)Aplicación de filtros**

En la imagen anterior la aplicación del filtro se hizo con un kernel para el opening cuadrado y con un tamaño de 1. Se puede observar que no hay un cambio significativo por lo que se decidió cambiar a un kernel cuadrado de tamaño 3, la elección del kernel cuadrado es porque el objetivo tiene bordes de aproximadamente 90 grados.



**Figura. 4.27.: A) Imagen original, B)Aplicación de filtros. Con size de 3**

Hubo ligeros cambios, solo desaparecieron algunos puntos que son ruido por lo que es conveniente aumentar el parámetro **size**.



**Figura. 4.28.: A) Imagen original,B) Espacio de color HSV C)Aplicación de filtros**

Con un size de 5 para el kernel del filtro opening se obtuvo un resultado que considero aceptable porque limpió el ruido externo, es decir, desapareció los puntos que no están asociados con nuestro objeto deseado.

## 4.8 Centroide

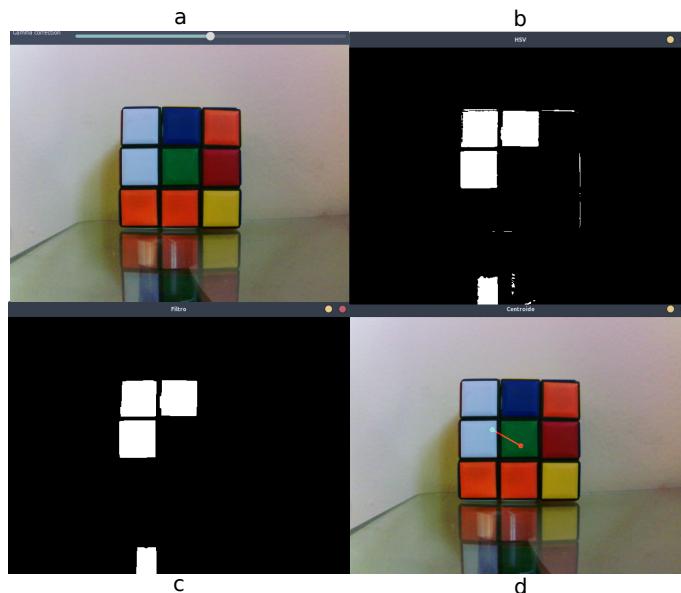
Para obtener el centroide de un objeto en una imagen binaria tenemos que hacer uso del teorema de green, visto en el capítulo 2, y de las librerías opencv.

En esta sección haremos uso de los métodos **centroid**, **drawCentroid** y **publishCoordinates** de la clase **objectTracking** descrita en la sección 4.3.2. El metodo centroid tiene los siguientes parametros

#### Función 4.8.1: Centroid

```
void centroid(Mat &_Image, uint16_t &x ,uint_y & y )
```

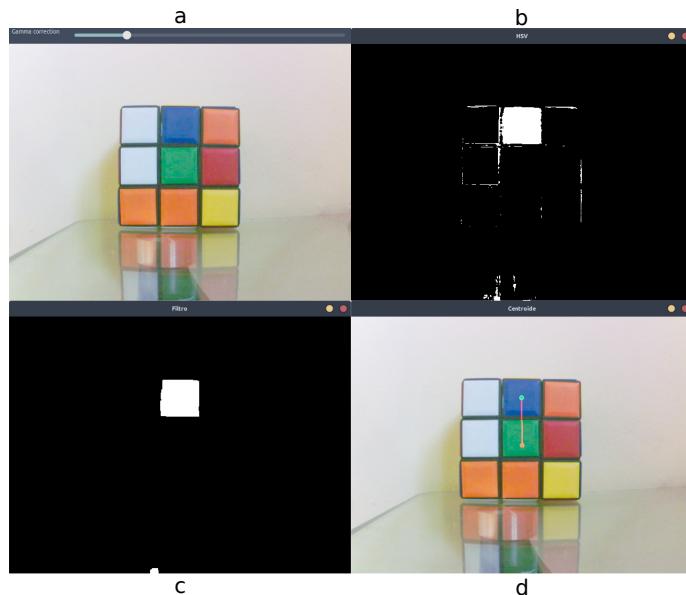
La salida del filtro morfológico es una de las entradas de la función 4.8.1, además de pedir dos variables de tipo uint16 donde se almacenaran las coordenadas (X, Y) del centroide de la figura.



**Figura. 4.29.:** Obtención del centroide sin corrección de  $\gamma$

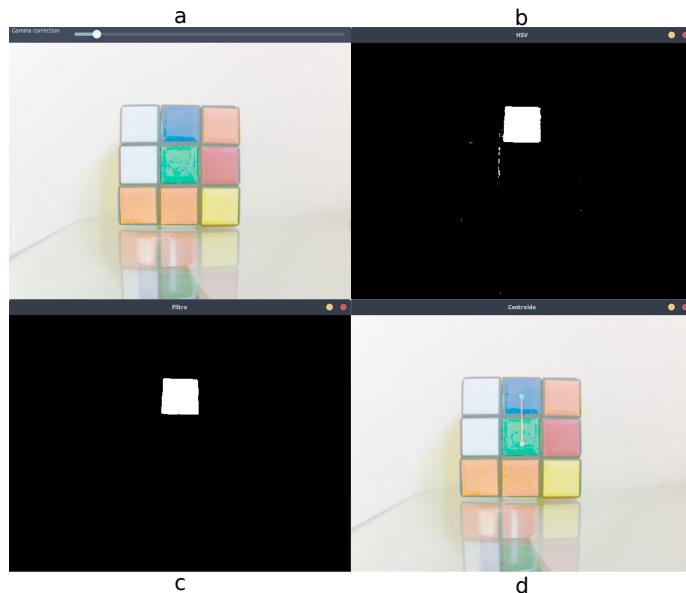
Como primera prueba tenemos en a) la ventana donde se puede modificar el valor de gamma para la corrección de brillo, que en la la figura anterior no tenemos ninguna corrección de  $\gamma$ , en b) tenemos la aplicación del algoritmo de HSV, pero podemos observar que detecta a los cuadros blancos como si fueran azules, en c) tenemos la imagen procesada por dos filtros: opening y closing y finalmente en d) observamos un punto rojo al centro del plano de la cámara y un punto verde que representa el resultado del teorema de green, que debería estar en el centro del recuadro azul, sin embargo, esta lejos del centroide esto debido a que con los valores que establecimos en HSV y con dada iluminación del ambiente el algoritmo toma algunos tonos de blanco como azules, para eso la solución propuesta es la modificación del valor de gamma para ajustar los tonos.

Para la siguiente prueba ahora modificamos a gamma en un valor de 0.4 y el resultado es el siguiente.



**Figura. 4.30.:** Obtención del centroide con corrección de  $\gamma$  en 0.4

Es notable la mejora que tuvo con una corrección de gamma en 0.4, ya que ahora podemos ver que en b) detecta por completo el recuadro azul con un poco de ruido añadido pero en c) el ruido es limpiado completamente para que en d) tengamos el círculo verde dibujado en el centroide del cuadro azul.



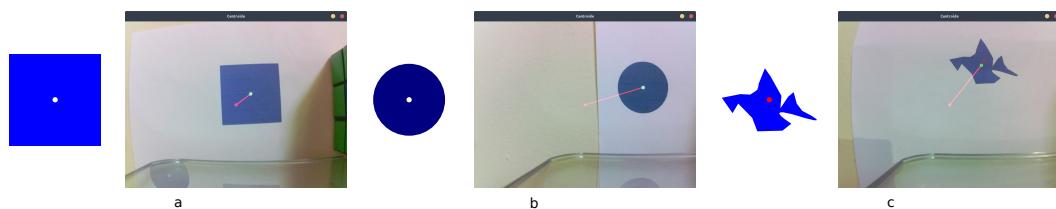
**Figura. 4.31.:** Obtención del centroide con corrección de  $\gamma$  en 0.2

Si se baja aún más el valor de gamma podemos observar que en la imagen anterior en b) la salida del proceso HSV sale con muy poco ruido, que después se limpia por completo en c) para tener como resultado el centroide

localizado y dibujado en el centro del cuadro azul.

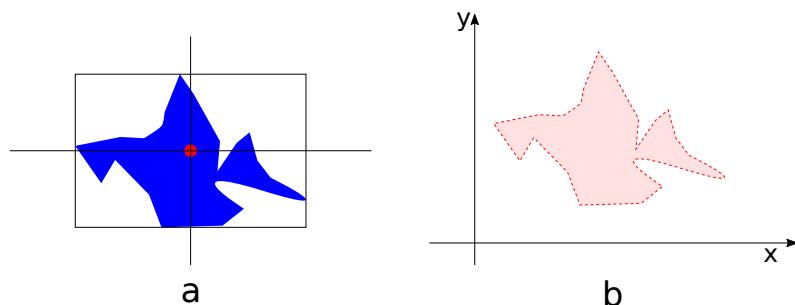
Con esto solucionamos los problemas de tener que ajustar el rango de 3 valores durante el proceso del cambio de espacio de color y así facilitamos al usuario la tarea de encontrar dichos valores. El valor de gamma podría ser ajustado por software con ayuda de un sensor de iluminación, pero eso sale de los rangos del proyecto, y queda como desarrollo a futuro.

Además de las pruebas anteriores se hicieron otras para conocer como funciona el algoritmo ante otras figuras geométricas.



**Figura. 4.32.:** Pruebas del centroide con diferentes formas geométricas

Podemos observar que las figuras a) y b) tienen un respectivo punto blanco, esto con la finalidad de poder saber con exactitud si el centroide que encuentra el sistema es el real, y como se puede observar el punto verde encaja justo con el punto blanco impreso en la figura, por lo que podemos afirmar que encuentra el centroide real, pero que pasa con figuras de geometría irregulares, como sucede en c) hay un punto rojo que no coincide con el punto verde creado por el sistema, esto es porque el punto impreso en la figura es el centro del rectángulo que toca los bordes de la figura, como se ilustra en la siguiente imagen

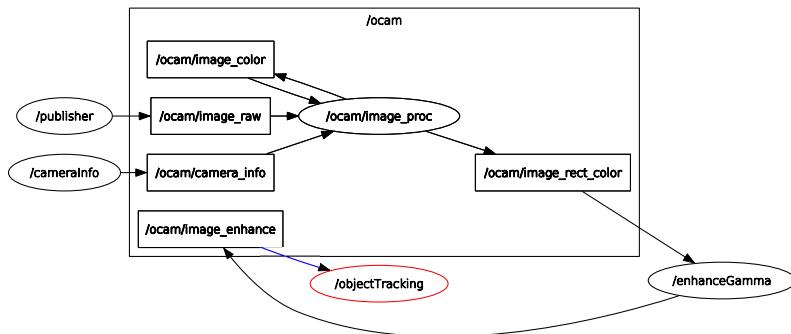


**Figura. 4.33.:** teorema de Green

En a) tenemos un centro que no es el correcto, hay que recordar que la búsqueda del centroide se basa en el teorema de Green, es decir que busca en centroide con base en momentos, esto es que pone a la figura en un plano b) de dos dimensiones y es entonces que se realiza una integral de línea en sentido antihorario por todo el contorno de la silueta, y con base en eso se

puede obtener el centroide real, el propósito de esta última prueba era la de verificar que el sistema no haga lo que no debe de hacer y es un tipo de prueba típica en el proceso de pruebas de software.

Visto desde nodos de ROS tenemos



**Figura. 4.34.:** Nodos de ROS

El nodo llamado `objectTracking` tiene la clase que se encarga de hacer el proceso de HSV del filtro y de la búsqueda del centroide, y publica dos variables de tipo `uint16` que son las coordenadas del centro de la figura.

# Sistema Embebido

**“** I do not fear computers. I fear lack of them.

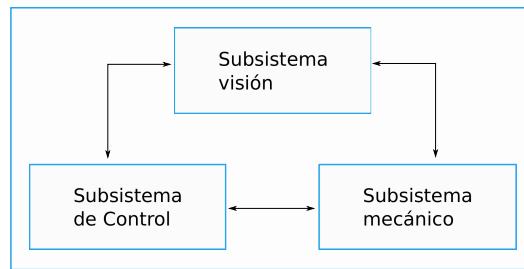
— Isaac Asimov

(American writer and professor of  
biochemistry)

En este capítulo se abordan los procedimientos de diseño del sistema embebido desde la parte mecánica hasta la obtención del controlador.

## 5.1 Arquitectura

La arquitectura general del sistema embebido está dividido en 3 diferentes sub-áreas que en su conjunto ayudan al seguimiento de un objetivo mediante visión artificial.

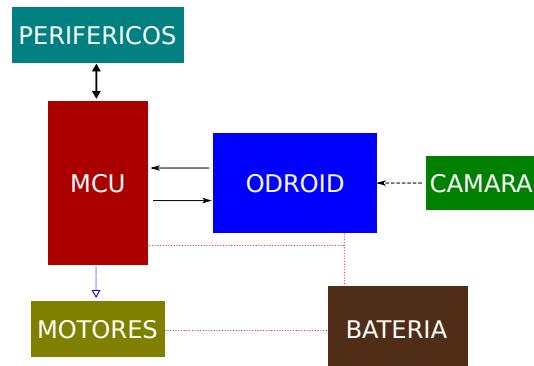


**Figura. 5.1.:** Modulos del sistema

Tal y como se ilustra en la figura anterior, dichas sub-áreas se relacionan entre si, compartiendo datos para tener un sistema integrado y robusto.

- Visión: Identifica el centroide del objetivo, el desarrollo del algoritmo se abordó en el Capítulo 4
- Control: Un micro-controlador es el encargado de interactuar con los datos de la cámara y retroalimentarlos con la salida deseada, para después controlar con un PID.
- Mecánico: Son los actuadores del sistema que hacen que la cámara tenga dos grados de libertad de movimiento, que son las rotaciones en Pitch y Yaw, el modelo se abordó en el Capítulo 3.

La comunicación de datos se da por varios buses de datos, entre todos los componentes del sistema, la siguiente imagen ilustra la interacción entre ellos y se presenta la arquitectura propuesta



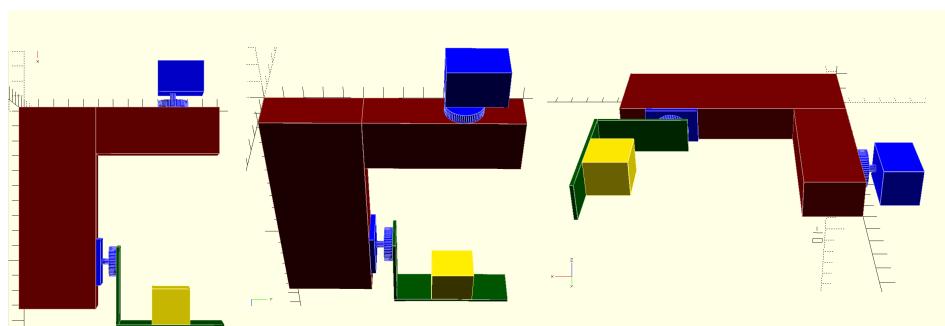
**Figura. 5.2.:** Arquitectura del sistema

El microcontrolador es de 32bits arquitectura RISC, las especificaciones detalladas de los motores y de la tarjeta de desarrollo ODROID puede ser consultada en el Apéndice B.

Este tipo de Arquitectura permite que la ODROID pueda comunicarse paralelamente con el MCU por medio del protocolo serial, que va a 57000 baudios, ademas de que permite que el microcontrolador tenga la posibilidad de publicar en ROS.

## 5.2 Mecanismo móvil

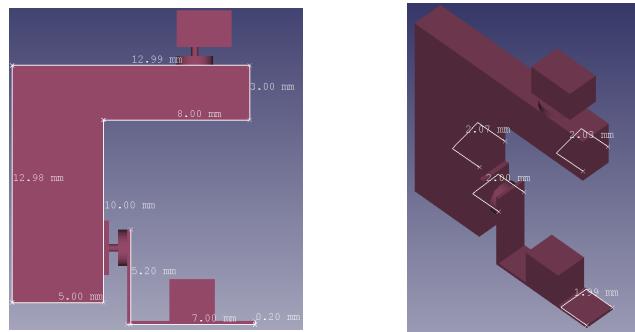
El diseño del sistema fue hecho mediante el uso de software libre llamado OpenSCAD, en el cual se trazó el prototipo del sistema utilizando dos motores tipo servo, es decir, que tienen su propio controlador de velocidad integrado vía PWM.



**Figura. 5.3.:** Diseño CAD del prototipo

Los motores estan representados en color azul, mienrmas que la camara esta de amarillo, como se puede apreciar la camra recae sobre un soporte de color verde, dicha camara esta sujetada por un par de ligas que ayudan a que esta no se caiga cuando hay movimientos de " Till " .

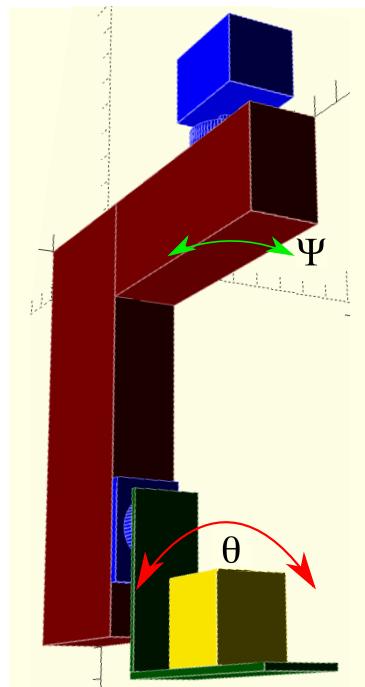
A continuación se presentan algunas medidas del diseño, las unidades estan en mm.



**Figura. 5.4.:** Dimensiones del CAD

Como se vio previamente en los capítulos 2 y 3 este sistema tendrá dos ejes de libertad, que en términos técnicos, a estos movimientos tambiéen se les conoce como " Till y " Pan que es no es mas que otra manera de decirle a la rotación sobre el eje Y y Z.

Es entonces que esos movimientos pueden ser visualizados como se presenta a continuación.



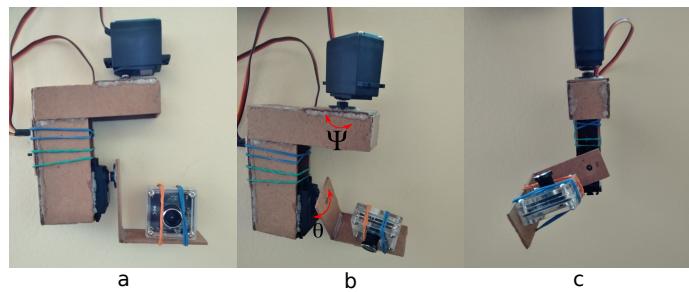
**Figura. 5.5.:** Grados de libertad del sistema

Donde

- $\theta$  representa el movimiento de "Till"
- $\psi$  representa el movimiento de "Pan"

### 5.3 Prototipo físico

Para la realización del prototipo se utilizó un material ligero para que no agregue mucho peso adicional al sistema.

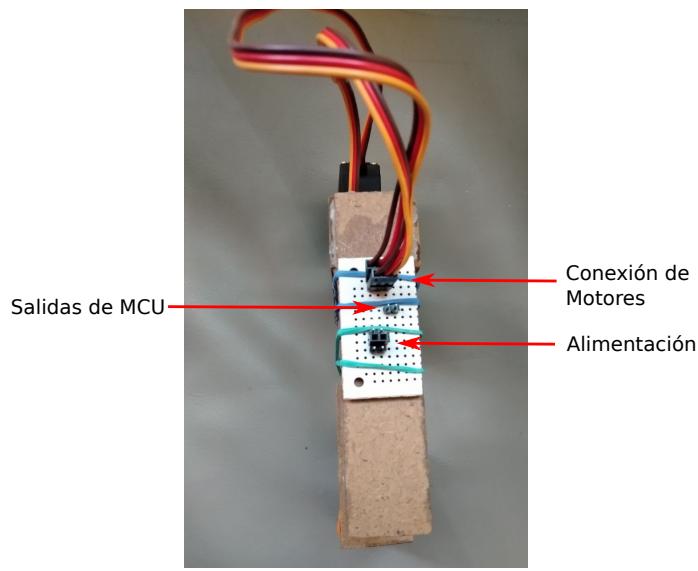


**Figura. 5.6.:** Prototipo del sistema hecho con madera

Como se puede observar la cámara esta sujetada a la base por medio de ligas que le dan mejor soporte y ayudan a que esta no se caiga al momento de seguir a los objetivos.

- Peso del sistema sin motores ni cámara : 70 g
- Peso con motores y cámara : 215 g

En b) y c) de la imagen anterior se ilustra el movimiento que puede realizar el sistema, y que será controlado por la cámara y actuado por los motores. Adicionalmente se diseñó un pequeño circuito que sirve como terminal para las conexiones de alimentación y señales, dicho circuito se encuentra en la parte lateral del sistema sujetado por ligas y pegamento.



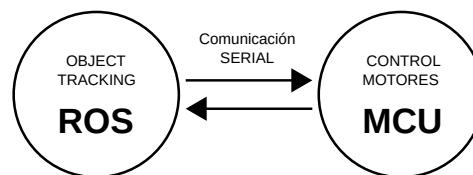
**Figura. 5.7.:** Circuito eléctrico

## 5.4 ROS-Microcontrolador

La parte mecánica encargada de hacer mover la cámara alrededor de dos ejes son los motores de corriente directa, que están conectados a un microcontrolador que es el encargado de ejecutar el controlador.

El control tiene como entrada las coordenadas obtenidas por el algoritmo descrito en el capítulo 4. Y con base en ese pixel, podemos calcular el error teniendo en cuenta que la referencia es el centro de la cámara.

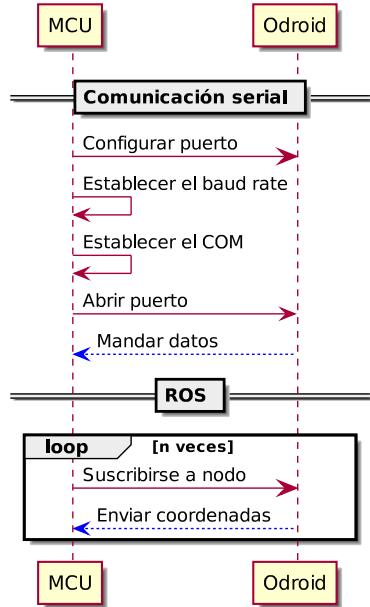
Una vez dicho lo anterior queda una pregunta clara, ¿Cómo comunicar el microcontrolador encargado de los motores, con la cámara que esta siendo ejecutada en ROS?



**Figura. 5.8.:** Protocolo de comunicación

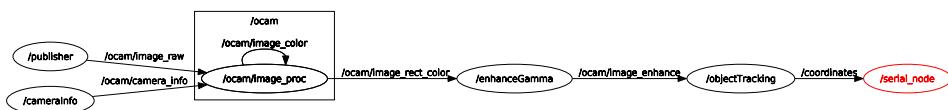
Como se ilustra en la imagen anterior el protocolo de comunicación que se utilizó es el serial, debido a su practicidad y que ya cuenta con librerías de ROS para el microcontrolador, que para la primera etapa de esta investigación se utiliza el Atmega328p.

La comunicación está basada en ser asíncrona, y se detalla mejor en el siguiente diagrama:



**Figura. 5.9.:** Diagrama de secuencia

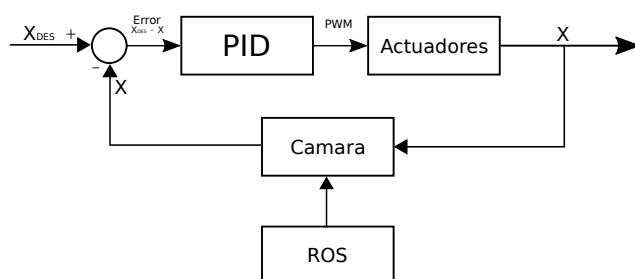
La velocidad de la comunicación es de 570000 baudios y se conecta a través de un cable usb. El microcontrolador tiene además la función de publicar el error con la finalidad de graficar dicho error y obtener conclusiones. Que visto desde el plano de ROS tenemos los siguientes Topics y Nodes:



**Figura. 5.10.:** Nodos y Topicos activos

## 5.5 Control

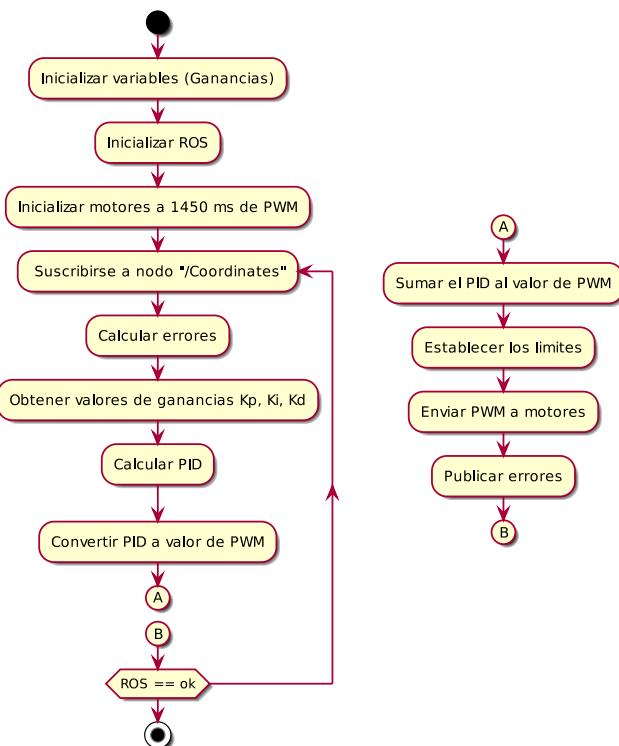
Durante el estudio de diseño del controlador, los parámetros del controlador PID se ajustan con el sistema no lineal, basado en la obtención de la función de transferencia de los motores de corriente directa, como se mostrará enseguida el lazo de control principal esta dado por una entrada que es la coordenada en algún eje y la deseada que se localiza en el pixel central de la cámara.



**Figura. 5.11.:** Lazo de control

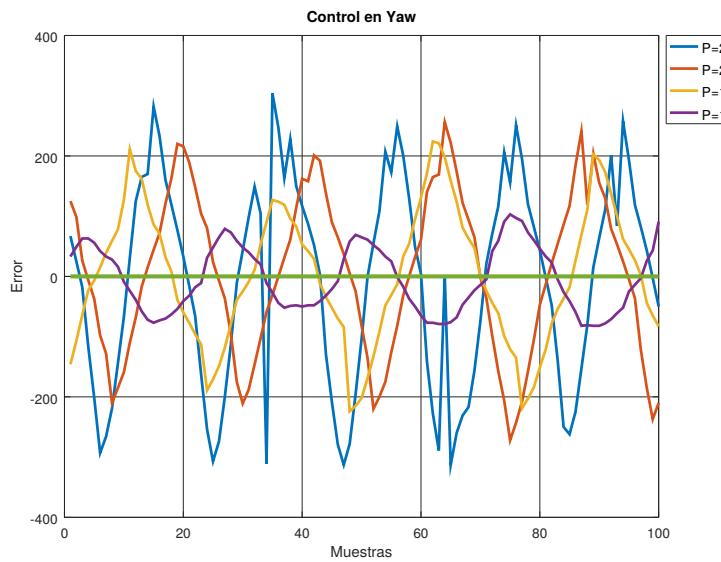
El lazo de control es aplicable también para la coordenada Y, como fue abordado en la sección 5.2 la comunicación se da mediante el protocolo serial, y debido a que la tarea que se encarga de realizar el algoritmo de control se debe suscribir al nodo de coordenadas, este se ve limitado, a un tiempo de muestreo de 60fps, o dicho de otro modo a 16.66 ms.

El algoritmo general del control esta descrito en los siguientes diagramas, basado en programación orientado a objetos donde hay dos puntos claves, la obtención de datos y la publicación de los errores para después ser graficados.



**Figura. 5.12.:** Lazo de control

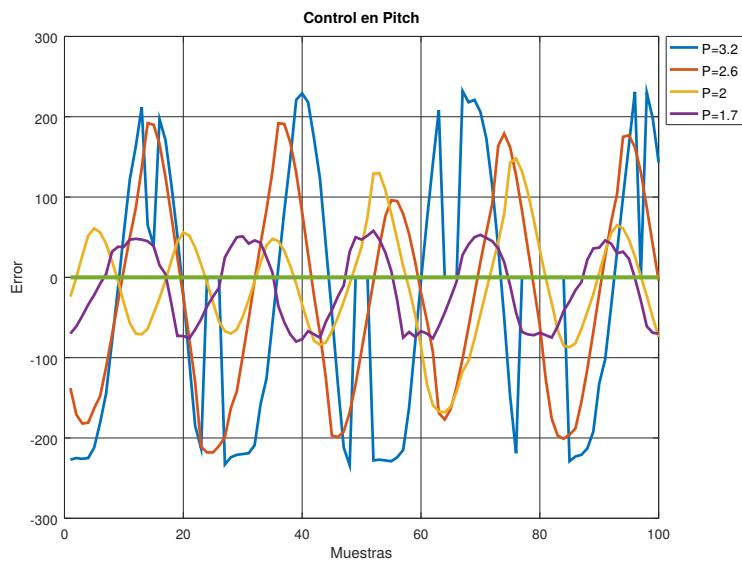
Lo primero que se realizó fue un controlador proporcional y analizar su respuesta, de la cual se grafico el error, teniendo como referencia el cero.



**Figura. 5.13.:** Control proporcional sobre el eje X

El primer control se hizo variando el valor de  $K_p$  para poder ver la respuesta de la oscilación del sistema, donde se puede ver claramente que la frecuencia aumenta conforme la ganancia lo hace, por lo que mete inestabilidad a la lectura.

Haciendo el mismo procedimiento en el eje Y variando las ganancias tenemos:

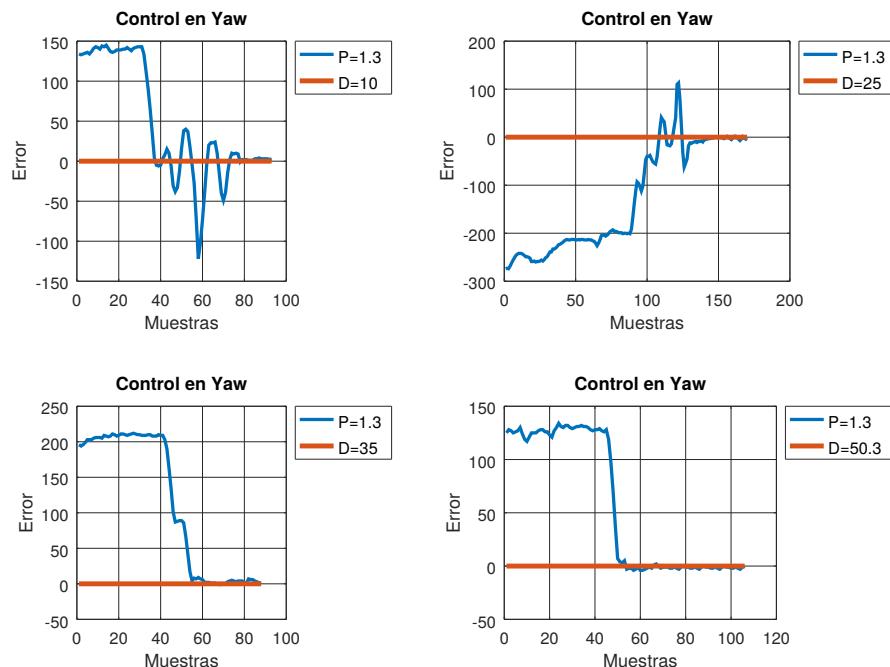


**Figura. 5.14.:** Control proporcional sobre el eje Y

Similar a lo que pasa en X ocurre a Y, donde conforme la oscilación aumenta a un nivel incontrolable.

El siguiente paso es reducir las oscilaciones utilizando el control Derivativo, que sirve como amortiguador para tener un mejor control. Empezando por agregar varios valores de Kd hasta lograr que pueda amortiguar y llegar a un punto donde las oscilaciones sean en estado estacionario.

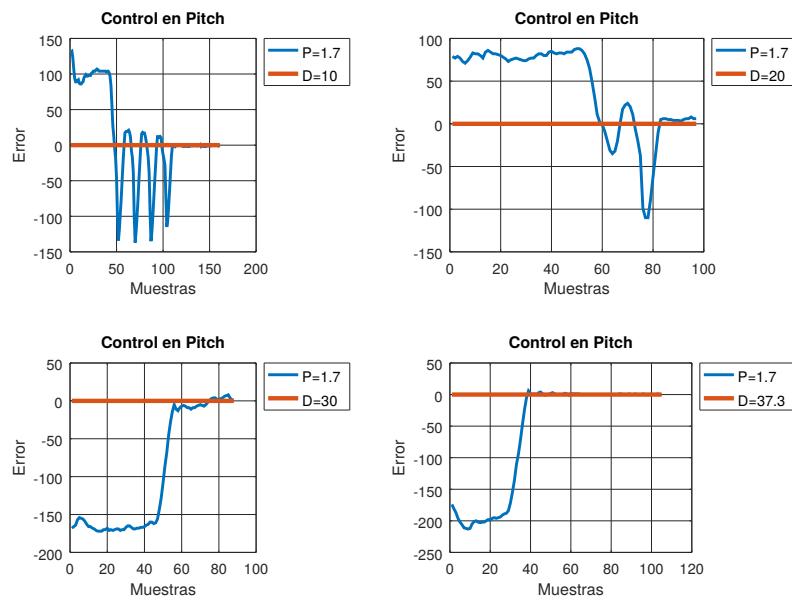
El metodo de sintonización esta basado en primero hacer oscilar al sistema por medio de la adcion de un control proporcional para despues amortiguar el sistema agregando un control derivativo, dejando entonces al sistema con pequeñas oscilaciones que pueden ser corregidas con un control integral. Las siguientes graficas ilustran las variaciones de Kd en el control del Yaw:



**Figura. 5.15.:** Control proporcional derivativo sobre el eje X

De donde podemos observar como es que la señal es amortiguada a tal nivel que el error se acerca a cero con +-3, que para ser un error de píxeles se vuelve un control deseado, debido a que la diferencia entre un pixel y otro esta en el orden de los milis.

Por su parte tenemos el control en Pitch donde se siguió el mismo procedimiento que en el otro eje descrito.



**Figura. 5.16.:** Control proporcional derivativo sobre el eje Y

# Bibliografía

- [Ani08] Anil Bharath. *Next Generation Artificial Vision System*. Artech House, 2008 (vid. pág. 8).
- [Cor11] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011 (vid. pág. 26).
- [Esc11] Arturo de la Escalera. *Visión por Computador. Fundamentos y métodos*. Prentice Hall, 2011 (vid. págs. 22, 33).
- [Flu09] Jan Flusser. *Moments and Moment Invariants in Pattern Recognition*. Wiley Sons Ltd, 2009 (vid. pág. 37).
- [Jäh97] Bernd Jähne. *Digital Image processing*. Springer, 1997 (vid. págs. 22, 25).
- [Jos05] José Ramón Mejía Vilet. *Apuntes de Procesamiento Digital de Imágenes*. First. Facultad de Ingeniería UASLP, ene. de 2005 (vid. págs. 10, 11, 16, 22, 23, 34, 35).
- [Jos18] Lentin Joseph. *Robot Operating System for Absolute Beginners: Robotics Programming Made Easy*. Apress, 2018 (vid. págs. 17, 21).
- [Kim00] Taiyi ChengHyun Wook ParkYongmin Kim. „IMAGE FILTERING IN HS COLOR SPACE“. En: *University of Washington* (2000) (vid. pág. 28).
- [Kue03] Rolf G. Kuehni. *Color Space and Its Divisions Color Order from Antiquity to the Present*. Wiley-Interscience, 2003 (vid. pág. 27).
- [Moh07] Mohinder S. Grewal. *Global Positioning Systems, inertial navigation, and integration*. Wiley, 2007 (vid. pág. 3).
- [Ort02] Francisco Gabriel Ortiz Zamora. *Procesamiento morfológico de imágenes en color: aplicación a la reconstrucción geodésica*. 2002 (vid. págs. 31-33).
- [Qui15] Morgan Quigley. *Programming Robots with ROS*. O'Reilly, 2015 (vid. pág. 20).
- [Raf02] Richard E. Woods Rafael C. Gonzalez. *Digital image processing*. Prentice Hall, 2002 (vid. pág. 27).

- [Ric08] A. J. Ricolfe Viala y Sánchez Salmerón. *Procedimiento completo para el calibrado de cámaras utilizando una plantilla plana*. 2008 (vid. pág. 16).
- [Ser84] Jean Serra. *Image Analysis and Mathematical Morphology, Volume 1*. Academic Press, 1984 (vid. pág. 31).
- [Sze11] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011 (vid. págs. 24, 35).
- [Tay06] Geoffrey Taylor. *Visual perception and robotic manipulation*. Springer-Verlag Berlin Heidelberg, 2006 (vid. pág. 16).
- [YAN94] LUREN YANG. *FAST AND EXACT COMPUTATION OF CARTESIAN GEOMETRIC MOMENTS USING DISCRETE GREEN'S THEOREM*. 1994 (vid. pág. 37).

## Webpages

- [@Ame17] American Academy of Ophthalmology". *Fotorreceptores*. 2017. URL: <https://www.aao.org/salud-ocular/anatomia/fotorreceptores%22> (visitado 10 de ene. de 2020) (vid. pág. 11).
- [@Har] HardKernel. *oCam : 5MP USB 3.0 Camera*. URL: <https://www.hardkernel.com/shop/ocam-5mp-usb-3-0-camera/> (visitado 10 de ene. de 2020) (vid. pág. 45).
- [@Har20] HardKernel. *ODROID XU-4*. 2020. URL: <https://www.hardkernel.com/shop/odroid-xu4-special-price/> (visitado 10 de ene. de 2020) (vid. pág. 18).
- [@MUN] MUNDIARIO. *Cómo el cerebro procesa las imágenes*. URL: <https://www.mundiario.com/articulo/sociedad/cerebro-procesa-imagenes/20190304191451147484.html> (visitado 10 de ene. de 2020) (vid. pág. 17).
- [@NASa] NASA. *gimbal bearing – Liquid Rocket Engines (J-2X, RS-25, general)*. URL: <https://blogs.nasa.gov/J2X/tag/gimbal-bearing/> (visitado 10 de ene. de 2020) (vid. pág. 4).
- [@NASb] NASA. *NASA's J-2X Rocket Engine Development*. URL: <https://www.nasa.gov/exploration/systems/sls/j2x> (visitado 10 de ene. de 2020) (vid. pág. 4).
- [@NAS17] NASA. *The Gimbal Rig Mercury Astronaut Trainer*). 2017. URL: <https://www.nasa.gov/centers/glenn/about/history/mastif.html> (visitado 10 de ene. de 2020) (vid. pág. 4).
- [@Opea] OpenCV. *About it*. URL: <https://opencv.org/about/> (visitado 10 de ene. de 2020) (vid. pág. 25).

- [@Opeb] OpenCV. *Color conversions*. URL: [https://docs.opencv.org/master/de/d25/imgproc\\_color\\_conversions.html](https://docs.opencv.org/master/de/d25/imgproc_color_conversions.html) (vid. pág. 30).
- [@Opec] Opencv. *Camera calibration With OpenCV*. URL: [https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera\\_calibration/camera\\_calibration.html](https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html) (visitado 10 de ene. de 2020) (vid. pág. 26).
- [@PHO] EASY BASIC PHOTOGRAPHY. *How Cameras Work-The Parts of a Camera*. URL: <http://www.easybasicphotography.com/the-camera.html> (visitado 10 de ene. de 2020) (vid. pág. 11).
- [@Pla] PlayList. *Gimbal*. URL: <https://playlists.net/artists/gimbal> (visitado 10 de ene. de 2020) (vid. pág. 3).
- [@ROS] ROS. *ROS/Concepts - ROS Wiki*. URL: <http://wiki.ros.org/ROS/Concepts> (visitado 10 de ene. de 2020) (vid. pág. 19).
- [@Ubu14] Ubuntu. *What is Ubuntu MATE?* 2014. URL: <https://ubuntu-mate.org/what-is-ubuntu-mate/> (visitado 10 de ene. de 2020) (vid. pág. 19).
- [@UCS19] IEEE UCSA. *Transformaciones Morfológicas con Visión Artificial*. Mayo de 2019. URL: <http://ucsa.ieeeparaguay.org/2019/05/15/transformaciones-morfologicas-con-vision-artificial/> (vid. pág. 32).



# Índice de figuras

1.1	Representación grafica del objetivo del proyecto . . . . .	2
1.2	Uso de una gimbal para un sensor inercial [Moh07] . . . . .	3
1.3	Uso de una gimbal en un motor de propulsión [@NASa] . . . . .	4
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 . . . . .	7
2.2	El camino que sigue la señal óptica dentro de la cabeza humana.	8
2.3	Formación de una imagen en el ojo . . . . .	10
2.4	Componentes de la camara . . . . .	12
2.5	Proceso general de captura de imagen por la cámara . . . . .	12
2.6	Captura de imagen . . . . .	13
2.7	Regla 1 . . . . .	14
2.8	Regla 2 . . . . .	14
2.9	Regla 3 . . . . .	14
2.10	Obtención geométrica de la imagen . . . . .	15
2.11	Modelo de lente fina . . . . .	15
2.12	Modelo pinhole tomado de [Tay06] . . . . .	16
2.13	Tarjetas procesadoras de datos . . . . .	17
2.14	Pruebas de performance . . . . .	18
2.15	Nodos conectados al Master . . . . .	20
2.16	Representación grafica de Topics . . . . .	21
2.17	Comunicación de mensajes . . . . .	21
2.18	Representación 2D de imágenes digitales mediante conjuntos de puntos discretos en una matriz rectangular . . . . .	23
2.19	Ilustración de cuantización[Jäh97].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. . . . .	24

2.20	Tomada de [Cor11]; a) Imagen con distorsión; b) Imagen después de calibrar . . . . .	26
2.21	Representación del espacio de color RGB . . . . .	27
2.22	Representación del espacio de color HSI . . . . .	29
2.23	Ejemplo de formas básicas de elementos estructurantes planos .	32
2.24	(a) Erosión de X por el elemento estructurante Y. (b) Los elementos conectados del conjunto X más pequeños que Y son eliminados. . . . .	33
2.25	(a) Dilatación de X por el elemento estructurante Y. (b) El conjunto X aumenta su definición. . . . .	34
2.26	(a) Apertura morfológica del conjunto X por el elemento estructurante Y. (b) Eliminación de objetos menores en tamaño al elemento estructurante. La apertura redondea las convexidades importantes. . . . .	35
2.27	(a) Apertura morfológica del conjunto X por el elemento estructurante Y. (b) El cierre redondea las concavidades importantes.	35
2.28	Area de D limitada por C . . . . .	36
3.1	Angulo de azimuth . . . . .	39
3.2	Seguimiento de un objetivo utilizando una cámara sujetada a una gimbal embebida en un vehículo aéreo no tripulado. . . . .	40
3.3	Marco de referencia inercial . . . . .	40
3.4	Rotación en 3 ejes. . . . .	41
3.5	Marco de referencia del cuerpo. . . . .	41
3.6	Marco de referencia de la cámara. . . . .	42
3.7	Marco de referencia Body y gimbal . . . . .	42
3.8	Rotación del marco de referencia Gimbal respecto al del cuerpo	43
4.1	Cámara 'Ocam' obtenida de [@Har] . . . . .	45
4.2	Nodos y topic . . . . .	47
4.3	Comunicación entre opencv y ROS . . . . .	47
4.4	Diagrama de flujo del programa publisher . . . . .	48
4.5	Diagrama de clases del subscriber . . . . .	48
4.6	Comunicación ROS con calibrador . . . . .	49
4.7	Proceso de calibración utilizando un tablero de ajedrez . . . . .	49
4.8	Proceso de calibración utilizando un tablero de ajedrez . . . . .	50
4.9	Diagrama de flujo para cambio de espacio de color . . . . .	51
4.10	a)Imagen RGB; b) Imagen HSV . . . . .	52
4.11	Espacio de color HSV, a)Hue; b)Saturation; c)Value . . . . .	52

4.12	Espectro de colores . . . . .	53
4.13	Separación de colores . . . . .	53
4.14	Separación de colores . . . . .	54
4.15	Separación de color azul . . . . .	54
4.16	Separación de color azul . . . . .	55
4.17	Separación de colores RGB utilizando el modelo HSV . . . . .	55
4.18	a) Imagen captada por la cámara; b) Corrección de contraste; c) Histograma de a y b . . . . .	56
4.19	a) Imagen captada por la cámara; b) Corrección de contraste en 1.5; c) Histograma de a y b . . . . .	57
4.20	a) Imagen captada por la cámara; b) Corrección de brillo en 50; c) Histograma de a y b . . . . .	58
4.21	a) Imagen captada por la cámara; b) Corrección de brillo en 100; c) Histograma de a y b . . . . .	58
4.22	a) Imagen captada por la cámara; b) Corrección de gamma <1; c) Histograma de a y b . . . . .	59
4.23	a) Imagen captada por la cámara; b) Corrección de gamma >1; c) Histograma de a y b . . . . .	60
4.24	Comunicación de nodos . . . . .	60
4.25	corrección de gamma, la imagen de la derecha es la corregida . .	60
4.26	A) Imagen original, B)Aplicación de filtros . . . . .	61
4.27	A) Imagen original, B)Aplicación de filtros. Con size de 3 . . .	62
4.28	A) Imagen original,B) Espacio de color HSV C)Aplicación de filtros	62
4.29	Obtención del centroide sin corrección de $\gamma$ . . . . .	63
4.30	Obtención del centroide con corrección de $\gamma$ en 0.4 . . . . .	64
4.31	Obtención del centroide con corrección de $\gamma$ en 0.2 . . . . .	64
4.32	Pruebas del centroide con diferentes formas geométricas . . . .	65
4.33	teorema de Green . . . . .	65
4.34	Nodos de ROS . . . . .	66
5.1	Modulos del sistema . . . . .	67
5.2	Arquitectura del sistema . . . . .	68
5.3	Diseño CAD del prototipo . . . . .	68
5.4	Dimensiones del CAD . . . . .	69
5.5	Grados de libertad del sistema . . . . .	69
5.6	Prototipo del sistema hecho con madera . . . . .	70
5.7	Circuito eléctrico . . . . .	71
5.8	Protocolo de comunicación . . . . .	71
5.9	Diagrama de secuencia . . . . .	72

5.10	Nodos y Topicos activos . . . . .	72
5.11	Lazo de control . . . . .	72
5.12	Lazo de control . . . . .	73
5.13	Control proporcional sobre el eje X . . . . .	74
5.14	Control proporcional sobre el eje Y . . . . .	74
5.15	Control proporcional derivativo sobre el eje X . . . . .	75
5.16	Control proporcional derivativo sobre el eje Y . . . . .	76
B.1	Odroid XU-4 . . . . .	93

# Índice de cuadros

4.1	Valores para cada rango de color . . . . .	54
4.2	Valores para cada rango de color en un ambiente iluminado por energía eléctrica . . . . .	55



# Lista de códigos

A.1	Nodo publicador de imagen en ros . . . . .	89
A.2	Código en c++ que cambia de espacio de color . . . . .	90



# Apendice A

**Listing A.1:** Nodo publicador de imagen en ros

```
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
39     {
40         ROS_INFO("Camara conectada en el COM %d",
41                 CAMCOM);
42         cv::Mat FRAME;
43         sensor_msgs::ImagePtr MSG;
44
45         ros::Rate loop_rate(SAMPLETIME); /*HZ*/
46
47         while (nh.ok())
48         {
49             VC >> FRAME;
50             if (!FRAME.empty())
51             {
52                 MSG = cv_bridge::CvImage(std_msgs::Header(),
53                                         "bgr8", FRAME).
54                                         toImageMsg();
55                 pub.publish(MSG);
56                 cv::waitKey(30);
57             }
58             ros::spinOnce();
59             loop_rate.sleep();
60         }
61     }

```

**Listing A.2:** Código en c++ que cambia de espacio de color

```

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" <<
26                 std::endl;
27             return -1;
28         }
29
30         cvtColor(image, imgHSV, COLOR_BGR2HSV);
31         inRange(imgHSV, Scalar(low_H, low_S, low_V),
32                 Scalar(high_H, high_S, high_V), imgHSV);
33
34         namedWindow("Escala de colores", 0); // Create a
35             window for display.
36         resizeWindow("Escala de colores", 683, 316);
37         namedWindow("HSV", 0); // Create a window for
38             display.
39         resizeWindow("HSV", 683, 316);
40
41         imshow("Escala de colores", image); // Show our
42             image inside it.
43         imshow("HSV", imgHSV); // Show our
44             image inside it.
45         waitKey(0); // Wait for a
46             keystroke in the window
47         return 0;
48     }

```



# Apendice B

## B.1 ODROID

El modelo XU-4 cuenta con las siguientes especificaciones: [hardkernel2017]

### ■ Procesador

Samsung Exynos5422 Cortex™-A15 2Ghz and Cortex™-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

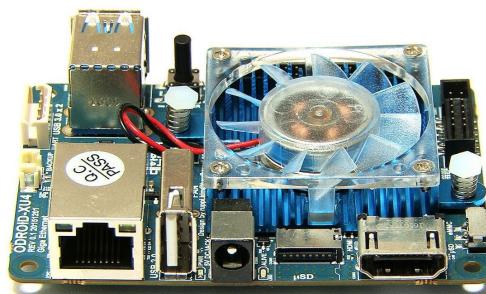
Hay dos tipos de almacenamiento de el sistema operativo. El primero es usando una microSD y el segundo es utilizando un modulo eMMC.

### ■ 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. B.1.:** Odroid XU-4

