

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 título de  
Ingeniero en Electrónica y Control de Sistemas de  
Aeronaves.

Marco Antonio Aguilar Gallardo

Dirige: M. en C. Antonio Flores Moreno

Municipio de Colón, Querétaro

25 de marzo de 2021



## 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      M. en C. Antonio Flores Moreno

Maestro en ciencias  
Asesor Técnico

Supervisores      Antonio Flores Moreno y María del Carmen

25 de marzo de 2021

**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, 25 de marzo de 2021

Asesores: M. en C. 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



Certificados en  
ISO 9001:2015

## AUTORIZACIÓN DE ENTREGA FINAL DEL TRABAJO PROFESIONAL

**Aguilar Gallardo Marco Antonio**

Matrícula 4064, Grupo IECSA02-A

Candidato al grado de Ingeniero en Electrónica y Control de Sistemas de Aeronaves

### P r e s e n t e

Se autoriza la entrega final del Trabajo Profesional con fecha **23 de octubre de 2020**, titulado: **Desarrollo y control de una gimbal de dos grados de libertad mediante visión artificial para el seguimiento de objetos**, realizado como parte del Proyecto en la UNAQ, el cual ha sido revisado y validado por el personal académico:



Ing. María del Carmen Alvaréz Lara

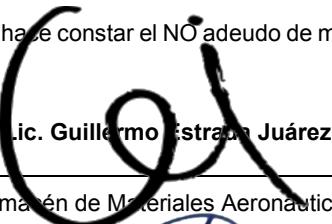
Profesor de la Asignatura de  
Proyecto Final



M. en C. Antonio de Jesus Flores Moreno

Profesor – Asesor Técnico

Además, se hace constar el NO adeudo de materiales, equipos, herramientas o documentos en las siguientes áreas.



Lic. Guillermo Estrada Juárez

Almacén de Materiales Aeronáuticos

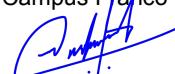


Lic. Laura Jiménez Camilo



Técnico Jorge David Guerrero Montero

Almacén del Campus Franco-Mexicano



Lic. Edgar Rangel Garcia

Centro de Información



Vinculación y Desarrollo Profesional

Lic. Cyntia Moreno Vázquez

Departamento de Servicios Escolares / Titulaciones

### A T E N T A M E N T E



M. en C. Felipe Augusto López Garduza

Subdirección de Ingeniería

c.c.p. Servicios Escolares

F061-A SIG / 10-Sep-2018 / Rev.02 / SING

Carretera Estatal 200  
No. 22154 Colón,  
Qro. México.

+ 52 // 442 // 101 6600

[www.unaq.edu.mx](http://www.unaq.edu.mx)

QUERÉTARO  
ESTÁ EN NOSOTROS



SECRETARÍA  
DE EDUCACIÓN



Estudiante



Certificados en  
ISO 9001:2015



Carretera Estatal 200  
No. 22154 Colón,  
Qro. México.

+ 52 // 442 // 101 6600

[www.unaq.edu.mx](http://www.unaq.edu.mx)

**QUERÉTARO**  
ESTÁ EN NOSOTROS



SECRETARÍA  
DE EDUCACIÓN

F061-A SIG / 10-Sep-2018 / Rev.02 / SING

# Agradecimientos

A mi Familia.

Mi Hermano Alexis que siempre ha creido en mi, gracias por llenarme de confianza.

Mi Padre que me dio la autonomia, gracias por dejarme tomar mis desiciones libremente.

Mi Madre que me da tranquilidad, gracias por abrigarme con tu inmerso amor.

Siempre estan conmigo.



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Tema de investigación . . . . .	1
1.2	Justificación . . . . .	2
1.3	Objetivo . . . . .	2
1.4	Objetivos específicos . . . . .	2
1.5	Estado de la cuestión . . . . .	3
1.6	Contribuciones . . . . .	6
1.7	Alcances . . . . .	7
<b>2</b>	<b>Marco Teórico</b>	<b>9</b>
2.1	Óptica . . . . .	10
2.1.1	Descripción general del sistema visual humano . . . . .	10
2.1.2	Estructura del ojo humano . . . . .	11
2.1.3	Formación de imágenes en el ojo . . . . .	12
2.1.4	Cámara digital . . . . .	14
2.1.5	Componentes de la cámara . . . . .	14
2.1.6	Formación de la imagen en la cámara . . . . .	15
2.1.7	Cálculo de la imagen . . . . .	17
2.1.8	Modelo de la cámara . . . . .	19
2.2	Procesamiento de datos . . . . .	21
2.2.1	Tarjeta procesadora . . . . .	21
2.2.2	ROS . . . . .	22
2.2.3	Procesamiento de imágenes . . . . .	25
2.2.4	Cuantización . . . . .	27
2.2.5	Librerías OPENCV . . . . .	27
2.2.6	Calibración de cámara . . . . .	28
2.2.7	Espacio de color . . . . .	29
2.2.8	Transformaciones morfológicas . . . . .	33

2.2.9	Contraste y brillo . . . . .	39
2.2.10	Teorema de Green . . . . .	39
2.3	Control . . . . .	41
2.3.1	Obtención de sistemas de control . . . . .	43
2.3.2	Lazo abierto y cerrado . . . . .	44
2.3.3	Respuesta en el tiempo . . . . .	45
<b>3</b>	<b>Modelo matemático</b>	<b>49</b>
3.1	Marco de referencia . . . . .	49
3.1.1	Marco de referencia Inercial [I] . . . . .	52
3.1.2	Marco de referencia del cuerpo [B] . . . . .	52
3.1.3	Marco de referencia de la gimbal [G] . . . . .	54
3.2	Matrices de rotación . . . . .	54
3.3	Ecuaciones de movimiento de una gimbal . . . . .	57
3.3.1	Rotación sobre Pitch . . . . .	59
3.3.2	Rotación sobre Yaw . . . . .	61
<b>4</b>	<b>Visión Artificial</b>	<b>67</b>
4.1	Cámara . . . . .	67
4.2	Algoritmo general . . . . .	68
4.3	Comunicación con ROS . . . . .	69
4.3.1	Publisher . . . . .	69
4.3.2	Subscriber . . . . .	71
4.4	Calibración de cámara . . . . .	71
4.5	Espacio de color . . . . .	74
4.5.1	Pruebas de campo . . . . .	78
4.6	Corrección de brillo . . . . .	80
4.6.1	Corrección de Gamma . . . . .	83
4.7	Filtro morfológico . . . . .	86
4.8	Centroide . . . . .	88
<b>5</b>	<b>Sistema Embebido</b>	<b>93</b>
5.1	Arquitectura . . . . .	93
5.2	Mecanismo móvil . . . . .	95
5.3	Prototipo físico . . . . .	97
5.4	ROS-Microcontrolador . . . . .	99
5.5	Control . . . . .	100

5.5.1	Control en Pitch . . . . .	101
5.5.2	Control en Yaw . . . . .	110
<b>6</b>	<b>Conclusiones</b>	<b>117</b>
	<b>Índice de figuras</b>	<b>123</b>
	<b>Índice de cuadros</b>	<b>127</b>
<b>A</b>	<b>Apendice A</b>	<b>131</b>
<b>B</b>	<b>Apendice B</b>	<b>135</b>
B.1	ODROID . . . . .	135
B.2	Atmega328P . . . . .	136



# Introducción

“ *El experimentador que no sabe lo que está buscando no comprenderá lo que encuentra.*

— Claude Bernard  
(Biólogo teórico, médico y fisiólogo francés.)

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.

Sensores como radar, láser y cámaras se han utilizado ampliamente en muchas aplicaciones, como el procesamiento de imágenes, sistemas de rastreo y sistemas de navegación. En este tipo de sensores, el eje del sensor óptico debe apuntar con precisión desde una base móvil a un objetivo fijo o móvil. En un entorno de este tipo, en el que el equipo suele estar montado en una plataforma móvil, mantener la orientación de los sensores hacia un objetivo es un serio desafío. Una plataforma de estabilización inercial es una forma

apropiada de resolver este desafío. Este trabajo se centra en el control de una gimbal de dos grados de libertad a través de una cámara montada sobre la plataforma interna de la misma gimbal.

## 1.2. Justificación

La presente investigación se enfocará 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 representará 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.

## 1.4. Objetivos específicos

- Definir el modelo matemático de una gimbal con base a los grados de libertad definidos.
- Capturar figuras geométricas predefinidas mediante el uso de una cámara digital y emplear algoritmos de visión artificial para la obtención de datos.
- Diseñar e implementar el sistema embebido que dará el soporte electrónico a la gimbal.

---

<sup>1</sup>Micro air vehicle

- Diseñar un controlador autónomo con base en el modelo matemático, previamente obtenido.

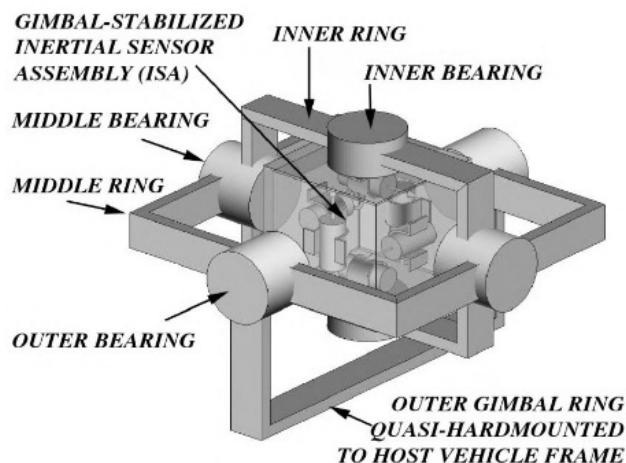
## 1.5. Estado de la cuestión

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 diseño un sistema que pudiera estabilizar la inyección de tinta. (PlayList, 2020)

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 la guiñada, inclinación y balanceo del barco.



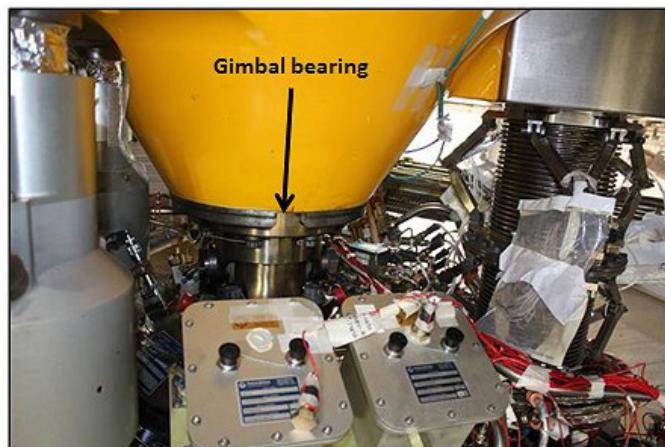
**Figura. 1.1.:** Uso de una gimbal para un sensor inercial (Mohinder S. Grewal, 2007)

En la figura 1.1, 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 gimbal 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. En la figura 1.2 se puede observar 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.(NASA, 2020b)



**Figura. 1.2.:** Uso de una gimbala en un motor de propulsión (NASA, 2020a)

#### ■ 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 gimbal proporcionó una capacitación valiosa para los siete astronautas del

Proyecto Mercurio. Cada uno experimentó unas cinco horas de tiempo de vuelo simulado.(NASA, 2017) EN la figura 1.3 se ilustra el sistema antes descrito, que permitió dar un gran paso hacia la capacitación de astronautas.



**Figura. 1.3.:** Jerrie Cobb, uno de los Mercury 13, da un giro en la plataforma gimbal.  
Créditos: NASA

#### ■ Estabilizador de Cámaras

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 gimbol 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 gimbol 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. 1.4.:** Primer uso de la steadicam

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

#### ■ Control de una gimbal

En la literatura se han empleado muchos métodos de control del sistema gimbal de dos ejes. En donde se han propuesto enfoques de agrupación de polos y control óptimo lineal (Salatun y Bainum, 1983),redes neuronales adaptativas de avance multicapa (Lin y Hsiao, 2001), controlador LQG/LTR (K. J. Seong y Lee, 2006) y controlador integral (PI) proporcional(S. B. Kim y Kwak, 2010).

## 1.6. Contribuciones

Actualmente la forma en que se desarrollan los sistemas gimbal es mediante el uso de una IMU y cámara, para posteriormente aplicar algún control, en este trabajo se enfoca en omitir la IMU y hacer la retroalimentación por medio de la cámara, esto nos permite tener un ahorro económico al momento de diseñar el sistema, además de dar otra aproximación al 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 Vehículos Aéreos

no Tripulados con la incorporación de una tarjeta monoprocesador con ROS.

## 1.7. Alcances

Se tendrá un sistema de control de dos grados de libertad capaz de seguir un objeto de color sólido y geometría conocida en un ambiente controlado, el sistema hará uso de filtros morfológicos (solo opening y closing) y se verá limitado por la corrección del brillo mediante software y no hardware.



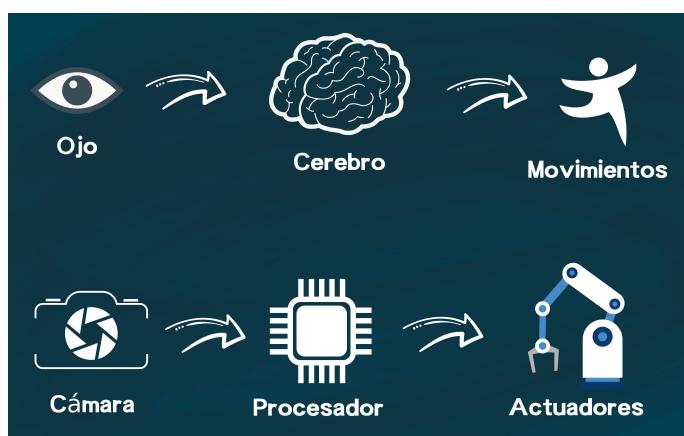
## Marco Teórico

“ Nunca memorices algo que puedes consultar.

— Albert Einstein  
(Físico teórico)

En este capítulo se 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 cuáles son las fases del proyecto y el porqué de ellas. La figura 2.1 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

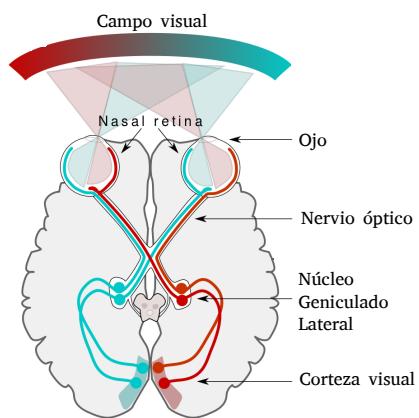
La visión artificial surge de un amplio estudio probabilístico y matemático del procesamiento de imágenes digitales, pero sobre todo de análisis humanos y de la intuición, ya que de estas últimas el ingeniero hace selección de entre una u otra técnica. Esta elección se basa usualmente en juicios visuales 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 del sistema óptico 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 núcleo 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. (Anil Bharath, 2008)



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

En términos generales, cuanto más nos alejamos del camino visual del ojo, menos entendemos lo que está sucediendo. En la figura 2.2 se ilustra el sistema óptico de un ser humano.

### 2.1.2. Estructura del ojo humano

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

El ojo está formado de dos componentes principales:

- **Componentes ópticos:**

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

- **Componentes neurológicos:**

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

De los cuales tenemos los siguientes subcomponentes.

- 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.

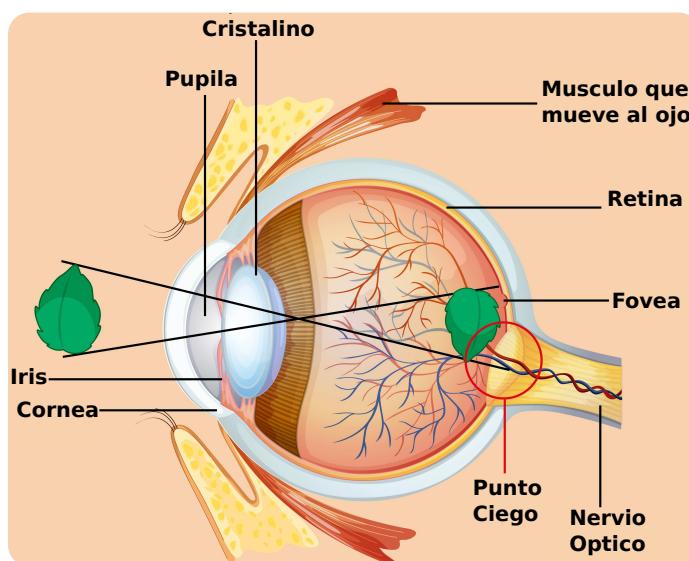
(José Ramón Mejía Vilet, 2005)

### 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.** Tomada de Google

El cerebro es quien realmente ve las imágenes. Tal y como se aprecia en la figura 2.3 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.(José Ramón Mejía Vilet, 2005)

## 2.1.4. Cámara digital

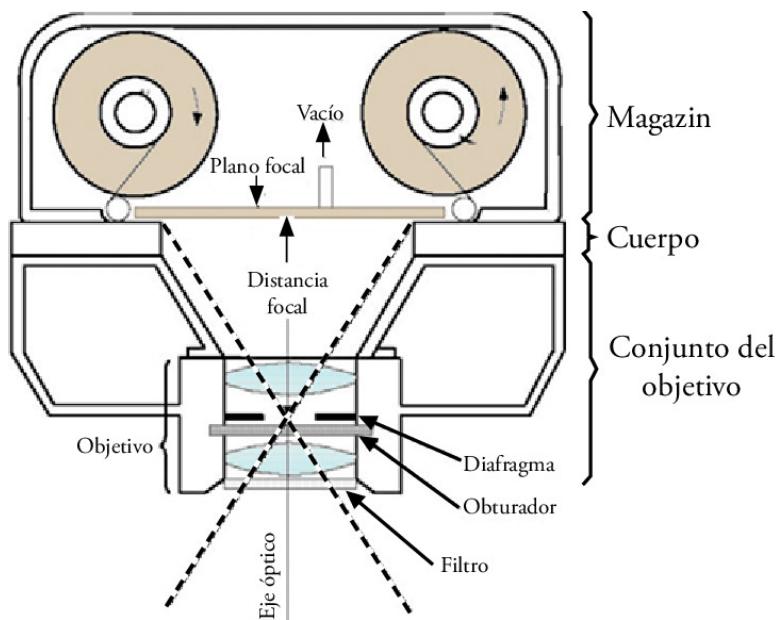
La cámara digital es uno de los cambios más importantes en esta era de la digitalización de la información. Es un invento tan revolucionario y que dista tanto de su predecesor análogo.

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 es 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. (PHOTOGRAPHY, 2020)

## 2.1.5. Componentes de la cámara

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

microscópicos de información de luz llamada "fotosites". Cada una de estas fotositas se conoce mejor como píxeles.



**Figura. 2.4.:** Componentes de la cámara. Tomada de (PHOTOGRAPHY, 2020)

En la figura 2.4 se ilustran todos los componentes antes descritos de la cámara.

## 2.1.6. Formación de la imagen en la cámara

En una cámara fotográfica se recibe la luz que traspasa el diafragma, pasa por los cristales de la cámara hasta llegar al CCD(Charge Coupled Device o, en español, Dispositivo de Carga Acoplada) o sensor, que es donde se forma la imagen correcta y se envía al procesador. Donde cada uno de los componentes están ilustrados en la figura 2.5

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 cuero geniculado, que es la parte del cerebro donde se produce la visión.



**Figura. 2.5.:** Proceso general de captura de luz por el sensor de la cámara digital

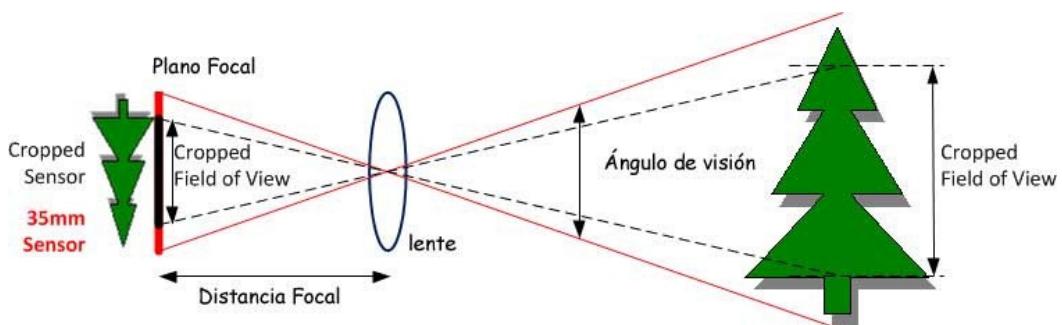
Cuando los rayos paralelos pasan a través de una lente convexa, convergen hacia un punto que se denomina punto focal.

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, como se muestra en la ecuación 2.1:

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



**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.

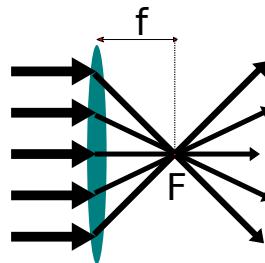
En las ópticas habrá que tener por tanto en cuenta su F mínimo, que indicará la máxima cantidad de luz que puede atravesar la óptica y que tendrá que estar en concordancia con la sensibilidad de la cámara.

## 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**

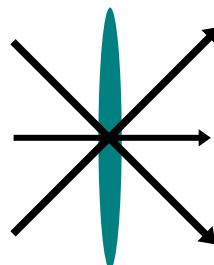
Cualquier rayo incidente paralelo al eje principal en la zona objeto sale pasando por el foco principal en la zona imagen.



**Figura. 2.7.:** Regla 1

- **Regla 2**

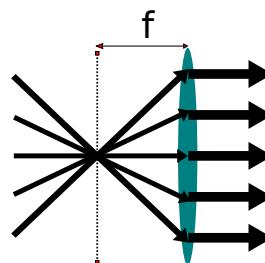
Cualquier rayo incidente que pasa por el centro de la lente sale con la misma dirección, es decir, no sufre desviación.



**Figura. 2.8.:** Regla 2

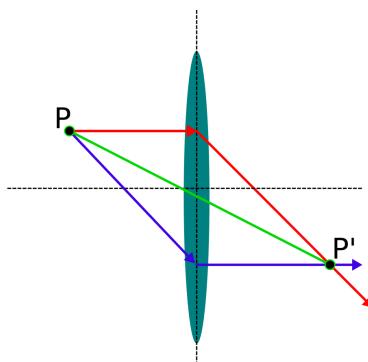
- **Regla 3**

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 2.10 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  $P$ .

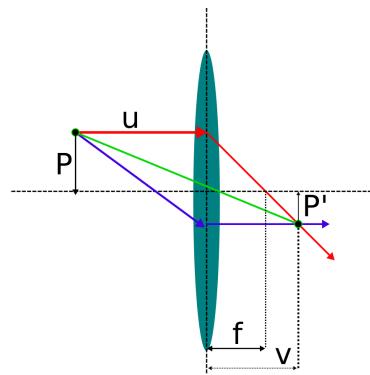


**Figura. 2.10.: Obtención geométrica de la imagen**

Estas tres reglas nos indican la trayectoria que seguirán tan solo tres rayos de todos los que genera el objeto. La imagen vendrá dada por el punto donde se intercepten 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.

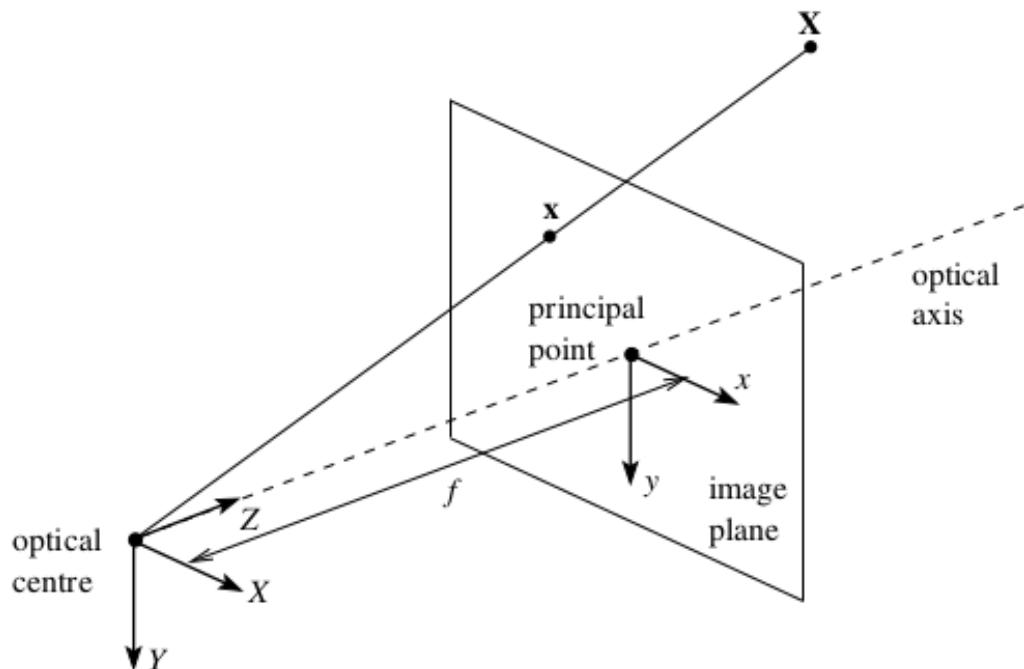


**Figura. 2.11.:** Modelo de lente fina

### 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. (Ricolfe Viala y Sánchez Salmerón, 2008)

La figura 2.12 ilustra la proyección del punto  $X$  de  $\mathbb{R}^3$  al punto  $x$  de  $\mathbb{R}^2$



**Figura. 2.12.:** Modelo pinhole tomado de (Taylor, 2006)

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). (José Ramón Mejía Vilet, 2005)

Despejando de las ecuaciones 2.2 y 2.3 obtenemos la distancia focal como

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

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

$$x' = \frac{fX}{z'}, y' = \frac{fY}{z'}, z' = Z \quad (2.5)$$

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 la lente presenta un tipo de distorsión y hay que corregirlo, en los próximos capítulos se hablará acerca del método de calibración de cámara.

## 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.(MUNDIARIO, 2020) 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. En la figura 2.13 solo se muestran algunas de las más importantes en el mercado actual.



Raspberry Pi



Odroid



FPGA'S



NUC de Intel

### **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. (Joseph, 2018) 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 más rápido en comparación con sus competidores. 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.(HardKernel, 2020a)

## **2.2.2. ROS**

Un sistema de comunicación es a menudo una de las primeras necesidades que surgen al implementar una nueva aplicación de robot. El framework ROS ahorra tiempo al administrar los detalles de la comunicación entre los nodos distribuidos a través del mecanismo 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).

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, 2020)

## 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>al</sup> principio para aquellos que han usado principalmente entornos de desarrollo gráfico en Windows o Mac OS X.(Quigley, 2015)

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.

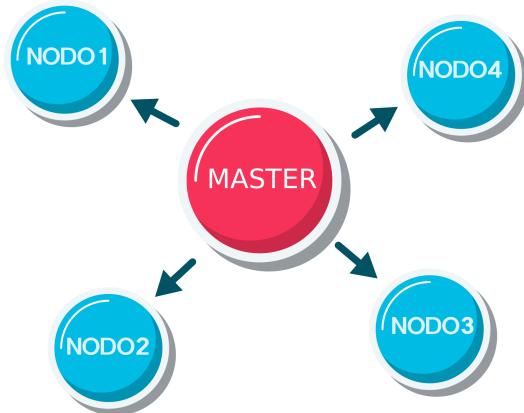
### ■ Nodo

Los nodos son procesos que realizan cálculos. ROS está diseñado para

ser modular entre nodo; un sistema de control de robot generalmente comprende muchos nodos.

- **Master**

Un programa intermedio que conecta nodos.



**Figura. 2.14.:** 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.

- **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.(Joseph, 2018)

## 2.2.3. 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.(Jähne, 1997)

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.(Escalera, 2011) 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.(José Ramón Mejía Vilet, 2005)

- **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

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

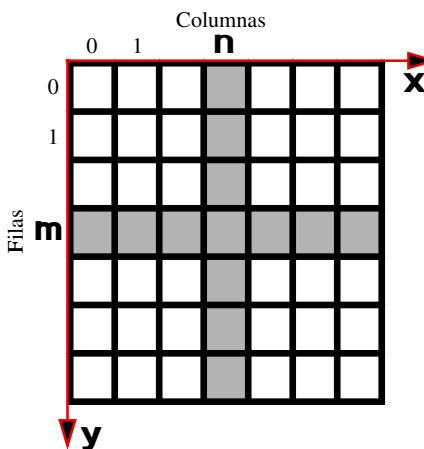
y

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

La ecuación 2.8 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.

Un píxel representa la irradiancia en la posición de cuadrícula correspondiente. En el caso más simple, los píxeles se encuentran en una cuadrícula rectangular como en la figura 2.15. 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. 2.15.:** 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.(José

Ramón Mejía Vilet, 2005)

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,  $x = (x, y) \in \mathbb{R}^2$ (Szeliski, 2011)

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

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.

## 2.2.4. 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.(Szeliski, 2011)

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).

## 2.2.5. Librerías OPENCV

Dentro del mundo del procesamiento de imágenes digitales encontramos una amplia variedad de desarrolladores que han construido librería especializada 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.(OpenCV, 2020) Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS.

## 2.2.6. 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.

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 basada en las medidas de las coordenadas de la plantilla con las ventajas de la auto calibración en la cual no es necesaria utilizar plantilla.

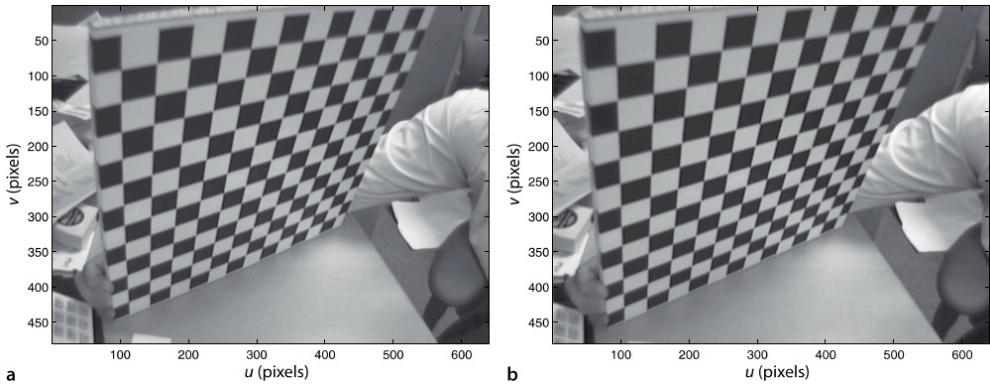
La distorsión geométrica es generalmente el efecto más problemático que encontramos para las aplicaciones robóticas, y comprende dos componentes: radial y tangencial. La distorsión radial hace que los puntos de la imagen se traduzcan a lo largo de líneas radiales desde el punto principal. (Corke, 2011). 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 como se puede observar en la figura 2.16 a)



**Figura. 2.16.:** Tomada de (Corke, 2011); a) Imagen con distorsión; b) Imagen después de calibrar

Las coordenadas del punto  $(u,v)$  después de la distorsión 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 término corresponde a la distorsión radial, mientras que el segundo corresponde a la distorsión tangencial.

## 2.2.7. 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.(Rafael C. Gonzalez, 2002)

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 múltiples espacios de color en la actualidad, siendo el espacio RGB el más 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.(Kuehni, 2003) 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.(T. C. W. P. Kim, 2000)

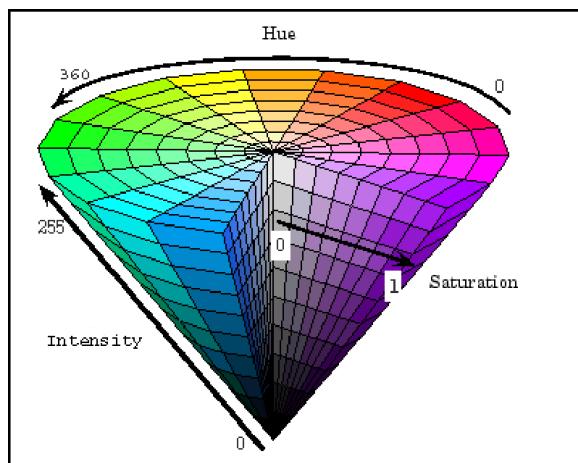
Las transformaciones matemáticas que permiten el paso del espacio RGB al HSI son realizadas normalmente mediante hardware específico. Las ecuaciones son:

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

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

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

Así como el espacio RGB se representa por un cubo, el HSI lo forman dos pirámides unidas por su base como en la figura 2.17. Dependiendo de la intensidad se tiene un corte a los conos. Variaciones de este espacio de color



**Figura. 2.17.:** Representación del espacio de color HSI. Tomada de Google

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

(OpenCV, s.f.)

## 2.2.8. 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 amplia, 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 (Serra, 1984)

## Conceptos elementales de teoría de conjuntos

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

Las siguientes definiciones fueron tomadas de (Ortiz Zamora, 2002)

**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 conmutativa, 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. (UCSA, 2019)

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. Como ejemplo se tienen las formas básicas en la figura 2.18.



**Figura. 2.18.:** 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. (Ortiz Zamora, 2002).

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. (Ortiz Zamora, 2002).

Entonces podemos definir matemáticamente a la erosión como en la ecuación 2.25:

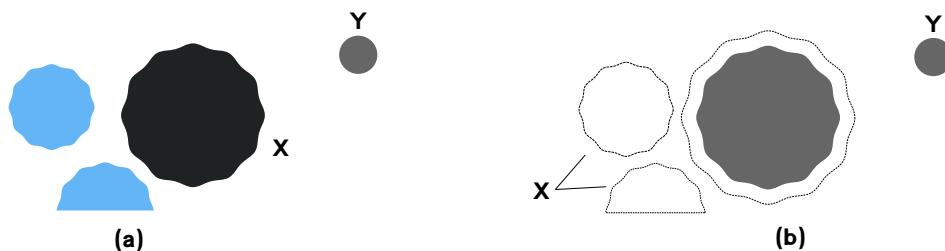
$$X \Theta \tilde{B} = \{x | B_x \subset X\} \quad (2.25)$$

de donde

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

Que en palabras del autor de la obra (Escalera, 2011) 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 2.25 podemos decir entonces que la erosión es más pronunciada con base en el tamaño del elemento estructural.



**Figura. 2.19.:** (a) Erosión de X por el elemento estructurante Y. (b) Los elementos conectados del conjunto X más pequeños que Y son eliminados.

De la figura 2.19 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.

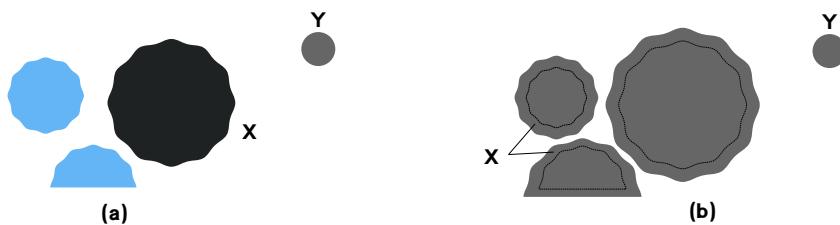
Es el crecimiento progresivo de uno de los campos (0 ó 1). Un elemento

del campo contrario a crecer será convertido si posee algún vecino perteneciente al campo que se expansiona. En caso contrario, permanecerá igual. Los elementos pertenecientes al campo a expandirán evidentemente no se modifican.(José Ramón Mejía Vilet, 2005)

Tomando en cuenta la dualidad antes mencionada, podemos obtener matemáticamente la dilatación como en la ecuación 2.26:

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

La figura 2.20 ilustra el resultado de aplicar la operación de dilatación de donde el elemento estructurante Y tiene forma de disco.



**Figura. 2.20.:** (a) Dilatación de X por el elemento estructurante Y. (b) El conjunto X aumenta su definición.

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

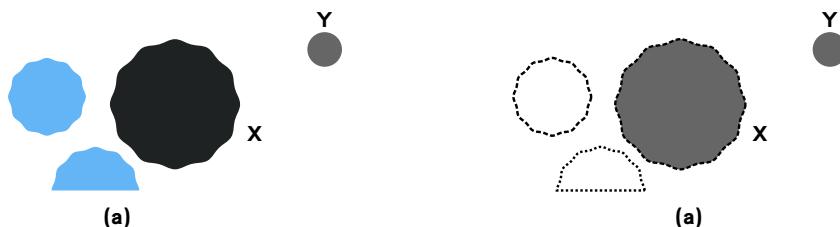
### Apertura y Cierre

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 de 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 una erosión y la última una dilatación, siempre con el mismo elemento estructural: (José Ramón Mejía Vilet, 2005). Ejemplo de comportamiento en la figura 2.21.

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

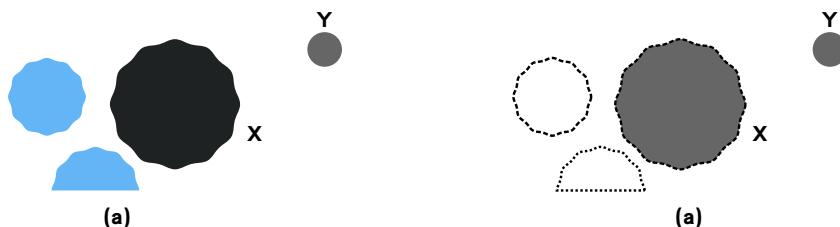


**Figura. 2.21.:** (a) Apertura morfológica del conjunto X por el elemento estructurante Y. (b) Eliminación de objetos menores en tamaño al elemento estructurante. La apertura redondea las convexidades importantes.

## Cierre

La dualidad de la operación de apertura es la del cierre(closing en literatura anglosajona), definido matemáticamente en la ec. 2.28:

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



**Figura. 2.22.:** (a) Apertura morfológica del conjunto X por el elemento estructurante Y. (b) El cierre redondea las concavidades importantes.

## 2.2.9. Contraste y brillo

Hablar de contraste y brillo es hablar de transformación de píxeles que a su vez hace referencia a otro término que aún no se ha abordado en este proyecto, que es el de operador de procesamiento de imagen. En la obra (Szeliski, 2011) 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 2.29 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 2.29 podemos acotar la ecuación 2.30 para dar paso a la ec. 2.32:

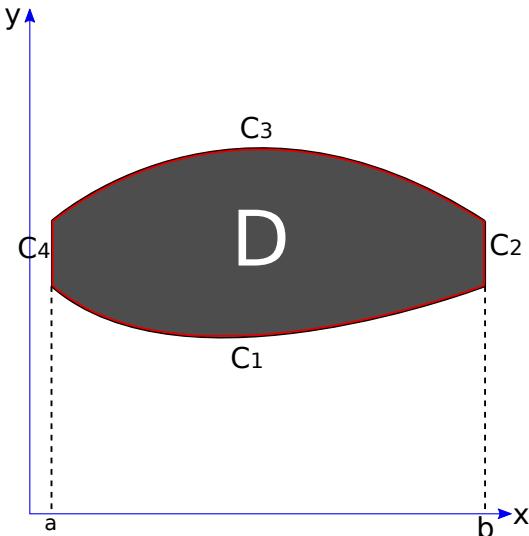
$$g(i, j) = \alpha * f(i, j) + \beta \quad (2.32)$$

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

## 2.2.10. 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 figura 2.23.



**Figura. 2.23.:** Area 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 (YANG, 1994) 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

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

De la ecuación 2.34 y suponiendo que tenemos imágenes binarias, o sea que 1 para objetos y 0 para el fondo la ecuación 2.34 se transforma en

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

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 (Flusser, 2009)  $p_{kj}(x, y) = x^k y^j$  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

## 2.3. Control

Una vez que nuestro sistema visual tiene ya la captación de imágenes y el procesamiento de la información, el siguiente paso que realiza es el envío de impulsos a través del cuerpo para poder generar movimientos. Estos movimientos podemos trasladarlos en un mecanismo con actuadores, donde dichos actuadores pueden ser válvulas, motores, pistones, etc.

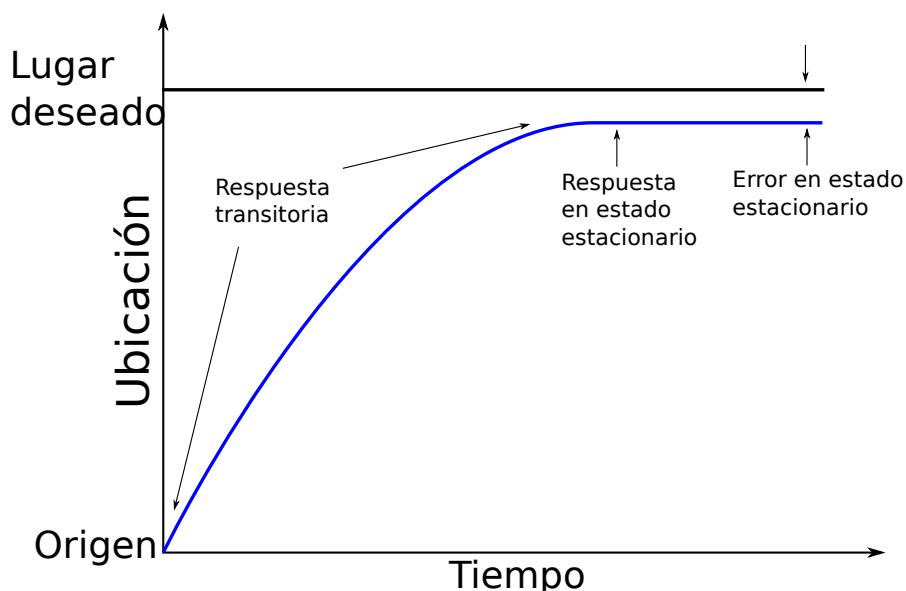
De (Nise, 2010) podemos definir que un sistema de control consiste en subsistemas y procesos (o plantas) ensamblados con el fin de obtener una salida deseada con el rendimiento deseado, dada una entrada especificada.



**Figura. 2.24.:** Diagrama de bloques de sistema de control

La figura 2.24 muestra un sistema de control en su forma más simple, donde la entrada representa una salida deseada.

A manera de ejemplo tenemos los sistemas de desarrollo de piloto automático en los automóviles de última generación, donde un usuario puede ingresar el destino deseado y el automóvil debe llevárselo. Al ubicar el punto en el mapa le damos al sistema una entrada que es la salida deseada, mostrada en la figura 2.28 y la respuesta del sistema se ilustra con la curva azul.



**Figura. 2.25.:** Diagrama de bloques de sistema de control

Dos medidas principales de rendimiento son evidentes: (1) la respuesta transitoria y (2) el error de estado estacionario. En nuestro ejemplo, la seguridad del pasajero y la paciencia del pasajero dependen de la respuesta transitoria. Si esta respuesta es demasiado rápida, se sacrifica la seguridad del pasajero; si es demasiado lento, se sacrifica la paciencia del pasajero. El error de estado estacionario es otra especificación de rendimiento importante, ya que la seguridad de llegar al destino correcto y la conveniencia del pasajero se sacrificarían si el automóvil no logra llegar al destino deseado.

### 2.3.1. Obtención de sistemas de control

Los ingenieros de control deben encontrar el modelo matemático que más se acerque al comportamiento del sistema, para poder realizar el diseño de un controlador se debe obtener primero el modelo matemático, puede estar en el dominio de la frecuencia o del tiempo, esto último es arbitrario.

El siguiente paso es desarrollar modelos matemáticos a partir de esquemas de sistemas físicos. Discutiremos dos métodos: (1) funciones de transferencia en el dominio de frecuencia y (2) ecuaciones de estado en el dominio de tiempo.

#### Función de transferencia

Esta función permitirá la separación de la entrada, el sistema y la salida en tres partes separadas y distintas, a diferencia de la ecuación diferencial. La función también nos permitirá combinar algebraicamente representaciones matemáticas de subsistemas para obtener una representación total del sistema. Esta función es útil porque nos permite conocer donde están las raíces de nuestra planta y por ende saber si es estable o no, además de que podemos diseñar un controlador partiendo de esta función.

Comencemos escribiendo una ecuación diferencial general de orden n, lineal, invariante en el tiempo siguiendo el procedimiento propuesto en la obra de (Nise, 2010)

$$a_n \frac{d^n c(t)}{dt^n} + a_{n-1} \frac{d^{n-1} c(t)}{dt^{n-1}} + \cdots + a_0 c(t) = b_m \frac{d^m r(t)}{dt^m} + b_{m-1} \frac{d^{m-1} r(t)}{dt^{m-1}} + \cdots + b_0 r(t) \quad (2.37)$$

Donde  $c(t)$  es la salida,  $r(t)$  es la entrada y las  $a'_i s$ ,  $b'_i s$  representan al sistema. Tomando la transformada de Laplace de ambos lados.

$$\begin{aligned} a_n s^n C(s) + a_{n-1} s^{n-1} C(s) + \cdots + a_0 C(s) + C.Ic(t) \\ = b_m S^m R(s) + b_{m-1} S^{m-1} R(s) + \cdots + b_0 R(s) + C.Ir(t) \end{aligned} \quad (2.38)$$

Donde C.I es la condición inicial. Pero si asumimos que todas las condiciones iniciales son cero, por lo tanto, la ecuación anterior se reduce a

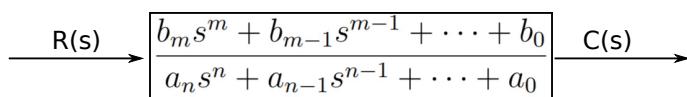
$$(a_n s^n + a_{n-1} s^{n-1} + \dots + a_0)C(s) = (b_m s^m + b_{m-1} s^{m-1} + \dots + b_0)R(s) \quad (2.39)$$

Y ahora obteniendo la relación de salida respecto a la entrada nos queda la siguiente ecuación.

$$\frac{C(s)}{R(s)} = G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0} \quad (2.40)$$

La ecuación anterior separa la salida de la entrada y nos da una relación entre ellas, llamamos a esta relación,  $G(s)$ , la función de transferencia y la evaluamos con condiciones iniciales cero.

La función de transferencia puede ser representada en un diagrama de bloques tal y como se ilustra en la siguiente figura.



**Figura. 2.26.:** Diagrama de bloques de la función de transferencia

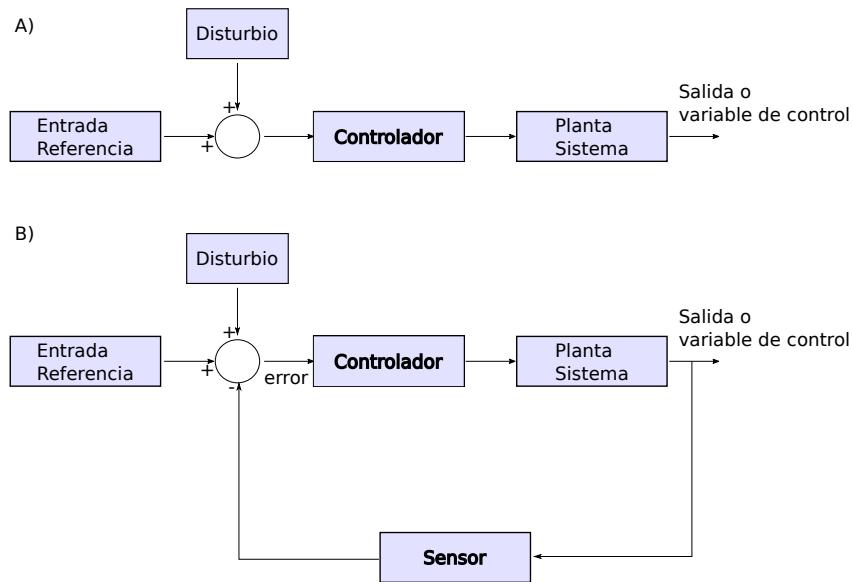
Observe que el denominador de la función de transferencia es idéntico al polinomio característico de la ecuación diferencial. Además, podemos encontrar la salida,  $C(s)$  usando

$$C(s) = R(s)G(s) \quad (2.41)$$

### 2.3.2. Lazo abierto y cerrado

Para poder tener la seguridad que nuestro sistema va a hacer lo que tiene que hacer, es importante aplicar un controlador al sistema este variará dependiendo de la aplicación y de los recursos disponibles.

La teoría de control se basa en dos principales ramas, de lazo abierto y lazo cerrado así como se ilustra en la figura 2.27.



**Figura. 2.27.: (a) Lazo abierto (b) Lazo cerrado**

La gran diferencia entre las dos maneras de control es la incorporación de un sensor que nos permite leer la salida del sistema y retroalimentar con la entrada, de esta manera se genera el error entre la salida deseada con la salida real. Por otra parte el lazo abierto solo nos permite tener una entrada no retroalimentada.

Para propósitos específicos de este proyecto vamos a emplear un sistema de control en lazo cerrado porque necesitamos controlar a partir del error generado por la diferencia de coordenadas del centroide de la figura a seguir.

### 2.3.3. Respuesta en el tiempo

La respuesta de salida de un sistema es la suma de dos respuestas: la respuesta forzada y la respuesta natural.

El uso de polos y ceros y su relación con la respuesta temporal de un sistema es una técnica de control. Aprender esta relación nos da un "Manejo cualitativo" de los problemas. El concepto de polos y ceros, fundamental para el análisis y diseño de sistemas de control, simplifica la evaluación de la respuesta de un sistema.(Nise, 2010)

## Polos de la función de transferencia

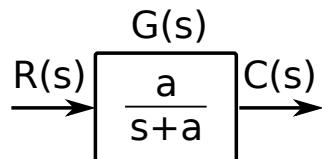
Los polos de una función de transferencia son (1) los valores de la variable de la transformada de Laplace,  $s$ , que hacen que la función de transferencia se vuelva infinita o (2) cualquier raíz del denominador de la función de transferencia que sea común a las raíces del numerador.

## Ceros en la función de transferencia

Los ceros de una función de transferencia son (1) los valores de la variable de la transformada de Laplace,  $s$ , que hacen que la función de transferencia sea cero, o (2) cualquier raíz del numerador de la función de transferencia que sea común a las raíces del denominador.

## Sistemas de primer orden

Un sistema de primer orden sin ceros puede describirse mediante la función de transferencia que se muestra en la figura 2.28



**Figura. 2.28.:** Función de transferencia de un sistema de primer orden

Si la entrada es un escalón unitario, es decir,  $R(s) = \frac{1}{s}$ , la salida que obtenemos es

$$C(s) = R(s)G(s) = \frac{a}{s(s+a)} \quad (2.42)$$

Aplicando la inversa de laplace de la ecuación 2.42, la respuesta escalón esta dada por

$$c(t) = 1 - e^{-at} \quad (2.43)$$

Examinemos el significado del parámetro  $a$ , el único parámetro necesario para describir la respuesta transitoria. Cuando  $t = 1/a$

$$e^{at}|_{t=1/a} = e^{-1} = 0,37 \quad (2.44)$$

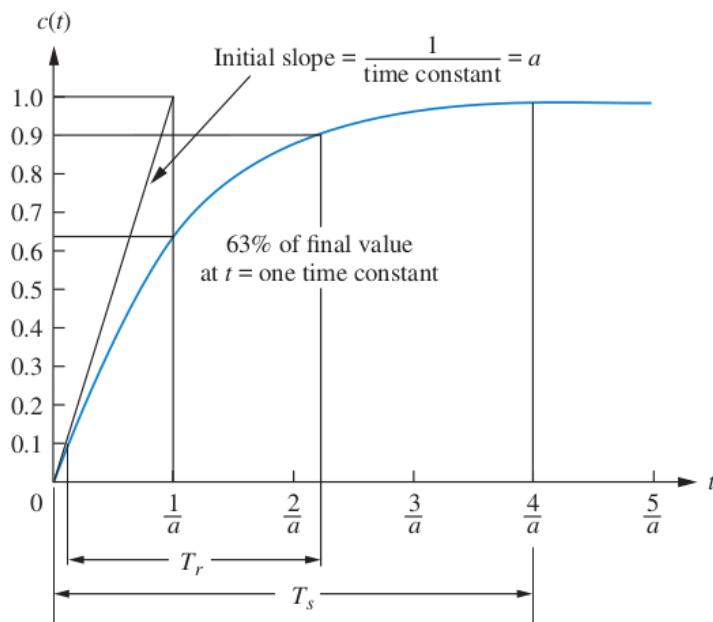
o

$$c(t)|_{t=1/a} = 1 - e^{-at}|_{t=1/a} = 1 - 0,37 = 0,63 \quad (2.45)$$

Ahora utilizamos las Ec. (2.43), (2.44) y (2.45) para definir tres especificaciones de rendimiento de respuesta transitoria: tiempo constante, tiempo de subida y tiempo de asentamiento.

#### ■ Constante de tiempo

Llamamos  $1/a$  a la constante de tiempo de la respuesta. A partir de la Ec. (2.44), la constante de tiempo puede describirse como el tiempo para que  $e^{-at}$  decaiga hasta el 37 % de su valor inicial. Alternativamente, a partir de la Ec. (2.44) la constante de tiempo es el tiempo que le toma a la respuesta escalón elevarse al 63 % de su valor final.



**Figura. 2.29.:** Respuesta de un Sistema de Primer orden a un escalón unitario, tomada de (Nise, 2010)

#### ■ Tiempo de subida

En la figura 2.33 la sección  $T_r$  hace referencia al tiempo de subida que se define como el tiempo para que la forma de onda pase de 0,1 a 0,9 de su

valor final. El tiempo de subida se encuentra resolviendo la Ec. (2.45) para la diferencia de tiempo en  $c(t) = 0.9$  y  $c(t)=0.1$ . Por lo tanto,

$$T_r = \frac{2,31}{a} - \frac{0,11}{a} = \frac{2,2}{a} \quad (2.46)$$

- Tiempo de asentamiento

En la figura 2.29  $T_s$  hace referencia al tiempo de asentamiento que se define como el tiempo para que la respuesta alcance, y se mantenga dentro, del 2 % de su valor final. Dejando  $c(t)=0.98$  en la Ec. (2.44) y resolviendo para el tiempo,  $t$ , encontramos que el tiempo de asentamiento es

$$T_s = \frac{4}{a} \quad (2.47)$$

### Sistemas de segundo orden

La mayoría de los sistemas de control se basan en la obtención de su sistema dinámico que se encuentra en el orden cuadrático, siendo este donde se realizan las diversas técnicas de diseño de control, por lo que si el sistema es mayor a segundo orden, lo que se hace es dividir el sistema para poder verlo en pedazos de segundo orden.

La forma general de una planta de segundo orden es la siguiente

$$\frac{b}{s^2 + as + b} \quad (2.48)$$

# Modelo matemático

“ En matemáticas no entiendes las cosas, te acostumbras a ellas.

— John von Neumann  
(Físico matemático)

En el presente capítulo se aborda el modelo matemático que fundamenta el diseño de control en el capítulo 5, pasando por la definición de los marcos de referencia del sistema, la obtención de sistema dinámico con base en investigaciones de otros autores.

## 3.1. Marco de referencia

Hay un concepto clave que ayuda establecer la bases para la comprensión del movimiento de un cuerpo rígido, que es el de marco de referencia.

En la figura 3.1 asociamos con cualquier posición y orientación un marco de referencia

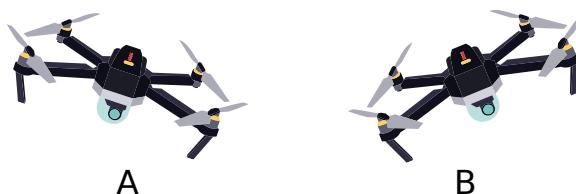


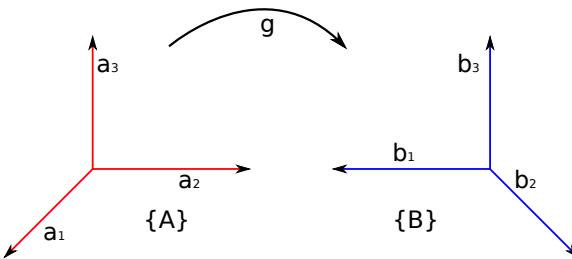
Figura. 3.1.: Drone con diferente grado de orientación

En el marco A de la figura 3.2, podemos encontrar 3 vectores linealmente independientes  $a_1$ ,  $a_2$  y  $a_3$  y de la misma manera en el marco B tenemos tres

vectores  $b_1$ ,  $b_2$  y  $b_3$ , podemos escribir cualquier vector como una combinación lineal del vector base en cualquier marco:

$$\mathbf{v} = v_1 \mathbf{a}_1 + v_2 \mathbf{a}_2 + v_3 \mathbf{a}_3 \quad (3.1)$$

Tales vectores pueden ser vistos como ortogonales y si aplicamos una rotación del marco A la transformación g queda ilustrada en la figura 3.2.



**Figura. 3.2.:** Desplazamiento cuerpo rígido

Podemos escribir ambos vectores ortogonales unitarios en un solo marco de referencia como una combinación lineal de ambos

$$\mathbf{b}_1 = R_{11} \mathbf{a}_1 + R_{12} \mathbf{a}_2 + R_{13} \mathbf{a}_3 \quad (3.2a)$$

$$\mathbf{b}_2 = R_{21} \mathbf{a}_1 + R_{22} \mathbf{a}_2 + R_{23} \mathbf{a}_3 \quad (3.2b)$$

$$\mathbf{b}_3 = R_{31} \mathbf{a}_1 + R_{32} \mathbf{a}_2 + R_{33} \mathbf{a}_3 \quad (3.2c)$$

Para este caso se escoge  $\mathbf{b}_1$ ,  $\mathbf{b}_2$  y  $\mathbf{b}_3$  como la combinación lineal para  $\mathbf{a}_1$ ,  $\mathbf{a}_2$  y  $\mathbf{a}_3$  y los coeficientes de R se pueden acomodar en un arreglo de 3x3 llamada matriz de rotación.

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (3.3)$$

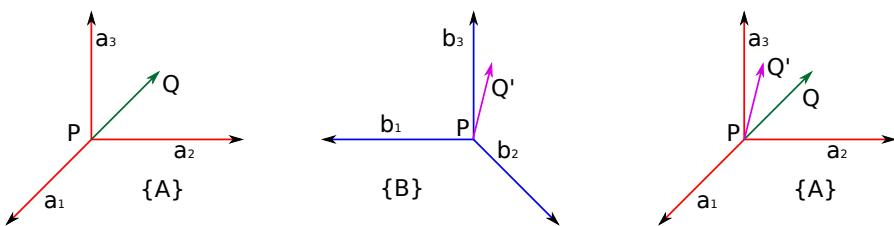
Y las propiedades de esta matriz son las siguientes

- Ortogonal
- Su determinante es +1

- El producto de dos matrices de rotación da como resultado otra matriz de rotación
- La inversa de una matriz de rotación es también una matriz de rotación

La importancia de las matrices de rotación radican en que con ayuda de software y hardware podemos aplicar técnicas de mapeo y simulación, además de que para el sistema de la gimbal entra en juego múltiples marcos de referencia que constantemente cambian su orientación y desplazamiento.

Ahora bien si trazamos un vector Q en cada marco de referencia como en la figura 3.3



**Figura. 3.3.: Vectores de rotación**

Si trasladamos el marco B en A podremos obtener de nuevo una combinación lineal pero con diferentes conjuntos de coeficientes

$$\overrightarrow{PQ} = q_1 \mathbf{a}_1 + q_2 \mathbf{a}_2 + q_3 \mathbf{a}_3 \quad (3.4)$$

$$\overrightarrow{PQ'} = q'_1 \mathbf{a}_1 + q'_2 \mathbf{a}_2 + q'_3 \mathbf{a}_3 \quad (3.5)$$

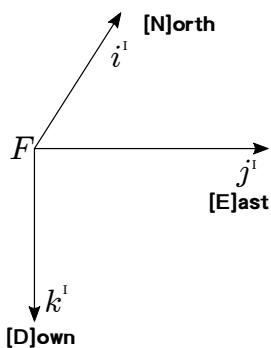
La matriz que conecta  $q_1, q_2$  y  $q_3$  con  $q'_1, q'_2$  y  $q'_3$  es la matriz de rotación vista en la ecuación 3.3

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} q'_1 \\ q'_2 \\ q'_3 \end{bmatrix} \quad (3.6)$$

Dicha matriz nos dice como se transforman los vectores de un marco de referencia a otro. La matrices de rotación nos van a servir, como se mencionó anteriormente, para poder hacer mapeos del entorno, por lo que necesitamos primero definir los marcos de referencia inercial y del cuerpo para posteriormente hacer sus matrices de transformación.

### 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 Figura 3.4 ilustra los ejes de este marco.

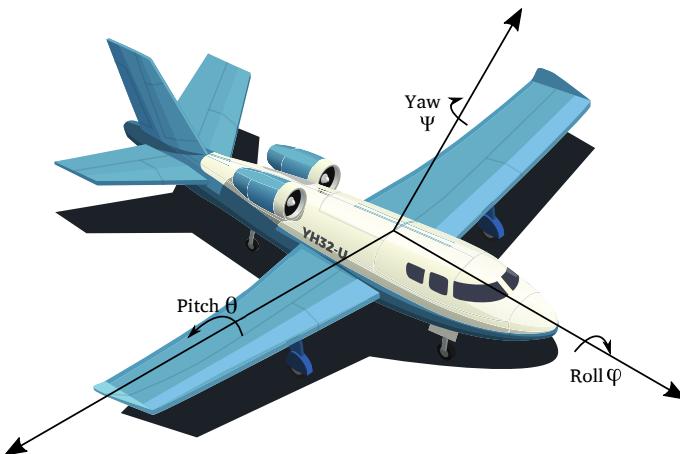


**Figura. 3.4.:** Marco de referencia inercial

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).

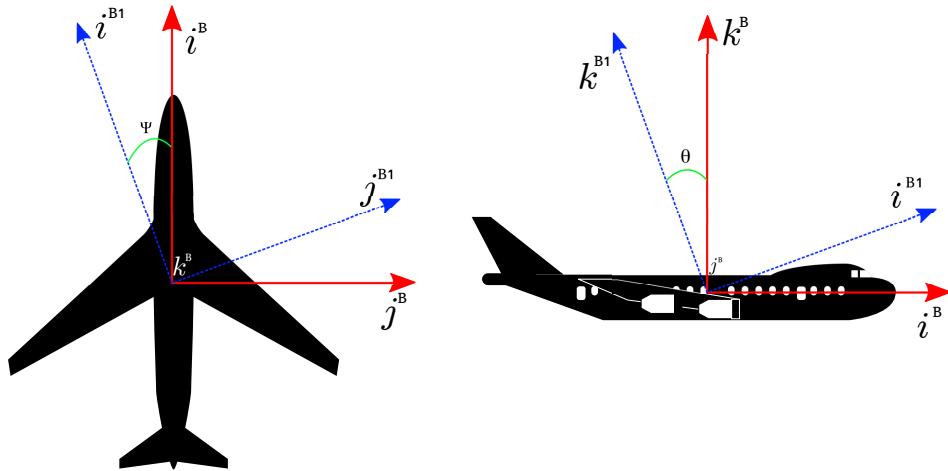
### 3.1.2. Marco de referencia del cuerpo [B]

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, dichas rotaciones pueden ser mejor detalladas en una imagen, por lo que en la ilustración 3.5 se grafica un plano 2D y se asigna nombre a cada rotación de los ejes.



**Figura. 3.5.:** Rotación en 3 ejes.Tomada de freepik

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 figura 3.6.

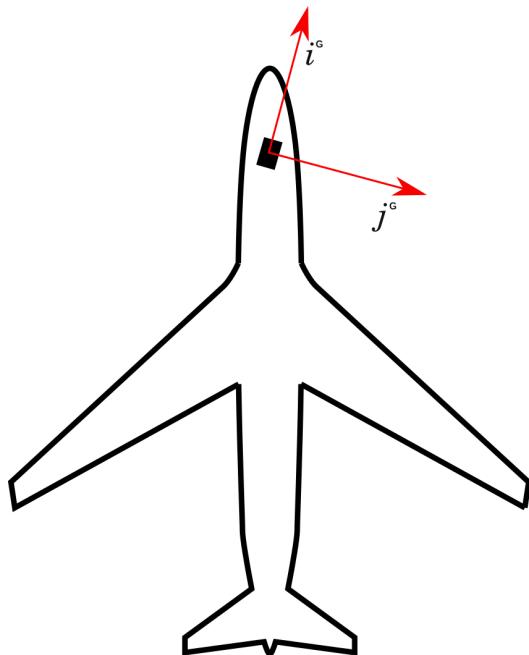


**Figura. 3.6.:** Marco de referencia del cuerpo.

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.

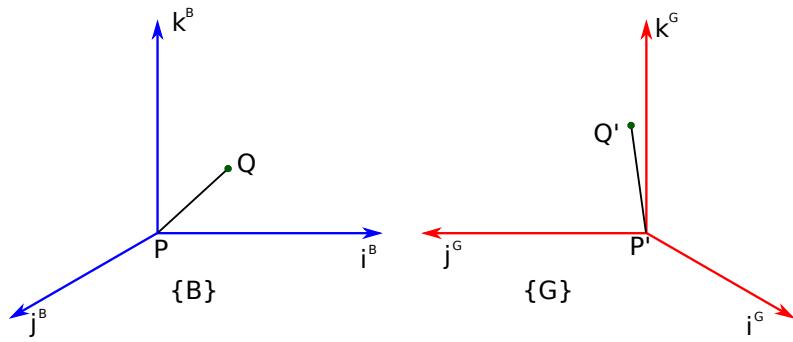


**Figura. 3.7.:** Marco de referencia del la cámara.

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. Matrices de rotación

Considerando el marco de referencia del cuerpo(B) y el de la Gimbal(G) tenemos los puntos Q y Q' Como se ilustra en la figura 3.8.



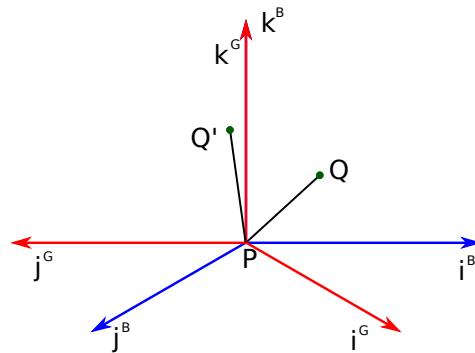
**Figura. 3.8.:** Marco de referencia Body y gimbal

Que podemos expresarlo en una combinación lineal para  $\overrightarrow{PQ}$  y para  $\overrightarrow{P'Q'}$

$$PQ = q_1 i^B + q_2 j^B + q_3 k^B \quad (3.7a)$$

$$P'Q' = q_1 i^G + q_2 j^G + q_3 k^G \quad (3.7b)$$

Trasladamos el marco de referencia del gimbal en el del cuerpo



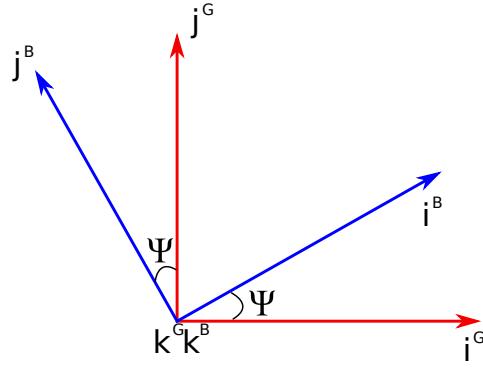
**Figura. 3.9.:** Marco de referencia del gimbal trasladado en el del cuerpo

Ahora tenemos un punto  $Q''$  que es el punto  $Q'$  trasladado en el marco de referencia del cuerpo

$$PQ'' = q_1 i^B + q_2 j^B + q_3 k^B = q_1'' i^G + q_2'' j^G + q_3'' k^G \quad (3.8)$$

$$\begin{bmatrix} q_1'' \\ q_2'' \\ q_3'' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (3.9)$$

La rotación sobre el eje  $k$  se ilustra en la figura 3.10.



**Figura. 3.10.:** Rotación en el eje k

Se obtiene el cambio para  $j^B$  y  $i^B$

$$i^B = |i^B| \cos(\psi) i^G + |i^B| \sin(\psi) j^G = \cos(\psi) i^G + \sin(\psi) j^G \quad (3.10)$$

$$j^B = -|j^B| \sin(\psi) i^G + |j^B| \cos(\psi) j^G = -\sin(\psi) i^G + \cos(\psi) j^G \quad (3.11)$$

Y de la ecuación 3.8 sustituimos  $j^B$  y  $i^B$

$$PQ'' = q_1(\cos(\psi) i^G + \sin(\psi) j^G) + q_2(-\sin(\psi) i^G + \cos(\psi) j^G) + q_3(k^G) \quad (3.12)$$

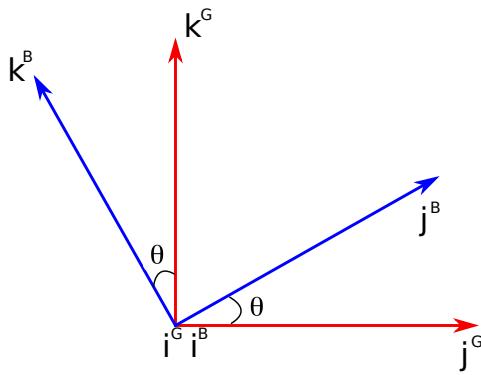
Simplificando la ecuación 3.12 tenemos

$$PQ'' = (q_1 \cos(\psi) - q_2 \sin(\psi)) i^G + (q_1 \sin(\psi) + q_2 \cos(\psi)) j^G + q_3(k^G) \quad (3.13)$$

Y acomodando la ecuación 3.13 en matrices tenemos

$$\begin{bmatrix} q_1'' \\ q_2'' \\ q_3'' \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (3.14)$$

Ahora si rotamos sobre el eje x como se muestra en la figura 3.11



**Figura. 3.11.:** Rotación en el eje x

Se obtiene el cambio para  $j^B$  y  $k^B$

$$j^B = |j^B| \cos(\theta) j^G + |j^B| \sin(\theta) k^G = \cos(\theta) j^G + \sin(\theta) k^G \quad (3.15)$$

$$k^B = -|k^B| \sin(\theta) j^G + |k^B| \cos(\theta) k^G = -\sin(\theta) j^G + \cos(\theta) k^G \quad (3.16)$$

Y de la ecuación 3.8 sustituimos  $j^B$  y  $k^B$

$$PQ'' = q_1(i^G) + q_2(\cos(\theta) j^G + \sin(\theta) k^G) + q_3(-\sin(\theta) j^G + \cos(\theta) k^G) \quad (3.17)$$

Simplificando la ecuación 3.18 tenemos

$$PQ'' = q_1(i^G) + (q_2 \cos(\theta) - q_3 \sin(\theta)) j^G + (q_2 \sin(\theta) + q_3 \cos(\theta)) k^G \quad (3.18)$$

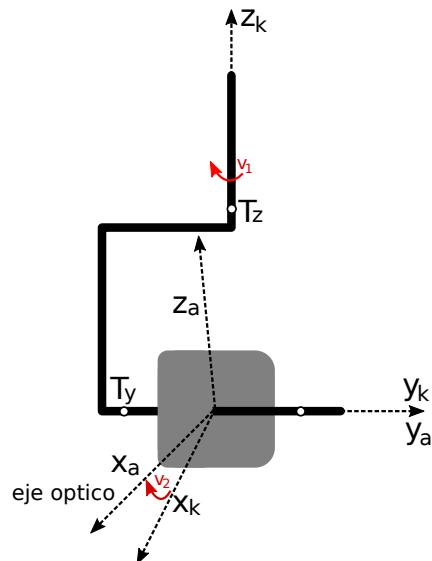
Y acomodando la ecuación 3.18 en matrices tenemos

$$\begin{bmatrix} q_1'' \\ q_2'' \\ q_3'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (3.19)$$

### 3.3. Ecuaciones de movimiento de una gimbal

Tres marcos de referencia han sido introducidos en este capítulo; el del cuerpo[B], el de la gimbal[G] y el inercial[I]. Podemos dividir el marco de

referencia de la gimbal en dos: Yaw[K] y Pitch[P], tal y como se ilustra en la figura 3.12.



**Figura. 3.12.:** Sistema gimbal de dos grados de libertad

El eje  $x_a$  coincide con el eje óptico de la cámara, el centro de rotación está en el origen del marco de referencia, que asumimos es el mismo para los tres marcos.

Podemos expresar las velocidades angulares de los marcos de referencia B,K y A en matrices de 3x1.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}; \begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix}; \begin{bmatrix} p_a \\ q_a \\ r_a \end{bmatrix} \quad (3.20)$$

Y el tensor de inercia para Pitch lo definimos como:

$$J_A = \begin{bmatrix} J_{ax} & D_{xy} & D_{xz} \\ D_{xy} & J_{ay} & D_{yz} \\ D_{xz} & D_{yz} & J_{az} \end{bmatrix} \quad (3.21)$$

Mientras que para Yaw tenemos

$$J_K = \begin{bmatrix} J_{kx} & d_{xy} & d_{xz} \\ d_{xy} & J_{ky} & d_{yz} \\ d_{xz} & d_{yz} & J_{kz} \end{bmatrix} \quad (3.22)$$

Para el signo de los productos de inercia, seguimos la definición de (Goldstein, 1980); en consecuencia, no se utilizan explícitamente signos negativos en las matrices. Se supone que el centro de masa de la gimbal está en el centro de rotación común, es decir, asumimos que la gimbal no tienen desequilibrio de masa.

En la figura 3.12 se ilustra los dos torques de entrada del sistema

$T_y$  = El torque alrededor del eje  $Y_a$

$T_z$  = El torque alrededor del eje  $Z_k$

### 3.3.1. Rotación sobre Pitch

La ecuación básica del movimiento en Pitch de la gimbal se obtiene directamente si pensamos en la gimbal como un cuerpo rígido por sí mismo y consideramos el hecho de que no hay desequilibrio, y recordando que el momento de Inercia es la resistencia que presenta un cuerpo a ser acelerado en rotación podemos hacer una analogía de la segunda ley de Newton equivalente para la rotación.

$$\tau = I\alpha \quad (3.23)$$

Donde

- $\tau$  es el torque Aplicando.
- $I$  es el momento de inercia.
- $\alpha$  es la aceleración angular

Recordando que el momento angular puede ser calculado mediante la ecuación

$$H = J\omega \quad (3.24)$$

Por lo que para Pitch tenemos

$$H_A = \begin{bmatrix} J_{ax}p_a + D_{xy}q_a + D_{yz}r_a \\ D_{xy}p_a + J_{ay}q_a + D_{yz}r_a \\ D_{xz}p_a + D_{yz}q_a + J_{az}r_a \end{bmatrix} = \begin{bmatrix} H_{ax} \\ H_{ay} \\ H_{az} \end{bmatrix} \quad (3.25)$$

La ecuación de momento para un marco giratorio es

$$T = \frac{dH}{dt} + \omega x H \quad (3.26)$$

Aplicando la derivada de (3.25) y el producto cruz de (3.20) con (3.25) la suma es igual a la ecuación (3.27)

$$T = \begin{bmatrix} \dot{H}_{ax} + q_a H_{az} - r_a H_{ay} \\ \dot{H}_{ax} + r_a H_{ax} - p_a H_{az} \\ \dot{H}_{ax} + q_a H_{ay} - q_a H_{ax} \end{bmatrix} \quad (3.27)$$

De la ecuación(3.27) y (3.23) llegamos a la siguiente expresión

$$J_{ay}\dot{q}_a = T_y + (J_{az} - J_{ax})p_a r_a + D_{xz}(p_a^2 - r_a^2) - D_{yz}(\dot{r}_a - p_a q_a) - D_{xy}(\dot{p}_a + q_a r_a) \quad (3.28)$$

Es una ecuación diferencial para la velocidad angular para Pitch  $q_a$  donde se incluye torque externo  $T_y$  velocidades angulares  $p_a q_a r_a$ . Como se presenta en el trabajo de (Yoon, 2001)

Desde el punto de vista de la estabilización, los "términos de inercia"de la derecha en la expresión 3.28 representan perturbaciones no deseadas. Entrarán en el sistema de control en el mismo punto que el torque externo; por consiguiente, pueden considerarse como perturbaciones del torque.

Vamos a suponer que los productos de la inercia pueden ser despreciados

$$D_{xy} = D_{xz} = D_{yz} = 0 \quad (3.29)$$

Asumiendo además que

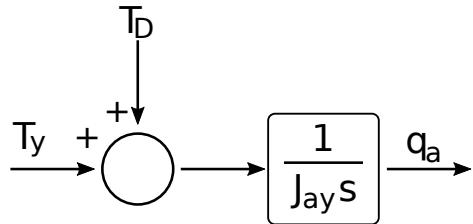
$$J_{ax} = J_{az} \quad (3.30)$$

La ecuación (3.28) se convierte en

$$J_{ay}\dot{q}_a = T_y \quad (3.31)$$

Si no tenemos en cuenta los torques de perturbación externa,  $T_y$  representa solo el par motor. Entonces, (3.31) es la ecuación de movimiento deseada, en el sentido de que no hay perturbaciones que influyan en el movimiento sobre el eje Pitch. No importa qué movimientos de cuerpo o de gimbal tengamos, la velocidad angular del gimbal  $q_a$  no se ve afectada.

La ecuación (3.28) puede ser representada con un diagrama de bloques como se ilustra en la figura 3.13



**Figura. 3.13.:** Representación de la ecuación de movimiento en el eje Pitch

Donde el término  $\tau_D$  entra como disturbio al sistema

$$\tau_D = (J_{az} - J_{ax})p_a r_a + D_{xz}(p_a^2 - r_a^2) - D_y z(\dot{r}_a - p_a q_a) - D_{xy}(\dot{p}_a + q_a r_a) \quad (3.32)$$

### 3.3.2. Rotación sobre Yaw

En la figura 3.12 podemos observar que B coincide con K para  $v_1 = 0$ . Asociado con esas rotaciones tenemos las siguientes transformaciones, donde la transformación de K respecto a B fue antes descrita en la ecuación (3.14)

$$L_{KB} = \begin{bmatrix} \cos(v_1) & \sin(v_1) & 0 \\ -\sin(v_1) & \cos(v_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.33)$$

$$L_{AK} = \begin{bmatrix} \cos(v_2) & 0 & -\sin(v_2) \\ 0 & 1 & 0 \\ \sin(v_2) & 0 & \cos(v_2) \end{bmatrix} \quad (3.34)$$

La ecuación (3.34) representa la transformación del marco de referencia del pitch[A] respecto a yaw[K].

El momento angular del sistema es la suma del momento angular de las rotaciones de Pitch y Yaw (Yoon, 2001)

El momento angular del sistema está dada por  $J\omega$  donde J es la matriz de

inerzia y  $\omega$  es la velocidad angular, por lo tanto con las matrices (3.21) y (3.25) y las velocidades (3.20) obtenemos

$$H = \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} = \begin{bmatrix} J_{kx}p_k + d_{xy}q_k + d_{yz}r_k \\ d_{xy}p_k + J_{ky}q_k + d_{yz}r_k \\ d_{xz}p_k + d_{yz}q_k + J_{kz}r_k \end{bmatrix} + L_{AK}^T \begin{bmatrix} J_{ax}p_a + D_{xy}q_a + D_{yz}r_a \\ D_{xy}p_a + J_{ay}q_a + D_{yz}r_a \\ D_{xz}p_a + D_{yz}q_a + J_{az}r_a \end{bmatrix} \quad (3.35)$$

El momento angular de la ecuación (3.35) es la suma del yaw y pitch expresado en el marco de referencia de Yaw.

Podemos aplicar de manera similar que en Pitch las ecuaciones (3.23) (3.24) y (3.26) para encontrar el torque total, después de algunos cálculos algebraicos, cuyos detalles se omiten, y que están desarrollados en (Yoon, 2001)

$$J_k \dot{r}_k = T_z + T_{d1} + T_{d2} + T_{d3} \quad (3.36)$$

Donde

$$J_k = J_{kz} + J_{ax} \sin^2(v_2) + J_{az} \cos^2(v_2) - D_{xz} \sin(2v_2) \quad (3.37)$$

$$T_{d1} = [J_k x + J_{ax} \cos^2(v_2) + J_{az} \sin^2(v_2) + D_{xz} \sin(2v_2) - (J_{ky} + J_{ay})] p_k q_k \quad (3.38)$$

$$T_{d2} = -[d_{xz} + (J_{az} - J_{ax}) \sin(v_2) \cos(v_2) + D_{xz} \cos(2v_2)] x (\dot{p}_k - q_k r_k) \quad (3.39)$$

$$-(d_{yz} + D_{yz} \cos(v_2) - D_{xy} \sin(v_2)) (\dot{q}_k + p_k r_k)$$

$$-(d_{xy} + D_{xy} \cos(v_2) + D_{yz} \sin(v_2)) (p_k^2 - q_k^2)$$

$$T_{d3} = \ddot{v}_2 (D_{xy} \sin(v_2) - D_{yz} \cos(v_2)) \quad (3.40)$$

$$+ \dot{v}_2 [(J_{ax} - J_{az})(p_k \cos(2v_2) - r_k \sin(2v_2))]$$

$$+ 2D_{xz}(p_k \sin(2v_2) + r_k \cos(2v_2))$$

$$+(D_{yz}\sin(v_2) + D_{xy}\cos(v_2))(q_a + q_k) - J_{ay}p_k]$$

Podemos tomar a  $\tau_{d1}$ ,  $\tau_{d2}$ ,  $\tau_{d3}$  como disturbios que entran al sistema. De manera similar que en Pitch haremos algunas suposiciones, donde decimos que la expresión (3.29) y (3.30) también son aplicables para la ecuación 3.36.

$$D_{xy} = D_{xz} = D_{yz} = 0 \quad (3.41)$$

$$J_{ax} = J_{az} \quad (3.42)$$

Además, si los productos de la inercia del gimbal en Yaw también pueden ser despreciados, es decir si

$$d_{xy} = d_{xz} = d_{yz} = 0 \quad (3.43)$$

Por lo que ahora obtenemos

$$J_k = J_{kz} + J_{az} \quad (3.44)$$

$$T_{d1} = [J_{kx} + J_{ax} - (J_{ky} + J_{ay})]p_kq_k \quad (3.45)$$

$$T_{d2} = 0 \quad (3.46)$$

$$T_{d3} = -J_{ay}\dot{v}_2p_k \quad (3.47)$$

Podemos incluso seguir eliminando disturbios de entrada, haciendo que

$$J_{kx} + J_{ax} = J_{ky} \quad (3.48)$$

Dando como resultado

$$T_d = T_{d1} + T_{d2} + T_{d3} \quad (3.49)$$

$$T_d = J_{ay}(p_kq_k - p_k\dot{v}_2) \quad (3.50)$$

Entre el marco de referencia B y K tenemos la rotación  $v_1$  sobre el eje z, podemos obtener de (Barfoot, 2020) la relación (3.51)

$$\begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix} = L_{KB} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{v}_1 \end{bmatrix} \quad (3.51)$$

Desarrollando (3.48) tenemos

$$p_k = p\cos(v_1) + q\sin(v_1)$$

$$q_k = -p\sin(v_1) + q\cos(v_1) \quad (3.52)$$

$$r_k = r + \dot{v}_1$$

De manera similar, entre el marco de referencia Yaw [K] y el Pitch [A] tenemos

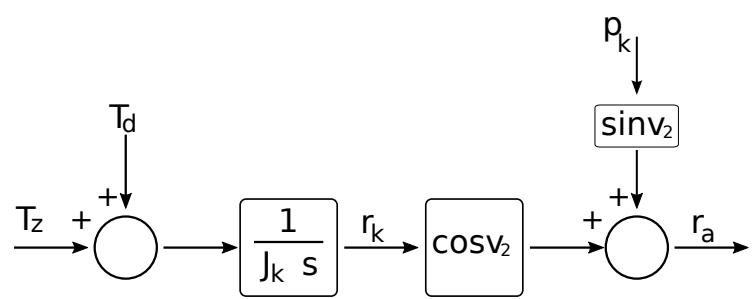
$$p_a = p_k\cos(v_2) - r_k\sin(v_2)$$

$$q_a = q_k + \dot{v}_2 \quad (3.53)$$

$$r_a = p_k\sin(v_2) + r_k\cos(v_2)$$

Hasta aquí hemos obtenido una ecuación de movimiento para Yaw en el marco de referencia K, es decir  $r_k$ . Sin embargo la salida es la velocidad angular  $r_a$  es decir la velocidad de rotación en el eje z respecto al marco de referencia A, por lo que las ecuaciones 3.53 nos ayudan a obtener la relación de  $r_a$  y  $r_k$ .

EL diagrama de bloques de la ecuación de movimiento de Yaw se puede observar en la figura 3.14



**Figura. 3.14.:** Representación de la ecuación de movimiento en Yaw



# Visión Artificial

“ Vamos a inventar el mañana, en lugar de preocuparnos por lo que sucedió ayer.

— Steve Jobs  
(Empresario y magnate americano)

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 (HardKernel, 2020b)

Que tiene las siguientes especificaciones.

- **Sensor:** CMOS sensor de imagen.
- **Lente:** Lente estándar 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.
- **Frame rate:**  
1920 x 1080 a 30fps, 1280 x 720 a45fps, 640 x 480 a30fps

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

## 4.2. Algoritmo general

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.

---

### 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 visión 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.

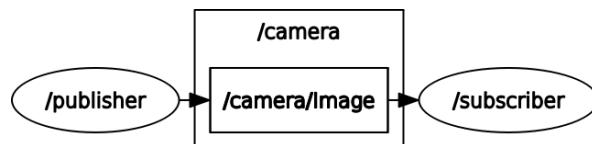


Figura. 4.2.: Nodos y topic

En la figura 4.2 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 será 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.

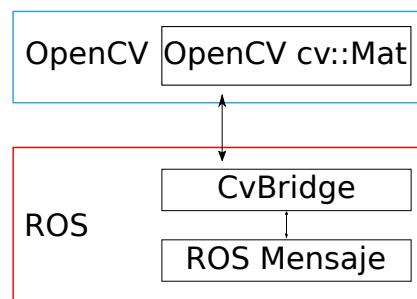


Figura. 4.3.: Comunicación entre opencv y ROS

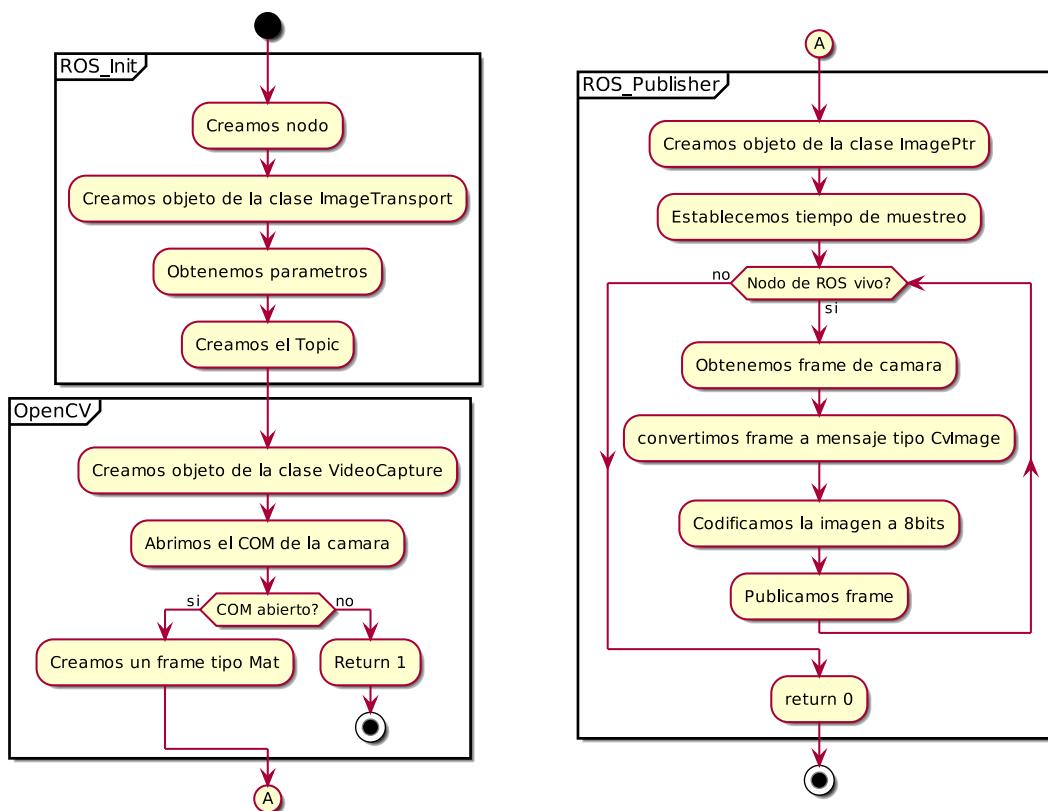
Al convertir un mensaje sensor\_msgs/Image en una imagen Cv, CvBridge reconoce dos casos de uso distintos:

- 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 diagrama de flujo de la figura 4.4:



**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 verá mas a detalle en las siguientes secciones de este capítulo.

Dichos procesos están incluidos en una clase llamada 'objectTracking' que se ilustra en el diagrama de clases de la figura 4.5.

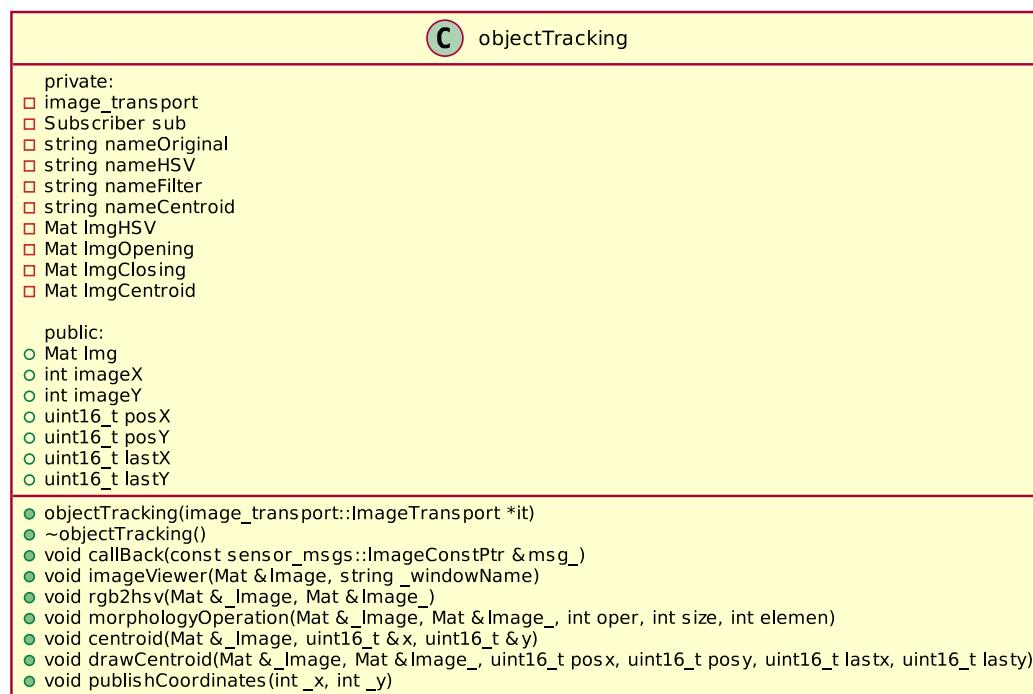


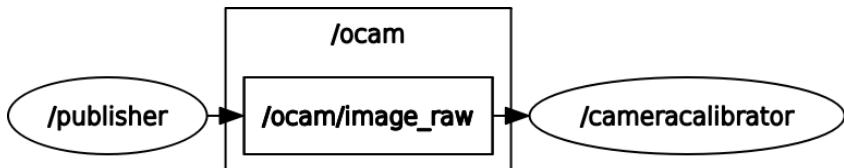
Figura. 4.5.: Diagrama de clases del subscriber

## 4.4. Calibración de cámara

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 cuadro.

En la figura 4.6 ilustra los nodos activos para el procedimiento de calibración de cámara.



**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 figura 4.7



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

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

$$D = [-0,439076 \quad 0,215870 \quad 0,001905 \quad 0,005790 \quad 0,000000] \quad (4.1)$$

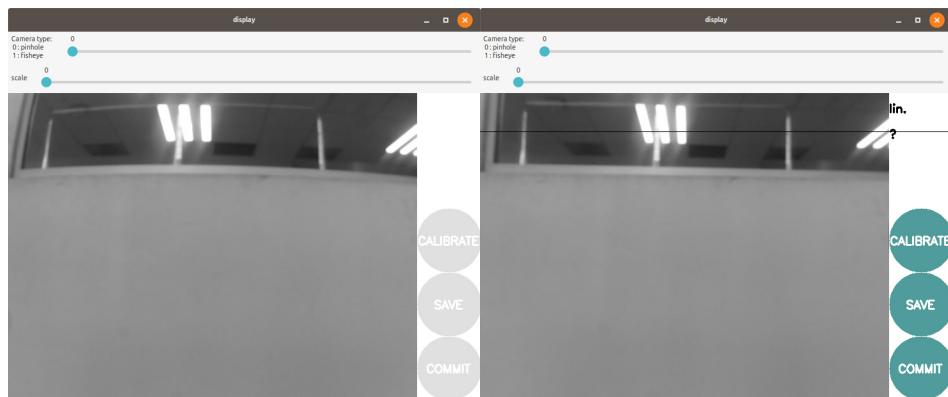
Matriz de parámetros intrínsecos de la cámara

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

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 parámetros anteriores al publicador da como resultado la figura 4.8

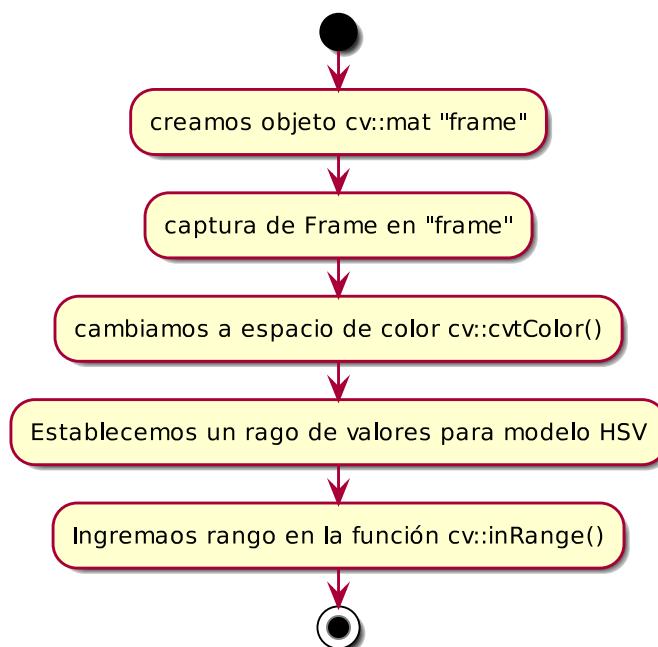


**Figura. 4.8.:** Proceso de calibración utilizando un tablero de ajedrez; corrección de distorsión

## 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 porqué 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 diagrama de la figura 4.9 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

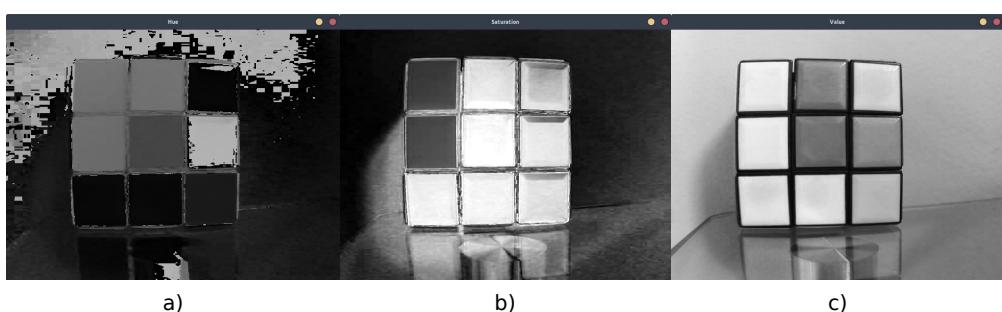
- **sdt**: Imagen de salida del mismo tamaño y profundidad que src.
- **code**: Código de conversión de espacio de color en este caso es COLOR\_BGR2HSV.
- **dstCn**: Número de canales en la imagen de destino; si el parámetro es 0, el número de canales se deriva automáticamente de src y código.

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 byte serían el segundo píxel (Azul, luego Verde, luego Rojo), y así sucesivamente.



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

En la figura 4.10 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 como se muestra en la figura 4.11.



**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()`

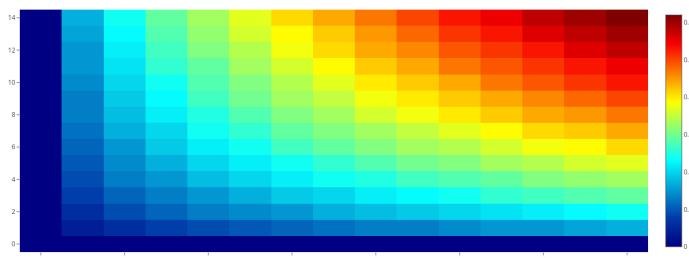
```
void cv::cvtColor(Mat &src, Mat &lowerb, Mat &upperb, Mat &dst )
```

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 figura 4.12 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 a partir del código 2 se obtuvieron los siguientes resultados.



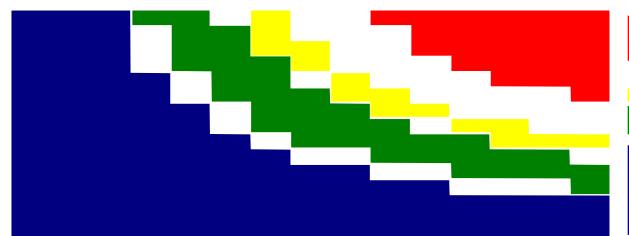
**Figura. 4.13.:** Separación de colores en 4 rangos

En la figura 4.13 se aprecia la separación de la figura 4.12 en 4 colores; Rojo, Amarillo, Verde y Azul. Con los siguientes rangos de valores.

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

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

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

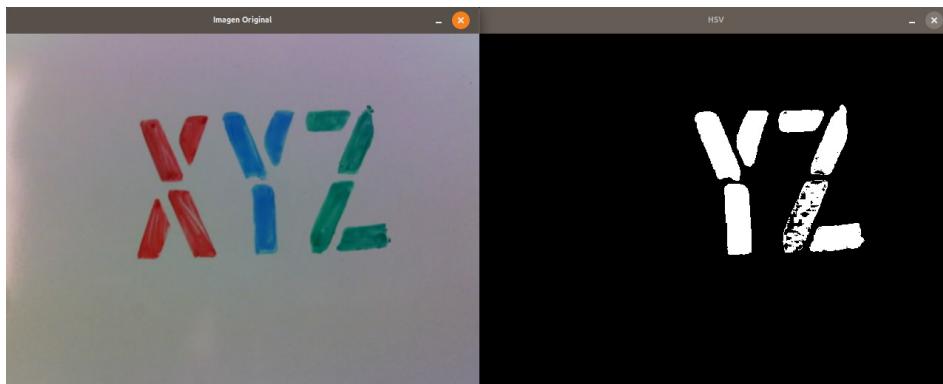


**Figura. 4.14.:** Separación de colores

#### 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.

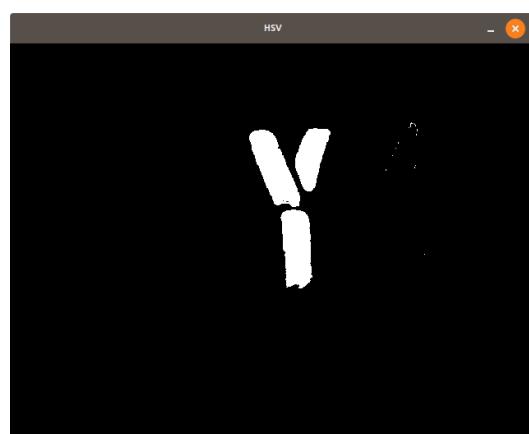
Las pruebas se hicieron con luz emitida por un foco, y con poca interacción con la energía solar, probando primero el rango de color azul.



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

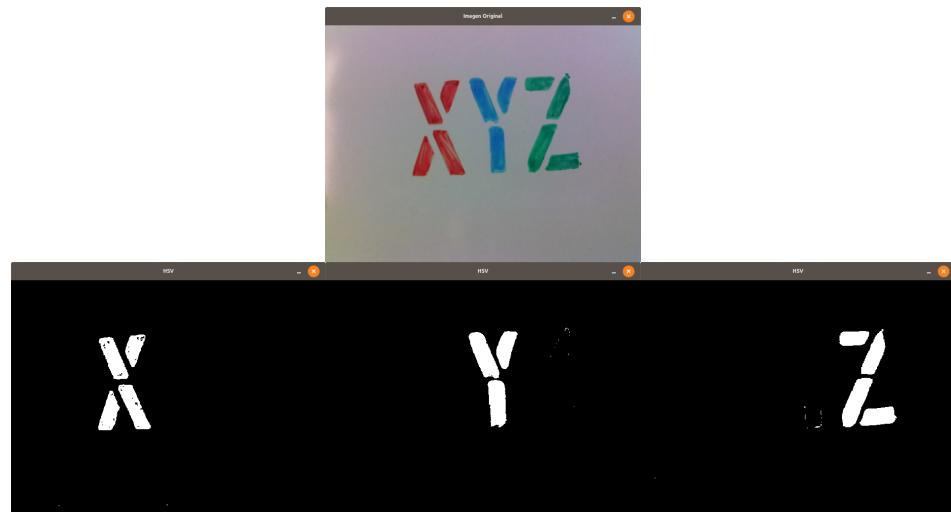
Como se puede observar en la figura 4.15 gran parte del color verde está encapsulada en lo que debería ser únicamente color azul, esto debido a que en las pruebas iniciales no se consideró la variación de iluminación en el entorno, por lo que reajustar valores se hace una tarea necesaria. Aunque los 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 figura 4.16.



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

Que al final tenemos la separación de los colores RGB como se muestra en la figura 4.17.



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

Un problema evidente 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 esté la cámara, es decir, que si hay alguna sombra que haga captar al algoritmo que el color es verde, cuando en realidad es azul, esto representa un ruido indeseado. Para las pruebas anteriores se obtuvieron de resultado los valores de la tabla 4.2.

**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 es la entrada de ruido, en las figuras 4.16 y 4.17 hay presencia de pequeño ruido tanto fuera como dentro de las letras. Esto se puede solucionar agregando un par de filtros básicos, llamados opening y closing.

## 4.6. Corrección de brillo

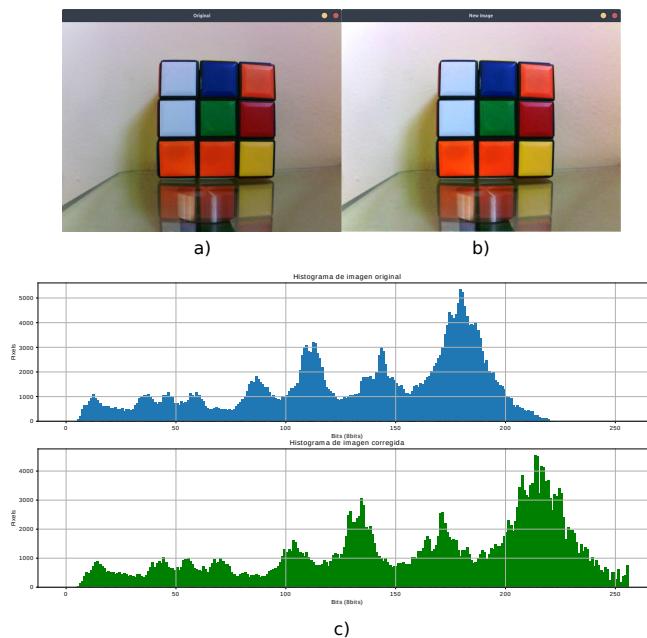
Como se pudo observar en pruebas anteriores el rango que se asigna a cada color en el modelo HSV está completamente ligado con la cantidad de iluminación de nuestra imagen, es por eso que predefinir un conjunto de valores se hace una tarea complicada, una solución sería poner 3 barras de interfaz para 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 ecuación 4.4

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

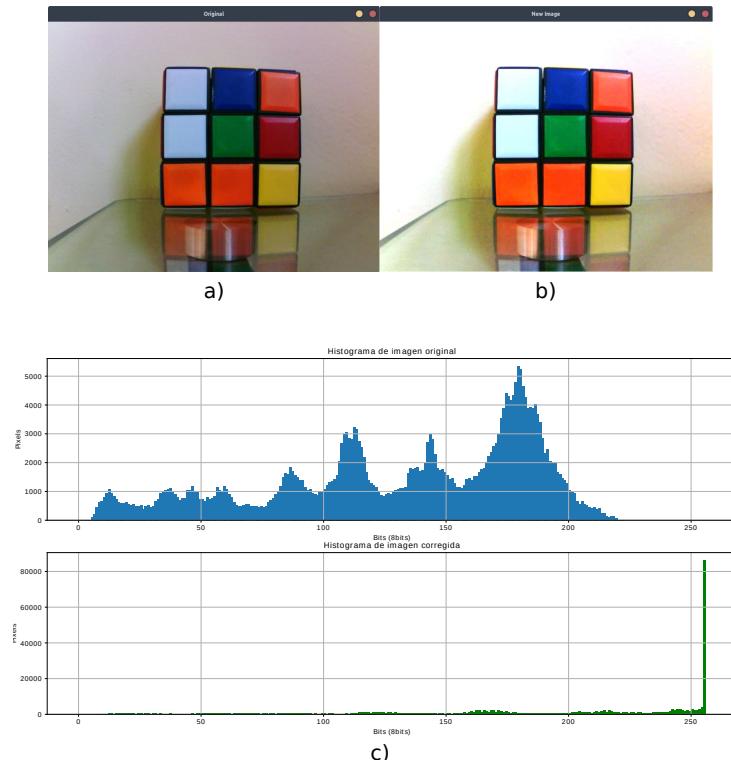
Donde  $\alpha$  es el encargado de ajustar el contraste, mientras que  $\beta$  se encarga del brillo. Los valores de  $\alpha$  van en el rango de 1 a 3, de tipo float y los de  $\beta$  van de 0 a 100 de tipo entero.



**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 4.18 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.

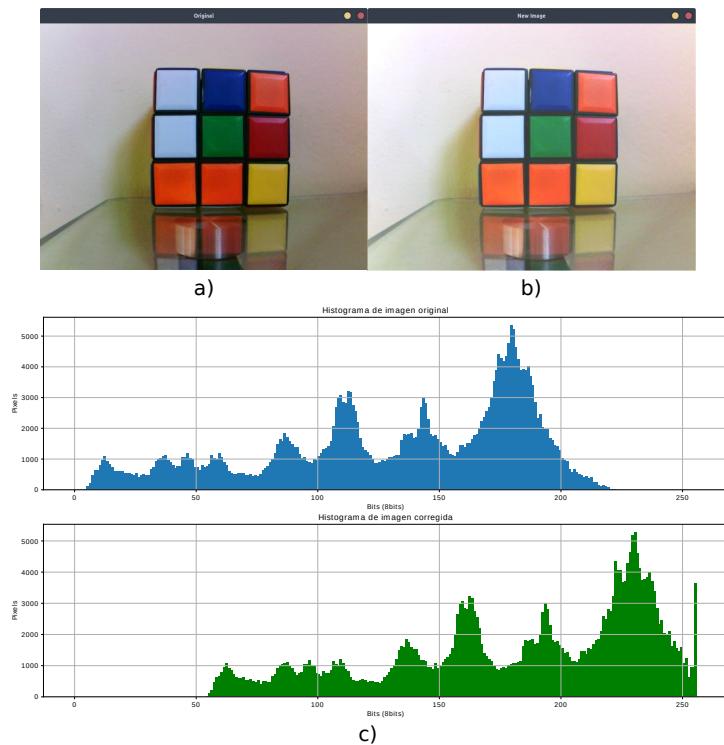


**Figura. 4.19.:** a) Imagen captada por la cámara; b) Corrección de contraste en 1.5; c) Histograma de a y b

Ahora observamos en la figura 4.19 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 blanco. Esto es con respecto a  $\alpha$ , pero ahora veamos que pasa si modificamos el brillo.

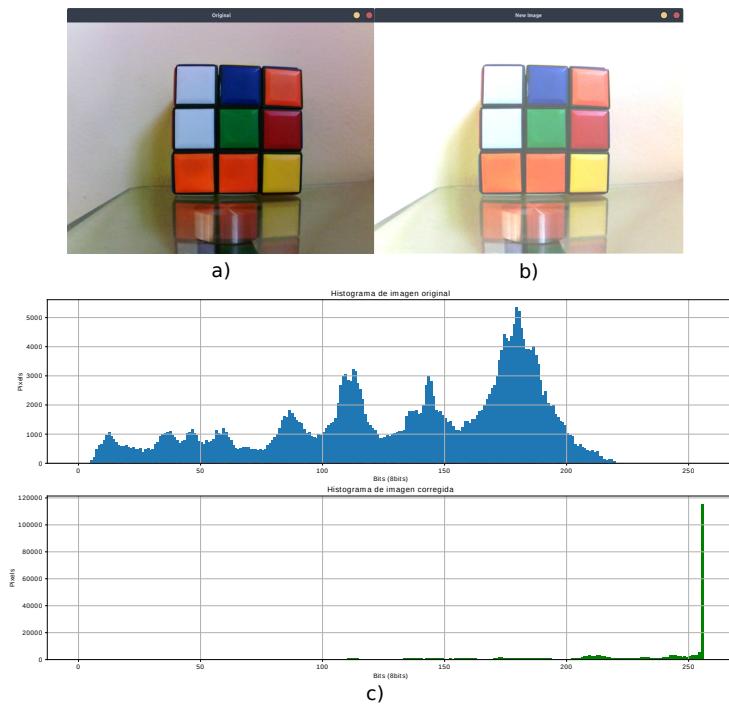
En la figura 4.20 mantenemos el valor de contraste en 0 pero el brillo lo modificamos a 50.

Como se puede apreciar la imagen de la figura 4.20 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 mostrada en la figura 4.21.



**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.

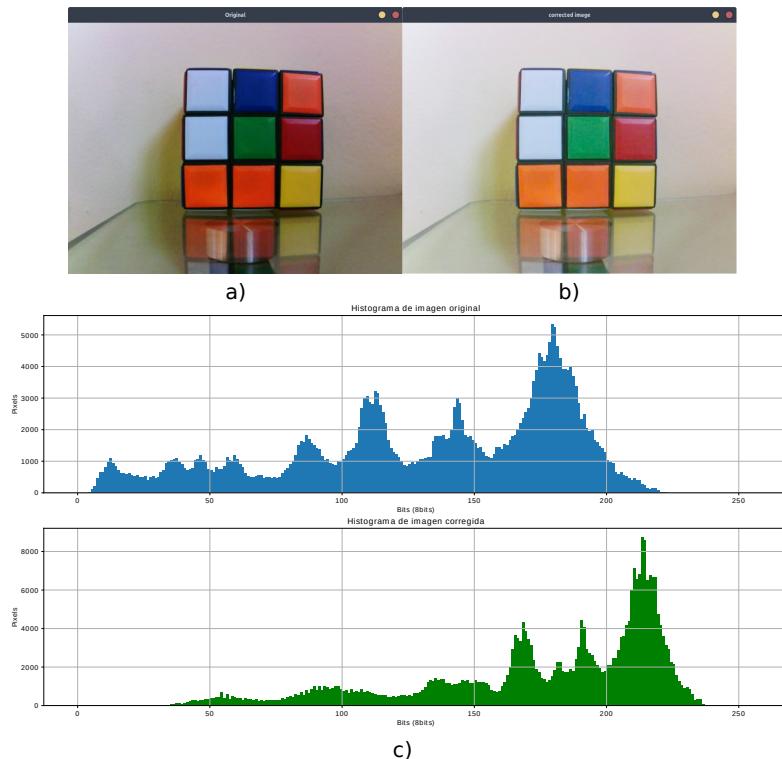
Puede ocurrir que jugar con el valor de  $\beta$  mejore el brillo, pero al mismo tiempo la imagen aparecerá con un ligero velo a medida que se reducirá el contraste. La ganancia  $\alpha$  se puede usar para disminuir este efecto, pero debido a la saturación, perderemos algunos detalles en las regiones 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.

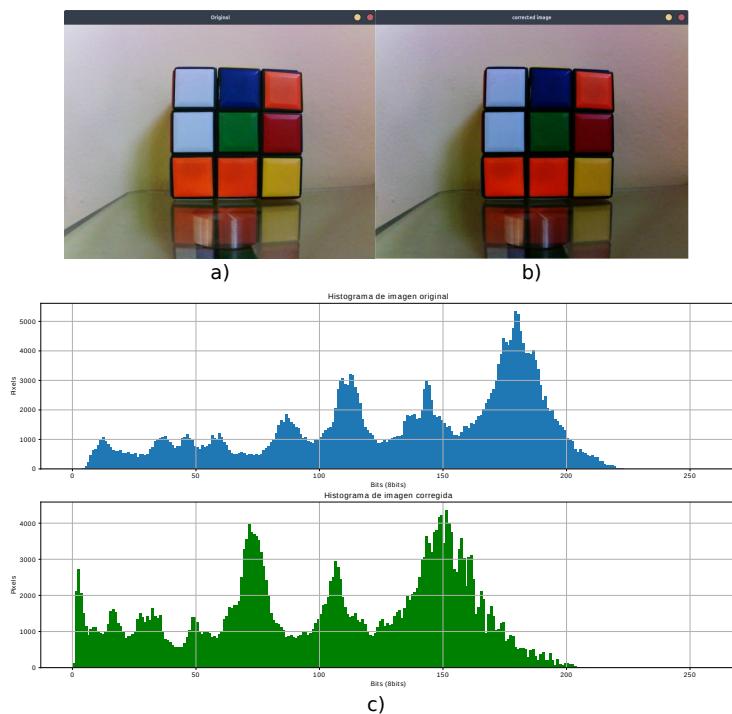
$$O = \left( \frac{I}{255} \right)^\gamma * 255 \quad (4.5)$$

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

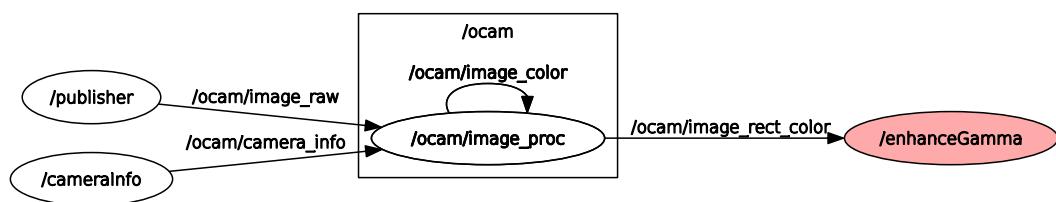
La figura 4.22 es el resultado de una modificación de gamma a 0.5, es decir, que hace que la imagen se vea con más brillo de lo que originalmente es, esto nos permite obtener mejores resultados al momento de aplicar técnicas de identificación de color cuando el entorno de la imagen es un tanto oscuro. 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.

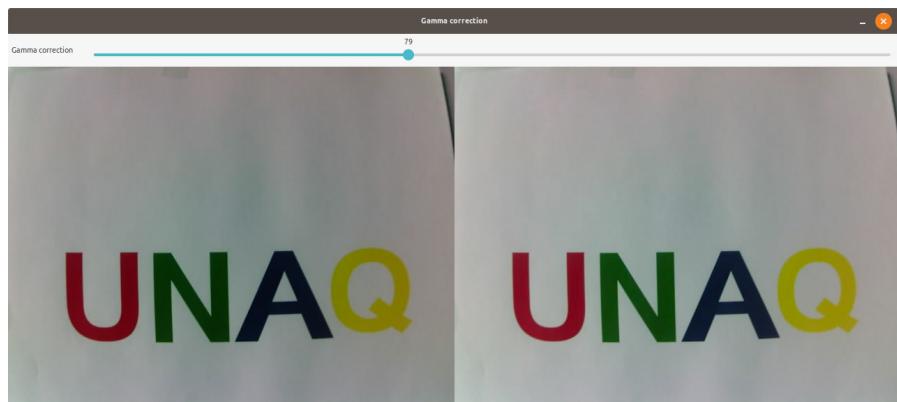
Los nodos y tópicos que están activados en ROS se muestran en la figura 4.24



**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 figura 4.25 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 segundo, claramente se percibe un realce de colores y el fondo se ve más claro al de la imagen original, que esto ayuda 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. Dichos 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

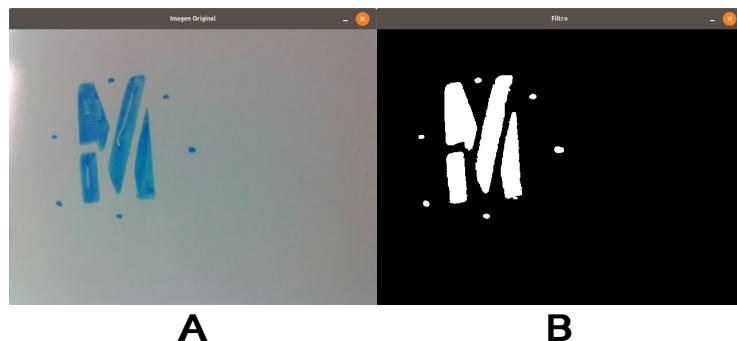
```
void morphologyOperation(Mat &_Image, Mat &Image_, int oper, int size, int elemen)
```

De la función anterior tenemos que:

- \_Image es la imagen de entrada.
- Image\_ es la imagen de salida

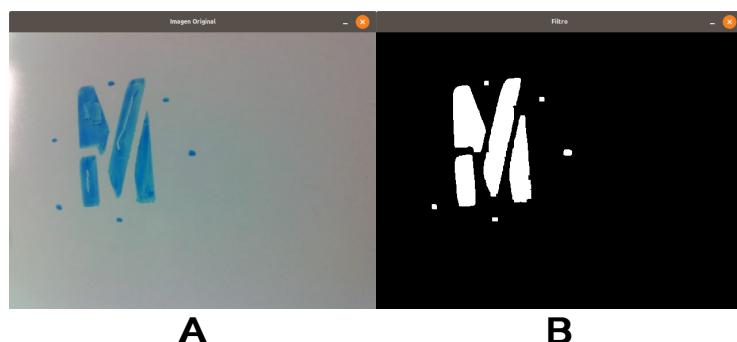
- **Oper** indica la operación Morfológica, 0 para opening y 1 para closing.
- **size** El tamaño del kernel.
- **elemn** La forma del kernel; 0 → Rectangular; 1 → Elipse y 2 → Cruz.

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



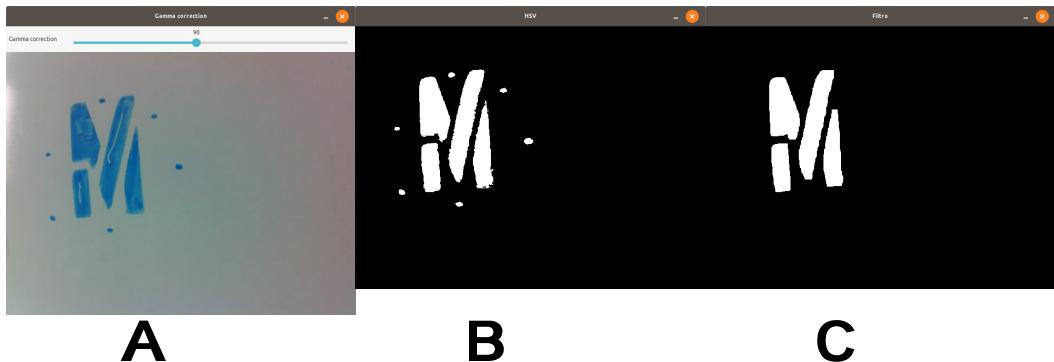
**Figura. 4.26.:** A) Imagen original, B)Aplicación de filtros

En la figura 4.26 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

En la figura 4.27 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

Para la figura 4.28 se modificó el size a 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

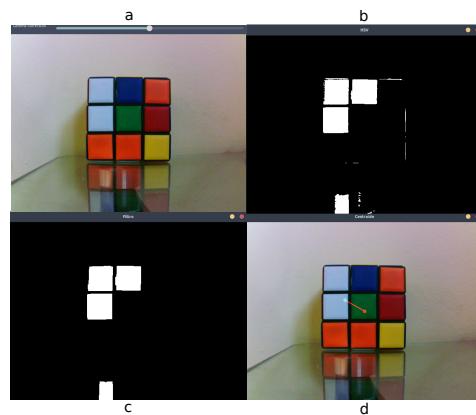
Para obtener el centroide de un objeto en una imagen binaria tenemos que hacer uso del teorema de green, visto en el capítulo 2, y de las librerías opencv.

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

### Función 4.8.1: Centroid

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

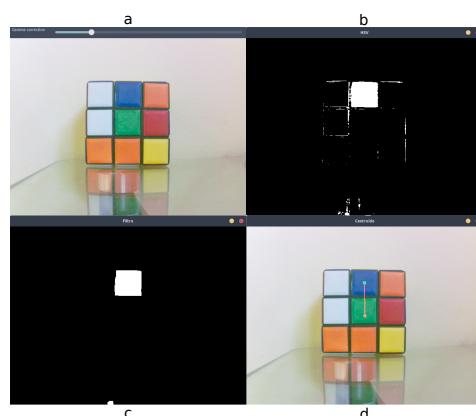
La salida del filtro morfológico es una de las entradas de la función 4.8.1, además de pedir dos variables de tipo uint16 donde se almacenarán las coordenadas (X, Y) del centroide de la figura.



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

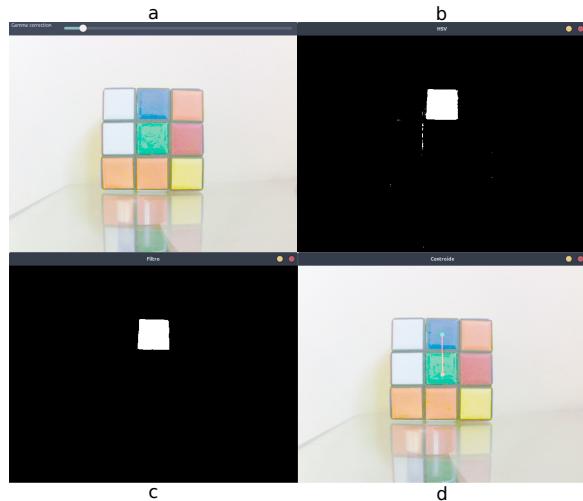
Como primera prueba tenemos en la figura 4.29 en a) la ventana donde se puede modificar el valor de gamma para la corrección de brillo, que en la figura anterior no tenemos ninguna corrección de  $\gamma$ , en b) tenemos la aplicación del algoritmo de HSV, pero podemos observar que detecta a los cuadros 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 se muestra en la figura 4.30.



**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.

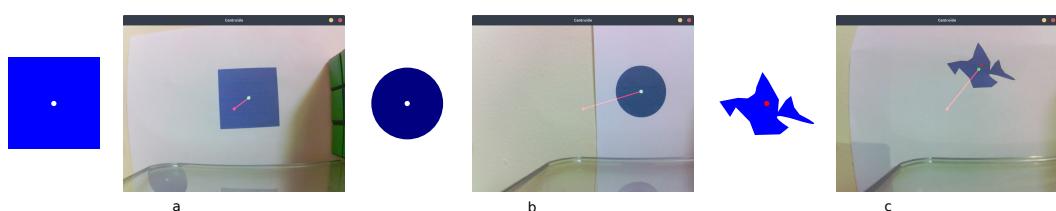


**Figura. 4.31.:** Obtención del centroide con corrección de  $\gamma$  en 0.2

Si se baja aún más el valor de gamma podemos observar que en la figura 4.31 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.

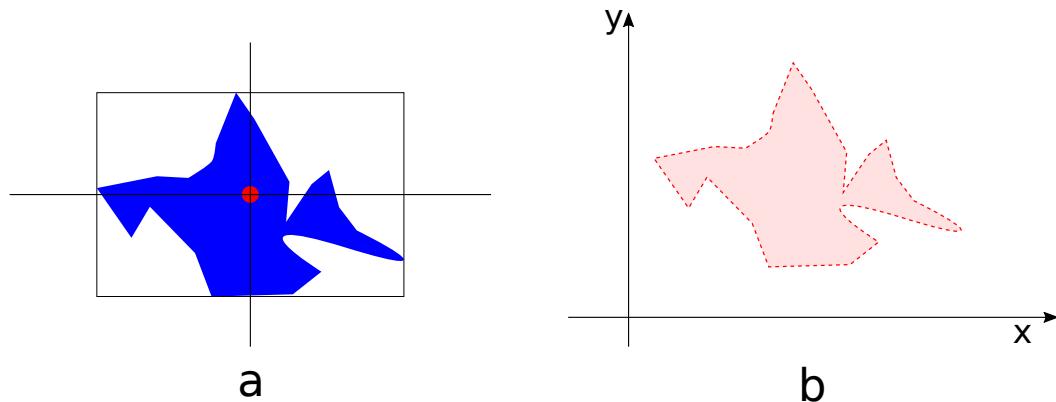
Con esto solucionamos los problemas de tener que ajustar el rango de 3 valores durante el proceso del cambio de espacio de color y así facilitamos al usuario la tarea de encontrar dichos valores. El valor de gamma podría ser ajustado por software con ayuda de un sensor de iluminación, pero eso sale de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

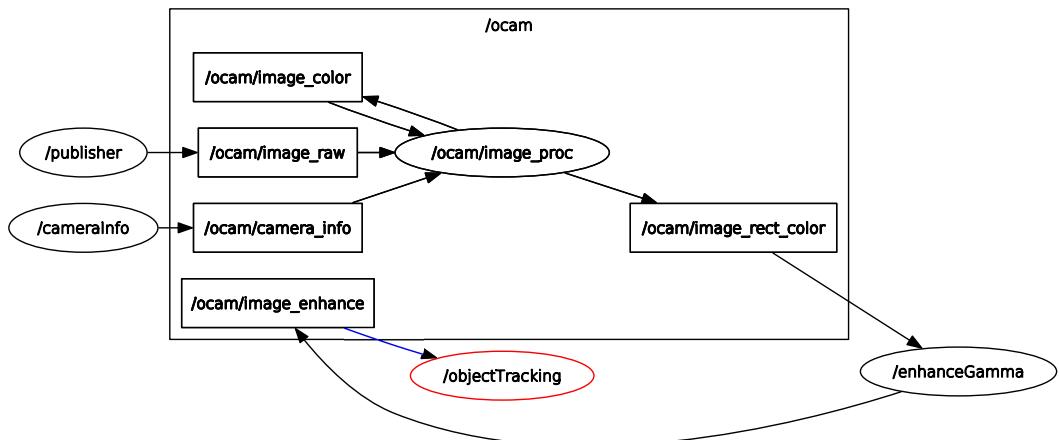
Podemos observar que en la figura 4.23 en a) y b) tienen un respectivo punto blanco, esto con la finalidad de poder saber con exactitud si el centroide que encuentra el sistema es el real, y como se puede observar el punto verde encaja justo con el punto blanco impreso en la figura, por lo que podemos 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.: Pruebas del centroide con figura de geometría irregular**

En la figura 4.33 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 el 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

” No me dan miedo los ordenadores. Temo la falta de ellos ”

— Isaac Asimov  
(Escritor americano)

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

## 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.

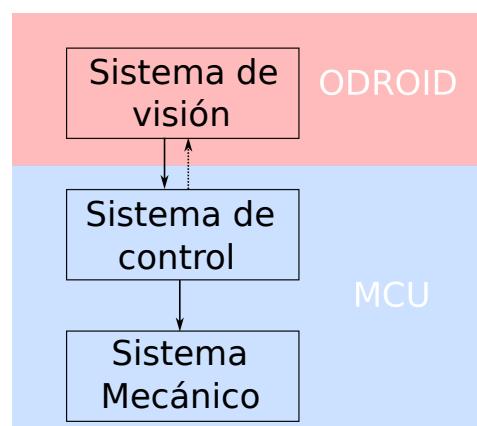
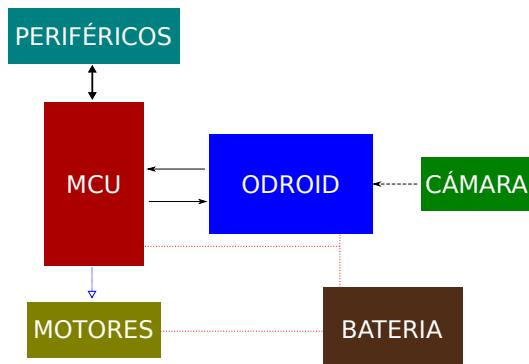


Figura. 5.1.: Módulos del sistema

Tal y como se ilustra en la figura 5.1, dichas 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. Interacciona con el sistema de control enviando la retroalimentación y recibe de este ultimo, datos del error.
- Control: Un microcontrolador es el encargado de interactuar con los datos de la cámara y retroalimentarlos con la salida deseada, para después controlar con una PI.
- 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 figura 5.2 ilustra la interacción entre ellos y se presenta la arquitectura propuesta.



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

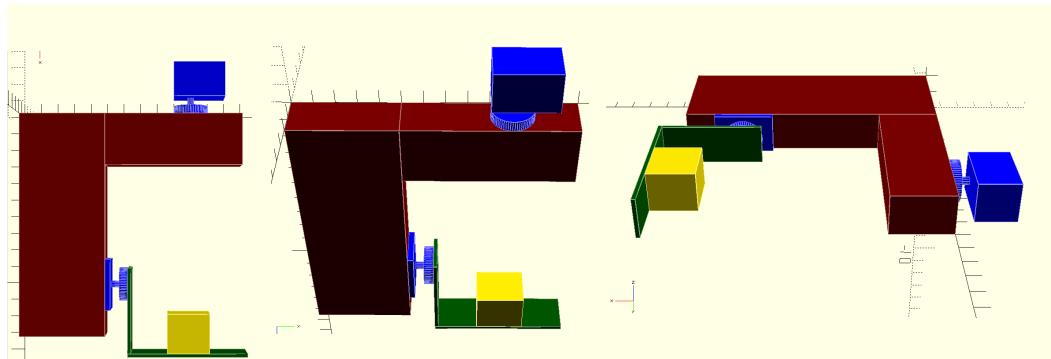
El microcontrolador es de 32 bits arquitectura RISC, las especificaciones detalladas de los motores y de la tarjeta de desarrollo ODROID puede ser consultada en el Apéndice B.

El Bus señalado con líneas punteadas en rojo representan la conexión de voltaje con los demás componentes, mientras que la línea punteada en negro entre la cámara y la odroid simboliza la conexión USB para la transferencia de datos, la comunicación entre ODROID y el MCU se da mediante el módulo RS-323 integrado en el microcontrolador que a su vez se comunica con los periféricos deseados utilizando sus salidas digitales.

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

## 5.2. Mecanismo móvil

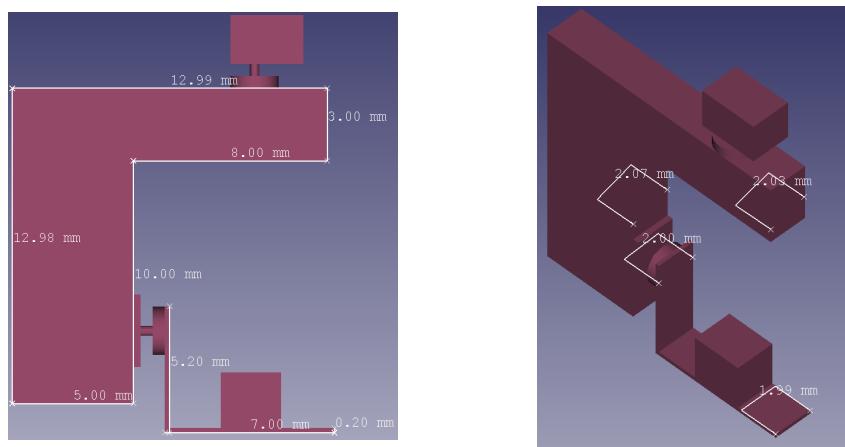
El diseño del sistema fue hecho con un programa de uso de software libre llamado OpenSCAD, en el cual se trazó el prototipo del sistema utilizando dos motores tipo servo, es decir, que tienen su propio controlador de velocidad integrada vía PWM.



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

En la figura 5.3 los motores están representados en color azul, mientras que la cámara esta de amarillo, como se puede apreciar la cámara recae sobre un soporte de color verde, dicha cámara está sujetada por un par de ligas que ayudan a que esta no se caiga cuando hay rotación en pitch.

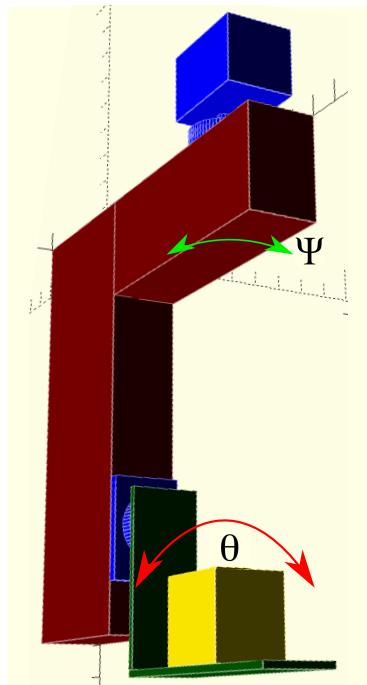
A continuación en la figura 5.4 se presentan algunas medidas del diseño, las unidades están en mm.



**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 también se les conoce como "Tilt" para los movimientos pitch y "Pan" para movimientos en yaw 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 en la figura 5.5.



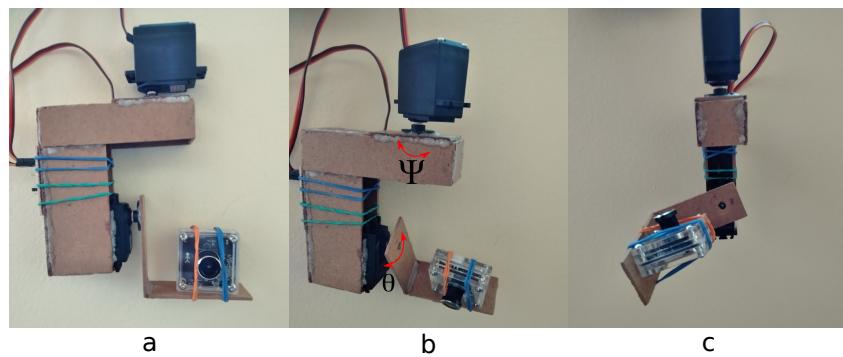
**Figura. 5.5.:** Grados de libertad del sistema

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.

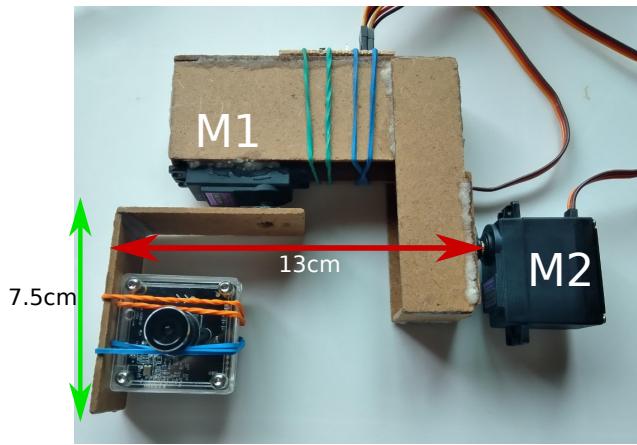


**Figura. 5.6.:** Prototipo del sistema hecho con madera

Como se puede observar en la figura 5.6 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

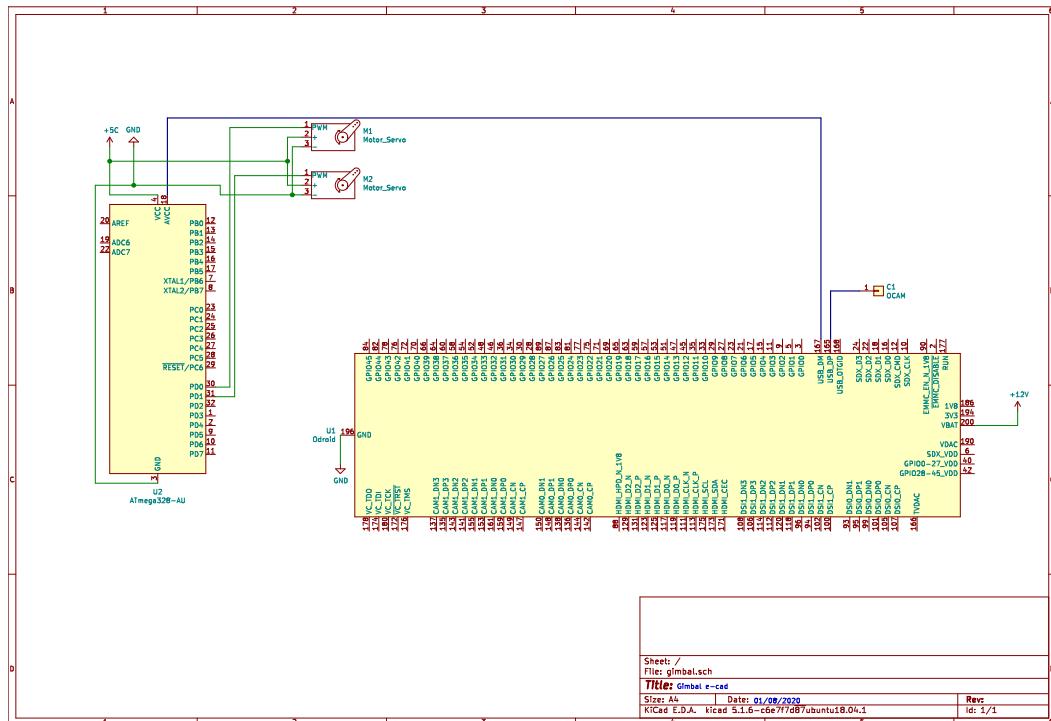
En b) y c) de la figura 5.6 se ilustra el movimiento que puede realizar el sistema, y que será controlado por la cámara y actuado por los motores.



**Figura. 5.7.:** Vista frontal del prototipo

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.

La conexiones eléctricas se ilustra en la figura 5.8. Es un esquemático que muestra la conexión entre los componentes principales: la cámara(OCAM), monoprocesador(ODROID), el microcontrolador (ATMEGA, ver características en apéndice B) y los motores (Servomotores).



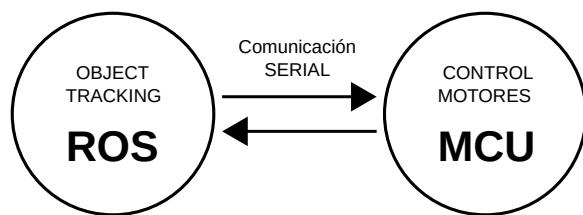
**Figura. 5.8.:** Circuito eléctrico

## 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 píxel, podemos calcular el error teniendo en cuenta que la referencia es el centro de la cámara.

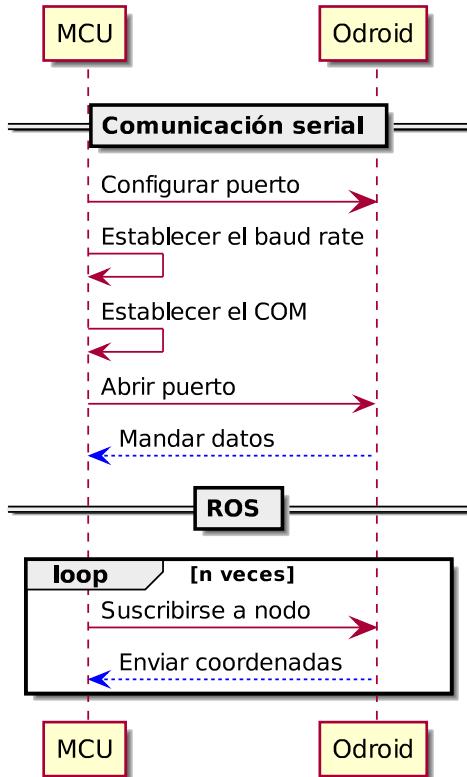
Una vez dicho lo anterior queda una pregunta clara, ¿Cómo comunicar el microcontrolador encargado de los motores, con la cámara que está siendo ejecutada en ROS?.



**Figura. 5.9.:** Protocolo de comunicación

Como se ilustra en la imagen 5.9 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 basado en ser asíncrona, y se detalla mejor diagrama 5.10.



**Figura. 5.10.:** 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 Tópicos y Nodos:



**Figura. 5.11.:** Nodos y Tópicos activos

## 5.5. Control

La idea principal del sistema de control es la de encontrar una función de entrada  $u(t)$  tal que la función de salida  $x(t)$  siga a la salida deseada  $x^{des}(t)$ .

Es decir la diferencia entre la salida deseada y la actual, tal y como se expresa en la ecuación 5.1.

$$e(t) = x^{des}(t) - x(t) \quad (5.1)$$

Donde  $e(t)$  representa el error. Por lo que la estrategia es encontrar un  $u(t)$  tal que

$$K_i \int e + k_p e = 0 \quad (5.2)$$

Donde  $k_i$  y  $k_p > 0$

La ecuación 5.2 representa un controlador proporcional-integral que se verá más a detalle a lo largo de este capítulo, la ec 5.3 se define el control en el tiempo.

$$u(t) = k_i \int e(t) + k_p e(t) \quad (5.3)$$

### 5.5.1. Control en Pitch

El sistema dinámico de Pitch obtenida en el capítulo 3 se trata de una ecuación de primer orden

$$J_{ay} \dot{q}_a = T_y \quad (5.4)$$

Pasando la ecuación 5.5 al dominio de Laplace, obtenemos

$$J_{ay} S Q_a(s) = T_y(s) \quad (5.5)$$

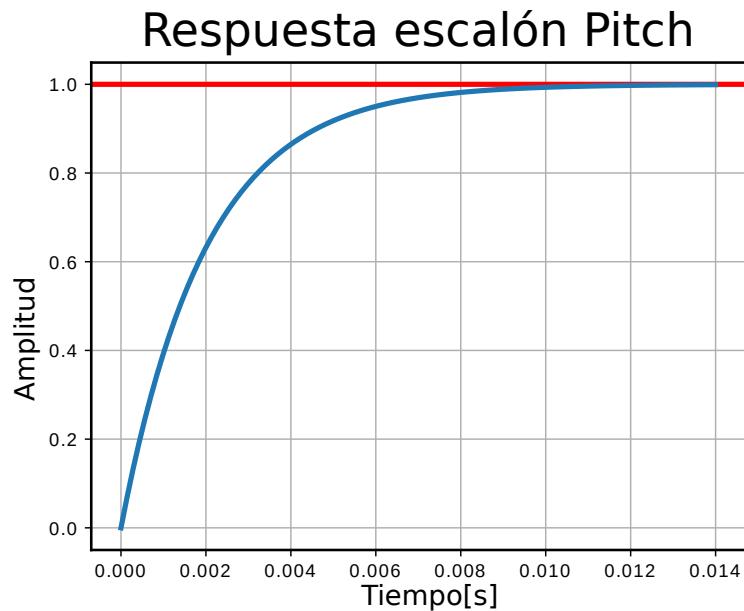
Dividimos entrada entre salida, o en otras palabras, obtenemos la función de transferencia en lazo abierto

$$G = \frac{Q_a(s)}{T_y(s)} = \frac{1}{J_{ay}s} \quad (5.6)$$

Y en lazo cerrado la ecuación 5.7 se transforma en

$$G_c = \frac{G}{1 + GH} = \frac{1}{J_{ay}s + 1} \quad (5.7)$$

La figura 5.13 es el resultado de la simulación del sistema dinámico de pitch en lazo cerrado en respuesta a una entrada escalón.

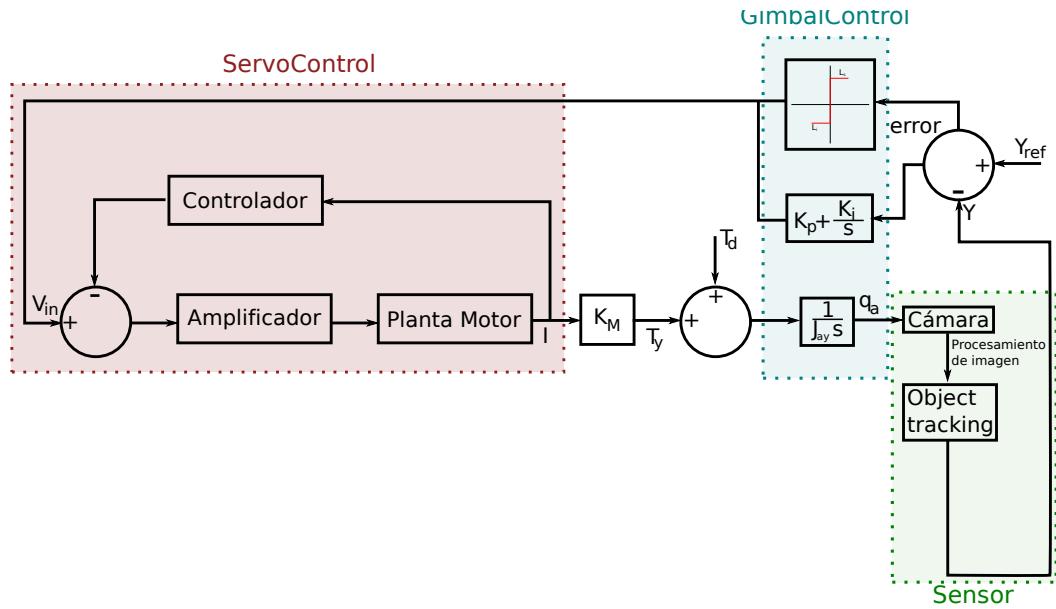


**Figura. 5.12.:** Simulación entrada escalón

Los valores para la ecuación 5.14 se obtuvieron del trabajo de (Abdo y Vali, 2013), debido a la similitud del sistema físico, donde  $J_{ay} = 0,002$

Como se abordó en el capítulo 3, el sistema dinámico de la gimbal es de primer orden para los movimientos en pitch y yaw, y como se introdujo en la sección 5.5 el control que se implementó es el Proporcional-Integral. Siendo este el necesario para hacer que el error converja a cero.

El diagrama de bloques de la figura 5.14 une todas las etapas de control, pasando por el control del servomotor, que hace una retroalimentación interna con base en la dinámica vista en la sección 5.1 y que obtenemos como salida el porqué de entrada de la función de transferencia de la dinámica de Pitch para posteriormente obtener la velocidad angular  $q_a$ , que después pasa por un integrador y así obtener la posición respecto al marco de referencia de la cámara, es decir, en píxeles de la coordenada Y.



**Figura. 5.13.:** Lazo de control para Pitch

### Diseño de controlador

El sistema tiene un controlador PI, donde tenemos la acción proporcional al error, que se encarga de acelerar la respuesta del sistema para llegar a la referencia y tendrá la acción integradora que se encarga de ajustar automáticamente el BIAS integrando el área bajo la curva del error presente en el proceso y de esa forma eliminar el error en estado estacionario. Tenemos que la función de transferencia del controlador PI viene dado por la expresión 5.9.

$$C(s) = K_p \left( 1 + \frac{1}{T_i s} \right) \quad (5.8)$$

Donde  $K_p$  es la ganancia proporcional y  $T_i$  es el tiempo integral. La técnica para el diseño del control se llama asignación de polos.

Aplicamos el lazo cerrado a la planta con el controlador, es decir  $G_c(s) = G(s)C(S)$

$$G_c(s) = \frac{500K_p s + \frac{500K_p}{T_i}}{s^2 + 500K_p s + \frac{500K_p}{T_i}} \quad (5.9)$$

De donde obtenemos la ecuación característica 5.11

$$E.C_1 = s^2 + 500K_p s + \frac{500K_p}{T_i} \quad (5.10)$$

Como todavía no hemos definido cuáles serán los parámetros del controlador PI, estos nos servirán para ubicar los polos en el lugar que nosotros queremos, es por eso que esta técnica es conocida como asignación de polos.

Como condiciones de diseño tenemos que considerar lo siguiente:

- El sistema teóricamente podría establecerse en un tiempo de 0.01 segundos acorde a la figura 5.13, sin embargo físicamente no es posible debido a dos factores, los motores ya tienen su lazo de control incluido como se detalló en la sección 5.5.1 y la velocidad sin carga es de 0.14s lo cual es una limitante y segundo el tiempo de muestreo de la cámara es mayor al de establecimiento del control, ya que está en 60Hz.
- Por lo anterior el tiempo de asentamiento  $T_s$  se establece en 0.2s
- El porcentaje de error se establece en 2 %
- Y el pico Máximo  $M_p$  no debe pasar del 2 %

La función deseada que cumpla los puntos anteriores es de la forma

$$G_d(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega + \omega_n^2} \quad (5.11)$$

Donde el factor de amortiguamiento está dado por

$$\zeta = \sqrt{\frac{\ln\left(\frac{M_p}{100}\right)^2}{\pi^2 + \ln\left(\frac{M_p}{100}\right)^2}} \quad (5.12)$$

Y la frecuencia natural viene dada por la expresión 5.14

$$\omega_n = \frac{3}{\zeta T_s} \quad (5.13)$$

A partir de las ecuaciones 5.13 y 5.14 y con los parámetros de diseño obtenemos la función de transferencia con los polos deseados

$$Gd(s) = \frac{1086}{s^2 + 60s + 1086} \quad (5.14)$$

La ecuación característica de la función de transferencia de la ecuación 5.15 viene dada por la expresión 5.16

$$E.C_2 = s^2 + 60s + 1086 = s^2 + \alpha_1 s + \alpha_2 \quad (5.15)$$

Donde los polos deseados se sitúan en  $s_{1,2} = -30, \pm 13,6437j$

Igualamos la ecuación 5.11 con 5.16 para calcular el valor de  $K_p$  y  $T_i$

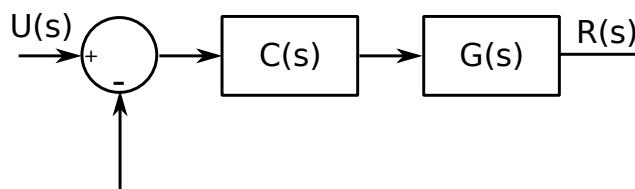
$$500K_p = \alpha_1 \quad (5.16)$$

$$\frac{500K_p}{T_i} = \alpha_2 \quad (5.17)$$

A partir de las ecuaciones 5.17 y 5.18 obtenemos los valores de  $K_p = 0,118$  y  $T_i = 0,054$ , por lo que nuestro controlador viene dado por la ecuación 5.19.

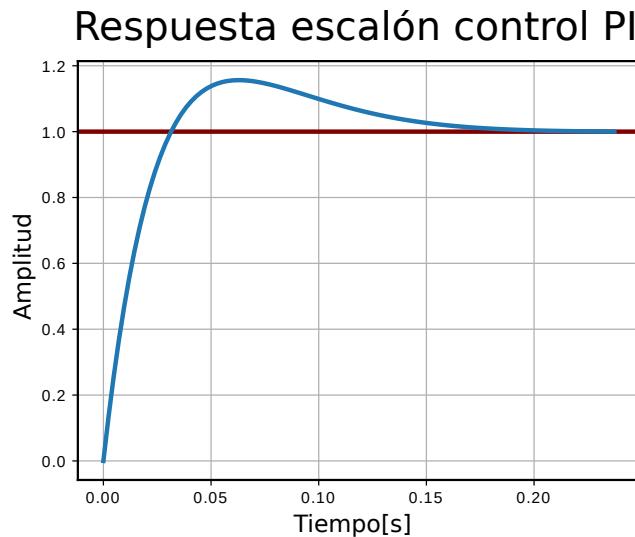
$$C(s) = 0,118 + \frac{2,1851}{s} \quad (5.18)$$

El diagrama de la figura 5.15 muestra el lazo cerrado de controladorPI y la planta para el movimiento en pitch



**Figura. 5.14.:** Lazo cerrado del sistema con control PI

Si aplicamos  $U(s) = 1$ , obtenemos la siguiente respuesta en el tiempo del sistema.



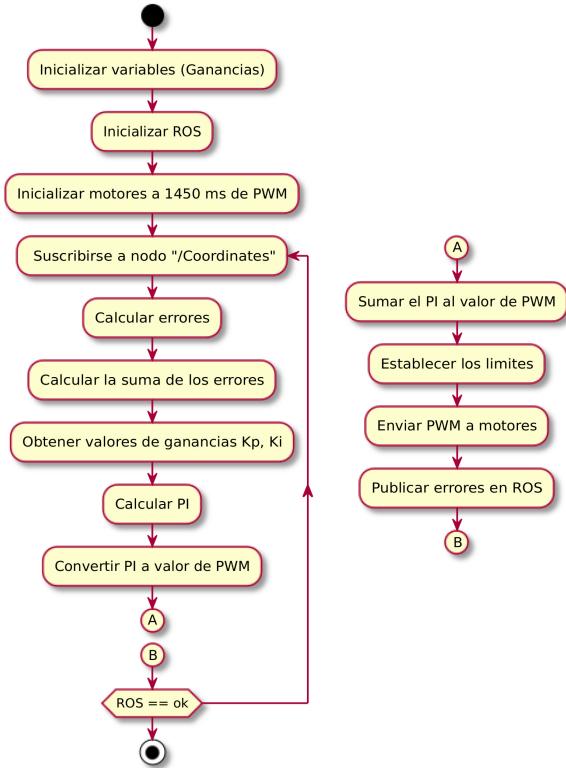
**Figura. 5.15.:** Simulación respuesta escalón en pitch

La simulación de la figura 5.16 muestra que el sistema tiene un tiempo de establecimiento de 0.2 segundos tal y como se planeó en la etapa de diseño, hay un pequeño sobre impulso que en la práctica no representara más del 2%. El código para la simulación está hecho en python y puede ser consultado en mi repositorio de [GitHub](#). La Ganancia Ki de la ecuación 5.25 se ve modificada en el microcontrolador debido a que se utiliza el método numérico para la integración del error, por lo que la ganancia se multiplica por el tiempo de muestreo que es de 0.016, entonces la ecuación 5.25 se convierte en:

$$C(s) = 0,118 + \frac{0,034}{s} \quad (5.19)$$

### Algoritmo de control

El algoritmo general del control tanto para pitch y para yaw está descrito en el diagrama 5.17, basado en programación orientado a objetos donde hay dos puntos claves, la obtención de datos de la cámara vía serial y la publicación de los errores para su posterior uso en gráficas.

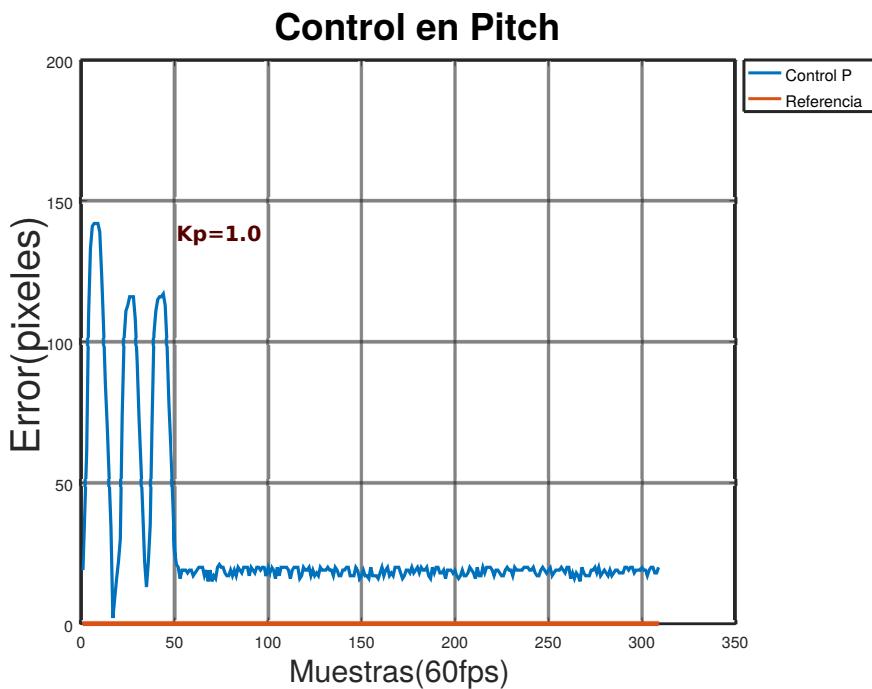


**Figura. 5.16.:** Diagrama de flujo de algoritmo de control

## Sintonización de control

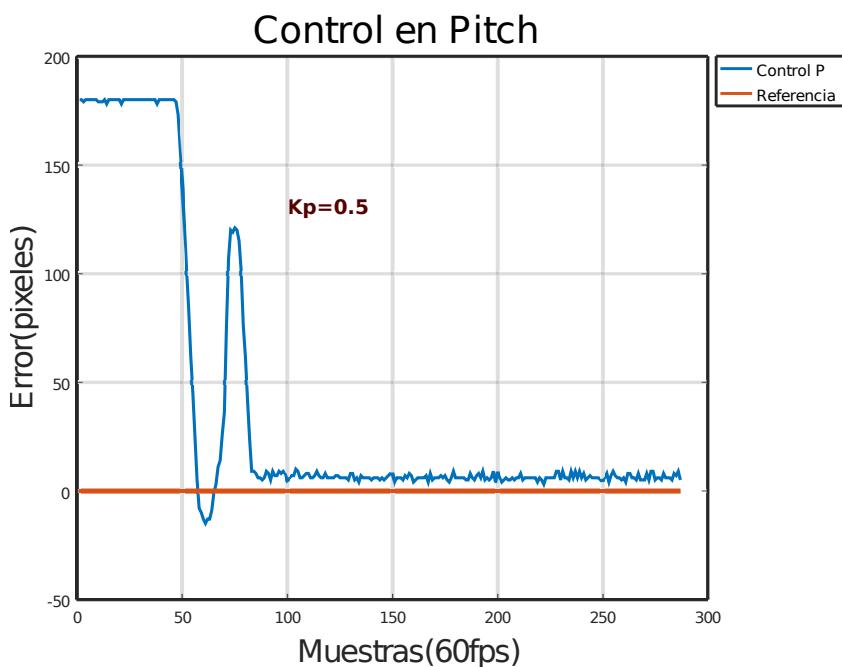
Lo primero que se realizó fue un controlador proporcional y se analizó su respuesta en el tiempo, de la cual se graficó el error, teniendo como referencia el cero.

La figura 5.18 muestra el resultado de sintonizar la ganancia K<sub>p</sub> a uno, como se puede observar el sistema comienza en las primeras 50 muestras para después dar paso a un reposo, el error se mantiene en 20, por lo que el siguiente paso fue reducir la ganancia K<sub>p</sub>



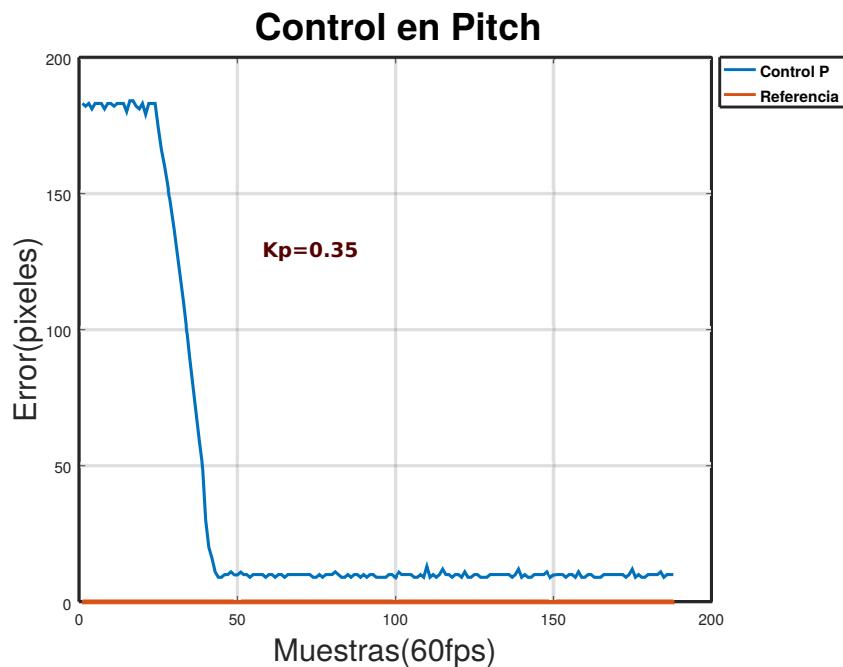
**Figura. 5.17.:** Gráfica del error en Pitch con ganancia  $K_p = 1$

Como abordó anteriormente las ganancias deseadas para el control PI se ubican por debajo de cero, por lo que la siguiente prueba fue bajar la ganancia  $K_p$ . La figura 5.19 muestra la respuesta del sistema con un controlador proporcional con ganancia  $K_p = 0.5$



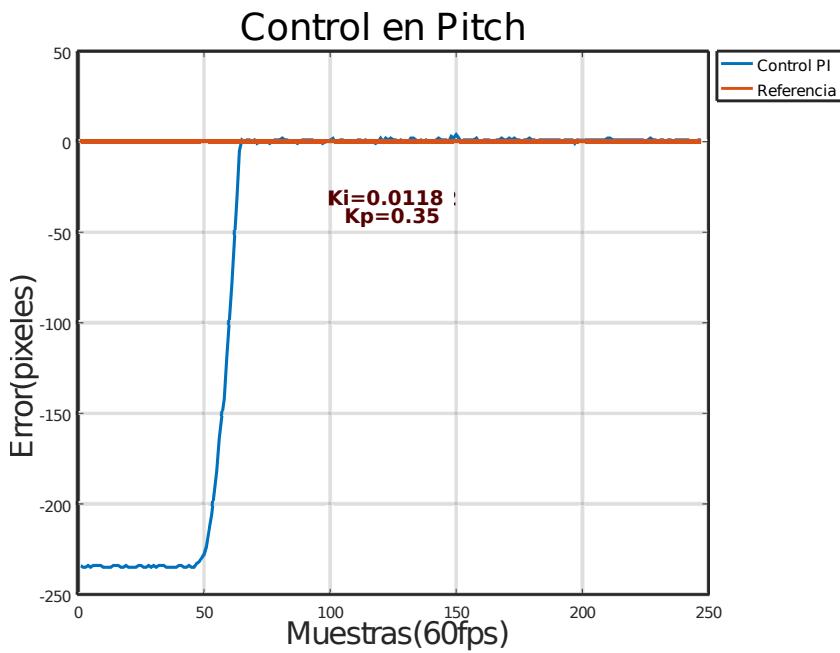
**Figura. 5.18.: Gráfica del error en Pitch con ganancia K<sub>p</sub> = 0.5**

Las oscilaciones antes de que se mantenga estable fueron menores en comparación con ganancia K<sub>p</sub>, por lo que, el valor de la ganancia se redujo a 0.35, obteniendo como resultado la figura 5.20



**Figura. 5.19.: Gráfica del error en Pitch con ganancia K<sub>p</sub> = 0.35**

Es claro que en la gráfica 5.20 el sobre impulso es menor al 2 % sin embargo el error no converge a cero, se mantiene en 10, esto se puede solucionar con un control Integral, reducir el error en estado estacionario agregando la suma de los errores multiplicada por una ganancia K<sub>i</sub>. Finalmente la gráfica de la figura 5.21 muestra el resultado de agregar un controlador PI a la planta en Pitch, donde el error en estado estacionario es eliminado por la ganancia K<sub>i</sub>, que a su vez ayuda a que el sistema converge en menor tiempo, donde se cumple una condición de diseño de  $T_s = 0,2s$ .



**Figura. 5.20.:** Gráfica del error en Pitch con control PI

### 5.5.2. Control en Yaw

El sistema dinámico en Yaw es de primer orden, la obtención de la ecuación diferencial fue obtenida en el capítulo 3, de donde se obtuvo

$$J_k \dot{r}_k = T_z + T_D \quad (5.20)$$

Pasando la ecuación 5.21 al dominio de Laplace obtenemos

$$J_k S R_k(s) = T_z(s) \quad (5.21)$$

Dividimos entrada sobre salida, o en otras palabras, obtenemos la función de transferencia en lazo abierto

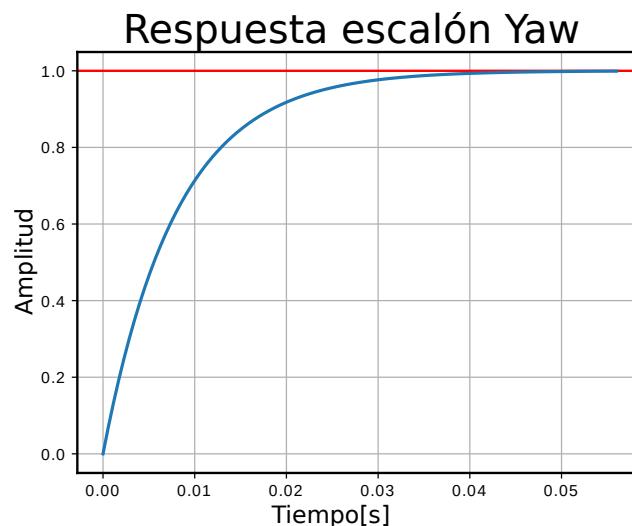
$$G = \frac{R_k(s)}{T_z(s)} = \frac{1}{J_k s} \quad (5.22)$$

Y en lazo cerrado la ecuación 5.22 se transforma en 5.23

$$G_c = \frac{G}{1 + GH} = \frac{1}{J_k s + 1} \quad (5.23)$$

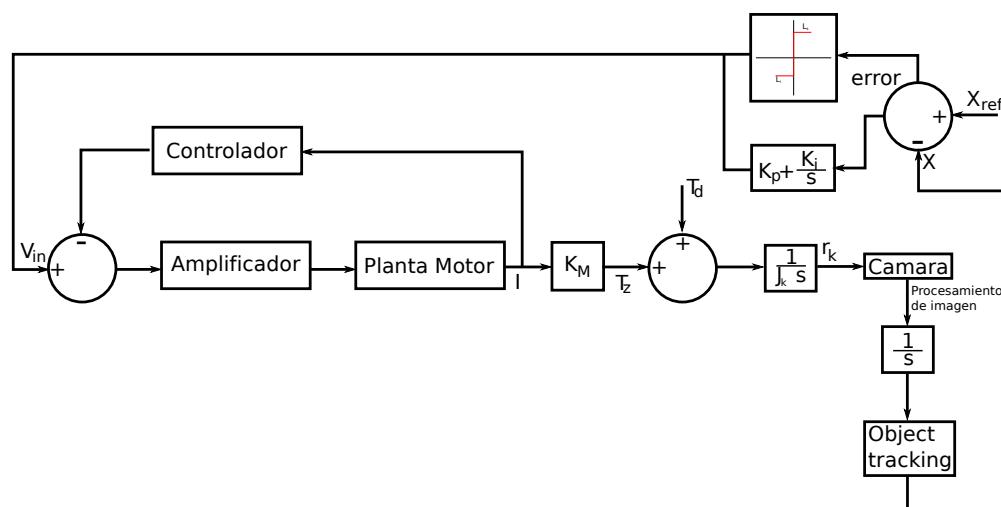
Donde

$J_k = J_{kz} + J_{az}$  La figura 5.22 es el resultado de la simulación del sistema dinámico de pitch en lazo cerrado ante una entrada escalón.



**Figura. 5.21.:** Simulación entrada escalón para yaw

Los valores para la ecuación 5.30 se obtuvieron del trabajo de (Abdo y Vali, 2013), debido a la similitud del sistema físico, donde  $J_k = 0,008$  De manera similar que en pitch tenemos un sistema de lazo cerrado donde se involucra el lazo de control del servomotor y la retroalimentación en la cámara como se observa en la figura 5.23



**Figura. 5.22.:** Lazo de control para Yaw

## Diseño de control

De manera similar que en el control de Pitch en Yaw se diseñó un control PI debido a que dichos sistemas comparten comportamiento similar y se modelaron como sistemas de primer orden. Para este sistema los parámetros de diseño son los siguientes

- $T_s = 0.3s$
- Error menor al 2 %
- El máximo sobre impulso no sobre pasa del 2 %

El procedimiento es el mismo que se abordó al diseñar el control para pitch, por lo que solo se agregan puntos clave del proceso. La función de transferencia deseada se satisface las condiciones de diseño está dada por la expresión 5.25

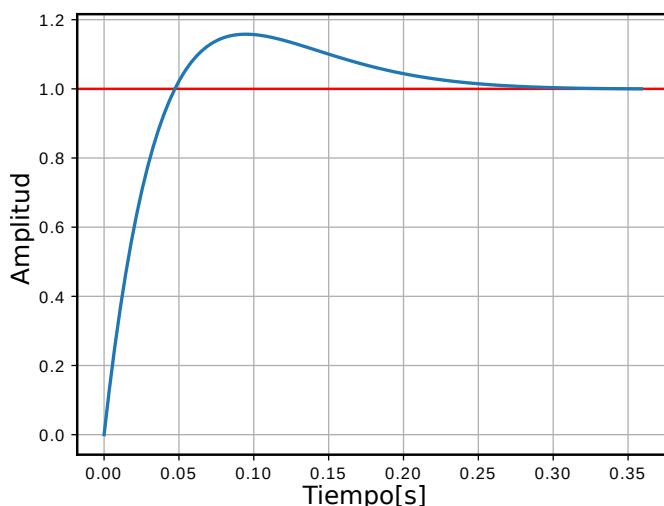
$$G_d = \frac{482,7}{s^2 + 40s + 482,7} \quad (5.24)$$

Donde los polos de la ecuación característica se sitúan en  $-20 \pm 9,0958j$ . Despues de calcular la ganancia  $K_p$  y el tiempo  $T_i$  con base en los polos deseados, obtenemos la función de trasferencia(ecuación 5.26)

$$G_c = 0,312 + \frac{3,8623}{s} \quad (5.25)$$

Aplicando el lazo cerrado de la figura 5.15 y aplicando una entrada escalón se simuló la respuesta en el tiempo de la planta y el control.

## Respuesta escalon control YAW

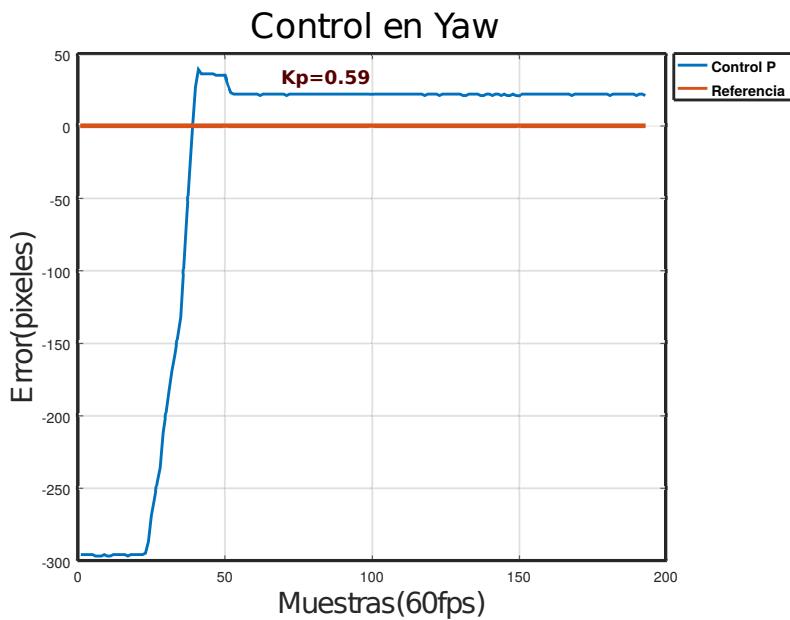


**Figura. 5.23.:** Simulación del control PI en Yaw

En la figura 5.24 podemos observar que el sistema simulado llega a la referencia en 0.3 segundos. Las ganancias que hacen que el sistema se comporte con los polos deseados son  $K_p = 0,312$  y  $K_i = 0,06179$ . El controlador sigue el mismo algoritmo descrito en el diagrama 5.17.

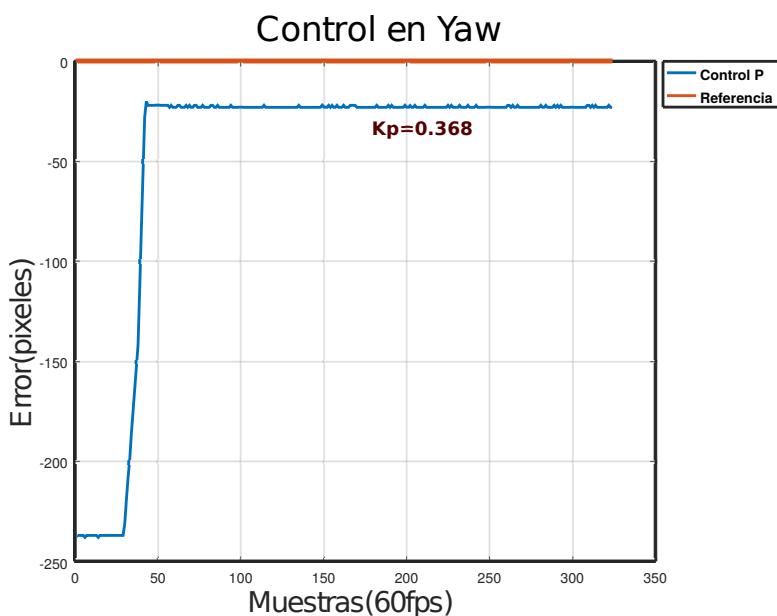
## Sintonización de control

Para sintonizar el control PI primero se aplica ganancia  $K_p$  y se observa el comportamiento del sistema, este método nos ayuda a llegar lo más cerca a la referencia, que en este caso es cero, ya que estamos controlando respecto a la dinámica del error.



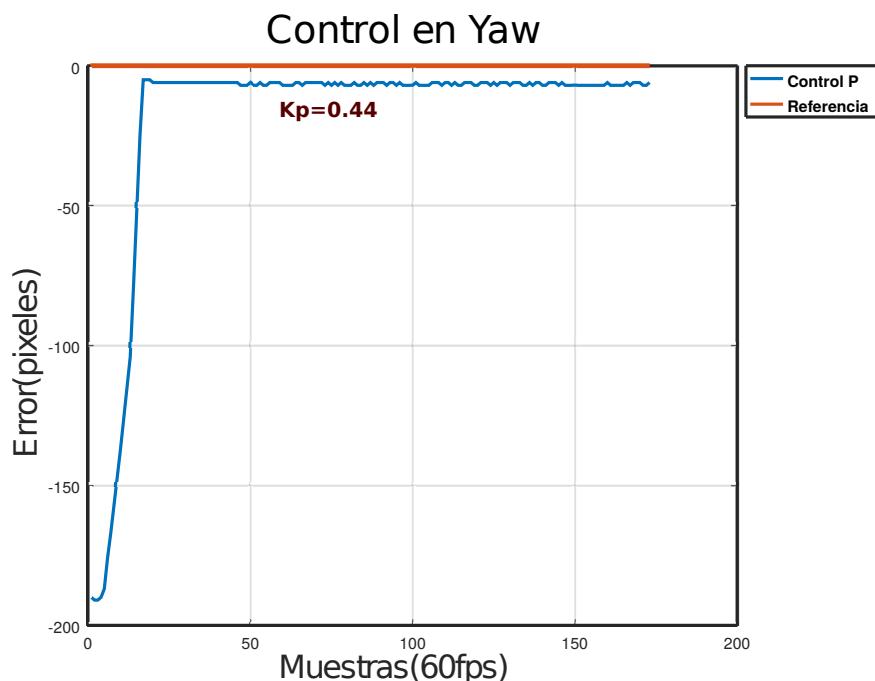
**Figura. 5.24.:** Gráfica del error en Yaw con ganancia  $K_p = 0.59$

La figura 5.25 muestra un control proporcional con ganancia mayor al 0.5, hay un sobre impulso que posteriormente hace que el sistema se mantenga con un error en 20 píxeles, le toma al sistema llegar a la referencia en 10 tiempos de muestreo, pero el control no se mantiene en error cero. Con base en la ganancia obtenida en la etapa de diseño del control la ganancia proporcional debe estar por debajo de 0.5.



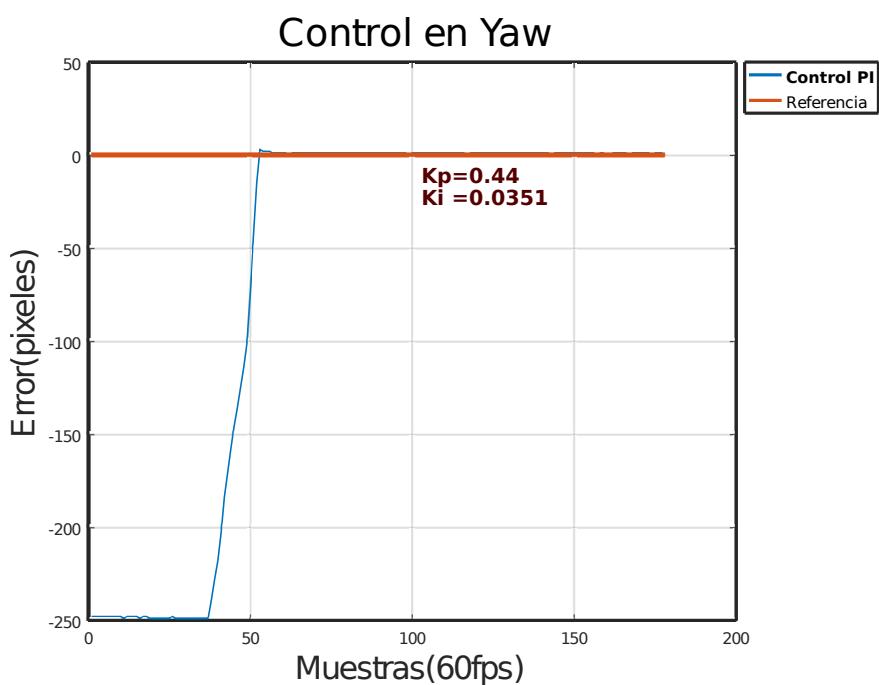
**Figura. 5.25.:** Gráfica del error en Yaw con ganancia  $K_p = 0.368$

Ahora la ganancia es menor que 0.5, lo que hace que el sistema no tenga un sobre impulso, pero tampoco llega a la referencia, se mantiene en un error de aproximadamente -20 píxeles, lo que tampoco es un comportamiento deseado. La ganancia en 0.44 hace que el error se mantenga cercano a cero (figura 5.24), con lo cual podemos aplicar la ganancia integral para erradicar el error en estado estacionario.



**Figura. 5.26.:** Grafica del error en Yaw con ganancia  $K_p = 0.44$

Con una ganancia integral de 0.0351 el sistema se comporta como las especificaciones de diseño marcan, llegar a la referencia y mantenerse estable le toma al sistema 16 tiempos de muestreo, que equivalen a 0.3s y además el sobre impulso se ubica en menos del 2 % por lo que podemos decir que el control satisface los requerimientos.



**Figura. 5.27.:** Gráfica del error en Yaw con control PI

## Conclusiones

En este trabajo, se propuso y formuló un sistema gimbal de dos grados de libertad utilizando la segunda ley de Newton. Se introdujo la obtención de imágenes y su debido procesamiento para detectar figuras dadas ciertas características predefinidas. Las ecuaciones de movimiento de los grados de libertad de la gimbal se derivaron para un sistema con masas equilibradas. Posteriormente se realizaron simulaciones utilizando python y con una propuesta de diseño proporcional-integral, el controlador está completamente diseñado a partir de la comparación de los datos de la simulación con los datos del experimento. Finalmente se probó el sistema físico y se graficó el desempeño que tiene el sistema para mantener el equilibrio. Basado en lo anterior se pueden sacar las siguientes conclusiones:

- Tal y como se esperaba la implementación del algoritmo de visión artificial en ROS hizo que se pudiera tener una mejor modularidad en el sistema debido a que se pudo hacer diversos nodos que podían ser ejecutados en tiempo real e incluso llegar a ser ejecutados en paralelo. ROS además permitió tener una interfaz gráfica para obtener la respuesta en el tiempo del control sin necesidad de migrar datos a un programa exterior, esto permitió que al sintonizar el control uno como diseñador tenga mejor noción del error y los valores de las ganancias del sistema en tiempo real.
- La implementación del sistema en un vehículo aéreo no tripulado requiere que el sistema emita señales de control, tal y como se planteó en un inicio de este trabajo, al implementarse con ROS permite que se pueda crear diversos nodos remotos vía UDP, y con la tarjeta monoprocesador ODROID en el drone, haría que la obtención de datos en tierra sea una tarea relativamente sencilla.
- En el algoritmo de visión artificial para el seguimiento de objetivos, una de las tareas que tomaron más tiempo de análisis fue la de clasificación

de colores, como se abordó en el capítulo 4 diversos factores ambientales afectaron a la percepción y clasificación de colores, la solución que muchos autores, citados en este trabajo, hicieron para lidiar con este problema es modificar los valores del espacio de color, esto fue probado en este trabajo pero los resultados no fueron los esperados, ya que debe de realizarse siempre en un espacio controlado, lo que no permitiría una correcta implementación en UAV, además de la cantidad de procesamiento que se tiene que hacer para modificar los valores cada frame, es por eso que la idea inicial de modificar los rangos de colores respecto al entorno quedo descartada dando pie a la implementación del control gamma de iluminación, lo que nos permitió tener que modificar solo un valor(gamma) y con esto se rejujo las variables a manipular de carga al procesador. Como se mencionó al inicio del actual trabajo se desconocía esta técnica de mejorar la nitidez de colores, dando incluso mejores resultados para el usuario final, además pensar en la implementación de un sensor de brillo ahora es posible, lo cual nos ahorraría modificar el valor de gamma manualmente.

- Diversos tipos de controladores están disponibles en la actualidad, debido a la linealidad del modelo presentado había dos caminos a tomar, la decisión de hacer un diseño PI se dio porque ya se había probado implementar un controlador PD, donde los resultados no fueron los esperados, ya que si bien es cierto se podía quitar el sobre impulso, el error en estado estacionario no convergía a cero. Por esta razón y dadas las condiciones del proyecto utilizar el controlador proporcional integral fue la mejor opción para controlar al sistema.
- Los resultados muestran que los errores en píxeles de los datos obtenidos experimentalmente están muy por debajo de lo permitido (2 %). Sin embargo, hay que reconocer que, por muy satisfactoria que sea la exactitud de la medición podría ser insuficiente para la implementación en un sistema aéreo a alta velocidad, debido a la limitada resolución de la cámara, y de la respuesta del propio control. Resolver el problema es uno de los objetivos a futuro.

# Bibliografía

- Abdo, Maher y Ahmad Reza Vali (2013). *Research on the Cross-Coupling of a two Axes Gimbal System with Dynamic Unbalance* (vid. págs. 102, 111).
- Anil Bharath (2008). *Next Generation Artificial Vision System*. Artech House (vid. pág. 10).
- Barfoot, Timothy D. (2020). *State Estimation for robotics*. Cambridge University Press (vid. pág. 64).
- Corke, Peter (2011). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer (vid. págs. 28, 29).
- Escalera, Arturo de la (2011). *Visión por Computador. Fundamentos y métodos*. Prentice Hall (vid. págs. 25, 36).
- Flusser, Jan (2009). *Moments and Moment Invariants in Pattern Recognition*. Wiley Sons Ltd (vid. pág. 41).
- Goldstein, Herbert (1980). *Calssical Mechanics*. Addison-Weasley (vid. pág. 59).
- Jähne, Bernd (1997). *Digital Image processing*. Springer (vid. pág. 25).
- José Ramón Mejía Vilet (ene. de 2005). *Apuntes de Procesamiento Digital de Imágenes*. First. Facultad de Ingeniería UASLP (vid. págs. 12, 13, 20, 25, 26, 37, 38).
- Joseph, Lentin (2018). *Robot Operating System for Absolute Beginners: Robotics Programming Made Easy*. Apress (vid. págs. 22, 24).
- K. J. Seong, H. G. Kang y H. P. Lee (2006). *The stabilization loop design for a two-axis gimbal system using LQG/LTR controller* (vid. pág. 6).
- Kim, S. B. e Y. K. Kwak (2010). *Robust control for two-axis gimbaled sensor system with multivariable feedback systems* (vid. pág. 6).
- Kim, Taiyi ChengHyun Wook ParkYongmin (2000). “IMAGE FILTERING IN HS COLOR SPACE”. En: *University of Washington* (vid. pág. 30).
- Kuehni, Rolf G. (2003). *Color Space and Its Divisions Color Order from Antiquity to the Present*. Wiley-Interscience (vid. pág. 30).

- Lin, C. L e Y. H. Hsiao (2001). *Adaptive feedforward control for disturbance torque rejection in seeker stabilizing loop* (vid. pág. 6).
- Mohinder S. Grewal (2007). *Global Positioning Systems, inertial navigation, and integration*. Wiley (vid. pág. 3).
- Nise, Norman S. (2010). *Control systems engineering*. Wiley (vid. págs. 41, 43, 45, 47).
- Ortiz Zamora, Francisco Gabriel (2002). *Procesamiento morfológico de imágenes en color: aplicación a la reconstrucción geodésica* (vid. págs. 34-36).
- Quigley, Morgan (2015). *Programming Robots with ROS*. O'Reilly (vid. pág. 23).
- Rafael C. Gonzalez, Richard E. Woods (2002). *Digital image processing*. Prentice Hall (vid. pág. 29).
- Ricolfe Viala y Sánchez Salmerón, A. J. (2008). *Procedimiento completo para el calibrado de cámaras utilizando una plantilla plana*. (Vid. pág. 19).
- Salatun, A. S. y P. M. Bainum (1983). *Analysis of a double gimbaled reaction wheel spacecraft attitude stabilization system* (vid. pág. 6).
- Serra, Jean (1984). *Image Analysis and Mathematical Morphology, Volume 1*. Academic Press (vid. pág. 33).
- Szeliski, Richard (2011). *Computer Vision: Algorithms and Applications*. Springer (vid. págs. 27, 39).
- Taylor, Geoffrey (2006). *Visual perception and robotic manipulation*. Springer-Verlag Berlin Heidelberg (vid. pág. 19).
- YANG, LUREN (1994). *FAST AND EXACT COMPUTATION OF CARTESIAN GEOMETRIC MOMENTS USING DISCRETE GREEN'S THEOREM* (vid. pág. 40).
- Yoon, Sungpil (2001). *Equations of Motion for a Two-Axes Gimbal System* (vid. págs. 60-62).

## Webpages

- HardKernel (2020a). *ODROID XU-4*. URL: <https://www.hardkernel.com/shop/odroid-xu4-special-price/> (visitado 10 de ene. de 2020) (vid. pág. 22).
- (2020b). *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. 68).

- MUNDIARIO (2020). *Cómo el cerebro procesa las imágenes*. URL: <https://www.mundiarario.com/articulo/sociedad/cerebro-procesa-imagenes/20190304191451147484.html> (visitado 10 de ene. de 2020) (vid. pág. 21).
- NASA (2017). *The Gimbal Rig Mercury Astronaut Trainer*). URL: <https://www.nasa.gov/centers/glenn/about/history/mastif.html> (visitado 10 de ene. de 2020) (vid. pág. 5).
- (2020a). *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).
  - (2020b). *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).
- OpenCV (2020). *About it*. URL: <https://opencv.org/about/> (visitado 10 de ene. de 2020) (vid. pág. 28).
- (s.f.). *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. 33).
- PHOTOGRAPHY, EASY BASIC (2020). *How Cameras Work-The Parts of a Camera*. URL: <http://www.easybasicphotography.com/the-camera.html> (visitado 10 de ene. de 2020) (vid. págs. 14, 15).
- PlayList (2020). *Gimbal*. URL: <https://playlists.net/artists/gimbal> (visitado 10 de ene. de 2020) (vid. pág. 3).
- ROS (2020). *ROS/Concepts - ROS Wiki*. URL: <http://wiki.ros.org/ROS/Concepts> (visitado 10 de ene. de 2020) (vid. pág. 23).
- UCSA, IEEE (mayo de 2019). *Transformaciones Morfológicas con Visión Artificial*. URL: <http://ucsa.ieeeparaguay.org/2019/05/15/transformaciones-morfologicas-con-vision-artificial/> (vid. pág. 35).



# Índice de figuras

1.1	Uso de una gimbal para un sensor inercial (Mohinder S. Grewal, 2007) . . . . .	3
1.2	Uso de una gimbal en un motor de propulsión (NASA, 2020a) .	4
1.3	Jerrie Cobb, uno de los Mercury 13, da un giro en la plataforma gimbal. Créditos: NASA . . . . .	5
1.4	Primer uso de la steadicam . . . . .	6
2.1	Similitudes entre humano y computadora . . . . .	9
2.2	El camino que sigue la señal óptica dentro de la cabeza humana.Tomada de Google . . . . .	11
2.3	Formación de una imagen en el ojo. Tomada de Google . . . . .	13
2.4	Componentes de la cámara. Tomada de (PHOTOGRAPHY, 2020)	15
2.5	Proceso general de captura de luz por el sensor de la cámara digital . . . . .	16
2.6	Captura de imagen . . . . .	16
2.7	Regla 1 . . . . .	17
2.8	Regla 2 . . . . .	17
2.9	Regla 3 . . . . .	17
2.10	Obtención geométrica de la imagen . . . . .	18
2.11	Modelo de lente fina . . . . .	19
2.12	Modelo pinhole tomado de (Taylor, 2006) . . . . .	19
2.13	Tarjetas procesadoras de datos . . . . .	21
2.14	Nodos conectados al Master . . . . .	24
2.15	Representación 2D de imágenes digitales mediante conjuntos de puntos discretos en una matriz rectangular . . . . .	26
2.16	Tomada de (Corke, 2011); a) Imagen con distorsión; b) Imagen después de calibrar . . . . .	29
2.17	Representación del espacio de color HSI. Tomada de Google . .	31
2.18	Ejemplo de formas básicas de elementos estructurantes planos .	35

2.19	(a) Erosión de X por el elemento estructurante Y. (b) Los elementos conectados del conjunto X más pequeños que Y son eliminados. . . . .	36
2.20	(a) Dilatación de X por el elemento estructurante Y. (b) El conjunto X aumenta su definición. . . . .	37
2.21	(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. . . . .	38
2.22	(a) Apertura morfológica del conjunto X por el elemento estructurante Y. (b) El cierre redondea las concavidades importantes. . . . .	38
2.23	Area de D limitada por C . . . . .	40
2.24	Diagrama de bloques de sistema de control . . . . .	42
2.25	Diagrama de bloques de sistema de control . . . . .	42
2.26	Diagrama de bloques de la función de transferencia . . . . .	44
2.27	(a) Lazo abierto (b) Lazo cerrado . . . . .	45
2.28	Función de transferencia de un sistema de primer orden . . . . .	46
2.29	Respuesta de un Sistema de Primer orden a un escalón unitario, tomada de (Nise, 2010) . . . . .	47
3.1	Drone con diferente grado de orientación . . . . .	49
3.2	Desplazamiento cuerpo rígido . . . . .	50
3.3	Vectores de rotación . . . . .	51
3.4	Marco de referencia inercial . . . . .	52
3.5	Rotación en 3 ejes.Tomada de freepik . . . . .	53
3.6	Marco de referencia del cuerpo. . . . .	53
3.7	Marco de referencia del la cámara. . . . .	54
3.8	Marco de referencia Body y gimbal . . . . .	55
3.9	Marco de referencia del gimbal trasladado en el del cuerpo . . . . .	55
3.10	Rotación en el eje k . . . . .	56
3.11	Rotación en el eje x . . . . .	57
3.12	Sistema gimbal de dos grados de libertad . . . . .	58
3.13	Representación de la ecuación de movimiento en el eje Pitch . . . . .	61
3.14	Representación de la ecuación de movimiento en Yaw . . . . .	65
4.1	Cámara 'Ocam' obtenida de (HardKernel, 2020b) . . . . .	67

4.2	Nodos y topic . . . . .	69
4.3	Comunicación entre opencv y ROS . . . . .	69
4.4	Diagrama de flujo del programa publisher . . . . .	70
4.5	Diagrama de clases del subscriber . . . . .	71
4.6	Comunicación ROS con calibrador . . . . .	72
4.7	Proceso de calibración utilizando un tablero de ajedrez . . . . .	72
4.8	Proceso de calibración utilizando un tablero de ajedrez; corrección de distorsión . . . . .	73
4.9	Diagrama de flujo para cambio de espacio de color . . . . .	74
4.10	a)Imagen RGB; b) Imagen HSV . . . . .	75
4.11	Espacio de color HSV, a)Hue; b)Saturation; c)Value . . . . .	75
4.12	Espectro de colores . . . . .	76
4.13	Separación de colores en 4 rangos . . . . .	77
4.14	Separación de colores . . . . .	77
4.15	Separación de color azul . . . . .	78
4.16	Separación de colore azul . . . . .	78
4.17	Separación de colores RGB utilizando el modelo HSV . . . . .	79
4.18	a) Imagen captada por la cámara; b) Corrección de contraste; c) Histograma de a y b . . . . .	80
4.19	a) Imagen captada por la cámara; b) Corrección de contraste en 1.5; c) Histograma de a y b . . . . .	81
4.20	a) Imagen captada por la cámara; b) Corrección de brillo en 50; c) Histograma de a y b . . . . .	82
4.21	a) Imagen captada por la cámara; b) Corrección de brillo en 100; c) Histograma de a y b . . . . .	83
4.22	a) Imagen captada por la cámara; b) Corrección de gamma <1; c) Histograma de a y b . . . . .	84
4.23	a) Imagen captada por la cámara; b) Corrección de gamma >1; c) Histograma de a y b . . . . .	85
4.24	Comunicación de nodos . . . . .	85
4.25	Corrección de gamma, la imagen de la derecha es la corregida .	86
4.26	A) Imagen original, B)Aplicación de flitros . . . . .	87
4.27	A) Imagen original, B)Aplicación de flitros. Con size de 3 . . . . .	87
4.28	A) Imagen original,B) Espacio de color HSV C)Aplicación de flitros	88
4.29	Obtención del centroide sin corrección de $\gamma$ . . . . .	89
4.30	Obtención del centroide con corrección de $\gamma$ en 0.4 . . . . .	89

4.31	Obtención del centroide con corrección de $\gamma$ en 0.2 . . . . .	90
4.32	Pruebas del centroide con diferentes formas geométricas . . . . .	90
4.33	Pruebas del centroide con figura de geometría irregular . . . . .	91
4.34	Nodos de ROS . . . . .	92
5.1	Módulos del sistema . . . . .	93
5.2	Arquitectura del sistema . . . . .	94
5.3	Diseño CAD del prototipo . . . . .	95
5.4	Dimensiones del CAD . . . . .	96
5.5	Grados de libertad del sistema . . . . .	96
5.6	Prototipo del sistema hecho con madera . . . . .	97
5.7	Vista frontal del prototipo . . . . .	98
5.8	Circuito eléctrico . . . . .	98
5.9	Protocolo de comunicación . . . . .	99
5.10	Diagrama de secuencia . . . . .	100
5.11	Nodos y Tópicos activos . . . . .	100
5.12	Simulación entrada escalón . . . . .	102
5.13	Lazo de control para Pitch . . . . .	103
5.14	Lazo cerrado del sistema con control PI . . . . .	105
5.15	Simulación respuesta escalón en pitch . . . . .	106
5.16	Diagrama de flujo de algoritmo de control . . . . .	107
5.17	Gráfica del error en Pitch con ganancia $K_p = 1$ . . . . .	108
5.18	Gráfica del error en Pitch con ganancia $K_p = 0.5$ . . . . .	108
5.19	Gráfica del error en Pitch con ganancia $K_p = 0.35$ . . . . .	109
5.20	Gráfica del error en Pitch con control PI . . . . .	110
5.21	Simulación entrada escalón para yaw . . . . .	111
5.22	Lazo de control para Yaw . . . . .	111
5.23	Simulación del control PI en Yaw . . . . .	113
5.24	Gráfica del error en Yaw con ganancia $K_p = 0.59$ . . . . .	114
5.25	Gráfica del error en Yaw con ganancia $K_p = 0.368$ . . . . .	114
5.26	Grafica del error en Yaw con ganancia $K_p = 0.44$ . . . . .	115
5.27	Gráfica del error en Yaw con control PI . . . . .	116
B.1	Odroid XU-4 . . . . .	136

# Índice de cuadros

4.1	Valores para cada rango de color . . . . .	77
4.2	Valores para cada rango de color en un ambiente iluminado por energía eléctrica . . . . .	79



# **Lista de códigos**

A.1	Nodo publicador de imagen en ros . . . . .	131
A.2	Código en c++ que cambia de espacio de color . . . . .	132



# Apendice A

```
1 #include <ros/ros.h>
2 #include <image_transport/image_transport.h>
3 #include <opencv2/highgui/highgui.hpp>
4 #include <cv_bridge/cv_bridge.h>
5 #include <sstream>
6 #include <iostream>
7
8 using namespace std;
9 using namespace cv;
10
11 image_transport::Publisher pub;
12 VideoCapture VC;
13 int CAMCOM;
14 int SAMPLETIME;
15 void publisher();
16
17
18 int main(int argc, char **argv)
19 {
20     ros::init(argc, argv, "image_publish");
21     publisher();
22
23     return 0;
24 }
25
26 void publisher()
27 {
28     ros::NodeHandle nh;
29     image_transport::ImageTransport imgt(nh);
30     nh.getParam("/com",CAMCOM);
31     nh.getParam("/sampletime",SAMPLETIME);
```

```

32     pub = imgt.advertise("camera/Image", 1);
33     VC.open(CAMCOM);
34     if (!VC.isOpened())
35     {
36         ROS_INFO("Camara no conectada");
37     }
38     else
39     {
40         ROS_INFO("Camara conectada en el COM %d",
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     }
62 }
```

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

```

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
35            a 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.

```

```
39     waitKey(0); // Wait for  
40     a keystroke in the window  
41     return 0;  
42 }
```

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

# 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

## B.2. Atmega328P

Es el microchip que esta embebido en el microcontrolador Arduino UNO. Cuenta con las siguientes características:

- Program memory size: 32KB
- SRAAM: 2,048 B
- serial communication, I2C y SPI

La elección de este microchip es por su compatibilidad con ROS-Serial, y además por su amplia documentación.

