Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Inteligencia Artificial

Odometría y mapeo de lugares de difícil acceso empleando técnicas SLAM

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Ciencias en Inteligencia Artificial

Presenta:
Ing. José Héctor León Chávez

Dirigida por
Dr. Juan Manuel Ramos Arreguin

Co-dirigida por
Dr. Sebastián Salazar Colores

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario – Septiembre 2020
México

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Inteligencia Artificial

Odometría y mapeo de lugares de difícil acceso empleando técnicas SLAM

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Ciencias en Inteligencia Artificial

Presenta:
Ing. José Héctor León Chávez
Dirigida por
Dr. Juan Manuel Ramos Arreguin
Co-dirigida por
Dr. Sebastián Salazar Colores

Dr. Juan Manuel Ramos Arreguin
Presidente
Dr. Sebastián Salazar Colores
Secretario
Dr. Jesús Carlos Pedraza Ortega
Vocal
Dr. Saúl Tovar Arriaga
Sinodal
Dr. Marco Antonio Aceves Fernández
Sinodal

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario – Septiembre 2020
México

Dedicatoria

A mi familia.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnolog´ıa por fomentar este proyecto de investigación apoyándome con recursos para mi formación académica como maestro en ciencias. De la misma forma agradezco a la Universidad Autónoma de Querétaro y a la dirección de Investigación y Posgrado por darme la oportunidad de continuar con mis estudios, a mis profesores que siempre me brindaron su apoyo y consejo no permitiéndome desistir de mis objetivos, en especial quiero agradecer al Dr. Juan Manuel Ramos Arreguin por haber participado en este proyecto como director de tesis. El apoyo y conocimiento del Dr. Sebastián Salazar Colores ayudo a que este proyecto tenga un mayor alcance. Finalmente quiero agradecer a mi familia por su apoyo en todo momento sin importar que tan dif´ıcil parec´ıa el panorama, definitivamente su parecencia hace que este objeto sea mas satisfactorio.

Índice

1. Introducción **12**

1.1. Objetivos 13

1.1.1. Objetivos generales 13

1.1.2. Objetivos particulares 13

1.2. Justificación y descripción del problema 14

1.3. Hipótesis 17

1.4. Antecedentes 17

1.4.1. Vehículos autónomos no tripulados 24

1.4.2. Aplicación de la segmentación en la literatura 24

1.4.3. Visión por computadora 26

1.4.4. Segmentación con CNN 27

1.4.5. Estimación de la posición 30

1.4.6. SLAM 32

2. Fundamentación Teórica **36**

2.1. Visión por computadora 36

2.1.1. Cámara 37

2.1.2. Imagen 37

2.1.3. Procesamiento de imágenes 39

2.2. Navegación 40

2.2.1. IMU 40

2.2.2. Sensores de distancia 41

2.2.3. Filtro Kalman 42

2.2.4. Modelo dinámico y cinemático de un drone 43

2.3. Sistemas de control 54

2.3.1. Acción de control 57

2.4. Algoritmos basados en CNN para la segmentación de imágenes 60

2.4.1. Operación convolución en imágenes 62
2.4.2. Métodos de segmentación 66
2.4.3. U-Net 68
2.4.4. Mobile-Net V2 72
2.4.5. Pix2Pix 75
2.4.6. Métricas de segmentación 78
2.5. SLAM 79
2.6. Semejanza entre curvas 81
2.7. Distancia de Fréchet 81

3. Metodolog´ıa 83

3.1. Plataforma de simulación 85
3.1.1. Simulador de drones 86
3.1.2. Comunicación con el controlador de vuelo 88
3.2. Bases de datos 88
3.2.1. Base de datos con imágenes reales 88
3.2.2. Base de datos de la simulación 89
3.2.3. Pre-procesamiento para segmentación 91
3.3. Algoritmos de segmentación 93
3.4. Conducción del drone 95
3.5. Comparación de similitud entre rutas 99

4. Resultados y discusión 101

4.1. Resultados con la base de datos real 102
4.1.1. Valor de pérdida 102
4.1.2. Comparación de imágenes 103
4.1.3. Comparación de IoU empleando conjunto de pruebas 105
4.2. Resultados con la base de datos creada 106
4.2.1. Valor de pérdida 106

4.2.2. Comparación de imágenes 107
4.2.3. Comparación 108
4.3. Resultados del sistema completo 109
4.3.1. Acción correctiva en curvas 109
4.3.2. Obtención del mapa 110

5. Conclusiones y trabajo futuro **113**
5.1. Trabajo futuro 114

Referencias **115**

6. Anexos **124**
6.1. Consideraciones éticas 124

Índice de figuras

1.1. *Sendero natural en Yelapa, Jalisco, México.* 14

1.2. *Algoritmos de segmentación.* 17

1.3. *División de regiones basado en árbol de cuadrantes*, (Gurusamy, Kannan, y Nalini, 2013) 21

1.4. *Arquitectura Mask R-CNN, adaptada de* (He, Gkioxari, Dollár, y Girshick, 2017) 28

2.1. *Diagrama del proceso llevado a cabo en un sistema de visión por compu- tadora* 36

2.2. Representación visual de cada canal de color y la imagen compuesta de los tres canales de color.....39

2.3. *Marco de referencia y fuerzas de un quad-rotor, modificado de* (Luukkonen, 2011). 44

2.4. *Sistema de control en lazo abierto y retroalimentado.*.....57

2.5. *Modelo de la neurona artificial, modificado de* (Rothman, 2018).....61

2.6. *Modelo de un perceptrón multicapa, modificado de* (Rothman, 2018). 62

2.7. *Operación de convolución, modificado de* (Dumoulin y Visin, 2016). 64

2.8. *Arquitectura CNN básica, basada en* (O'Shea y Nash, 2015). 65

2.9. *Operación pooling, modificado de* (Dumoulin y Visin, 2016). 66

2.10. *Arquitectura U-Net basado en* (Ronneberger, Fischer, y Brox, 2015).....69

2.11. *Arquitectura del bloque de cuello de botella residual basado en* (Sandler, Howard, Zhu, Zhmoginov, y Chen, 2018) 73

2.12. *Modelo básico de la arquitectura GANs, basado en* (Özgen y Ekenel, 2020). 75

2.13. *Arquitectura Pix2Pix basado en* (Isola, Zhu, Zhou, y Efros, 2017).....77

2.14. *Problema esencial SLAM, modificado de* (Durrant-Whyte y Bailey, 2006). 80

3.1. Metodolog´ıa propuesta del sistema de seguimiento de senderos.....84

3.2. *Interacción entre los componentes del sistema* 87

3.3. *Escena a) y Segmentación b) de base de datos tomada de* (Valada, Oliveira, Brox, y Burgard, 2016) 89
3.4. Imágenes que componen la base de datos. 90
3.5. *Imágenes pre-procesadas resaltando una etiqueta* 91
3.6. *Imágenes pre-procesadas resaltando una etiqueta* 93
3.7. *Metodolog´ıa empleada para el control del drone* 96
3.8. *Desv´ıo lateral.* 98
3.9. *Desv´ıo angular* 98
3.10. *Obtención de similitud entre rutas del mismo sistema.* 99
4.1. *Valor de pérdida en cada modelo.* 103
4.2. *Comparación de imágenes de entrada y generadas por cada arquitectura empleando base de datos de* (Valada y cols., 2016). 104
4.3. *Comparación de desempeño empleando IoU empleando el conjunto de pruebas de la base de datos real.* 105
4.4. *Valor de perdida de cada modelo empleando base de datos propia.* 106
4.5. *Comparación de imágenes de entrada y generadas por cada modelo empleando base de datos propia* 107
4.6. *Comparación de desempeño empleando IoU empleando el conjunto de pruebas de la base de datos propia.* 108
4.7. *Acción correctiva de los controladores.* 110
4.8. *Comparación de ruta 1 generada de manera manual y autónoma.* 111
4.9. *Comparación de distancia de Fréchet empleando la ruta 1.* 112

Índice de tablas

1. Comparación de métodos de segmentación. 23
2. Arquitectura U-Net basada en (Ronneberger y cols., 2015) 70
3. Arquitectura original Mobile-Net V2.....74
4. Arquitectura del Discriminador 78
5. Comparación de características de los modelos basados en CNN. 102

Resumen

Los vehículos autónomos tienen una gran importancia en el campo de la inteligencia artificial. Se aplican redes neuronales entrenadas para adaptarse a entornos no conocidos. Algunas redes neuronales convolucionales (CNN por sus siglas en inglés) tuvieron gran impacto en diferentes campos de la ciencia, como la biolog´ıa. Sin embargo, su eficiencia en otros campos es poco explorada. En la actualidad podemos emplear modelos computacionales para simular el funcionamiento de diferentes componentes de la vida diaria como veh´ıculos, tráfico o el clima con la finalidad de llevar a cabo experimentos sin ningún riesgo inherente. Este trabajo presenta el comportamiento de los modelos U-NET, Mobile-Net y Pix2Pix para la segmentación de caminos en un entorno virtual. Se aprovechan las facultades del entorno virtual para generar la base de datos que compartiremos. Finalmente se muestran los resultados estadísticos de cada modelo empleando como base de comparación la métrica intersección sobre unión (IoU por sus siglas en inglés) así como la cantidad de imágenes procesadas por segundo.

Abstract

Autonomous vehicles are of great relevance in the field of artificial intelligence. Trained neural networks are used to adapt to unfamiliar environments. Some convolutional neural networks (CNN) had a great impact in different fields of science, such as biology or medicine. However, its efficiency in other fields is less explored. Today we can use computational models to simulate the performance of different components of daily life such as vehicles, traffic or weather in order to develop experiments without any inherent risk. This work presents the behavior of the U-Net, Mobile-Net and Pix2Pix models in the task of trail segmentation in a virtual environment. Taking advantage of the characteristics of the virtual environment, we create a database that we can share. Finally, the statistical results of each model are shown taking as a metric the intersection-over-union (IoU) as well as the number of images processed per second.

1. Introducción

La segmentación de imágenes es la separación de elementos en la imagen mediante algún proceso de agrupación (Yuheng y Hao, 2017). Es una de las tareas más importantes en el área de visión por computadora y tiene una gran relevancia en campos de estudio que van desde la medicina hasta la industria bélica (Shan, 2018; Kim, Kim, y Ro, 2017). La información contenida en una imagen puede ser reagrupada, dependiendo de los componentes con una misma clasificación o cada componente por separado y esto corresponde al tipo de segmentación empleada que puede ser semántica o por instancias.

La segmentación semántica busca asignar una etiqueta a cada elemento en la imagen agrupando a aquellos con características similares. La segmentación por instancias busca dos cosas la detección de objetos y la asignación de etiquetas a dicho objeto separando incluso a aquellos con características similares. En los últimos años con el uso de las redes neuronales convolucionales (CNN) se ha logrado un gran avance en la tarea de segmentar tanto de manera semántica como por instancias tanto en sistemas con capacidad de cómputo alto como en sistemas embebidos para su incorporación en vehículos autónomos (Sultana, Sufian, y Dutta, 2020).

La reconstrucción 3D es uno de los procesos con mayor campo de acción en la actualidad ya que se utiliza en aplicaciones médicas, industriales y militares. Una de las áreas de reconstrucción 3D más solicitadas hoy en día es el mapeo de entornos abiertos debido al auge que están teniendo los vehículos autónomos. Además, una de las necesidades que cubre el mapeo 3D de entornos es la interacción hombre-maquina en la industria ya que lo que se busca es que ambos puedan trabajar independientes y en armon´ıa (Hong y Kim, 2016a).

Una de las bases de la reconstrucción 3D es la odometría, que trata de estimar la posición de un sistema (ego) en el espacio. La palabra odometr´ıa tiene sus ra´ıces en el griego *hodos* que significa trayecto y *metron* que significa medida (Fernandez y Price, 2004), por ello la odometría se encarga de la estimación de la posición en un trayecto de modo que los robots autónomos deben mantener conocimiento sobre su posición para alcanzar la autonomía en navegación. El error en la exactitud se considera como la mayor desventaja para el cálculo

de posición por odometría como se menciona en (Aqel, Marhaban, Saripan, y Ismail, 2016). El uso de vehículos autónomos de diferente índole, como son aéreos, marítimos o terrestres, requieren de la comprensión del medio para determinar las diferentes acciones de control en la detección de caminos u obstáculos (Wang y cols., 2019; Zhou, Kong, Wei, Creighton, y Nahavandi, 2016). La segmentación de caminos permite separar los diferentes elementos de la imagen, como pueden ser vegetación y cielo del objeto de interés. Para ello se emplean técnicas de agrupamiento o de inteligencia artificial.

Algunos modelos de inteligencia artificial basados en redes convolucionales, han mostrado buenos resultados en tareas de segmentación (Mortazi y Bagci, 2018), por ello tienen gran relevancia en la actualidad. Sin embargo, algunos de los modelos fueron diseñados para segmentación de componentes específicos y no se han probado en tareas como lo puede ser segmentación de caminos.

Empleando estos diferentes aspectos se busca trabajar con una plataforma que permita desarrollar diferentes algoritmos que unan la inteligencia artificial con vehículos autónomos con la capacidad de moverse a través de un entorno no conocido esto con la finalidad de exportar el sistema a un entorno real con el cual se pueda dimencionar senderos y permita mantener un registro de los mismos.

1.1. Objetivos

1.1.1. Objetivos generales

Diseñar un sistema con la capacidad de recorrer un trayecto no definido mediante técnicas de inteligencia artificial con la finalidad de modelar la trayectoria y obtener la distancia.

1.1.2. Objetivos particulares

- Diseñar un algoritmo con la capacidad de distinguir un sendero mediante técnicas de inteligencia artificial con la finalidad de mantenerse sobre el sendero.
- Seleccionar y acondicionar una plataforma para generar un mapa tridimensional de

entornos cerrados y/o abiertos para la generación de un mapa 3D utilizando una técnicas de mapeo como podr´ıan ser ORB-SLAM2, Octo-Slam, RTAB-Map entre otras.

- Evaluar el sistema en entornos abiertos para validar su funcionalidad en el seguimiento de un sendero.
- Comparar resultados mediante diferentes experimentos para determinar estad´ısticamente la repetibilidad.

1.2. Justificación y descripción del problema

El enfoque de esta investigación permite la incursión en el estudio de vehículos autónomos ya que muchas de las bases para asegurar la autonomía serán aplicadas. En la actualidad los vehículos autónomos son utilizados en agronomía, entrega de paquetería, situaciones de emergencia, búsqueda de personas, control de incendios, entre otras.
El sistema tendr´ıa que ser capaz de reconocer el sendero, el cual es un camino marcado por el uso constante de personas, como se muestra en la Figura 1.1. Además, tendríamos que asegurar que se tiene un monitoreo constante del medio para evitar dañar el ambiente.

Figura 1.1: *Sendero natural en Yelapa, Jalisco, México.*

En México no existe un registro de las rutas de los senderos, por lo que muchas veces la

transmisión de las características de la ruta se hace de viva voz. En muchos casos, estos datos serian de gran importancia para determinar que zonas tur´ısticas son aptas para transitar o quien puede transitar dichas zonas.

Si el sendero tiene caracter´ısticas que significaran un riesgo a la integridad de una persona se volvería más viable enviar un vehículo no tripulado para la identificación del trayecto para una posterior evaluación, en cualquier modo el desconocimiento del trayecto, sea de riesgo o no, significa un peligro para cualquier persona que lo cruce.

Un sistema como el propuesto tiene múltiples aplicaciones y particularmente destacamos aquellas posibles contribuciones a los ámbitos: tecnológicos, sociales, económicos y académicos.

- Tecnológico. Es necesario contar con un sistema capaz de reconocer un sendero, y dimensionar la longitud del mismo. El sistema podrá estar formado por una cámara de profundidad, as´ı como un sistema embebido basado en microprocesador.

- Social. Puede servir como gu´ıa para turistas en determinados parques, al orientarlo sobre la ruta a seguir. El sistema ayudar´ıa a que las personas conozcan de mejor manera su entorno en que pasean al proveer una herramienta de mapeo de lugares de difícil acceso, perjudiciales o de larga extensión. Con la inclusión de esta herramienta, una institución podría proporcionar más datos sobre las características de senderos en lugares no muy poblados pero si tur´ısticos, con la finalidad de que menos personas se pierdan en estos lugares, o que en primer instancia evalúen si sus capacidades físicas les serán suficientes para recorrerlo.

- Económico. Existen equipos que pueden realizar algo parecido a lo que se propone, sin embargo, sus costos son bastante elevados y son de arquitectura cerrada. Entre ellos se encuentra el AgrasMG-1 que es empleado en arquitectura pero puede seguir una ruta determinada, este drone puede alcanzar más de 260,000 pesos mexicanos.

- Académico. El desarrollo de este proyecto va a permitir aplicar diversos algoritmos de inteligencia artificial, para realizar el reconocimiento del sendero. Algunos algoritmos

pueden ser: Redes neuronales convolucionales, segmentación, entre otros.

Para asegurar el funcionamiento de un vehículo autónomo es necesario conocer en todo momento la posición en el espacio, así como conocer el entorno en el cual el vehículo autónomo se encuentra ya que si este debe generar una trayectoria tendría que saber su posición respecto a los obstáculos.

Entre las dificultades que se tiene al buscar la autonom´ıa de un veh´ıculo se encuentra:

- Ruido en los sensores

  Dependiendo del tipo de cámara, se vera limitada por el entorno en el que opera ya que pueden ser susceptibles a luz infrarroja, no tener un ángulo de visión adecuado o la imagen puede ser distorsionada. Los sensores IMU pueden integrar un magnetómetro, mismo que es susceptible a campos magnéticos además, los giroscopios deben incorporar un filtro pasa bajas para asegurar que no sean susceptibles a movimientos repentinos.

- Error acumulado

  Los sensores IMU tienen el inconveniente de añadir un error de derivada, este error es causado por la acumulación del error de compensación de sesgo en la lectura del giroscopio. Dicho error debe ser reducido mediante algún algoritmo externo al sensor IMU. Es posible emplear diferentes configuraciones de cámaras como lo son monoculares y estereoscópicas entre otras, pero es necesario efectuar un cálculo de profundidad, dicho cálculo añade un error en la estimación de profundidad.

- Recursos computacionales limitados

  Es deseado que el sistema mantenga un trayecto a una velocidad constante. Por lo que, la tasa de refresco debe ser suficiente para este efecto. Otro proceso que debe ser asegurado es el mapeo, por lo que también debe ser considerada una cantidad de memoria suficiente para este proceso.

Lo que se busca es que un veh´ıculo no tripulado pueda circular por un camino no conocido (sendero) en un entorno abierto y que devuelva una reconstrucción 3D del mismo. A par-

tir del trabajo de (Szklarski, Ziemiecki, Szałtys, y Ostrowski, 2019) podemos deducir que técnicas de reconstrucción 3D son más viables en este proyecto.

1.3. Hipótesis

Es posible mejorar la precisión en la medición de longitud y obtención de trayectoria de un sendero utilizando técnicas de procesamiento de imágenes y algoritmos de inteligencia artificial con el objetivo de dimensionarlo y reconstruirlo con técnicas SLAM.

1.4. Antecedentes

La detección de caminos podría ser considerada inicialmente como una tarea simple que se resuelve como una detección de color, por ejemplo para determinar el limite de un carril se puede buscar el color amarillo en las señales de piso. Sin embargo, el problema se vuelve más complejo a medida que disminuimos el contraste entre carriles o simplemente no existe una diferencia clara entre el camino a seguir y el limite del mismo.
Un metodología más robusta se basa en el uso de segmentación de imágenes con la finalidad obtener una imagen procesada resaltando los atributos buscados en la imagen, a continuación, en la Figura 1.2 se muestran algunos de los algoritmos de segmentación más significativos.

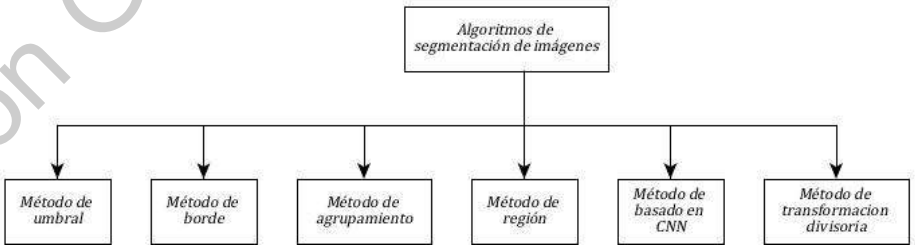


Figura 1.2: *Algoritmos de segmentación.*

Método de umbral Es el método más simple para la segmentación de imágenes. Este método consiste en analizar cada píxel y separarlos según su intensidad. Este método es

efectivo cuando existe una clara diferencia entre el objeto a segmentar y el fondo. Puede desarrollarse de manera manual o automática, añadiendo información de las características de la imagen. Existen tres tipos de umbrales (Lindeberg y Li, 1997):

1. Umbral global: Se logra usando un umbral apropiado de valor T. El valor de T puede considerarse constante en toda la imagen. Empleando T se obtiene una imagen de salida q(x, y) a partir de la imagen de entrada p(x, y) de la siguen te manera.

$$q(x,y) = \begin{cases} 1, & si \quad p(x,y) > T \\ 0, & si \quad p(x,y) \leq T \end{cases} \tag{1.1}$$

2. Umbral variable: En este tipo de segmentaci ón el valor del umbral varia a lo largo de la imagen. Podemos definir dos tipos de umbral variable:

   - Umbral local: En este caso el valor de T depende de los vecinos de x y y.
   - Umbral adaptativo: En este caso el valor de T es una funcion de x y y.

3. M últiples umbrales: En este tipo de umbral se tienen diferentes valores de umbral por ejemplo para T_0 y T_1 se tendría la función mostrada a continuación:

$$q(x,y) = \begin{cases} m, & si \quad p(x,y) > T_1 \\ n, & si \quad p(x,y) \leq T_1 \\ 0, & si \quad p(x,y) \leq T_0 \end{cases} \tag{1.2}$$

Método de detección de borde La segmentación basada en el método de bordes consiste en la detección de un cambio rápido de intensidad de una imagen dado que un solo cambio de intensidad no proporcionaría la información adecuada. La técnica de detección de bordes localiza los bordes donde la primer derivada de intensidad es mayor a un umbral determinado o donde la segunda derivada tiene cruce por cero. En los métodos de segmentación por bordes primero se buscan los bordes para después conectarlos entre si de modo que se puedan determinar los limites del objeto y as´ı segmentar las regiones requeridas. Se determinan

dos técnicas básicas de segmentación por bordes las cuales son: el método de histograma de grises y el método basado en gradiente. Para determinar los bordes pueden emplearse técnicas básicas como el operador Sobel, el operador Canny o el operador de Robert. Estas son técnicas basadas en detección de discontinuidades, (Al-Amri, Kalyankar, y Khamitkar, 2010).

Método de agrupamiento El método de segmentación basado en agrupamiento (Clustering) es una técnica que forma grupos de píxeles que comparten características. La agrupación de datos es un método que divide los elementos en grupos de modo que los elementos en el grupos on más similares entre si que entre otros grupos. Hay básicamente dos métodos de agrupamiento: método jerárquico y método basado en particiones. Los métodos jerárquicos se basan en el concepto de arboles, en el cual la raíz representa toda la base de datos y los nodos internos representan las agrupaciones. Los métodos basados en particiones emplean métodos de optimización de forma iterativa con la finalidad de minimizar una función objetivo. Podemos determinar dos tipos básicos de agrupación, (Dehariya, Shrivastava, y Jain, 2010).

- Hard clustering: Es una técnica de agrupamiento simple que divide la imagen en diferentes grupos con la finalidad de que un p´ıxel solo pueda pertenecer a un grupo. Estos métodos emplean funciones de pertenencia con valores de 1 o 0, osea que un píxel puede pertenecer a un grupo particular o no. Un ejemplo de esta técnica es el agrupamiento de k-medias, en la cual primero se calculan los centros y después se asigna cada píxel a su centro más cercano para después recalcular el nuevo centro.
- Soft clustering: El método de agrupamiento suave se considera más apegada a la realidad ya que determinar una pertenencia total es más difícil de conseguir. El agrupamiento suave considera valores de pertenencia parcial, es decir que un elemento puede pertenecer a dos grupos con diferente nivel de pertenencia. Un ejemplo de esta técnica es el algoritmo Fuzzy C-means en el cual un píxel puede pertenecer tiene un

valor de pertenencia para cada grupo siendo 1 un valor de pertenencia máximo y un valor de pertenencia m´ınimo.

Método de región Existen dos técnicas básicas basadas en segmentación por región en los cuales se divide la imagen en diferentes regiones de caracter´ısticas diferenciables, (Angelina, Suresh, y Veni, 2012).

1.M étodo de crecimiento de región: Los métodos basado en crecimiento de región se emplean para segmentar la imagen en diferentes regiones según la crecimiento de las semillas iniciales (p´ıxeles iniciales). Las semillas pueden elegirse de manera arbitraria (según el conocimiento previo del sistema) o de manera automática. Después, el cultivo de las semillas se controla mediante los p´ıxeles vecinos y con ayuda delo conocimiento previo del problema. Los pasos básicos par el método de crecimiento de región (conectado-8) se muestra a continuación:

   Si $p(x, y)$ es la imagen original a segmentar y $s(x, y)$ es la imagen binaria donde las semillas se localizan. Tomando a T como el predicado que sera probado para cada posición (x, y).

   - Primeramente, todos los componentes junto a s son erosionados.
   - Calcular una imagen binaria P_T. Donde $P_T(x, y) = 1$, si $T(x, y) = \text{Verdadero}$
   - Calcular una imagen binaria q. Donde $q(x, y) = 1$, si $P_T(x, y) = 1$ y (x, y) esta conectado-8 a la semilla en s.

   Los componentes en q son las regiones segmentadas.

2.M étodo de división y fusión de regiones (spitting and merging): Los métodos de segmentación de imágenes basado en división y fusión de regiones emplea dos técnicas básicas, dividir y combinar para segmentar en diferentes regiones. Spitting significa dividir iterativamente una imagen en regiones que tienen caracter´ısticas similares y Merging se encarga de combinar las regiones similares adyacentes. La Figura 1.3

muestra el diagrama formado al emplear una división de árbol cuádruple. Los pasos básicos del algoritmo de crecimiento de región y fusión se muestran a continuación:

Teniendo a p como la imagen original y a T como al predicado seleccionado.

- Primeramente R_1 es igual a p.
- Cada región es dividida en cuadrantes para cada $T(R_i) = Falso$.
- Si para cada región $T(R_i) = Verdadero$, luego combinar las regiónes adyacentes R_i y R_j de modo que $T(R_i \cup R_j) = Verdadero$
- Repetir el paso anterior hasta que la unión sea imposible.

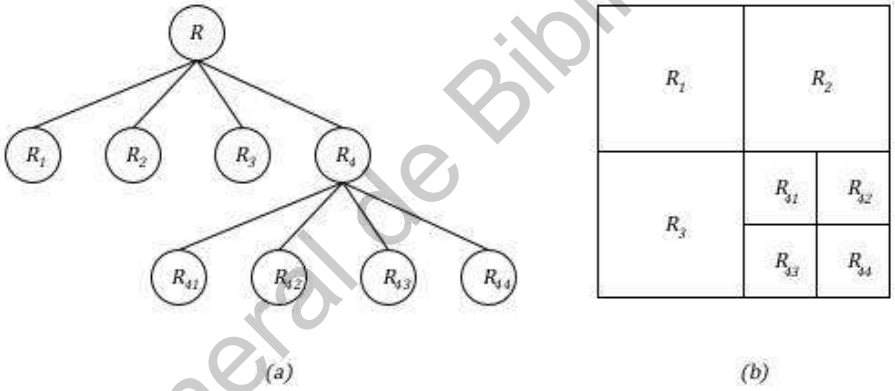


Figura 1.3: *División de regiones basado en árbol de cuadrantes*, (Gurusamy y cols., 2013)

En la Figura 1.3 se muestran los incisos *(a)* y *(b)*. *(a)* muestra la correspondencia de las divisiones y subdivisiones y *(b)* muestra la relación que tienen las subdivisiones en una imagen.

Método basado en CNN El método de segmentación basado en redes neuronales convolucionales (CNN) de imágenes tiene dos paso básicos: la extracción de características y segmentación por redes neuronales, (Senthilkumaran y Rajesh, 2009). En general los métodos basados en redes neuronales tratan de imitar la forma de aprendizaje del cerebro humano.

La forma en como se extrae información de las imágenes es aplicando una serie de convoluciones a la imagen de modo que la información de la misma se vea transformada, se reduce la longitud de y se amplia la profundidad, después aplicando las operaciones opuestas se reduce la profundidad y se regresa la imagen al tamaño original. En un proceso de entrenamiento de la red se requiere del conocimiento del comportamiento que la red debe tener, es decir es un proceso supervisado, se requiere de una imagen de entrada y una imagen objetivo. Al terminar el proceso de la red convolución se tendrá una imagen generada y los pesos de la red se ajustaran de acuerdo a la relación existente entre la imagen generada y la imagen objetivo, (Senthilkumaran y Rajesh, 2009).

Método de transformación divisoria Los métodos de segmentación de imágenes basados en transformación divisoria (Watershed) emplean conceptos de interpretación topológica. De modo que los valores más bajos o cuencas se diferencian por ser los puntos donde se derrama el agua y cuando el agua llega al borde de ka cuenca, las cuencas adyacentes se fusionan. Los limites de la segmentación representan la separación entre cuencas y para mantener dicha separación se requieren presas. Las presas se forman con la dilatación. Los métodos de transformación divisoria emplean el gradiente de imagen como superfi- cie topológica, los píxeles que tienen más gradiente se representan como limites continuos, (Kang, Yang, y Liang, 2009).

Comparación entre métodos de segmentación En la Tabla 1 se muestra una comparación de las ventajas y desventajas más notorias de cada método de segmentación.

Tabla 1: Comparación de métodos de segmentación.

Técnica de segmentación	Ventaja	Desventaja
Método de umbral	No requiere de información previa.Es de implementación simple.	Es altamente dependiente de la existencia de picos. Puede considerarse un método de prueba y error.
Método de borde	Es efectivo con imágenes de altocontraste entre objetos.	No es efectivo si se tienen muchos bordes.
Método de agrupamiento	Tiene un funcionamiento más parecido a la realidad al usar membres´ıa parcial (fuzzy).	Determinar una función de membresía adecuada no es simple.
Método de región	Es menos susceptible a ruido. Es más efectivo si se puede determinar un criterio de similitud.	Es un método caro en términos de tiempo y memoria.
Método basado en CNN	Se adapta al funcionamiento deseado ya que trata de minimizar una función de pérdida que interactúa con la imagen objetivo.	Funcionamiento no conocido con entradas fuera de lo esperado. El procesamiento de las imágenes puede ser lento por la cantidad de operaciones requeridas.
Método de transformación divisoria	El resultado suele ser más estable.Los límites detectados son continuos.	El cálculo de gradientes es complejo.

1.4.1. Vehículos autónomos no tripulados

En la actualidad una de las aplicaciones de la robótica con más auge son los vehículos no tripulados, entre las aplicaciones se encuentran:

- UAV: Vehículos aéreos no tripulados.
- USV: Veh´ıculos terrestres
- UUV: Vehículos acuáticos no tripulados
- ROV: Veh´ıculos operados remotamente

Destacamos los UAVs por su practicad y mayor aplicación en áreas como la industria militar o incluso en aplicaciones civiles. Este tipo de veh´ıculos son empleados en tareas como monitoreo de trafico, búsqueda y rescate, entrega de producto, vigilancia, agricultura de precisión e inspección de infraestructuras (Shakhatreh y cols., 2019).

1.4.2. Aplicación de la segmentación en la literatura

Uno de los trabajos más relacionados con este trabajo es el mostrado en (Smolyanskiy, Kamenev, Smith, y Birchfield, 2017), donde se emplea dos redes convolucionales profundas (CNN), dedicadas a la detección de caminos y detección de objetos. La red que se encarga de detectar el camino para determinar la acción necesaria para alinear el centro del camino con el centro de la imagen. Emplea una arquitectura s-ResNet-18 (He, Zhang, Ren, y Sun, 2016) para determinar si el veh´ıculo se encuentra dentro o fuera del camino, as´ı como realizar las acciones necesarias.

El mapeo del área silvestre tiene gran importancia en la comunidad científica, ya que se puede registrar información del deterioro o progreso del medio ambiente. Debido a esto, trabajos como el de (Pierzchała, Giguère, y Astrup ((2018))) (Pierzchała y cols., 2018) empleando un algoritmo de localización y mapeo simultáneo (SLAM por sus siglas en inglés), se generan mapas de bosque. Los sensores empleados son un LIDAR (Detección y rango

de imágenes láser) (Velodyne VLP-16), una cámara estéreo, una IMU (Unidad de medición inercial) y un GPS (Sistema de posicionamiento global). El mapa resultante, en nube de puntos 3D, fue evaluado en términos de precisión y exactitud teniendo un error de estimación promedio de 2cm. El algoritmo de mapeo empleado fue graph-SLAM, debido a que reduce la complejidad computacional respecto a un $O(n^2)$ del filtro Kalman, que además requiere del mapa m todos los n puntos de referencia. Para el enfoque graph-SLAM cada nodo del grafo representa una posición particular del robot, mientras que los bordes codifican restricciones de odometr´ıa o cierres de bucle.

En el trabajo de (Chen y cols. ((2020)))(Chen y cols., 2020) se busca conocer la especie y el diámetro de vegetación en un bosque, mostrando que métodos no se desempeñan de manera satisfactoria en este tipo de entornos, por ejemplo LIDAR capta mucho ruido. El algoritmo propuesto en este trabajo es SLOAM (Semantic Lidar Odometry and Mapping). El propósito del algoritmo de odometr´ıa lidar es estimar el movimiento r´ıgido de 6-DOF (grados de libertad) del lidar dentro de un barrido de 360 grados, mientras que el propósito del algoritmo de mapeo lidar es estimar la posición de 6-DOF del lidar en el mundo, y registrar la nube de puntos en el marco mundial.

El trabajo de (Yang y cols. ((2019))) (Yang y cols., 2019) también tiene gran relevancia para este proyecto, tanto por la orientación que tomó (ADAS, Sistema Avanzado de Asistencia al Conductor), como por los resultados obtenidos. En este trabajo proponen una combinación de estructuras CNN (Redes Neuronales Convolucionales) y LSTM (Memoria de largo y corto plazo) mostrando previamente el desempeño de ambas por separado. Se probó con la base de datos KITTI (Geiger, Lenz, y Urtasun, 2012) obteniendo un precisión promedio de 91.6 %. Una de las desventajas del modelo propuesto es la reducción de su desempeño en condiciones de poca luz y cuando hay presencia de sombras.

Anteriormente la forma común de estimar la posición de un cuerpo en un trayecto era mediante encoders pero existía un error asociado a la diferencia de retracción entre las llantas, que se volvía aun más notorio en entornos abiertos. Una de las soluciones propuestas fue la integración de múltiples tecnologías como lo es el radar, GPS, giroscopios y acelerómetros,

entre otras, (Fernandez y Price, 2004).

Una aplicación de la visión computacional es la navegación automática de vehículos y robots ya que en un uso efectivo para conseguir la detección de obstáculos y localización, lo cual esta sumamente relacionado con el SLAM (Simultaneus Localisation and Mapping) (Nister, Naroditsky, y Bergen, 2004), por lo que es esencial incorporar este tipo de algoritmos para asegurar que el funcionamiento del sistema es el adecuado ante percances como obstáculos en el camino.

También el uso de computadoras de vuelo se convierte en una necesidad, pese a las caracter´ısticas limitadas respecto a computadoras convencionales, para dividir la carga de operaciones ya que los controlados de vuelo actúan como cajas negras que responden de determinada manera ante ciertos parámetros de entrada como giro en determinado eje. Lo que se busca es que la parte robusta del sistema sea llevada por la computa adora de vuelo y que el control de los actuadores sea llevado por el controlador de vuelo.

1.4.3. Visión por computadora

La visión por computadora consiste en la extracción automática de información en las imágenes. La información puede ser variada desde patrones repetitivos hasta incluso la profundidad de los elementos en la imagen. De cierto modo la visión por computadora trata de imitar la visión humana pero con diferencias muy específicas ,por ejemplo, se sabe que el cerebro humano procesa la información visual principalmente en el espacio semántico extrayendo características semánticamente significativas mientras que las computadoras tienen que extraer información visual en el espacio de datos formado por características robustamente detectables pero menos significativas como colores o texturas (B. Zhang, 2010).

La investigación sobre la visión por computadora lleva más de 60 años y se tienen múltiples aplicaciones, a continuación se muestran algunas recopiladas de (Csurka, 2017):

- Inspección robotizada: Es una inspección rápida de las piezas para garantizar la calidad de los componentes de fabricación, requiere de iluminación especializada.
- Imágenes médicas: Se emplea para detectar tumores o diferentes afecciones a partir

de radiograf´ıas o encefalogramas.

- Vigilancia: Como monitoreo de intrusos, análisis de la vialidad y monitoreo de piscinas para v´ıctimas de ahogamiento.
- Detección de caras: Se logra resaltando características de los rostros tales como dimensiones entre nariz y ojos.

El gran avance que a tenido la ciencia computacional dio paso a nuevas técnicas tales como las redes neuronales y redes convolucionales que actúan como cajas negras las cuales mediante procesos de entrenamientos permiten a los sistemas adaptarse para reconocer patrones en las imágenes. A continuación, abordan dos temas fundamentales para llevar a cabo el reconocimiento y detección de objetos.

1.4.4. Segmentación con CNN

La segmentación una técnica de separación de estructuras de interés de un fondo y entre ellas (Rogowska, 2000). Típicamente, la segmentación de un objeto es alcanzada mediante la separación de los píxeles o vóxeles que componen el objeto o localizando los píxeles que rodean al objeto.

La segmentación no es propia de imágenes también existen técnicas de segmentación de datos tales como el agrupamiento (clustering) y técnicas como el fuzzy c-mean clustering surgieron en la década de los 80’s. En el trabajo de (Zhao y cols., 2019) se aborda la segmentación de vegetación endémica del interior de china empleando agrupamiento k-means y una técnica de algoritmos genéticos llamada AFSA (Artificial Fish Swarm Algorithm) con la finalidad de mejorar la precisión en la segmentación corrigiendo la sensibilidad inicial al centro y la optimización local de K-means mediante la optimización global de AFSA.

Entre las técnicas actuales de segmentación se encuentran aquellas que emplean redes convolucionales. Entre las arquitecturas de este estilo destacamos Mask R-CNN (He y cols., 2017) que pese a ser una arquitectura presentada en el 2017 sigue siendo una referencia para arquitecturas de actualidad. Mask R-CNN es un método de segmentación de instan-

cias de objetos. El algoritmo detecta los objetos en la imagen y simultáneamente genera una máscara de segmentación. El método es una extensión del método Faster R-CNN (He y cols., 2017). Alcanza un procesamiento de 5 imágenes por segundo. El diagrama de conexión entre módulos del método Mask R-CNN se muestra en la Figura 1.4.

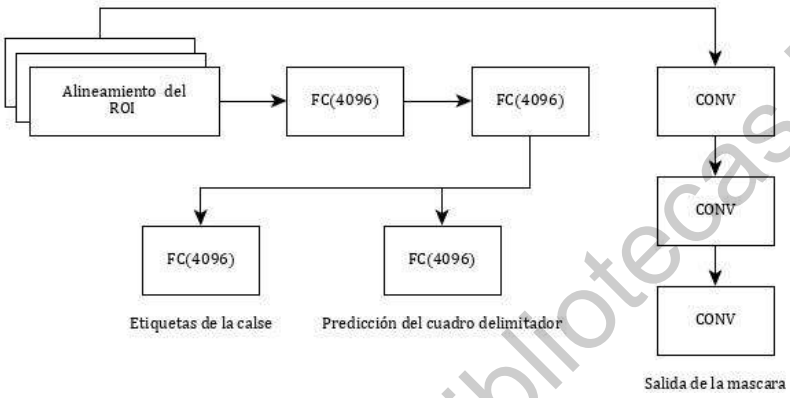


Figura 1.4: *Arquitectura Mask R-CNN, adaptada de* (He y cols., 2017)

La Figura 1.4 puede desglosarse en los siguientes pasos:

1. Extracci ón de características de la imagen. Mandada a un modelo de red de extracción pirámide multiescala para obtener un mapa de característica para después filtrarla.

2. Generar ROI (Region of Interest, Regi ón de Interés). Teniendo un número predeterminado de regiones de interés para cada punto en el mapa de características, siendo cada ROI un candidato.

3. Red de regi ón propuesta (RPN). La región de interés candidata es enviada a red de región propuesta para realizar el cálculo del cuadro delimitador, también se filtran las regiones de interés sin objetos.

4. Alineaci ón del cuadro delimitador. Un algoritmo bilinear de interpolación es adaptado para completar la alineación del cuadro delimitador con la finalidad de obtener la segmentación píxel-nivel.

5. Clasificaci ón y regresión. La ROI candidata obtenida es clasificada, delimitada por el cuadro y su máscara es generada.

Entre los trabajos que emplean este método se encuentra (Songhui, Mingming, y Chufeng, 2019) que une el algoritmo de detección y localización basado en Mask RCNN y visión estéreo. Lo que se buscaba era emplear un algoritmo que identificara el tipo de fruta y la ubicación de la misma de manera precisa y rápida. El sistema fue implementado en una Nvidia Jetson TX1 obteniendo mejores resultados en precisión a distancias de 0.6 metros. Uno de los datos que más destacan es que, pese a que la detección de plátanos a 0.6 metros alcanza una precisión de 99 %, el tiempo que tarda en reconocerlo es superior al de las otras frutas.

Otra aplicación de Mask RCNN es la reducción de congestión vehicular identificando daños vehiculares mediante segmentación. Lo que se busca es localizar en que parte de un vehículo se tiene daño (Q. Zhang, Chang, y Bian, 2020). En el trabajo se menciona que, a pesar de que la literatura menciona mejores resultados con un backbone resnet101 de 101 capas convolucionales, se implemento resnet50 como backbone debido a que demasiadas capas reducirían en gran medida la velocidad de la estructura de la red. Además de este cambio a la arquitectura Mask RCNN también se unieron los módulos de extracción de características y la FPN (Feature Pyramid Network) teniendo como resultado mayor precisión a diferentes distancias respecto a la MAsk RCNN propuesta por (He y cols., 2017).

Finalmente una aplicación de esta arquitectura en un vehículo aéreo no tripulado se muestra en (Subash, Srinu, Siddhartha, Harsha, y Akkala, 2020), el proyecto emplea un drone comercial para segmentar y clasificar objetos, se hace mucho énfasis que que las bajas condiciones de luminosidad fueron un reto significativo pero no un impedimento obteniendo detecciones a una frecuencia de 7.12 cuadros por segundo y una precisión de detección máxima de 0.99 y promedio de 0.956. Los objetos detectados fueron muebles y personas.

También es posible segmentar una imagen empleando una arquitectura encoder-decoder las cuales, mediante entrenamiento, pueden interpretar una imagen de entrada como una imagen objetivo. El trabajo de (Isola y cols., 2017)., posterior a la publicación del software

Pix2Pix, es considerado una traslación de imagen a imagen ya que se reinterpreta una imagen de entrada de manera similar a la reinterpretación que se le da al escuchar una lengua extranjera, manteniendo significado.

Uno de los trabajos más relacionados con este proyecto es el mostrado en (Smolyanskiy y cols., 2017) dado que emplea dos redes convolucionales profundas (CNN) dedicadas a acciones diferentes detección de caminos y detección de objetos. La red que se encarga de detectar el camino determina la acción necesaria para alinear el centro del camino con el centro de la imagen. Aunque no se espec´ıfica la arquitectura empleada, se puede deducir las caracter´ısticas de misma destacando los dos procesos necesarios:

- Segmentación del camino: Separación de la información en la imagen que este asociada a las caracter´ısticas propias del camino.
- Posición del camino respecto a la imagen: La posición del camino determinará la acción necesaria.

Los procesos de segmentación y posicionamiento no necesariamente debe estar separados ya que si la base de datos contiene imágenes etiquetadas como ”Muy a la derecha”, Çentro”, etc puede resolverse cono un problema de extracción de características con una capa final de que reduzca la dimensionalidad al de la clasificación.

El trabajo de (Yang y cols., 2019) también tiene gran relevancia para este proyecto tanto por la orientación que tomó (ADAS) como por los resultados obtenidos. En este trabajo proponen una combinación de estructuras CNN y LSTM mostrando previamente el desempeño de ambas por separado. Se probó con la base de datos KITTI (Geiger y cols., 2012) obteniendo un precisión promedio de 91.6 %. Una de las desventajas del modelo propuesto es la baja sensitividad en condiciones de poca luz así como la alteración cuando hay sombras.

1.4.5. Estimación de la posición

Se puede realizar la estimación de la posición de diferentes maneras, mostradas a continuación, pero todas los métodos de estimación de la posición tienen un inconveniente y es la

susceptibilidad al ruido de diferentes fuentes. Un magnetómetro cerca de un motor eléctrico presentará una alteración en las lecturas, un GPS dentro de un edificio no podrá conectarse a ningún satélite y no podrá mostrar ningún registro o un giroscopio montado en una superficie plana puede mostrar un error de deriva. Todas estas afecciones deben minimizarse mediante estimadores como el filtro Kalman que determinan el nivel de confianza de una lectura y al comprarlo con la estimación se determina cual tiene un mayor nivel de verdad. El trabajo de (Dai y cols., 2020) muestra que una forma de robustecer el sistema de estimación de la posición es mediante la fusión de diferentes sensores, esto no solo minimiza el error en las mediciones sino que además ayuda contra fallas en los sensores.

Se puede clasificar a los métodos de estimación de posición por fusión de sensores en dos categorías: basados en filtros y basados en optimización. Los métodos basados en filtros como lo son el filtro Kalman (KF) y su extendido (EKF) y las variantes unscented (UKF, sin traducción directa) normalmente se basan en estimaciones posteriores máximas. Los métodos basados en filtros emplean diseños modulares para manejar la sincronización y retraso entre los diferentes sensores lo que garantiza un desempeño en tiempo real. Sin embargo los métodos basados en filtros comúnmente suponen que el estado satisface la hipótesis de Markov de primer orden, la cual es que el estado actual esta relacionado con el estado anterior, por otro lado los estados más antiguos son superfluos para estimar el estado actual. Los métodos basados en filtros son ventajosos cuando procesan sensores globales, pero funcionan de manera subestima en comparación con los métodos basados en optimización, (Dai y cols., 2020).

Por otro lado los métodos basados en optimización asumen que los estados actuales satisfacen la hipótesis multi orden de Markov, lo cual es que todos los estados pasados pueden ser usados para estimar los estados actuales. Entre los algoritmos basados en optimización se encuentra OKVIS, VINS-Mono, mientras que el algoritmo Rehder logra la estimación de posición global fusionando GPS y visión estéreo, este método mejora la precisión de la estimación.

1.4.6. SLAM

Los sistemas basados en SLAM se han convertido en tendencia con el desarrollo de teor´ıas de localización, (Z. Zhang y Wan, 2018). Los sensores que actualmente son empleados para la adquisición de información del entorno son cámaras monoculares, RGB-D y cámaras binoculares. También se explica como las cámaras monoculares, al no entregar la profundidad del píxel, es prácticamente imposible determinar la escala real del movimiento del robot, las cámaras RGB-D, que si entregan la profundidad del píxel, son susceptibles a errores pa- ra profundidades muy grandes y son afectadas por la luz infrarroja además el consumo de energía es grande y las cámaras binoculares o cámaras estéreo calculan la profundidad del p´ıxel de manera indirecta lo que conlleva un error pero no tienen problemas de consumo de energ´ıa ni se ven afectadas por la luz infrarroja. En el mismo articulo se aborda la odometr´ıa visual, lo cual es una parte importante de los algoritmos SLAM.

Un sistema que integra más de una tecnología de medición es propuesto en (Szklarski y cols., 2019), en el que la incorporación de odometría inercial y visual, solventa los bajos recursos computacionales con que cuenta una placa de desarrollo comercial. En este trabajo se menciona que la combinación de sensores visuales e IMU (Inertial Motion Unit, sistema basado en un acelerómetro y/o un giroscopio) aún presentan problemas de desviación pero esto puede ser solventado incorporando algún proceso de mapeo y localización .

Entre las formas de reproducir el entorno en 3D es el uso de representaciones volumétricas del espacio divididos en vóxeles, donde cada vóxel describe alguna propiedad espacial, esta técnica es especialmente buena en cuestiones de gestión de memoria y velocidad de acce- so a memoria. Cabe resaltar que los componentes de este proyecto fueron dos IMUs, una cámara de ojo de pez y una cámara de profundidad, (Lim y Suter, 2009).

Un trabajo similar al anterior que incorpora visión monocular y sensores de inercia para el cálculo de la odometría es mostrado en (Bloesch, Omari, Hutter, y Siegwart, 2015), donde menciona que la incorporación de mediciones de inercia en la estimación aumenta significativamente la robustez del sistema y adaptando el filtro Kalman Extendido (EKF) propuesto por (Davison, 2003), que puede ser integrado de manera relativamente simple dentro de la

estimación ego-motriz. Otro beneficio es que la calibración de parámetros pueden ser co-estimados en línea. Finalmente, como parte de la conclusión se propone la integración de métodos heurísticos para evitar la divergencia.

Un sistema basado en deep learning es propuesto en (Muller y Savakis, 2017). El sistema fue entrenado con la base de datos de odometr´ıa visual KITTI, (Geiger, Lenz, Stiller, y Urtasun, 2013). Además, se probaron diferentes configuraciones de arquitecturas de redes neuronales para mejorar la exactitud. El sistema arrojó un error traslacional promedio de 10.77 % y un error rotacional de 0.0623 grados por metro.

En (Cheng y cols., 2019) se propone la integración de Red de aprendizaje Kalman (LKN por sus siglas en inglés) basada en odometría visual monocular que a comparación del fil- tro Kalman, los parámetros óptimos del modelo pueden ser obtenidos de salidas de la red neuronal dinámicas y determinísticas. La principal desventaja de este sistema es el alto consumo de recursos computacionales.

Entre los trabajos de mapeo de entornos abiertos se encuentra (Lee, Park, y Song, 2011), quienes proponen un diseño de localización y mapeo simultáneo (SLAM) basado en correspondencia de mapa 3D, ya que es necesario lidiar con la complejidad de los entornos, dado que la cantidad de datos que se pueden adquirir es grande, y lograr una correspondencia de mapeo precisa se convierte en una tarea desafiante. Para solventar esto proponen un método para seleccionar los datos 3D útiles para la estimación de la posición y así generar descriptores que sean invariantes a cambios de rotación. Este método es particularmente bueno para robots autónomos ya que reduce la cantidad de datos a procesar y con esto el tiempo de procesamiento también se ve reducido.

En el trabajo de (Nüchter, Lingemann, Hertzberg, y Surmann, 2007) para mapeo de entornos abiertos se consideran los tres ejes (x, y y z) así como tres ángulos (guiñada, alabeo y cabeceo) llamado 6D-SLAM. en este trabajo, mediante heur´ıstica se alinearon los escaneos. El mapa 3D final fue optimizado mediante una forma cerrada requiriendo de 77 corridas y cada corrida estaba compuesta de aproximadamente 100000 puntos. Los resultados fueron comparados con imágenes aéreas.

Una serie de aplicaciones como lo es la conducción autónoma, robots de rescate, robots de servicio entre otras son alcanzables mediante el mapeo de entornos y localización (Dubé y cols., 2019) además, conociendo la ubicación precisa se puede garantizar robustez y operaciones seguras. En este trabajo se presenta un diferente enfoque para la representación de mapas para los problemas de localización y mapeo para nubes de puntos 3D LIDAR (Light Detection and Ranging). La técnica SegMap (Dubé y cols., 2020) se forma a partir de la partición de nubes de puntos en segmentos de conjuntos descriptivos. Esta técnica combina las ventajas de los descriptores de caracter´ısticas locales y globales entregando la localización de 6-Dof (seis grados de libertad) global precisa en tiempo real. El entrenamiento se logró mediante la base de datos de odometría KITTI, (Geiger y cols., 2013). Esta técnica entrega la reconstrucción 3D del entorno y la extracción de información semántica.

El uso de cámaras estéreo también es considerado en (Brand, Schuster, Hirschmüller, y Suppa, 2014) y (G. Zhang, Lee, Lim, y Suh, 2015) mediante pruebas en entornos abiertos y cerrados. En el trabajo de (G. Zhang y cols., 2015) se muestra mejores resultados en reconstrucción que los sistemas SLAM basados en puntos. Además, emplea diferentes técnicas para obtener mayor precisión como son: la estimación de movimiento, la optimización de posición y ajuste de paquetes. En el trabajo de (Brand y cols., 2014) se muestra que en entornos abiertos se tiene una precisión de 0.22m de error promedio usando diferentes cámaras (ángulo estrecho y amplio), además de una desviación de posición final menor al 0.08 %.

Entre los algoritmos SLAM más sobresalientes en la actualidad se encuentran ORB-SLAM2 (Mur-Artal y Tardós, 2017) y RTAB-Map (Labbé y Michaud, 2019), ambos discutidos en (Ragot, Khemmar, Pokala, Rossi, y Ertaud, 2019) quienes emplearon una cámara de profundidad Realsense 435D para realizar la comparación de algoritmos. Entre los resultados se destaca que el algoritmo ORB-SLAM2 pierde puntos durante el mapeo al emplear la cámara monocular. En entornos abiertos la cámara estéreo y RGB-D mostraron resultados precisos pero en entornos abiertos la cámara estéreo mostró mejores resultados. Entre la comparación de resultados de ambos algoritmos se destaca que ambos funcionan bien para estimación de trayectorias pero en la estimación de odometría es mejor ORB-SLAM2.

Entre los trabajos más relacionados a la propuesta de proyecto se encuentran aquellos que implementan un vehículo no tripulado, ya sea aéreo o marítimo (UAV o UUV respectivamente) los trabajos más destacados son (Hong y Kim, 2016b) quien propone un algoritmo SLAM para la inspección autónoma de estructuras bajo el agua. Mientras que (Dou, Huo, Liu, y Wang, 2019) orienta su investigación a los UAV (vehículos aéreos no tripulados), en este sistema la estimación se hace mediante una algoritmo SLAM empleando una cámara monocular as´ı como odometr´ıa visual. El sistema fue probado con el conjunto de datos TUM, (Sturm y cols., 2011).

Entre los sistemas comerciales podemos encontrar funciones similares a las propuestas pero limitado en al menos dos aspectos, precio elevado y que sean de arquitectura cerrada. Esto conlleva que sean dif´ıciles de adquirir y que el procesamiento de los datos no pueda ser optimizado por el usuario. Entre los fabricantes destacamos tres:

- slamCore. El cual es un sistema montado en un veh´ıculo a radio control que tiene la capacidad de devolver la ruta por la cual ha pasado, además de un mapa de puntos que solo permite determinar si el vehículo tiene un obstáculo muy grande cerca. El precio no es conocido.

- DJI. DJI fabrica drones de alta calidad y prestaciones de punta. Los sistemas de DJI poseen cámaras de alta resolución para el mapeo aéreo y los modelos diseñados para la agricultura poseen la función de recorrer una trayectoria desganada para regar la plantación. Los precios fluctúan entre 2,500 y 13,000 dolares americanos.

- Foxtechfpv. Fsxtechfpv provee veh´ıculos destinados para el mapeo, el inconveniente es que este tipo de vehículos no puede ser empleado en áreas reducidas o con curvas muy pronunciadas ya que su utilidad es el mapeo aéreo. El precio de estos sistemas es de 7,599 dolares americanos.

2. Fundamentación Teórica

En esta sección se muestra evidencia científica de que respalde los diferentes componentes de este trabajo y se busca ampliar el entendimiento de los mismos. Entre los componentes podemos separar visión por computadora, navegación/censado y arquitecturas de segmentación basadas en CNN.

2.1. Visión por computadora

Visión por computadora es un proceso en el cual un componente tecnológico, maquina o computadora, procesa una imagen de modo que pueda determinar que hay en la imagen, (Snyder y Qi, 2017). La Figura 2.1 muestra un diagrama fundamental de como se lleva a cabo este proceso.

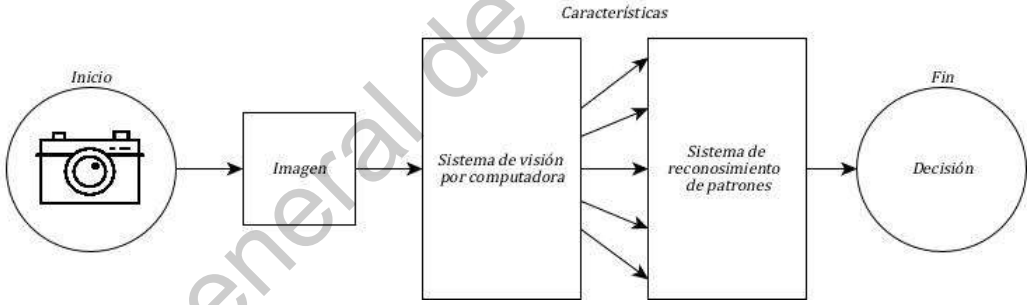


Figura 2.1: *Diagrama del proceso llevado a cabo en un sistema de visión por computadora*

En un sistema de *Comprensión de imagen* como el mostrado en la Figura 2.1 se tiene una imagen como entrada pero no una imagen a la salida por lo tanto a la salida de un sistema de reconocimiento de patrones se entrega una descripción de la imagen resaltando aquellas características más deseadas. En este sentido la decisión determinada por el sistema de reconocimiento de patrones esta basada en una serie de medidas hecha por el mismo, estas medidas son interpretaciones de las caracter´ısticas en la imagen.

2.1.1. Cámara

Se pueden catalogar tres tipos de cámaras de profundidad.

- Monocular

  Los píxeles de estas cámaras no tienen asignado un valor de profundidad por lo que al calcularlo implica incorporar un error.

- RGB-D

  Los p´ıxeles si tienen asignado su valor de profundidad pero tiene un alto consumo de energ´ıa y es susceptible a luz infrarroja

- Estereoscópica.

  Calcula la profundidad mediante las dos cámaras, no es susceptible a luz infrarroja, tiene un consumo energético moderado.

2.1.2. Imagen

Una imagen puede definirse como una compilación de información real o imaginaria representada de manera visual. Una imagen de un componente real debe ser captada por un instrumento que capture los rayos de luz reflejados en el objeto dentro de la imagen. Una imagen imaginaria puede ser producida por diferentes sistemas técnicas informáticas, como ejemplo la realidad virtual o aumentada. Una imagen también esta asociada a una componente temporal ya que captura la luminosidad reflejada por el objeto en un momento especifico el cual con bastante seguridad se puede afirmar que no volvera´ a suceder, (Bhabatosh y cols., 1977).

Imagen analógica Una imagen analógica es una representación visual de un objeto obtenida de manera remota la cual es obtenida por medio de instrumentos con la capacidad de grabar la luz reflejada por el objeto. La escena de la imagen analógica previo a ser capturada muestra una variación continua de tonalidad y luminosidad.

Imagen digital Una imagen digital es una conversión de una imagen analógica en su forma discreta de modo pueda ser interpretado por un sistema computacional, (Bhabatosh y cols., 1977). Una imagen se define como una función de dos dimensiones $f(x, y)$ el la cual tanto x como y corresponden a las coordenadas de un plano formado por todos los puntos y $f(x, y)$ es la amplitud en el punto (x, y) a la cual se le llama intensidad, (Gonzalez y Woods, 2003). Una imagen se compone por un número finito de elementos y cada elemento tiene una localidad y valor de intensidad distintivo. Estos elementos se consideran como la unidad m´ınima de medida en una imagen y son llamados p´ıxeles.

La resolución es uno de los parámetros esenciales al momento de decidir trabajar con una imagen. Este parámetro estará vinculado a la calidad de la imagen (cantidad de píxeles) así como al tamaño de la imagen (espacio requerido para almacenarla) y posteriormente al tiempo necesario en procesarla. Este parámetro se expresa como la cantidad de píxeles que tiene la imagen de anchura por la cantidad de p´ıxeles que tiene la imagen de altura, (Gonzalez y Woods, 2003).

Imágenes a color Una imagen a color es una composición de diferentes tres diferentes imágenes que comparten el mismo contexto. Para cada imagen que compone una imagen a color se le asocia un canal de color. Un ejemplo de la codificación que se le asigna a las imágenes a color es RGB (Red-Green-Blue) correspondientes a los colore rojo, verde y azul. En la Figura 2.2 se muestra como a partir de tres imágenes codificadas en formato RGB se puede formar una imagen a color.

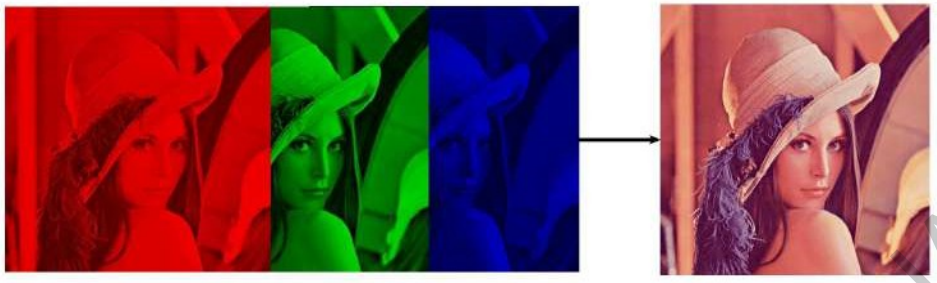


Figura 2.2: *Representación visual de cada canal de color y la imagen compuesta de los tres canales de color.*

En la Figura 2.2 se puede ver como al al sobreponer cada canal de color sobre los demás se forma un color diferente debido a la suma de intensidades de cada color. Cada p´ıxel en cada canal de color tiene un tamaño de hasta 8 bits dependiendo de la calidad de la imagen. Esto nos da aproximadamente 16.7 millones (2^{24}) de combinaciones de color.

2.1.3. Procesamiento de imágenes

El procesamiento de imágenes es un proceso diferente a visión por computadora en el cual se trata de mejorar o resaltar ciertos aspectos de la imagen para encentar los procesos posteriores por lo tanto los pasos de este proceso dependen meramente del problema que se quiere resolver. Principalmente se busca solventar tres problemas:

- Digitalizar y codificar: Con el propósito de facilitar la transmisión, representación y almacenamiento de las imágenes.
- Mejora y restauración: Resaltando las características que produzcan una mejor interpretación de la imagen.
- Descripción y segmentación: Paso previo a la visión por computadora que interpretara dichas métricas proporcionando un valor numérico propio del análisis desarrollado.

Podemos dividir el procesamiento de imágenes en tres grupos, (Gonzalez y Woods, 2003):

- Algoritmos de dominio espacial: Son los métodos que procesan una imagen píxel por píxel, o la interacción de un píxel con los píxeles vecinos.
- Algoritmos en dominio de frecuencia: Tomando en cuenta los resultados de un cálculo previo de la Transformada de Fuouier es posible aplicar correcciones a la imagen, por ejemplo aplicar filtros.
- Algoritmos de extracción de características: Estos algoritmos están enfocados en un análisis de la imagen en el cual se intenta resaltar atributos y regiones de interés (ROI) as´ı como separar elementos del fondo, detectar bordes, etc.

2.2. Navegación

El sistema de navegación es un componente empleado por diferentes vehículos con la finalidad de llevar a cabo tareas que van desde mantener la comunicación con el usuario hasta mantener la estabilidad del vehículo. Este sistema también se encarga de administrar los diferentes sensores empleados y transmitir los datos medidos.

2.2.1. IMU

El sensor Unidad de Medición Inercial, también llamado IMU (Inertial Measurement Unit) es muy empleado en aplicaciones de de manufactura, navegación y robótica, (Ahmad, Ghazilla, Khairi, y Kasi, 2013). Actualmente sistemas micro-electromecánicos IMU se encuentran a un costo bajo, en tamaños compactos y con bajos niveles de consumo energético.
Los elementos que constituyen a los sensores IMU son: acelerómetros y giroscopios. El acelerómetro se usa para medir la aceleración inercial mientras que el giroscopio se emplea para medir la rotación angular. A continuación se muestran las diferentes tecnologías de IMUs.

- IMU con dos tipos de sensores

  Consiste en la incorporación de un acelerómetro y un giroscopio. Comúnmente cada sensor tiene entre dos y tres grados de libertad que en conjunto pueden alcanzar hasta

seis grados de libertad. Los ángulos pueden ser medidos de manera separada, por ello los datos pueden ser calibrados. A pesar de combinar ambas tecnolog´ıas, es posible que la exactitud no aumente debido al ruido de los sensores y al problema de desviación del giroscopio

- IMU con tres tipos de sensores

  Este tipo de sensores incorporan un giroscopio, un acelerómetro y un magnetómetro. En total pueden alcanzar nueve grados de libertad. El magnetómetro se emplea para medir el ángulo de rotación *yaw*. La desventaja de emplear un magnetómetro es que el sistema se vuelve susceptible a campos magnéticos.

2.2.2. Sensores de distancia

Los sensores de distancia o transductores de posición lineal son dispositivos que permiten conocer la distancia que existe entre el sensor y un objeto. La clasificación de los sensores de distancia se divide de la siguiente manera.

- Grandes distancias.
  - Radar.
  - Ultrasónico.
  - Láser.
  - Magnetoresistivo.
- Cortas distancias.
  - LVTD.
  - Láser de triangulación.
  - Potenciométrico.

2.2.3. Filtro Kalman

El filtro Kalman es considerado un estimador o un filtro, dependiendo del retraso de las muestras. Sea considerado filtro o estimador, el ruido es minimizado, (Bravo, Arias, y Cardenas, 2013). El algoritmo aproxima el estado instantáneo de un sistema linear dinámico perturbado por ruido blanco. El filtro Kalman provee un promedio para inferir la información pérdida de medidas indirectas, (Grewal y Andrews, 2001). En esencia, el filtro Kalman es una serie de ecuaciones que implementan un estimador tipo predictor - corrector que minimiza el error estimado de la covarianza.

Algoritmo El filtro Kalman es un algoritmo que se basa en el modelo de espacio de estados de un sistema para estimar el estado futuro y la salida futura. El algoritmo tiene las siguientes caracter´ısticas, (Grewal y Andrews, 2001):

- No requiere de una frecuencia de corte espec´ıfica.
- Se basa en un método recursivo.
- Sus ecuaciones dependen de una muestra anterior y la muestra actual.

Ahora, supongamos la ecuación principal (2.1).

$$Z_t = H_t * x_t + u_t \tag{2.1}$$

Donde Z_t es un vector de variables observables, x_t es un vector de variables de estado y H_t es la matriz que reacciona a las variables observables con la de estado. El termino u_t se incluye para albergar la posibilidad de que las variables de que las variables observables pueden contener algún error de medición.
Ahora supongamos que la dinámica de las variables de estado viene determinada por la ecuación de estado dada en (2.2).

$$x_t = A_t * x_{t-1} + e_t \tag{2.2}$$

Donde e_t es un vector de errores que se incluye al modelar la incertidumbre de las variables de estado.

La ecuación de predicción esta dada por $x_{t,opt}$ un estimador óptimo del vector y P_t la varianza del vector de estados como muestran las ecuaciones (2.3) y (2.4):

$$x^p_{t+1,opt} = A_t * x_{t,opt} \tag{2.3}$$

$$P^p_{t+1} = A_t * P_t * A_t^{'} + Q \tag{2.4}$$

Estos términos son únicamente las predicciones de las variables atendiendo la dinámica que se ha supuesto que siguen.

Se mejoran los datos predichos utilizando información de las variables observables para $t+1$, ya conocida.

De (2.3),

$$x_{t+1,opt} = x^p_{t+1,opt} + K_{t+1} * (Z_{t+1} - Z^p_{t+1}) \tag{2.5}$$

$$x_{t+1,opt} = K_{t+1} * (Z_{t+1} - H_{t+1} * x^p_{t+1,opt} \tag{2.6}$$

De (2.4),

$$\hat{P}_{t+1} = (ID - K_{t+1} * H_{t+1}) * P^p_t \tag{2.7}$$

Donde k_t es el termino conocido como ganancia de Kalman y esta dado por:

$$K_t = P^p{}_t * H^{|}{}_t * (H_t * P^p_t * H^{|}_t + R_t)^{-1} \tag{2.8}$$

2.2.4. Modelo dinámico y cinemático de un drone

El uso de aeronaves controladas a distancia o de manera autónoma han tenido un gran impacto en diferentes áreas de interés entre ellas la agricultura, la vigilancia y en tareas de búsqueda y rescate tanto por su bajo coste como por su versatilidad de movimiento, (Dayoub,

Birech, Haghbayan, Angombe, y Sutinen, 2020; Hwang, Kim, y Lee, 2021; Mishra, Garg, Narang, y Mishra, 2020). Tanto los vehículos aéreos autónomos (UAV) como los controlados remotamente comparten los mismos componentes principales: un sistema que desarrolle en vuelo (frame, motores, controladores, etc), comunicación entre el vehículo y tierra, modulo de control en tierra. Es importante entender el funcionamiento de la estabilización de un drone o UAV para determinar las mejores acciones que permitan el vuelo pese a que no se busca desarrolla el controlador de vuelo.

De entre las diferentes configuraciones en que se pueden encontrar las UAV se eligió trabajar con la llamada *Quad-rotor* por ser una de las más exploradas. La configuración *Quad-rotor* consta de cuatro motores colocados en cada punta de un cuerpo en forma de x, tal como se muestra en la Figura 2.3. Para encontrar el modelo del mismo se trabajó con el método de aproximación Euler-Lagrange.

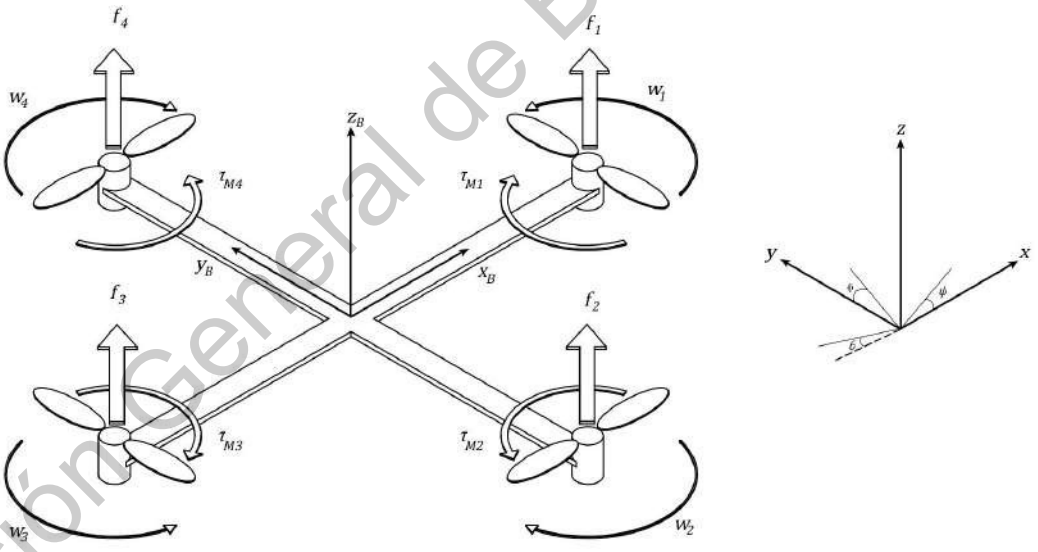


Figura 2.3: *Marco de referencia y fuerzas de un quad-rotor, modificado de* (Luukkonen, 2011).

En la Figura 2.3 se muestran x_B, y_B y z_B como el sistema de coordenadas móvil respecto al la tierra. En este sistema se muestra a x_B como el eje normal de ataque del drone, es decir

la dirección en la que el drone avanza. El sistema de coordenadas muestra los ejes x, y y z a los ejes fijos respecto a la tierra, (Luukkonen, 2011).

De acuerdo con el funcionamiento del drone, las hélices generan el empuje aerodinámico creado por la interacción entre la la rotación de las mismas sobre el fluido viscoso. Dicho empuje es necesario para mantener la aeronave en elevación. Por tanto tanto el empuje como la velocidad de rotación de los motores se relacionan mediante la siguiente expresión, en la cual k_i representa la constante de empuje y ω_i representa la velocidad angular del motor i.

$$f_i = k_i \cdot \omega_i^2 \tag{2.9}$$

Considerando que los motores están perfectamente alineados con respecto al sistema de coordenadas móvil, la dirección de empuje de las hélices será en el eje z del sistema de referencia fijo, en la cual la componente en los demás ejes es nula. Por lo tanto la suma de las fuerzas provocadas por las hélices será el empuje total que provocara el desplazamiento vertical del UAV. As´ı el empuje total se expresa en (2.10) mientras que el vector de fuerza traslacional de expresa en (2.11), (Luukkonen, 2011).

$$f = \sum_{i=1}^{4} f_i = k_i \cdot \sum_{i=1}^{4} \omega_i^2 \tag{2.10}$$

$$F = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \tag{2.11}$$

Para que el UAV gire sobre su propio eje cambiando su sentido de orientación es necesario que exista un momento. Este momento se desarrolla gracias al arrastre aerodinámico en el eje z. El giro es propiciado por la diferencia de velocidad de giro de los rotores, gracias a la fricción de las hélices con el aire generando un momento o torque en el sentido contrario a la dirección de giro de los rotores, (Luukkonen, 2011). De este modo el torque de giro en el eje z se ve expuesto en (2.12).

$$\tau_{Mi} = K_d \cdot \omega^2 + I_m \dot{\omega}_i \tag{2.12}$$

Donde K_d es la constante de arrastre y I_m es la inercia del motor. En algunos casos se desprecia el termino $I_m \omega_i$ dado que el efecto de ω_i en el sistema es muy pequeño, (Luukkonen, 2011). El valor de k_d esta´ dado por diferentes factores como la densidad del aire, la superficie del ala (hélice) y la velocidad de giro de la misma, (Quintana, Hassanalian, y Abdelkefi, 2018). De este modo reduciendo la primera derivada de la velocidad angular ($\dot{\omega}$) tenemos (2.13):

$$\tau_{Mi} = K_d \cdot \omega^2 \tag{2.13}$$

De este modo el momento total quedara´ denotado por la resultante del momento de cada rotor pero es necesario tomar en cuenta el sentido de giro de cada roto, por ello basándose en la Figura 2.3 consideramos que existen dos momentos positivos (2 y 4, dextrósum) y dos negativos (1 y 3, sinistrósum). De este modo el momento total τ_φ queda como (2.14):

$$\tau_\varphi = \sum_{i=1}^{4} \tau_{Mi} = K_d \cdot (-\omega^2{}_1 + \omega^2{}_2 - \omega_3^2 + \omega_4^2) \tag{2.14}$$

Una vez determinada la ecuación de momento para uno de los ejes del marco de referencias es posible determinar los momentos correspondientes para cada uno de los ángulos en el marco móvil (2.15). Para la cual es necesario considerar la distancia entre el rotor y el centro de más del UAV que denotaremos como l.

$$\tau = \begin{bmatrix} \tau_\varphi \\ \tau_\vartheta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l \cdot (f_4 - f_2) \\ l \cdot (f_3 - f_1) \\ \sum_{i=1}^{4} \tau_{Mi} \end{bmatrix} \tag{2.15}$$

Ahora nombramos ξ al vector donde alojaremos las coordenadas traslacionales o de posición lineal absoluta, y η al vector de coordenadas rotacionales o de posición angular para cada uno de los ángulos de Euler llamados y connotados por *Roll* (alabeo) φ, *Pitch* (cabeceo) ϑ y *Yaw* (guiñada) ψ.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \eta = \begin{bmatrix} \varphi \\ \vartheta \\ \psi \end{bmatrix} \tag{2.16}$$

Las matrices empleadas para representar un solido en rotación, o matrices básicas de rotación de un sistema espacial de tres dimensiones se muestran en 2.17, (Slabaugh, 1999).

$$R(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos_{\varphi} & -sin(\varphi) \\ 0 & sin(\varphi) & cos_{\varphi} \end{bmatrix}$$

$$R(\vartheta) = \begin{bmatrix} cos_{\vartheta} & 0 & sin(\vartheta) \\ 0 & 1 & 0 \\ -sin(\vartheta) & 0 & cos_{\vartheta} \end{bmatrix} \tag{2.17}$$

$$R(\psi) = \begin{bmatrix} cos_{\psi} & -sin(\psi) & 0 \\ sin(\psi) & cos_{\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A partir de las ecuaciones de rotación de los ángulos se puede calcular la matriz de rotación completa del marco de referencia del cuerpo respecto al marco de referencia fijo, la ecuación se muestra en 2.18. Se tomara en cuenta al $\cos(x)$ como c_x y al $\sin(x)$ como s_x.

$$R(\varphi, \vartheta, \psi) = R(z, \psi) \cdot R(y, \vartheta) \cdot R(x, \varphi)$$

$$R = \begin{bmatrix} c_\psi c_\vartheta & -s_\psi c_\varphi + c_\psi s_\vartheta s_\varphi & s_\psi s_\varphi + c_\psi s_\vartheta c_\varphi \\ s_\psi s_\vartheta & c_\psi c_\vartheta + s_\psi s_\vartheta s_\varphi & -c_\psi s_\varphi + s_\psi s_\vartheta c_\varphi \\ -s_\vartheta & c_\vartheta s_\varphi & c_\vartheta c_\varphi \end{bmatrix} \tag{2.18}$$

La matriz de rotación $\mathbf{R}$ es ortogonal de modo que $\mathbf{R}^{-1} = \mathbf{R}^{\mathsf{T}}$ que es la matriz de rotación del marco inercial del macro del cuerpo, (Luukkonen, 2011). De este modo se puede llegar

a la matriz identidad de I_3 de acuerdo a (2.19).

$$R^T R = I_3 \tag{2.19}$$

De acuerdo a (Craig, 1989) aplicando la derivada respecto al tiempo a (2.19) obtenemos:

$$R^T \dot{R} + \dot{R}^T R = 0_3 \tag{2.20}$$

Definiendo a:

$$S = R^T \dot{R} \tag{2.21}$$

De modo que se aprecia la anti-simetr´ıa de S, es decir que si A es una matriz cuadrada anti-simétrica, entonces debe cumplir que $A^T = -A$. Así, podemos asignar a la matriz S los términos mostrados en (2.22) de acuerdo a las propiedades relacionadas a las matrices ortogonales y anti-simétricas, (Craig, 1989). Las variables mostradas en (2.22) esta mostrada en términos del vector de velocidad angular Ω también mostrados en términos de η (2.23).

$$S(\Omega) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.22}$$

$$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\varphi} - \dot{\psi} s_\vartheta \\ \dot{\vartheta} c_\varphi + \dot{\psi} s_\varphi c_\vartheta \\ -\dot{\vartheta} s_\varphi + \dot{\psi} c_\varphi c_\vartheta \end{bmatrix} \tag{2.23}$$

Calculando la matriz de transformación ω_n nos permitirá obtener la relación entre velocidades angulares de modo que queden separados:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s_\vartheta \\ 0 & c_\varphi & s_\varphi c_\vartheta \\ 0 & -s_\varphi & c_\varphi c_\vartheta \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{bmatrix} \tag{2.24}$$

$$\Omega = \omega_n \dot{\eta}$$

Para el cálculo de la dinámica del quadrotor se emplea la metodología de Euler-Lagrange tomando como base conceptos de energía mecánica. Esta metodología emplea el vector de coordenadas generalizadas q para describir el movimiento de traslación y rotación de un sistema de seis grados de libertad (Naidoo, Stopforth, y Bright, 2011).

$$q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} \tag{2.25}$$

En (2.25) ξ representa al vector de coordenadas traslacionales del centro de masa del UAV respecto al marco inercial (x, y, z) y η representa al vector de coordenadas rotacionales de la orientación del vehículo empleando los ángulos $(\varphi, \vartheta, \psi)$ como se mostró en (2.16). De este modo q puede reescribirse como:

$$q = \begin{bmatrix} x, y, z, \varphi, \vartheta, \psi \end{bmatrix}^T \tag{2.26}$$

El modelo de energ´ıa se obtiene mediante el modelo del Lagrangiano. A partir de (Naidoo y cols., 2011), podemos definir a la energ´ıa potencial como la altura del quadrotor mientras que la energía cinética queda determinada por los movimientos de traslación y rotación del sistema. Así, se define al Lagrangiano como una función del vector de coordenadas generalizadas y su primer derivada respecto al tiempo.

$$L(q, \dot{q}) = T_{tras} + T_{rot} - U \tag{2.27}$$

desarrollando los componentes de (2.27) tenemos que la energía cinética traslacional queda expresada como:

$$T_{tras} = \frac{1}{2} m \dot{\xi}^2 = \frac{1}{2} m \dot{\xi}^T \dot{\xi} \tag{2.28}$$

La energ´ıa potencial tiene efecto unicamente en el eje z ya que depende de la altura del UAV de modo que queda representada de la siguiente manera:

$$U = mgz \tag{2.29}$$

En (2.29), m representa la masa del quadrotor, g la constante de aceleración gravitacional y z es representa a la altura del quadrotor. Con esto podemos definir al Lagrangiano en términos de las coordenadas de traslación como:

$$L(\xi, \dot{\xi}) = \frac{1}{2} m \dot{\xi}^T \dot{\xi} - mgz \tag{2.30}$$

La energía cinética rotacional del cuerpo rígido esta definida como:

$$T_{rot} = \frac{1}{2} I \omega^2 \tag{2.31}$$

Es necesario determinar un tensor de inercia para poder reunir todos los momentos de inercia existentes en el cuerpo dado que este posee diferentes ejes, (Bresciani, 2008). De este modo definimos al tensor de inercia como:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{2.32}$$

Asumiendo una simetría perfecta entre los rotores del UAV ubicados en los cuatro brazos de la estructura alineados con los ejes (x, y), podemos definir un tensor de inercia del cuerpo rígido como una matriz diagonal. Los elementos de dicha matriz se conocen como los principales momentos de inercia, (Craig, 1989).

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.33}$$

A continuación se muestra la representación de la energía cinética rotacional del sistema:

$$T_{rot} = \frac{1}{2} \Omega^T I \Omega \tag{2.34}$$

Conociendo (2.34) podemos expresar la energía cinética rotacional y el Lagrangiano en términos de las coordenadas rotacionales η.

$$T_{rot} = \frac{1}{2}\dot{\eta}^T J \dot{\eta}$$

$$L(\eta, \dot{\eta}) = \frac{1}{2}\dot{\eta}^T J \dot{\eta} \tag{2.35}$$

La siguiente expresión muestra el modelo dinámico completo del quadrotor a partir de las ecuaciones de Euler-Lagrange y las fuerzas generalizadas externas, (Luukkonen, 2011).

$$\begin{bmatrix} F_\xi \\ \tau \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \tag{2.36}$$

En (2.36) se expresa a τ como los movimientos *Roll*, *Pitch* y *Yaw* como los pares resultantes del movimiento rotacional, mientras que F_ξ representa la fuerza resultante del movimiento traslacional. En (2.37) se muestra la relación existente entre F_ξ y el vector de fuerza traslacional $\hat{F}$.

$$F_\xi = R \cdot F$$

$$F = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \tag{2.37}$$

Empleando las ecuaciones de Euler-Lagrange desarrollamos la fuerza resultante de traslación:

$$F_\xi = \frac{d}{dt}\left[\frac{\partial L(\xi, \dot{\xi})}{\partial \dot{\xi}}\right] - \frac{\partial L(\xi, \dot{\xi})}{\partial \xi} \tag{2.38}$$

$$F_\xi = m\ddot{\xi} + mg(\hat{k}) \tag{2.39}$$

Para conocer las aceleraciones lineales del sistema podemos reescribir (2.38) en función del vector de estado ξ.

$$\ddot{\xi} = \frac{1}{m} R \cdot \hat{F} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$\begin{aligned} \ddot{x} &= \frac{f}{m}(c_\varphi s_\vartheta c_\psi + s_\varphi s_{psi}) \\ \ddot{y} &= \frac{f}{m}(c_\varphi s_\vartheta s_\psi - s_\varphi c_{psi}) \\ \ddot{z} &= \frac{f}{m}(c_\varphi c_\vartheta - g \end{aligned} \tag{2.40}$$

Las ecuaciones de Euler-Lagrange del movimiento rotacional pueden quedar descritas como se muestra en (2.41) o bien evaluando el lagrangiano (2.42).

$$\tau = \frac{d}{dt}\left[\frac{\partial L(\eta, \dot{\eta})}{\partial \dot{\eta}}\right] - \frac{\partial L(\eta, \dot{\eta})}{\partial \eta} \tag{2.41}$$

$$\tau = \frac{d}{dt}\left[\frac{1}{2}\frac{\partial}{\partial \dot{\eta}}\left[\dot{\eta}^T J \dot{\eta}\right]\right] - \frac{1}{2}\frac{\partial}{\partial \eta}\left[\dot{\eta}^T J \dot{\eta}\right] \tag{2.42}$$

Aplicando la derivada respecto al tiempo del primer termino obtenemos:

$$\tau = J\ddot{\eta} + \dot{J}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}\left[\dot{\eta}^T J \dot{\eta}\right] \tag{2.43}$$

De este modo definimos la matriz Coriolis como:

$$C(\eta, \dot{\eta}) = \dot{J} - \frac{1}{2}\frac{\partial}{\partial \eta}\left[\dot{\eta}^T J\right] \tag{2.44}$$

$$C(\eta, \dot{\eta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.45}$$

Donde los valores de C quedan definidos de la siguiente manera:

$$
\begin{aligned}
C_{11} &= 0 \\
C_{12} &= (I_{yy} - I_{zz})\left(\dot{\vartheta}c_{\varphi}s_{\varphi} + \dot{\psi}s_{\varphi}^2 c_{\vartheta}\right) + (I_{zz} - I_{yy})\dot{\psi}c_{\varphi}^2 c_{\vartheta} - I_{xx}\dot{\psi}\, c_{\vartheta} \\
C_{13} &= (I_{zz} - I_{yy})\,\dot{\psi}c_{\varphi}s_{\varphi}c_{\vartheta}^2 \\
C_{21} &= (I_{zz} - I_{yy})\left(\dot{\vartheta}c_{\varphi}s_{\varphi} + \dot{\psi}s_{\varphi}c_{\vartheta}\right) + (I_{yy} - I_{zz})\dot{\psi}c_{\varphi}^2 c_{\vartheta} + I_{xx}\dot{\psi}\, c_{\vartheta} \\
C_{22} &= (I_{zz} - I_{yy})\dot{\varphi}c_{\varphi}s_{\varphi} \\
C_{23} &= -I_{xx}\dot{\psi}s_{\vartheta}c_{\vartheta} + I_{yy}\dot{\psi}s_{\varphi}^2 s_{\vartheta}c_{\vartheta} + I_{zz}\dot{\psi}c_{\varphi}^2 s_{\vartheta}c_{\vartheta} \\
C_{31} &= (I_{yy} - I_{zz})\dot{\psi}c_{\vartheta}^2 s_{\varphi}c_{\varphi} - I_{xx}\dot{\vartheta}c_{\vartheta} \\
C_{32} &= (I_{zz} - I_{yy})(\dot{\vartheta}c_{\varphi}s_{\varphi}s_{\vartheta} + \dot{\varphi}s_{\varphi}^2 c_{\vartheta}) + (I_{yy} - I_{zz})\dot{\varphi}c_{\varphi}^2 c_{\vartheta} + I_{xx}\dot{\psi}s_{\vartheta}c_{\vartheta} - I_{yy}\dot{\psi}s_{\varphi}^2 s_{\vartheta}c_{\vartheta} - I_{zz}\dot{\varphi}c_{\varphi}^2 s_{\vartheta}c_{\vartheta} \\
C_{33} &= (I_{yy} - I_{zz})\dot{\varphi}c_{\varphi}s_{\varphi}c_{\vartheta}^2 - I_{yy}\dot{\vartheta}s_{\varphi}^2 c_{\vartheta}s_{\vartheta} - I_{zz}\dot{\vartheta}c_{\varphi}^2 c_{\vartheta}s_{\vartheta} + I_{xx}\dot{\vartheta}c_{\vartheta}s_{\vartheta}
\end{aligned}
\tag{2.46}
$$

A partir de (2.43) podemos llegar a las ecuaciones diferenciales para las aceleraciones angulares, (Luukkonen, 2011)

$$\ddot{\eta} = J^{-1}(\tau_B - C(\eta, \dot{\eta})\dot{\eta}) \tag{2.47}$$

De acuerdo a (Garcia, Rubio, y Ortega, 2012), para desarrollar las ecuaciones en (2.44) es necesario considerar un equilibrio en el vuelo de modo que los ángulos de Euler se mantengan en un valor nulo en al menos los ángulos *Pitch* y *Roll*. Siguiendo tal metodología es posible simplificar en las siguientes expresiones:

$$
\begin{aligned}
\ddot{\varphi} &= \frac{I_{xx} + I_{yy} - I_{zz}}{I_{xx}}\dot{\psi}\dot{\vartheta} + \frac{\tau_{\varphi}}{I_{xx}} \\
\ddot{\vartheta} &= \frac{-I_{xx} - I_{yy} + I_{zz}}{I_{yy}}\dot{\psi}\dot{\varphi} + \frac{\tau_{\vartheta}}{I_{yy}} \\
\ddot{\psi} &= \frac{I_{xx} - I_{yy} + I_{zz}}{I_{zz}}\dot{\vartheta}\dot{\varphi} + \frac{\tau_{\psi}}{I_{zz}}
\end{aligned}
\tag{2.48}
$$

Las ecuaciones anteriores pueden ser reducidas si consideramos no solo el equilibrio en vuelo para cada uno de los ángulos de Euler sino que además consideramos las velocidades

angulares nulas. De este modo llegamos a que el modelo dinámico del quadrotor en términos de movimientos rotacionales y traslacionales se puede simplificar a las ecuaciones en (2.49)

$$\ddot{z} = \frac{f}{m}, \quad \ddot{\varphi} = \frac{\tau_\varphi}{I_{xx}}$$
$$\ddot{\vartheta} = \frac{\tau_\vartheta}{I_{yy}}, \quad \ddot{\psi} = \frac{\tau_\psi}{I_{zz}} \tag{2.49}$$

Para finalizar el análisis del modelo del quadrotor, se muestran las funciones de transferencia de a partir de (2.49) únicamente aplicando la transformada de Laplace. Estas funciones de transferencia nos permitirán conocer comprender la naturaleza del sistema así como obtener la respuesta transitoria para posteriormente poder aplicar un control.

$$G_z(s) = \frac{1}{ms^2}, \quad G_\varphi(s) = \frac{1}{I_{xx}s^2}$$
$$G_\vartheta(s) = \frac{1}{I_{yy}s^2}, \quad G_\psi(s) = \frac{1}{I_{zz}s^2} \tag{2.50}$$

2.3. Sistemas de control

La teoría de control utilizadas comúnmente son la teoría de control clásico o también llamada teor´ıa de control convencional, la teor´ıa de control moderno y la teor´ıa de control robusto. En general el control automático ha tenido una gran relevancia en el avance de las ciencias e ingeniería ya que soluciona aspectos en diferentes ramas como la implementación de sistemas auto ajustables en control de movimiento o temperatura por mencionar alguna, (Katsuhiko, 2010). El primer trabajo significativo se remonta hasta el siglo XVIII. Este trabajo consistió en el control de velocidad de una maquina de vapor y fue desarrollado por James Watt.

La teoría de control clásica trata dejó de ser potente con forme los sistemas se fueron haciendo más complejos (más de una entrada y más de un salida), fue hasta 1960 que gracias al la disponibilidad de las computadoras digitales fue posible realizar análisis de dominio

del tiempo de sistemas más complejos. La teoría de control moderna se desarrollo para trabajar con la creciente complejidad de los sistemas tomando asumiendo un mejor rendimiento en presunción. Los sistemas de control moderno se generan a partir de un análisis en el dominio del tiempo así como la síntesis en función de las variables de estado, (Katsuhiko, 2010).

La teoría de control moderno simplifica en gran proporción el diseño del sistema de control ya que este se basa en un modelo del sistema real a controlar. Por otra parte la teor´ıa de control moderno depende de una baja variabilidad entre el sistema real y el modelo del mismo. En algunas ocasiones perturbaciones no observadas en la creación del modelo podrían derivar en un funcionamiento incorrecto. De este modo se crea el control robusto siguiendo una metodolog´ıa distinta en la que primero se determina el rango de posibles errores que podrían afectar al sistema para después proceder a desarrollar el controlador. De esta manera se tendrá un rango de funcionamiento más amplio que un sistema de control moderno, (Katsuhiko, 2010).

Los elementos que componen un sistema de control son: variable a controlar, planta, procesos, sistema, perturbación y control retroalimentado, a continuación se proporciona un breve definición de cada uno de los componentes. En la Figura 2.4 se muestra como los diferentes componentes interactúan en el sistema de control en lazo abierto y retroalimentado, a continuación se muestran los diferentes componentes del sistema de control (Katsuhiko, 2010).

Variable a controlar y variable de control. La variable a controlar es la cantidad o condición que se mide y se controla. La variable de control o variable manipulada es la cantidad o condición que se modificara con la finalidad de que la variable a controlar alcance y se estabilice en en valor deseado. Determinamos la definición de *Controlar* como la acción de medir el valor de una variable para aplicar un ajuste correctivo que elimine o limite la diferencia entre el valor medido y el valor deseado.

Planta. Se determina como planta al comportamiento de cualquier sistema o conjunto de sistemas que va a ser controlado. La planta tendr´ıa que ser un elemento de al menos una

entrada y al menos una salida y el comportamiento se considerar´ıa como para una determinada entrada se tiene una determinada salida. Como ejemplo un horno eléctrico que para un determinado valor de voltaje a la entrada se tendra´ como respuesta un valor de temperatura.

Procesos. Se define al proceso como a un conjunto de operaciones o acciones continuas que derivan en un propósito definido. La forma en como se llega al propósito es de manera gradual por lo que existen puntos intermedios antes de llegar a ese punto.

Sistema. Conjunto de componentes actuando de manera ordenada para completar un objetivo. los sistemas pueden ser desde la interacción entre seres vivos hasta la interacción de diferentes componentes electrónicos.

Perturbación. Las perturbaciones son señales que afectan de manera negativa la salida de un sistema. Si la perturbación es inherente al sistema se considera *interna* mientras que si forma parte del medio y se propaga en sistema por medio de una entrada se considera *externa*.

Control retroalimentado. El control retroalimentado es una operación en la cual se compara el valor de la variable a controlar y se compara con el valor deseado aplicando una acción correctiva tomando en cuenta esta diferencia.

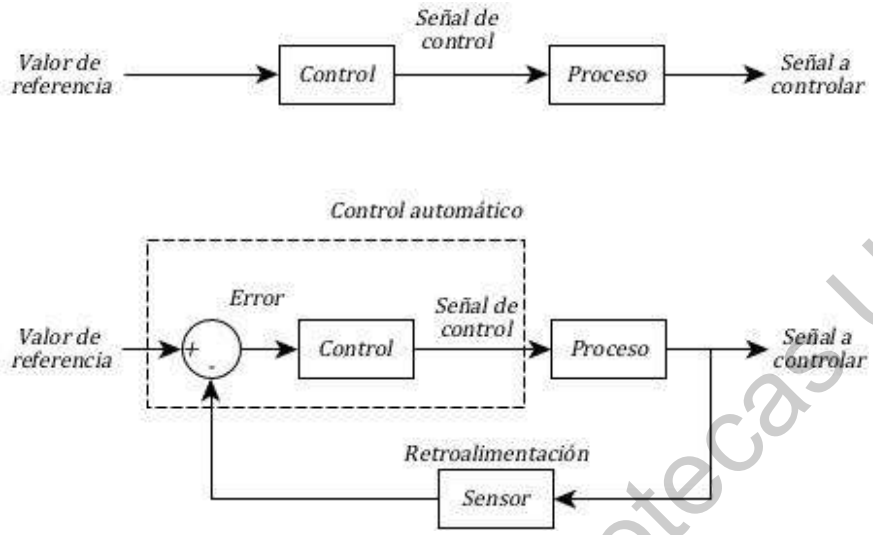


Figura 2.4: *Sistema de control en lazo abierto y retroalimentado.*

Un sistema de control de lazo abierto y uno de lazo cerrado pueden funcionar de manera similar bajo ciertas condiciones pero para que un sistema de lazo abierto iguale el funcionamiento de un sistema de lazo cerrado seria necesario invertir en un componente que invulnerable a perturbaciones externas y sin perturbaciones internas es decir un sistema prácticamente aislado y determinista. Por ello un sistema en lazo cerrado es más viable ya que es posible emplear componentes no tan precisos y por ello más baratos.

2.3.1. Acción de control

La acción de control es la forma en que un controlador automático produce la acción correctiva. Es un proceso dependiente de la lectura del sensor y la comparación del valor leído y el valor deseado. Los diferentes controladores se pueden clasificar de acuerdo a se acción de control, (Katsuhiko, 2010).

- Controladores on-off.
- Controladores proporcionales.

- Controladores integrales.
- Controladores proporcionales-integrales.
- Controladores proporcionales derivativos.
- Controladores proporcionales-integrales derivativos.

Controladores on-off. Son sistemas de dos posiciones en los cuales uno de los estados permite el paso completo (*Encendido*) mientras que el otro lo limita por completo (*Apagado*). Generalmente son sistemas eléctricos accionados por solenoides en los que el paso no puede ser limitado en puntos intermedios. En (2.51) se expresa manera matemática el comportamiento de un controlador on-off, para una salida $u(t)$, un error $e(t)$ y donde U_1 y U_2 son constantes.

$$u(t) = \begin{cases} U_1, & si \quad e(t) > 0 \\ U_2, & si \quad e(t) < 0 \end{cases} \tag{2.51}$$

Existen sistemas que implementan un mecanismo para reducir la frecuencia en que el conmutador (acción de control) cambia de estado, este mecanismo se llama brecha diferencial y lo que ocasiona es que el conmutador se active al estar brevemente desplazado del valor de referencia.

Acción de control proporcional. Para un control proporcional la relación que tiene la salida $u(t)$ y la señal del error $e(t)$ se muestra en (2.52).

$$u(t) = k_p e(t) \tag{2.52}$$

Aplicando la transformada de Laplace, la función de transferencia de un controlador proporcional se muestra en (2.53)

$$\frac{U(s)}{E(s)} = K_p \tag{2.53}$$

Tanto en (2.52) como en (2.53) K_p se considera como la ganancia proporcional. En cualquier caso se puede considerar al controlador proporcional como un sistema que amplifica de manera ajustable al error.

Acción de control integral. En un controlador integral se tiene una proporción de la suma de los valores previos a lo largo del tiempo. De modo que para una salida $u(t)$ y un error $e(t)$ la proporción esta dada por:

$$\frac{du(t)}{dt} = K_i e(t) \tag{2.54}$$

Visto de otro modo

$$u(t) = k_i \int_0^t e(t)dt \tag{2.55}$$

Donde K_i es la constante integral. La función de transferencia queda denotada de la siguiente manera:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{2.56}$$

Acción de control proporcional-integral. La acción de control proporcional-integral (PI) queda definida de la siguiente manera:

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt \tag{2.57}$$

Donde T_i se denomina como tiempo integral. La función de transferencia queda definida como:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s}\right) \tag{2.58}$$

Acción de control proporcional-derivativa. La acción del control proporcional-derivativo (PD) se define a continuación:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} \tag{2.59}$$

Su función de transferencia es

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s) \tag{2.60}$$

Donde T_d es el tiempo derivativo.

Acción de control proporcional-integral-derivativa. Un control proporcional-integral-derivativa combina las acciones de un control proporcional, un control integral y un control derivativo como se muestra en (2.61). Al combinar las tres acciones podemos obtener las ventajas de cada uno.

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \tag{2.61}$$

La función de transferencia queda de la siguiente manera:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.62}$$

Donde K_p es la constante proporcional, T_d es el tiempo derivativo y T_i es el tiempo proporcional.

2.4. Algoritmos basados en CNN para la segmentación de imágenes

Las técnicas basadas en redes neuronales intentan emular el comportamiento de la neurona humana desde la sinapsis (conexión e interacción entre neuronas) así como la manera en que aprende de diferentes errores o aciertos. El modelo de una red neuronal fue propuesto por McCulloch y Pitts en 1943 donde se propon´ıa a que las neuronas funcionaban como dispositivos booleanos. En el modelo de la Figura 2.5 se pueden observar los componentes

propios de la neurona: unión entre neuronas y agregación de información, (Rothman, 2018). La función lineal representativa de la neurona artificial se muestra en (2.63).

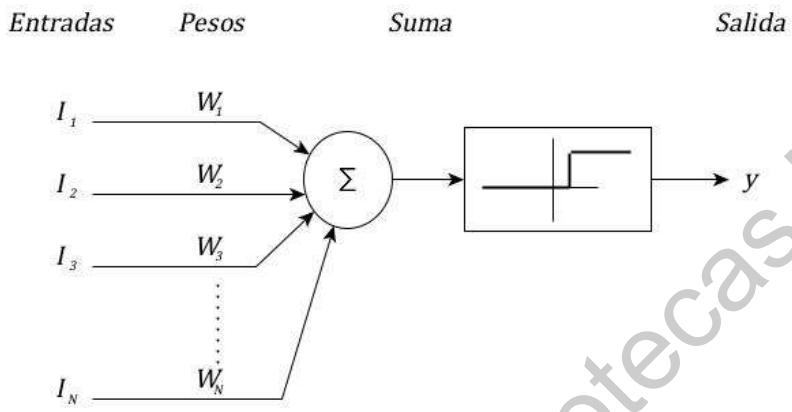


Figura 2.5: *Modelo de la neurona artificial, modificado de* (Rothman, 2018).

$$y = \sum_{i=0} I_i W_i + b \tag{2.63}$$

En (2.63) I_i representa a los datos de entrada, W_i son los pesos sinápticos y b es un factor de desface o bias. Antes de la salida en y se tiene una operación booleana lo cual da como resultado a la salida 1 o 0.

El uso de múltiples neuronas organizadas en diferentes arreglos ha llevado a que en los últimos 20 años se tengan registrados numerosos avances en el desarrollo de arquitecturas para la segmentación de imágenes. La agrupación de una gran cantidad de neuronas llevó al diseño del perceptrón multicapa (MLP) para derivar posteriormente en lo que conocemos actualmente como redes neuronales convolucionales (CNN). La principal causa del desuso del MLP en tareas de visión por computadora son las necesidades actuales de velocidad de procesamiento así como la cantidad de memoria utilizada. La topología lograda al unir múltiples neuronas se muestra en la Figura 2.6. En ella se puede observar un sistema de múltiples entradas y múltiples salida el cual podría ser adaptado a problemas de complejidad cada vez mayor, (Rothman, 2018).

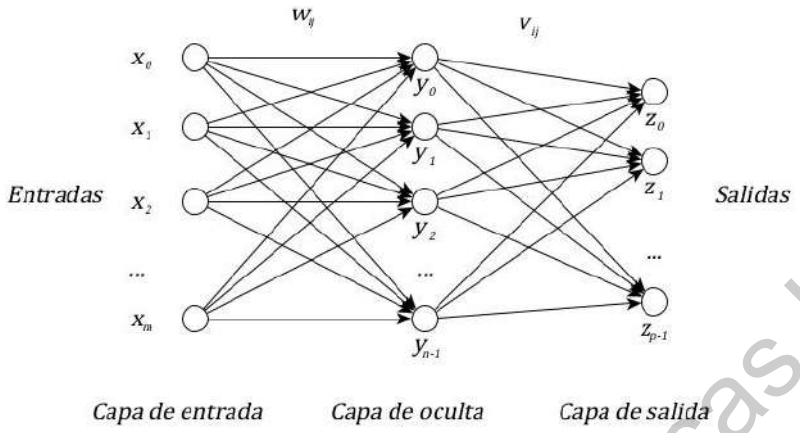


Figura 2.6: *Modelo de un perceptrón multicapa, modificado de* (Rothman, 2018).

Los resultados obtenidos con CNN ha sentado bases en diferentes áreas de la visión por computadora, en tareas como clasificación de imágenes, aumento de calidad en las mis- mas y en segmentación semántica, (Kayalibay, Jensen, y van der Smagt, 2017), sin embar- go algunas técnicas de agrupamiento también han demostrado resultados similares, (Naz, Majeed, y Irshad, 2010). Como se menciono en anteriores capítulos la segmentación de k-medias resuelve en gran parte el problema de segmentación. A continuación se muestra tanto el funcionamiento de la operación convolución sobre un imagen así como las diferentes arquitecturas de segmentación de imágenes encontradas en la literatura.

2.4.1. Operación convolución en imágenes

La operación de convolución es empleada en visión computacional para difuminar o aclarar imágenes así como para clasificación de imágenes y detección de objetos. Se considera que la operación convolución matemática mide la superposición de dos funciones de modo que su representación es el resultado de la combinación que integra do multiplicación punto a punto de diferentes conjuntos de datos, tal como se muestra en (2.64), (Dumoulin y Visin, 2016).

$$r(i) = (s \cdot k)(i) = \int s(i-n)k(n)dn \tag{2.64}$$

La cual vista de manera discreta queda denotada de la siguiente manera:

$$r(i) = (s \cdot k)(i) = \sum s(i-n)k(n) \tag{2.65}$$

Es necesario definir el tamaño del filtro ya que la desnacionalizad del filtro estará definida por las caracter´ısticas de la imagen a procesar. En la capa de entrada esas caracter´ısticas corresponden a el canal de color RGB y en los canales escondidos esas caracter´ısticas representan mapas de entidades ocultas que codifican varios tipos de formas en la imagen.

La operación convolucional sobre imágenes esta definida de la siguiente manera, el pth filtro en la qth capa tiene parámetro denotados por le tensor de tres dimensiones $W^{(p,q)} = \left[w^{(p,q)}_{ijk} \right]$, donde los índices i, j, k indican la posición a lo largo de la longitud, altura y profundidad del filtro.

Los mapas de caracter´ısticas en la qth capa esta representado por un tensor de tres dimensiones $H^{(q)} = \left[h^{(q)}_{ijk} \right]$ Para el caso en que el valor de q es 1, el caso especial corresponde a la notación H^1 que simplemente representa la capa de entrada. Entonces, la operación convolucioal desde la qth capa hasta la (q + 1)th capa esta definida en ((2.66)).

$$h^{(q+1)}_{ijp} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{k=1}^{d_q} w^{(p,q)}_{rsk} h^{(q)}_{i+r-1,j+s-1,k} \tag{2.66}$$

$$\forall i \epsilon \{1, ..., L_q - F_q + 1\}$$

$$\forall j \epsilon \{1, ..., B_q - F_q + 1\}$$

$$\forall p \epsilon \{1, ..., d_{q+1}\}$$

Esta operación se puede interpretar como un producto punto a lo largo todo el volumen del filtro, el cual se repite a lo largo de todas las posiciones espaciales (i, j) y filtros.

En la Figura 2.7 podemos apreciar como se realizan los corrimientos a lo largo de la matriz de entrada, dicha matriz resultante de los corrimientos es multiplicada celda por celda con la matriz filtro.

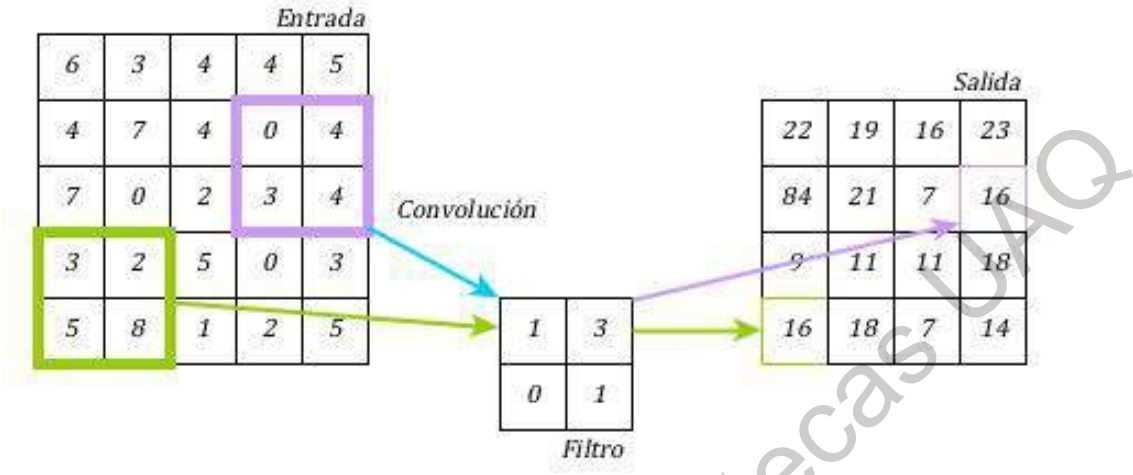


Figura 2.7: *Operación de convolución, modificado de* (Dumoulin y Visin, 2016).

Redes neuronales convolucionales Las redes neuronales convolucionales son modelos ampliamente usados en tareas de aprendizaje profundo resolviendo tareas que van desde segmentación hasta clasificación las cuales, igual que las redes neuronales, se inspiran en sistemas biológicos, en este caso estructuras de corteza visual, (Fukushima y Miyake, 1982).

Los componentes principales de una CNN son las capas convolucionales y las capas completamente conectadas (fully connected). Se le llama mapa de caracter´ısticas al resultado obtenido al procesar una entrada por una capa convolucional, este resultado es usualmente un tensor de tres dimensiones. El filtro empleado por la capa convolucional tiene la misma función que un filtro empleado en operaciones de procesamiento de imágenes, la diferencia radica en que los pesos del filtro en procesamiento de imágenes son elegidos de manera precisa para realizar la operación mientras que los pesos en los filtros de las capas convolucionales se van modificando conforme a la red se entrena.

El mapa de caracter´ısticas resultante de la capa convolucional es el resultado corresponde al campo receptivo en el mapa de entrada y su tamaño esta determinado por el núcleo o *kernel* y la dilatación. Para realizar las conexiones completas es necesario realizar un aplanamiento

(*flatten*). Estas capas completamente conectadas tienen son similares a las redes neuronales ya que aprenden las relaciones lineares y no lineares entre caracter´ısticas extra´ıdas clasificando las muestras en dos o más clases, en la Figura 2.8 se muestra un modelo simple de la arquitectura CNN.

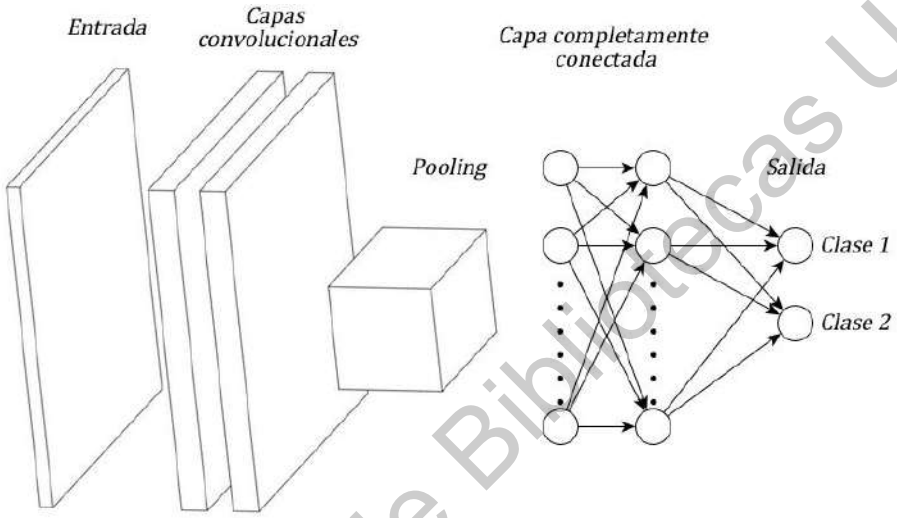


Figura 2.8: *Arquitectura CNN básica, basada en* (O'Shea y Nash, 2015).

Pooling El pooling es una operación empleada en redes redes convolucionales. Esta operación reduce la cantidad de elementos en una red propiciando la reducción espacial de la misma y controlando el sobreajuste. En la Figura 2.9 se muestran los resultados de las operaciones *max-pooling* y *avg-pooling*empleando un kernel de 2×2, (Dumoulin y Visin, 2016).

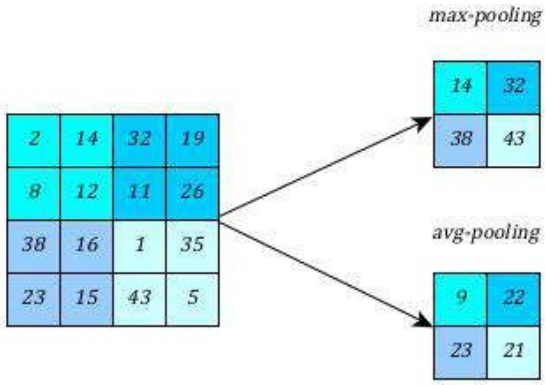


Figura 2.9: *Operación pooling, modificado de* (Dumoulin y Visin, 2016).

En la operación *max-pooling* se busca el elemento de mayor valor en el espacio delimitado por el kernel, en la operación *avg-pooling* se hace un promedio de los valores dentro del espacio del kernel dividido por la cantidad de elementos en el kernel.

2.4.2. Métodos de segmentación

La segmentación es un proceso de la visión computacional que conlleva la división de una entrada visual en segmentos para simplificar el análisis de la imagen. La segmentación representa objetos o partes de objetos que comprenden conjuntos de p´ıxeles, los niveles de análisis de imagen son los siguientes, (Khan, 2014):

- Clasificación. Categoriza la imagen completa en una clase.
- Detección de objetos. Detecta al objeto con un rectángulo alrededor de el.
- Segmentación. Identifica partes de la imagen y entiende a que objeto pertenece.

Existen diferentes técnicas de segmentación que requieren intervención humana y un algoritmo no adaptable, entre ellas se encuentran, (Khan, 2014):

- Thresholding. Divide la imagen en primer plano y fondo. Un valor limite especifico divide los p´ıxeles entre uno o dos niveles para aislar los objetos. Thresholding convierte la imagen de escala de grises a binaria.

- Agrupamiento K-medias. Es un algoritmo que identifica grupos en los datos, con la variable K representando el numero de grupos. El algoritmo asigna cada punto de datos, o p´ıxel, a uno de los grupos basado en similitud de caracter´ısticas. En lugar de analizar grupos predefinidos, la agrupación funciona de manera iterativa para formar grupos de forma organizada.
- Detección de bordes. Identifica cambios bruscos o discontinuidades en el brillo, generalmente implica organizar puntos de discontinuidad en segmentos de lineas, curvas o bordes.

Las técnicas actuales de segmentación son más poderosas por el uso de técnicas de aprendizaje profundo. Para la segmentación se emplean métodos como:

- Redes Neuronales Convolucionales (CNN). La segmentación de imágenes con CNN implica alimentar segmentos de una imagen como entrada a una neurona convolucional que etiqueta los p´ıxeles. La CNN no puede procesar toda la imagen a la vez, escanea la imagen usando un filtro de algunos p´ıxeles cada vez hasta que haya mapeado toda la imagen, (Kayalibay y cols., 2017).
- Redes Convolucionales Completas (FCN). Ya que las CNN tradicionales no pueden administrar diferentes tamaños de entrada, las FCN usan capas convolucionales para procesar diferentes tamaños de entrada y pueden funcionar más rápido. La capa de salida tiene un gran campo receptivo y corresponde a la altura y ancho de la imagen, mientras que el número de canales corresponde al número de clases. Las capas convolucionales clasifican cada p´ıxel para determinar el contexto de la imagen, incluida la ubicación de los objetos, (Lu, Chen, Zhao, y Chen, 2019).
- SegNet. una arquitectura basada en encoders y encoders profundos, también conocido como segmentación semántica a píxeles. Implica codificar la imagen de entrada en dimensiones bajas y luego recuperarla con capacidades de invariancia de orientación en el decodificador. Esto genera una imagen segmentada en el extremo del decodificador, (Badrinarayanan, Kendall, y Cipolla, 2017).

2.4.3. U-Net

La arquitectura U-Net propuesta en (Ronneberger y cols., 2015) es un modelo encoder-decoder que busca la extracción de características de una imagen mediante un proceso de contracción y un proceso de expansión con la finalidad de obtener la segmentación semántica de la misma. Fue originalmente diseñada para la segmentación de imágenes biomédicas, sin embargo su simplicidad y eficiencia ha sido probada para numerosas tareas ajenas a la original. Esta arquitectura esta dividida en dos procesos: el proceso de contracción y el de expansión. Con el proceso de expansión (Encoder) se busca obtener diferentes niveles de características a partir de múltiples capas convolucionales. En el proceso de la contracción (Decoder) se aplican deconvoluciones de capa múltiple en el mapa de características generado en el proceso de expansión además de combinar diferentes niveles de características hasta obtener un mapa de características del tamaño de la imagen de entrada y de este modo resolver el problema de segmentación semántica.

La razón por la cual se combinan los niveles de características de los proceso de expansión y contracción es que el proceso tiende a ser muy agresivo dado que el tamaño de la ima- gen se ve afectado hasta reducirlo a un valor muy pequeño por ello en (Ronneberger y cols., 2015) proponen conexiones entre la parte de contracción y la parte de expansión de modo que se pueda rescatar una mayor cantidad de información. La figura 2.10 muestra gráficamente la arquitectura U-Net.

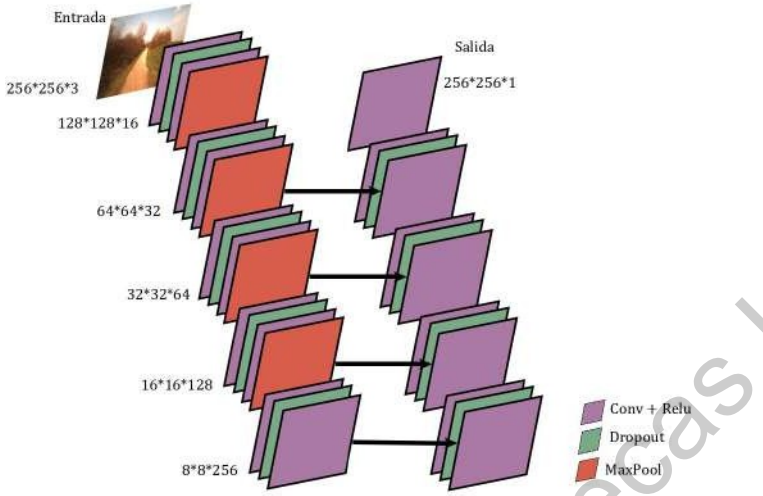


Figura 2.10: *Arquitectura U-Net basado en* (Ronneberger y cols., 2015).

En la Figura 2.10 se puede notar a la izquierda la ruta de contracción encargada de la reducción de resolución a mediante operaciones convolucionales y de Max-pooling (agrupación máxima). Esta ruta consta de cuatro bloques constituidos por tres convoluciones y una reducción de resolución de agrupación máxima. Después de cada bloque de contracción el número de mapas de características se multiplica por 2. Por otra parte, a la derecha se tiene la ruta de expansión el cual esta constituido por cuatro bloques. Antes de cada bloque es necesario multiplicar el mapa de caracter´ısticas por dos para posteriormente aplicar deconvoluciones de modo que este reduzca a la mitad. En la tabla 2 se muestra a detalle las diferentes capas de la arquitectura as´ı como la salida de las mismas

Tabla 2: Arquitectura U-Net basada en (Ronneberger y cols., 2015)

Capa	Tipo	Salida
Entrada		128 ×128 × 3
Conv	2 @ conv 16 @ 3 ×3	128 ×128 × 16 c1
Max-pool	Filtro 2 ×2 strides = 2	64 ×64 × 16
Conv	2 @ conv 32 @ 3 ×3	64 ×64 × 32 c2
Max-pool	Filtro 2 ×2 strides = 2	32 ×32 × 32
Conv	2 @ conv 64 @ 3 ×3	32 ×32 × 64 c3
Max-pool	Filtro 2 ×2 strides = 2	16 ×16 × 64
Conv	2 @ conv 128@ 3 ×3	16 ×16 × 128 c4
Max-pool	Filtro 2 ×2 strides = 2	8 ×8 × 128
Conv	2 @ conv 256 @ 3 ×3	8 ×8 × 256 c5
Drop-Out	50 %	8 ×8 × 256
Deconv	Upsample	16 ×16 × 128 u1
Conv	u1 + c4 2 @ conv 128 @ 3 ×3	16 ×16 × 128
Deconv	Upsample	64 ×64 × 32 u3
Conv	u3 + c2 2 @ conv 32 @ 3 ×3	64 ×64 × 32
Deconv	Upsample	128 ×128 × 16 u4
Conv	u4 + c1 2 @ conv 16 @ 3 ×3	128 ×128 × 16
Conv	1 @ conv 16 @ 1 ×1	128 ×128 × 1
Salida		128 ×128 × 1

En la tabla 2 se muestra como tipo de capa convolucional *Upsample*, esta capa esta constituida por una capa convolucional transpuesta cuyos parámetros corresponden al doble de la profundidad, ancho y largo que la convolución anterior y se obtiene una imagen de mismo largo y ancho pero la mitad de profundo de la convolución anterior.

Tal como se establece en (Ronneberger y cols., 2015), la función de pérdida empleada en el entrenamiento de la red es la función *softmax* de cada píxel con respecto al número de clases en combinación a la función de *entropía cruzada*. En (2.67) se muestra la función de pérdida *softmax* para la cual, $y_k(x)$ corresponde a la activación del píxel x

$$p_k(x) = \frac{e^{y_k(x)}}{\sum_j e^{y_j(x)}} \tag{2.67}$$

Además esta función representa la probabilidad de que que el píxel x pertenezca a la clase k y la suma de los valores para cada clase es 1.

$$\begin{aligned} P[x = k] &= p_k(x) \\ \sum_i p_{k=i}(x) &= 1 \end{aligned} \tag{2.68}$$

Paro cual se aplica la función de entropía cruzada mostrada en (2.69):

$$J(x) = \sum_k a_k(x) \cdot log(p_k(x)) \tag{2.69}$$

Donde $a_k(x)$ corresponde al valor real de activación del píxel x para la clase k. De modo que la función de coste resultante para todos los píxeles queda denotada de la siguiente manera:

$$J = \sum_x log(P_{true}(x)) \tag{2.70}$$

En la cual $P_{true}(x)$ es la predicción realizada por la función *softmax* correspondiente a la clase a la cual pertenece el píxel. La finalidad de esta función es penalizar a cada píxel la desviación del valor predicho para la clase real $P_{true}(x)$.

2.4.4. Mobile-Net V2

Mobile-Net V2 (Sandler y cols., 2018) es una mejora de Mobile-Net (Howard y cols., 2017), esta arquitectura mejorada fue diseñada para desarrollar aplicaciones de baja latencia como visión por computadora e internet de las cosas (IoT). Una de las principales mejoras logradas en Mobile-Net y Mobile-Net V2 respecto a las arquitecturas basadas en CNN convencionales es la reducción significativa de esfuerzo computacional lo cual permite que dispositivos móviles y computadoras de bajas prestaciones puedan trabajar con estas arquitecturas. De manera simplificada, la arquitectura de Mobile-Net V2 tiene una serie de capas ocultas basadas en bloques de cuello de botella y una convolución separable en profundidad que logra reducir considerablemente el número de parámetros entrenables logrando una red más liviana respecto a las redes basadas en CNN convencionales. La convolución convencional se remplazo por una convolución en profundidad con un solo filtro seguido de una devolu- ción puntual denominado convolución divisible en profundidad.

La primer capa del bloque también es una convolución 1x1. Su propósito es expandir el número de canales en los datos antes de que entren en la convolución en profundidad. Por lo tanto, esta capa de expansión siempre tiene más canales de salida que canales de entrada; hace prácticamente lo contrario de la capa de proyección.

La segunda novedad del componente básico de Mobile-Net V2 es la conexión residual. Esto funciona como en ResNet (He y cols., 2016) y existe para ayudar con el flujo de gradientes a través de la red. La conexión residual solo se utiliza cuando el número de canales que entran en el bloque es el mismo que el número de canales que salen de él. La Figura 2.11 muestra gráficamente el bloque de cuello de botella residual de Mobile-Net V2.

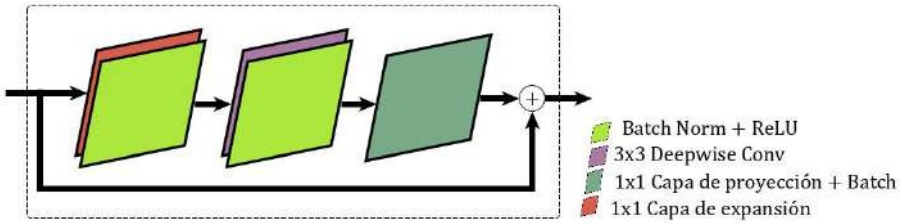


Figura 2.11: *Arquitectura del bloque de cuello de botella residual basado en* (Sandler y cols., 2018)

En la Figura 2.11 se muestra el bloque de cuello de botella con conexión residual. Este bloque cuenta con tres capas convolucionales, los dos últimos son los mismos que se mostraron en la anterior generación de Mobile-Net: una convolución en profundidad que filtra la entrada seguida de una capa convolucional puntual de 1×1. La idea principal de Mobile-Net es reemplazar los filtros convolucionales normales de 3x3 por filtros convolucionales separables en profundidad de 3x3 seguido de una convolución de 1x1. Este proceso de filtrado y combinación produce el mismo resultado que se obtendría con convoluciones normales pero requiere menos operaciones y parámetros.

En Mobile-Net, la convolución puntual mantenía el mismo número de canales o los duplicaba. En Mobile-Net V2 se busca lo contrario, reducir el número de canales. Esta es la razón por la que esta capa ahora se conoce como la capa de proyección: proyecta datos con un gran número de dimensiones (canales) en un tensor con un número mucho menor de dimensiones. Las dos caracter´ısticas nuevas en Mobile-Net V2 son:

- La capa de expansión: Es una convolución de 1x1 que amplia el numero de canales en los datos de la imagen previo a la convolución de profundidad, es decir que salida de capa de expansión siempre tiene más canales que a la entrada. Este proceso tiene el funcionamiento opuesto a capa de proyección.

- Bloque con conexión residual: Mostrado en la Figura 2.11. Este bloque ayuda en el flujo de gradientes a lo largo de la red. Los canales de caracter´ısticas se incrementan en un factor t.

La arquitectura original de Mobile-Net V2 (Sandler y cols., 2018) se muestra en la tabla 3. Cada renglón representa una secuencia de una o más capas identicas (stride) repetidas n veces con un factor de expancion t. Las capas en la misma secuencia tienen el mismo numero de canales de salida c. En la primer capa de cada secuencia se tiene un stride s las demás emplean un stride. Los kernels usados para todas las convoluciones espaciales es de 3x3.

Tabla 3: Arquitectura original Mobile-Net V2.

Entrada	Operador	t	c	n	s
2242×3	conv2d	-	32	1	2
1122×32	Cuello de botella	1	16	1	1
1122×16	Cuello de botella	6	24	2	2
562×24	Cuello de botella	6	32	3	2
282×32	Cuello de botella	6	64	4	2
142×64	Cuello de botella	6	96	3	1
142×96	Cuello de botella	6	160	3	2
72×160	Cuello de botella	6	320	1	1
72×320	conv2d 1×1	-	1280	1	1
72×1280	avgpool 7×7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1×1	-	k	-	

La arquitectura mostrada en Mobile-Net tiene como objetivo optimizar la latencia as´ı como permitir que redes redes pequeñas trabajen de manera eficiente sin importar el tamaño de la entrada. Se ha demostrado un mejor resultado en tiempo de inferencia que las arquitecturas ShuffleNet (X. Zhang, Zhou, Lin, y Sun, 2018) y NASNET (Zoph, Vasudevan, Shlens, y Le, 2018).

2.4.5. Pix2Pix

Las redes GANs son empleadas en tareas de procesamiento de imágenes como generación de imágenes, edición de imágenes y mapeo de imagen a imagen. Estas arquitecturas se basan en la interacción de dos modelos. El modelo Generador $G : Z \rightarrow Y$ mapea ruido aleatorio $z \in Z$ a una imagen de salida $y \in Y$. El otro modelo llamado discriminador $D : Y \rightarrow [0, 1]$ determina si una muestra es perteneciente del conjunto de entrenamiento o del generador, es decir que la salida del discriminador es la probabilidad de que la entrada sea 'real' o 'falsa', (Goodfellow y cols., 2014). La interacción entre los modelos generador y discriminador se muestra en la Figura 2.12. La función de pérdida empleada en el entrenamiento esta definida de la siguiente manera, como se menciona en (Özgen y Ekenel, 2020):

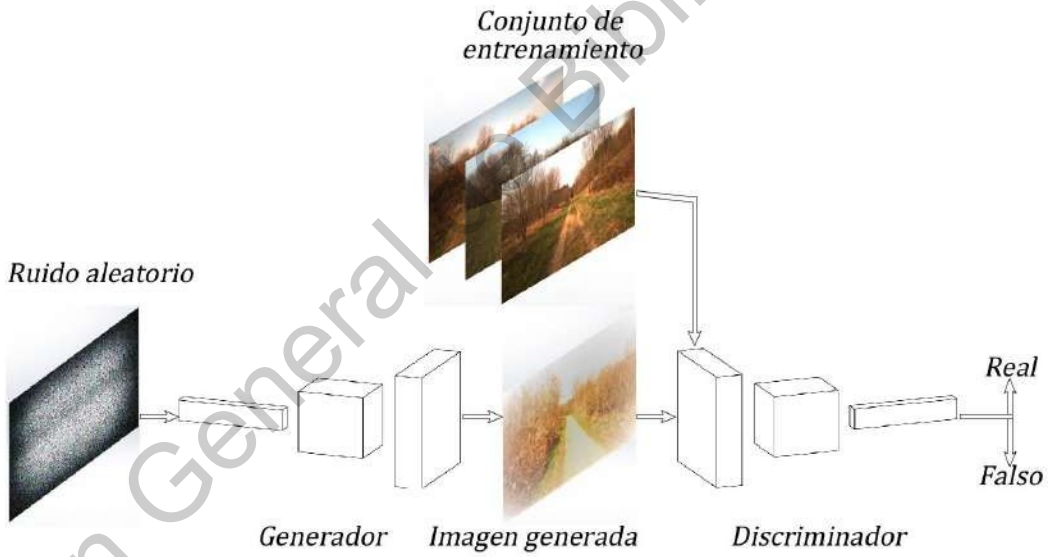


Figura 2.12: *Modelo básico de la arquitectura GANs, basado en* (Özgen y Ekenel, 2020).

$$L_{GAN}(G, D) = E_y \log(D(y)) + E_z \log(1 - D(G(z))) \qquad (2.71)$$

Donde E_z es el valor esperado para cada una de las instancias del conjunto de entrenamiento, E_z es el valor esperado para cada una de las entradas aleatorias en el generador. Esta función de pérdida muestra el comportamiento de los dos modelos, el generador se entrena para

engañar al discriminador, generando imágenes más reales, mientras que el discriminador se entrena para cada vez distinguir de mejor manera las imágenes reales de las falsas. Este comportamiento se interpreta de la siguiente manera:

$$G^* = \arg\min_G \max_D L_{GAN}(G, D) \tag{2.72}$$

De este modo las imágenes procesadas por el generador se parecerán cada vez más a las imágenes en el conjunto de entrenamiento. Si el generador o discriminador son condicionados el modelo pasa a ser CGAN (Conditional Generative Adversarial Networks), la principal diferencia entre los modelos GANs y CGANS radica en que tanto el generador como el discriminador emplean información adicional combinada con la entrada original. La arquitectura Pix2Pix (Isola y cols., 2017) se basa en un modelo CGAN. esta arquitectura resuelve el problema tradicional de mapeo de imagen a imagen (image-to-image translation) mediante la combinación de la pérdida adversaria (*adversarial loss*) con la pérdida estándar a nivel de píxel L_1. La distancia de pérdida L_1 esta definida de la siguiente manera:

$$L_{L1} = \mathbb{E}_{x,y,z} \|y - G(x, z)\|_1 \tag{2.73}$$

En esta arquitectura se emplea tanto la imagen de entrada como el ruido aleatorio para obtener una imagen de salida, es decir $G : X \times Z \rightarrow Y$ (para una imagen de entrada X, ruido aleatorio Z e imagen de salida Y). En otras palabras se genera una imagen de salida a partir de una imagen de entrada y el ruido aleatorio mientras que el discriminador determina si dicha imagen generada es real o falsa. De este modo la función objetivo que se busca optimizar se muestra en (2.74).

$$G^* = \arg\min_G \max_D L_{GAN}(G, D) + \lambda L_{L1}(G) \tag{2.74}$$

En el articulo se demostró que si no combinaba la imagen de entrada con el ruido aleatorio el sistema tendía a ser determinista, sin embargo se reportó que el generador entrenado tiene una predisposición a ignorar el ruido aleatorio. El discriminador esta basado en una arquitectura U-Net a la cual se le aplicaron capas de *Dropout*, esto para minimizar el problema

encontrado. La figura 2.13 muestra un diagrama de como interactúan ambas arquitecturas. En la tabla 4 se muestra la arquitectura empleada para el discriminador. El generador se mostró en la tabla 2.

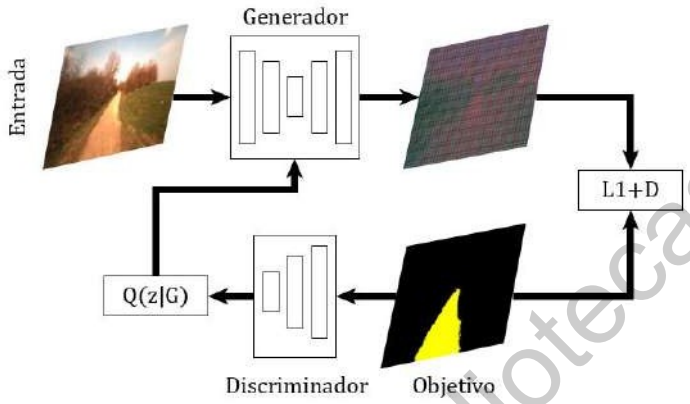


Figura 2.13: *Arquitectura Pix2Pix basado en* (Isola y cols., 2017).

Tabla 4: Arquitectura del Discriminador

Capa	Tipo	Salida
Entrada	Concat inp + tar	$256 \times 256 \times 6$
	64 @ 4	
Downsample	strides = 2	$128 \times 128 \times 64$
	Bach_norm = false	
	128 @ 4	
Downsample	strides = 2	$64 \times 64 \times 128$
	Bach_norm = true	
	256 @ 4	
Downsample	strides = 2	$32 \times 32 \times 256$
	Bach_norm = true	
Zero-padding		$34 \times 34 \times 256$
Conv	512 @ 4	$31 \times 31 \times 512$
	strides = 1	
bach_norm		
Leaky_ReLu		
Zero-padding		$33 \times 33 \times 512$
Conv	1 @ 4	$30 \times 30 \times 1$
	strides = 1	Salida

2.4.6. Métricas de segmentación

La métrica de segmentación empleada para las cuatro arquitecturas es el coeficiente Dice, (Tustison y Gee, 2009), mostrado en la ecuación 2.75.

$$D_c = \frac{2 * |A| \cap |B|}{|A| + |B|} \tag{2.75}$$

Donde A y B corresponden a las imágenes a comparar. Cave resaltar que este coeficien-

te funciona para imágenes binarias siendo $|A| \cap |B|$ la cantidad de píxeles que en ambas imágenes tienen un valor de uno y $|A| + |B|$ la cantidad de píxeles que tienen valor de uno en para cada imagen.

Una métrica con características similares al coeficiente Dice es la intersección sobre la unión (IoU) mostrada en la ecuación (2.76), sin embargo ambas pueden ser usadas de manera indiscriminada mostrando valores numéricos representativos del parentesco entre dos imágenes como se muestra en 2.77.

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.76}$$

$$IoU = \frac{TP}{TP + FP + FN} = \frac{a}{b}$$
$$D_c = \frac{2 * TP}{TP + FN + FP + TN} = \frac{2 * a}{a + b} = \frac{\frac{2 * a}{b}}{\frac{a}{b} + 1} = \frac{2 * IoU}{IoU + 1} \tag{2.77}$$

2.5. SLAM

El Mapeo y Localización Simultanea (SLAM por sus siglas en inglés) es un proceso en el cual se busca la verdadera autonomía de los robots móviles. El proceso trata de ubicar el robot móvil en un entorno desconocido a la vez que crea un mapa de dicho entorno, (Durrant-Whyte y Bailey, 2006). El sistema tiene como entrada di frentes sensores: giroscopio, Lidar, cámara, etc. y como salida la estimación del mapa del entorno y la ubicación del robot en el mismo. En la Figura 2.14 se muestra el ejemplo de un robot moviéndose a través de un entorno tomando observaciones relativas de un algunos puntos de referencia desconocidos.

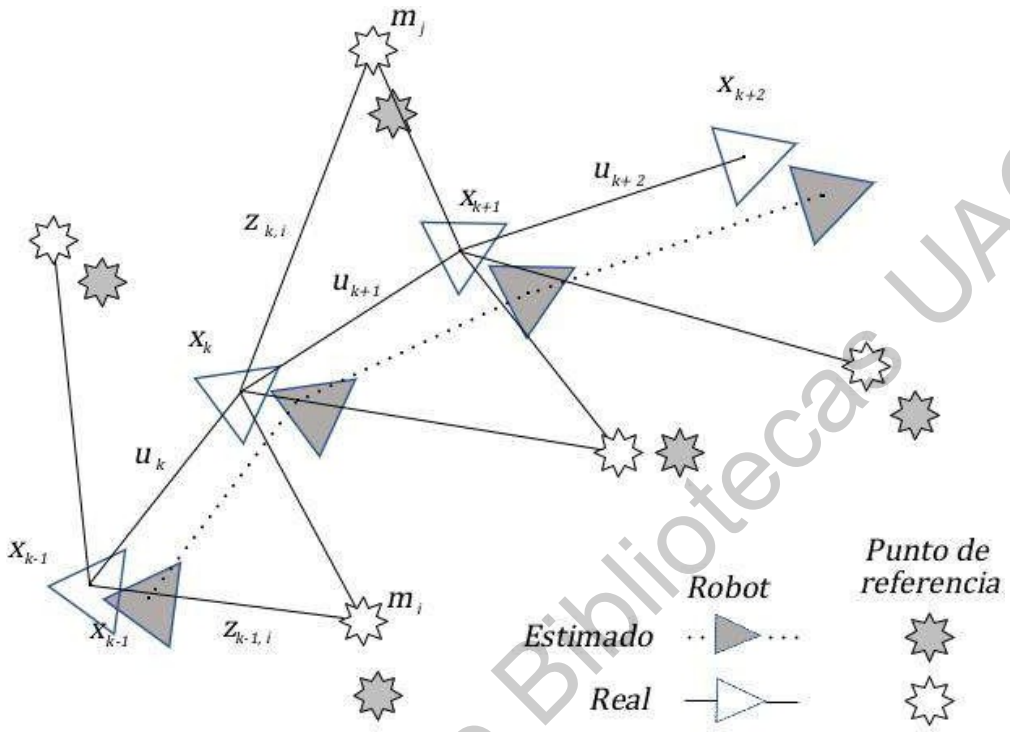


Figura 2.14: *Problema esencial SLAM, modificado de* (Durrant-Whyte y Bailey, 2006).

Las diferentes variables en la Figura 2.14 se describen a continuación, adicionando variables necesarias para llevar a cabo un correcto historial de las posiciones de los diferentes elementos.

- x_k: Es el vector de posición y orientación del vehículo.
- u_k: Es el vector de control, aplicado en el tiempo $k-1$ para conducir el veh´ıculo al estado x_k al tiempo k.
- m_i: Es el vector de las posición del i-ésimo punto de referencia cuya posición se asume es invariante en el tiempo.
- z_{ik}: Es una observación tomada por el vehículo de la posición del i-ésimo punto de referencia en el tiempo k. Si el punto de referencia no es relevante o si se tienen múltiples puntos de referencia al mismo tiempo este se simboliza como z_k.

- $X_{0:k} = \{x_0, x_1, \cdots, x_k\} = \{X_{0:k-1}, x_k\}$: Es el historial de posiciones del vehículo.
- $U_{0:k} = \{u_0, u_1, \cdots, u_k\} = \{U_{0:k-1}, u_k\}$: Es el historial de entradas de control.
- $Z_{0:k} = \{z_0, z_1, \cdots, z_k\} = \{Z_{0:k-1}, z_k\}$: es el historial de observaciones de los puntos de referencia.

2.6. Semejanza entre curvas

Para determinar la semejanza entre dos corvas planas se tienen diferentes técnicas, sin embargo en todas se requiere determinar la distancia que existe entre ambas, (Alt y Godau, 1995). Con una métrica de este estilo se puede determinar si un sistema de mapeo tiene un buen valor de repetibilidad al encontrar que tan diferentes son los recorridos registrados.

2.7. Distancia de Fréchet

La distancia de Fréchet es una métrica de similitud entre curvas, este proceso para determinar la similitud fue propuesto por Fréchet en 1906 (Fréchet, 1906). La métrica contempla el orden de los puntos y la posición de los mismos, de manera simple se describe como la distancia más corta necesaria para que dos puntos de curvas diferentes puedan mantener su orden, (Eiter y Mannila, 1994).

Tal como se muestra en (Fréchet, 1906), la distancia de Fréchet se describe de la siguiente manera: dado una curva como un mapa continuo $f : [a, b] \rightarrow V$, donde $a, b \in R$, $a \leq b$ y (V, d) es una métrica de espacio.

Dadas dos curvas $f : [a, b] \rightarrow V$ y $g : [a', b'] \rightarrow V$, su distancia de Fréchet se define como:

$$\delta_F(f, g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} d(f(\alpha(t)), g(\beta(t))) \tag{2.78}$$

Donde α y β son funciones arbitrarias continuas y no decrecientes que van desde $[0, 1$ hasta $[a, b$ y d es es una función de distancia de V.

Si interpretamos a t como el orden de los elementos de cada curva, (pomo ejemplo el tiempo). Entonces $f(\alpha(t))$ y $g(\beta(t))$ son los puntos correspondientes de cada curva en el momento t y $d(f(\alpha(t)), g(\beta(t)))$ es la distancia entre estos dos puntos. Se busca el m´ınimo de todas las reperametrizaciones de $[0, 1]$correspondientes a elegir la ruta para la cual la distancia entre las curvas se minimiza. Dicho de otro modo, la progresión de puntos en una de las curvas puede ir más lento que la otra dependiendo de que opción minimice la distancia entre puntos. Además, la restricción en α y β limitan que la progresión de puntos sea de manera contraria.

3. Metodolog´ıa

En este capítulo de describe el proceso llevado a cabo en este proyecto de investigación. Como se mencionó en la Introducción la finalidad es demostrar de manera estadística si el sistema basado en CNN integrado en un drone tiene un mejores resultados que un sistema que emplea únicamente el expertiz de un usuario convencional al transitar por una sendero no conocido. Para esto se desarrollaron y probaron diferentes algoritmos de segmentación en una plataforma que permita simular el funcionamiento de un drone convencional. Los elementos y el orden con el que se procedera´ con la metodolog´ıa se muestran a continuación:

- Plataforma de simulación: Se describe la plataforma en la cual se desarrolla la simulación y los elementos en ella. Por otra parte se mostrará la interacción de la plataforma con el simulador de drones.
- Bases de datos real y de la simulación: Se muestra como están constituidas cada una de las bases de datos empleadas.
- Algoritmos de segmentación:]Se proporciona una breve explicación de cada uno de los algoritmos empleados.
- Conducción del drone: Se muestra la metodología propuesta para mantener el drone sobre el camino.

Se realizaron dos etapas de prueba divididas de la siguiente manera

- Pruebas de segmentación: Realizadas con los conjuntos de imágenes reales y creados a partir de la simulación.
- Pruebas en la ruta obtenida: Análisis estadístico de las rutas obtenidas.

El sistema propuesto sigue la metodología mostrada en 3.1, en el cual interactúan los diferentes componentes listados anteriormente.

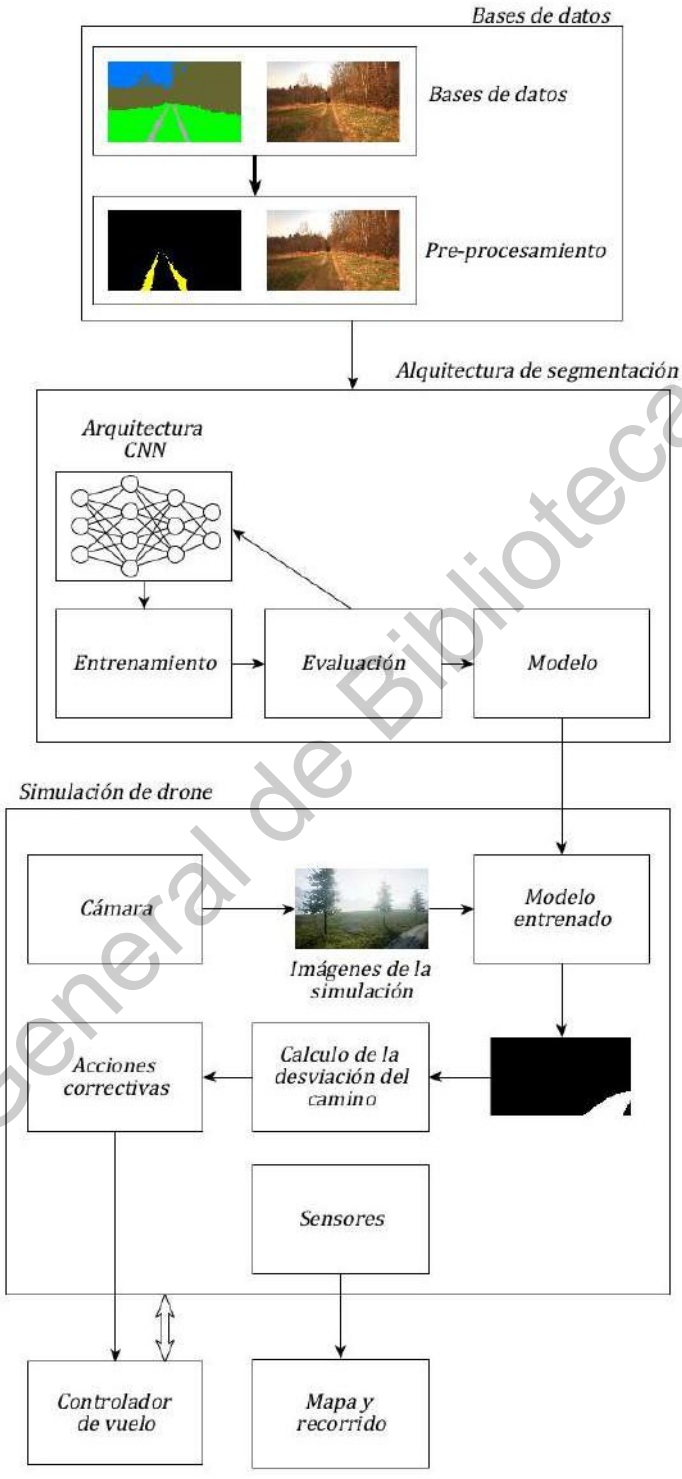


Figura 3.1: Metodolog´ıa propuesta del sistema de seguimiento de senderos.

3.1. Plataforma de simulación

La plataforma de simulación en la que se desarrolló el sistema esta basado en Unity tomando como base el proyecto *Mountain Landscape*, (Epic Games, s.f.). A dicho proyecto se le añadieron diferentes elementos que permiten la adición de caminos de distancia y forma que el usuario requiera. Los elementos que forman parte del entorno se enumeran a continuación:

- Mallas: Son los diferentes elementos en la simulación ajenos al suelo.
  - Arboles: Con y sin nieve. Se puede modificar las posición en la que aparecerá y el color del árbol (no modificado). El tamaño no es modificable.
  - Puentes. Un solo tipo de puente. Se puede modificar el tamaño, posición y orientación.
  - Piedras: Un solo tipo. Se puede modificar el tamaño y la posición.
  - Jugadores: Se configura la posición y orientación inicial del jugador, la cantidad de cámaras y su orientación además del punto de vista que esta tendrá.
- Lanscape: Se considera en esta categor´ıa a todos los elementos que formen el suelo.
  - Montañas: Modificando el landcape se puede elevar o depreciar la superficie así como añadir relieves a la misma. Se puede añadir pasto (verde) o nieve (blanca) en diferentes proporciones.
  - Caminos: El camino se ajusta a la forma de la superficie en la cual se añade únicamente se modifica la localización del siguiente punto.
  - Agua: Tiene el mismo comportamiento que las montañas, no puede ser atravesada.
- Camino (propio): Basado en una Spline con la cual se puede formar un camino abierto o cerrado. Se da forma al camino añadiendo porciones del mismo, estirándolo y dando ángulo.

3.1.1. Simulador de drones

Para asegurar una plataforma estable para realizar diferentes pruebas se optó por realizar- las sobre un entorno virtual realista. El sistema se compone por un mundo virtual descrito anteriormente empleando como base los componentes de *Mountain Landscape* proporcionado por (Epic Games, s.f.) y un simulador de drones llamado AirSim (Shah, Dey, Lovett, y Kapoor, 2017) que funciona sobre el mundo virtual.

Se creó un componente externo al entorno suministrado por Unreal Engine (UE4). Dicho componente está basado en splines y ayuda a generar un camino de manera simple. Únicamente tiene opciones como crear una spline cerrada o abierta, mantener visible la spline o el camino. Este componente fue necesario ya que los caminos generados en el Landscape son parte del mismo y no pueden ser segmentados mientras que el componente creado forma parte de las mallas.

El drone simulado es un cuadricóptero que emplea diferentes sensores como lo son Lidar, IMU, GPS, cámara de distancia, entre otros. Para las pruebas se empleó un sensor de distancia orientado a -90 grados en el eje de vuelo de elevación (cabeceo), para determinar la altura del drone, respecto al piso en la simulación, dado que la posición del drone es relativa al centro del mapa. La cámara incorporada en el drone tiene como características un ángulo de visión de 90 grados, orientado a $(0,\ 0,\ 0)$ grados respecto al cuerpo de la aeronave, velocidad de exposición de 100 y gama objetivo de $1{,}5$. Las imágenes que arroja son de segmentación de las mallas en la imagen y la imagen del panorama en un formato RGB y un tamaño de 360×420 píxeles.

También se incorporó un controlador de vuelo PixHawk 2.4.8, empleando HIL (Hardware-In-the-Loop), con la finalidad de incorporar mayor estabilidad en el vuelo y para incorporar un transmisor de radio. El controlador de vuelo tiene conexión con el sistema vía USB, la comunicación entre la simulación y el controlador de vuelo requiere de un programa extra que interprete los comandos en el protocolo MAVLink. El controlador de vuelo se encarga de realizar la estimación en la posición del drone a partir de los diferentes sensores incorporados, y ejecuta un filtro Kalman extendido (EKF) para la realización de dicha estimación.

Otra función del controlador de vuelo es determinar los valores de PWM suministrados a los motores en la simulación, ya que para las diferentes acciones se requiere un comportamiento especifico de los motores. En Fig. 3.2 se muestra las conexiones entre el simulador de drones, el mundo virtual y el controlador de vuelo.

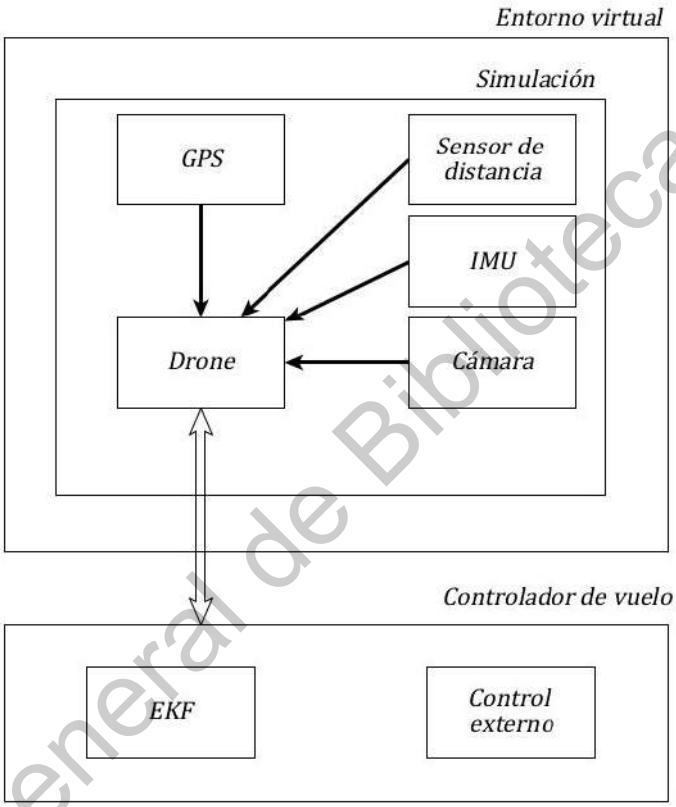


Figura 3.2: *Interacción entre los componentes del sistema*.

En Fig. 3.2 se muestra como los sensores que emplea el simulador de drones adquiere las mediciones del mundo virtual, es decir que las lecturas del GPS estarán referenciadas a un punto del mundo virtual que en este caso es el punto $(0, 0, 0)$. Estas mediciones se env´ıan al controlador de vuelo el cual regresa la estimación de la posición de la aeronave así como las señales de los motores.

3.1.2. Comunicación con el controlador de vuelo

La comunicación entre el PC o alguna computadora de vuelo y el controlador de vuelo se logra mediante el uso de un software de código abierto llamado *MavProxy* la cual actúa como interfaz entre comandos y señales enviadas. Para enviarlo es necesario determinar por que medio se enviaran y recibirán lo mensajes y por que medio se interpretaran los mensajes. Las opciones son USB, tty, TCP, UDP.

3.2. Bases de datos

Se emplearon dos bases de datos: una empleando imágenes reales tomada de (Valada y cols., 2016) y la otra generada a partir de la simulación. El conjunto de imágenes reales nos permitirá pre-entrenar los modelos para obtener algunas métricas de interés mientras que la base de datos creada nos permitira´ desarrollar el sistema completo.

3.2.1. Base de datos con imágenes reales

La base de datos con imágenes reales consta de 360 imágenes de 870×490 píxeles. La base de datos consta de diferentes elementos, entre ellos *scene* (escena) y *segmentation* (segmentación). Las imágenes en escena son RGB del entorno y las imagenes de segmentación son etiquetas con elementos como vegetación, camino, vació, etc. Esta base de datos corresponde a un bosque en Alemania. Esta base de datos es pública. Las imágenes se obtuvieron de un recorrido en un bosque con un sendero. Se mantuvo la altura de la cámara como se observa en la Figura 3.3. No se tiene un orden en las imágenes, es decir no se puede apreciar de manera clara que las imágenes correspondan a un recorrido completo.

(a) Escena.

(b) Segmentación.

Figura 3.3: *Escena a) y Segmentación b) de base de datos tomada de* (Valada y cols., 2016)

Las imágenes en esta base de datos son variadas, permitiéndonos dividirlas en tres casos:

- Contraste correcto: Imágenes con un nivel de luminosidad adecuado permitiendo distinguir claramente el sendero.
- Contraste bajo: Imágenes con un nivel de luminosidad bajo en las cuales no se puede distinguir claramente el sendero, sin embargo en la etiqueta de la imagen el sendero si fue segmentado.
- Imágenes sin sendero: Imágenes de en las cuales no aparece ningún sendero por lo tanto no se tiene una etiqueta.

La base de datos es bastante completa ya que cuenta con anotaciones en profundidad y de segmentación, sin embargo los dos conjuntos que tomamos para desarrollar las pruebas son *scene* y *RGB_annotated*. Una de las imágenes pertenecientes a escena es la mostrada en 3.3a la cual podriamos catalogar como contraste bajo ya que la sombra en la parte inferior de la imagen no permite ver claramente el sendero, su correspondiente imagen segmentada se muestra en 3.3b.

3.2.2. Base de datos de la simulación

Se decidió crear una base de datos lo suficientemente grande y significativa como para lograr el entrenamiento satisfactorio del sistemas. La base de datos se generó a partir de las

imágenes de escenario y segmentación. Se inició el mundo virtual y la simulación de drone para posteriormente llevar a cabo la adquisición de dichas imágenes y almacenarlas en formato JPG.

Se tomaron cuatro fases para la generación de la base de datos. En una se intentaba que el centro del camino y el centro de la imagen se mantuvieran alineados (funcionamiento óptimo), en otra se procuraba que el centro del camino se mantuviera en el extremo izquierdo o en el extremo derecho y la tercer fase se procuraba que el camino no apareciera en la imagen.

En total se generaron 5000 imágenes de 360×240 píxeles con la características antes mencionadas y un mapa de puntos del recorrido en tres ejes. En la Figura 3.4 se muestra tanto la imagen de escena como su segmentación.

(a) Escenario. (b) Segmentación.

Figura 3.4: Imágenes que componen la base de datos.

En la Figura 3.4a podemos apreciar diferentes componentes que afectan la imagen, como son iluminación del sol en un punto del panorama y sombras de los diferentes componentes. En la Figura 3.4b se aprecian cuatro componentes segmentados, siendo *landscape* el más extenso, ya que abarca todo el mundo virtual y únicamente los componentes compuestos por mallas serán segmentados.

3.2.3. Pre-procesamiento para segmentación

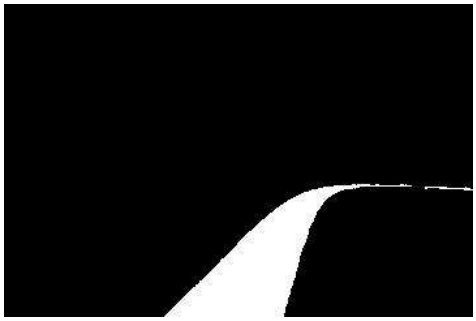
Fue necesario aplicar diferentes pre-procesamientos a las imágenes empleadas ya que no todas las etiquetas fueron significativas y en otros casos por las caracter´ısticas de las arquitecturas de segmentación. De manera general se obtuvieron las imágenes RGB y sus correspondientes imágenes objetivo también llamadas *ground-truth*.

Obtención de etiquetas de interés Para dividir las etiquetas se empleo un algoritmo de *thresholding* o de umbral ya que cada una de las etiquetas corresponde a un color definido. El algoritmo empleado mostrado en (1.1) fue empleado para cada una de las imágenes en el conjunto de datos de segmentación. La figura 3.5 muestra los resultados en el pre-procesamiento de imágenes para segmentación en ambas bases de datos.

(a) Imagen original, base de datos real.

(b) Imagen procesada, base de datos real.

(c) Imagen original, base de datos creada.



(d) Imagen procesada, base de datos creada.

Figura 3.5: *Imágenes pre-procesadas resaltando una etiqueta*

Normalización La normalización es el re-escalamiento de valores, en procesamiento de imágenes se mapea el valor de intensidad de color para cada píxel. Este proceso es necesario ya que si no se tiene un definido un valor máximo y mínimo para cada imagen las operaciones entre im'agenes no corresponderían a valores concretos. La normalización *minmax* empleada en este proyecto esta defienda a continuación:

$$y = (\text{máx}_o - \text{mín}_o) * \frac{(x - \text{mín}_i}{(\text{máx}_i - \text{mín}_i} \tag{3.1}$$

Donde mín_i y máx_i son los valores mínimos y máximos del rango de valores de x y mín_o y máx_o son los valores del rango de salida de y.

Este proceso cambia la escala del rango de cada imagen de entrada en un rango de $[0, 1]$. Este proceso permite que la red neuronal determine caracter´ısticas relevantes en la imagen sin cambiar la relación entre píxeles de la imagen acelerando el proceso de entrenamiento.

Aumento de datos Debido a que la base de datos con imágenes reales no es muy grande se opto por realizar algunas técnicas de aumento de datos. Estas técnicas consisten en alterar tanto las imágenes de entrada como las imágenes objetivo de la misma manera de modo que ambas tengan la misma alteración. Este aumento de datos se realiza de manera aleatoria en un 20 % de probabilidad. A continuación se lista las técnicas de aumento de datos empleadas, de mismo modo en 3.6 se muestra un ejemplo de cada técnica.

- Random cropping: En español corte aleatorio. Consiste en aumentar el tamaño de la imagen para después cortar una sección de la imagen con tamaño original. Este proceso ayuda en el robustecimiento de las redes entrenadas con bases de datos pequeñas.
- Random flip: El giro aleatorio en una técnica simple en la que la imagen resultante tiene un efecto espejo.

(a) Imagen original. (b) Random Cropping. (c) Random flip.

Figura 3.6: *Imágenes pre-procesadas resaltando una etiqueta*.

3.3. Algoritmos de segmentación

Se emplearon tres arquitecturas basadas en redes neuronales convolucionales y una método de comparación basado en agrupamiento. Las arquitecturas seleccionadas son U-Net (Ronneberger y cols., 2015), Pix2Pix (Isola y cols., 2017) y Mobile-Net V2 (Sandler y cols., 2018). La razón principal es encontrar el mejor funcionamiento al comparar una red que se ha demostrado es funcional en el campo de segmentación, una red que emplea características innovadoras y una red que optimizada para realizar la segmentación con baja latencia.

U-Net Esta arquitectura es ampliamente referenciada en tareas de segmentación semántica. La arquitectura, tal como se muestra en la Fundamentación teórica, cuenta con dos caminos: el de expansión y el de contracción.

- Contracción: Empleando filtros convolucionales de 3×3 se amplia la profundidad de los mapas de características y con operaciones *Max-pool* se reduce el tamaño de la imagen de entrada a la mitad.
- Expansión: Se emplean capas llamadas *Upsample* compuesta por una capa convolucional transpuesta que reduce la profundidad del mapa de caracter´ısticas.

El proceso de contracción y extracción simple tiende a ser muy extremo por lo que es frecuente una pérdida de información importante, por ello en la arquitectura U-Net se añaden inter-conexiones entre los dos procesos.

Pix2Pix Pix2Pix es una red generativa adversaria condicional. Esta red es empleada en tareas de segmentación, reconstrucción de imágenes, reducción de ruido, etc. La red aprende a mapear imágenes de entrada a una imagen objetivo. Esta red cuenta con dos modelos entrenados de manera simultanea. El modelo llamado generador se entrena para generar una imagen lo más parecido a la imagen objetivo y el modelo llamado discriminador se entrena para distinguir una imagen generada de una objetivo. El modelo generador se basa en una U-Net y el modelo discriminador se muestra en la tabla 4.

Mobile-Net V2 Mobile-Net es una arquitectura que emplea convoluciones separables en profundidad que reduce significativamente el coste convolucional del modelo. Esta red fue desarrollada para operaciones de baja latencia y es funcional en dispositivos móviles y de bajas prestaciones. Del manera similar a U-Net, Mobile-Net V2 emplea bloques de cuello de botella en los que se amplia el número de canales del mapa de características. El modelo empleado se muestra en la tabla 3.

Agrupamiento El método empleado es el agrupamiento por k-medias. Este algoritmo fue propuesto por MacQueen en 1967, es considerado uno de los algoritmos de aprendizaje no supervisado más simple. El procedimiento se inicia con la colocación de los k centroides, uno para cada grupo, estos deben colocarse de manera que puedan repartirse a lo largo de la base de datos es decir tan lejos como se pueda uno de otro. El siguiente paso es asociar cada punto de la base de datos con el centroide más cercano empleando una ecuación de distancia por ejemplo 3.2, una vez determinado a que centroide pertenece cada punto se debe recalcular el centroide a partir de los puntos asociados a el para después nuevamente determinar que puntos se asocian con los nuevos centroides.

$$d_E(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.2}$$

Este algoritmo tiene como objetivo minimizar la función objetivo mostrada en 3.3, para este caso una función de error cuadrado.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_i \right\|^2 \quad (3.3)$$

Donde $\left\| x_i^j - c_i \right\|^2$ es la distancia entre el centroide elegido c_i y el punto $x_i^{(j}$ de los datos de entrada.

El cálculo de los nuevos centroides se logra con la ecuación 3.4.

$$c(i) = mean(x(i)) \quad (3.4)$$

Donde el nuevo centroide es igual a la media de los puntos asociados a ese centroide.

3.4. Conducción del drone

Con el procedimiento anterior se procedió a determinar que píxeles en la imagen son parte del camino por lo que podemos considerar que la imagen generada a partir de los modelos de segmentación corresponde al camino, a dicha imagen la llamaremos máscara. Con la máscara como entrada se lleva a cabo el proceso en la Figura 3.7.

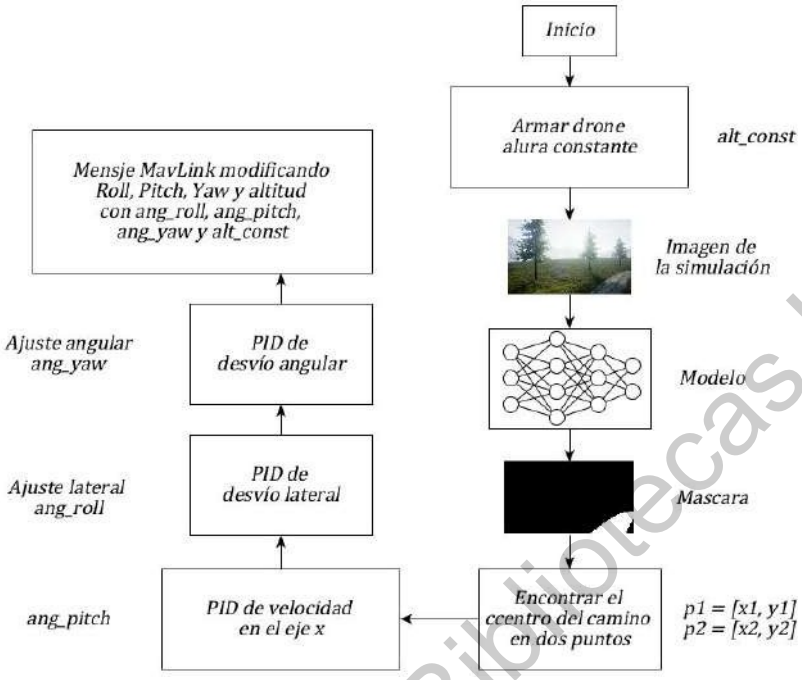


Figura 3.7: *Metodolog´ıa empleada para el control del drone.*

En el proceso mostrado en 3.7 el primer paso es inicializar el drone para posteriormente armarlo (encender motores y mantenerlo en una posición definida). Posteriormente comienza la transición de vídeo y la segmentación del camino. Se requiere una velocidad constante de modo que el drone se mantenga siempre en movimiento. A continuación se procede con los controles PID y finalmente se manda el mensaje recopilando todas las variables calculadas.

Control PID Se elaboro una función que lleva a cabo los tres controles. Cada uno de los controles tiene una variable de control diferente: PID de velocidad controla la velocidad en el eje `x` a partir del ángulo *YAW*. El PID de desvío lateral mantiene el centro del camino en el centro de la imagen controlando el ángulo *ROLL* y el PID de desvío angular mantiene el camino alineado con el drone controlando el ángulo *PICH*. El algoritmo empleado se muestra a continuación.

```
%%%PID %%%
kp, ki, kd       %variablesdelcontrolador
```

```
a_val, l_val        %valor anterior, valor actual
set_point           %valor deseado
ts                  %tiempo de  muestreo
sat_max, sat_min  %saturaci n  m xima  y  m nima

defPID(kp, ki, kd, a_val, l_val, set_point, ts, sat_max, sat_min):
    err = set_point - a_val
    d_err = err - (set_point - l_val)
    pid_ = (kp * err) + (kd * d_err) + (ki * ts * err)

    ifpid_ > sat_pos:
        pid_ = sat_pos
    elifpid_ < sat_neg:
        pid_ = sat_neg

    returnpid_
```

Listing 1: Python example

Desvío lateral El desvío lateral corresponde al valor numérico de desplazamiento del camino respecto al centro de la imagen, tal como se muestra en la figura 3.8. La corrección de este desvío se realiza al variar el ángulo *ROLL* en un intervalo de $-0{,}005, 0{,}005$ radianes.

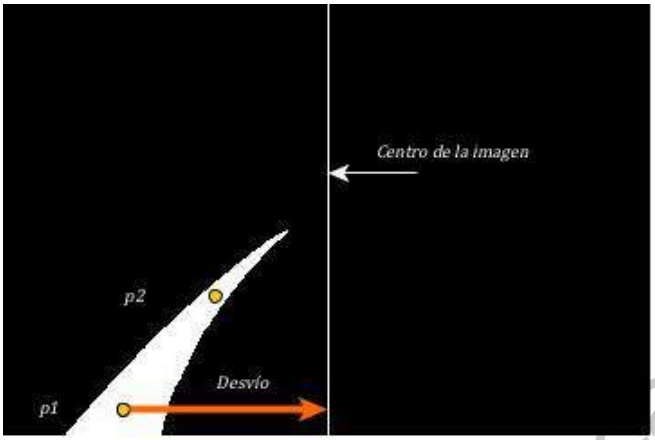


Figura 3.8: *Desv´ıo lateral*.

Desv´ıo angular Se da un desv´ıo angular cuando los dos puntos que marcan el centro del camino no se encuentran paralelos respecto a la imagen, es decir el ángulo que se forma ϑ_a es diferente de $\pi/2$, este desvío se compensa variando el ángulo *YAW*. En la Figura 3.9 se muestra de manera gráfica.

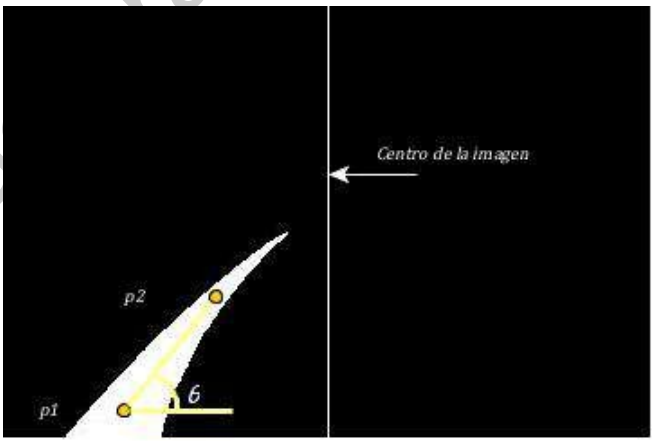


Figura 3.9: *Desv´ıo angular*.

3.5. Comparación de similitud entre rutas

Este procedimiento ayudó a determinar que sistema tendría una mayor repetibilidad mediante la similitud entre trayectos efectuados por el mismo sistema. El valor de similitud está dado por la distancia de Fréchet mostrada en el Marco teórico. El proceso se muestra en la Figura 3.10.

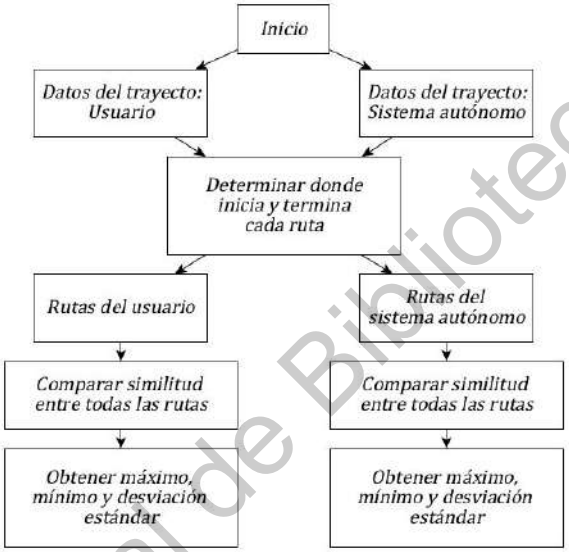


Figura 3.10: *Obtención de similitud entre rutas del mismo sistema.*

El proceso mostrado en 3.10 consta de dos procesos cr´ıticos: Determinar donde inicia y termina cada ruta del mismo sistema y encontrar la similitud entre rutas. Como se mencionó anteriormente, la similitud esta dada por la distancia entre rutas alternando de manera arbitraria la velocidad en que se toman puntos de cada ruta de la comparación. Para determinar los puntos iniciales y finales de cada ruta se procedió con los siguientes pasos:

1. Tomar tomar el punto inicial de la serie de datos $D(x, y)$ como l_1.

2. Avanzar en los datos 50 puntos.

3. Recorrer los datos de uno en uno desminando la distancia entre cada punto $D(x_i, y_i)$ y l_1.

4. Sea $D(x_j, y_j)$ el punto con la distancia más corta, tomar $D(x_{j-1}, y_{j-1})$ el punto final de la ruta.

5. Tomar como punto inicial $D(x_j, y_j)$.

6. Iniciar el proceso nuevamente en el paso 2, terminar el proceso si se esta al final de los datos.

4. Resultados y discusión

En este cap´ıtulo se mostraran los resultados obtenidos de las diferentes pruebas. Primeramente los resultados empleando la base de datos reales para continuar con los resultados empleando la base de datos de la simulación así como el desempeño del sistema completo. Todos los resultados se mostrados se obtuvieron con una laptop con las siguientes caracter´ısticas:

- Procesador: Intel I7 10700H, 6 hilos, 2,6GHz – 5GHz. Tarjeta gráfica: RTX2060 con 6Gb VRAM.
- Memoria RAM: 16Gb.
- Unidad de estado solido: M.2 de 256 Gb.

Los programas y herramientas empleadas fueron:

- Tensorflow
- Pycharm
- Matlab

Se comparo el desempeño de diferentes modelos de segmentación, tres basados en redes convolucionales y uno basado en agrupación. Las características de los modelos basados en redes convolucionales se muestra en la tabla 5. Las métricas mostradas se realizaron con el uso del resumen de la red y al medir el tiempo que tarda en procesar una cantidad fija de imágenes. La cantidad de imágenes procesadas por segundo puede ser intuida por la cantidad de parámetros entrenables ya que un modelo muy pesado tiende a ser lento. Sin embargo la cantidad de parámetros entrenables no refleja la capacidad del modelo para segmentar ya que influye la manera en que el sistema esta´ siendo entrenado.

Tabla 5: Comparación de características de los modelos basados en CNN.

Arquitectura	Parámetros entrenables (M)	Imágenes procesadas por segundo
Pix2Pix	54.414	24
U-Net	1.941	37
Mobile-Net	0.409	34

Posterior a la comparación de los modelos se procedió a compara el desempeño del sistema completo mediante la la similitud que tiene un recorrido obtenido de manera automática y manual.

4.1. Resultados con la base de datos real

Los resultados mostrados en esta sección son tanto del desempeño en de la red en entrenamiento como en diferentes pruebas. La métrica para determinar el funcionamiento de los modelos es la intersección entre la unión (IoU). Los valores mostrados en el valor de pérdida únicamente validan que la red fue entrenada de manera correcta,

4.1.1. Valor de pérdida

La Figura 4.1 muestra de manera gráfica el comportamiento de la función de perdida en el entrenamiento de cada una de las redes comparadas empleando el 70 % de la base de datos de (Valada y cols., 2016). Por las caracter´ısticas de cada una de las redes se tienen diferentes funciones de pérdida. U-Net emplea entropía cruzada binaria con perdida logística, M-Net emplea el coeficiente Dice y el generador de Pix2Pix emplea la pérdida G mostrada en el capitulo Fundamentación Teórica.

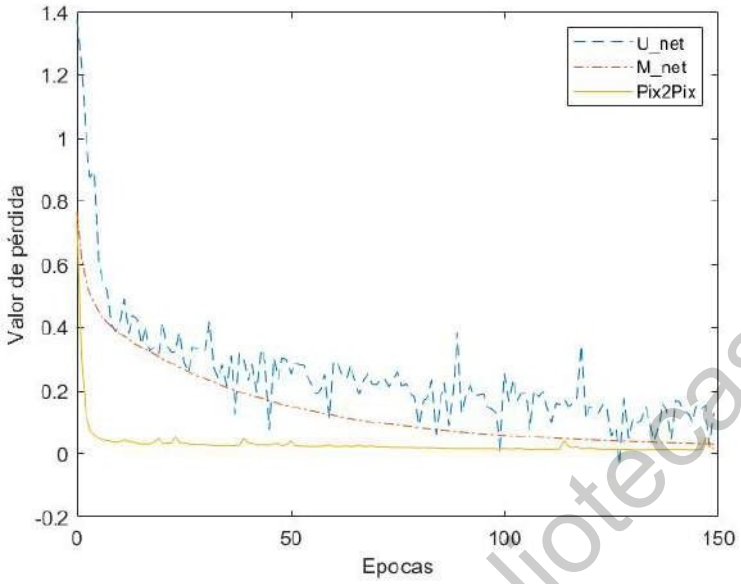


Figura 4.1: *Valor de pérdida en cada modelo.*

En la Figura 4.1 se pueden apreciar tres comportamientos diferentes. El modelo generador de Pix2Pix basado en U-Net tiene un comportamiento acelerado ya que en menos de 10 épocas se alcanza un valor mínimo estable pero se aprecian crestas en las que se tendría una mejor´ıa del modelo discriminador. Contrario a Pix2Pix, la arquitectura U-Net tiene valores de perdida variables y nada suavizados, podemos atribuirlo a la diferencia en la función de pérdida. El modelo Mobile-Net tiene una curva menos acelerada que Pix2Pix pero igualmente constante.

4.1.2. Comparación de imágenes

En la Figura 4.2 se muestran 5 imágenes de la base de datos de imágenes reales. Las 5 imágenes fueron separadas de los conjuntos de entrenamiento y pruebas. La comparación se realizó entre los modelos U-Net, Mobile-Net, Pix2Pix y la agrupación por k-medias. En la columna *Entrada* se tienen imágenes RGB de $256 \times 256 \times 3$ y es la imagen a segmentar. En la columna *Objetivo* se tienen imágenes en escala de grises de 256×256 y es la segmentación buscada.

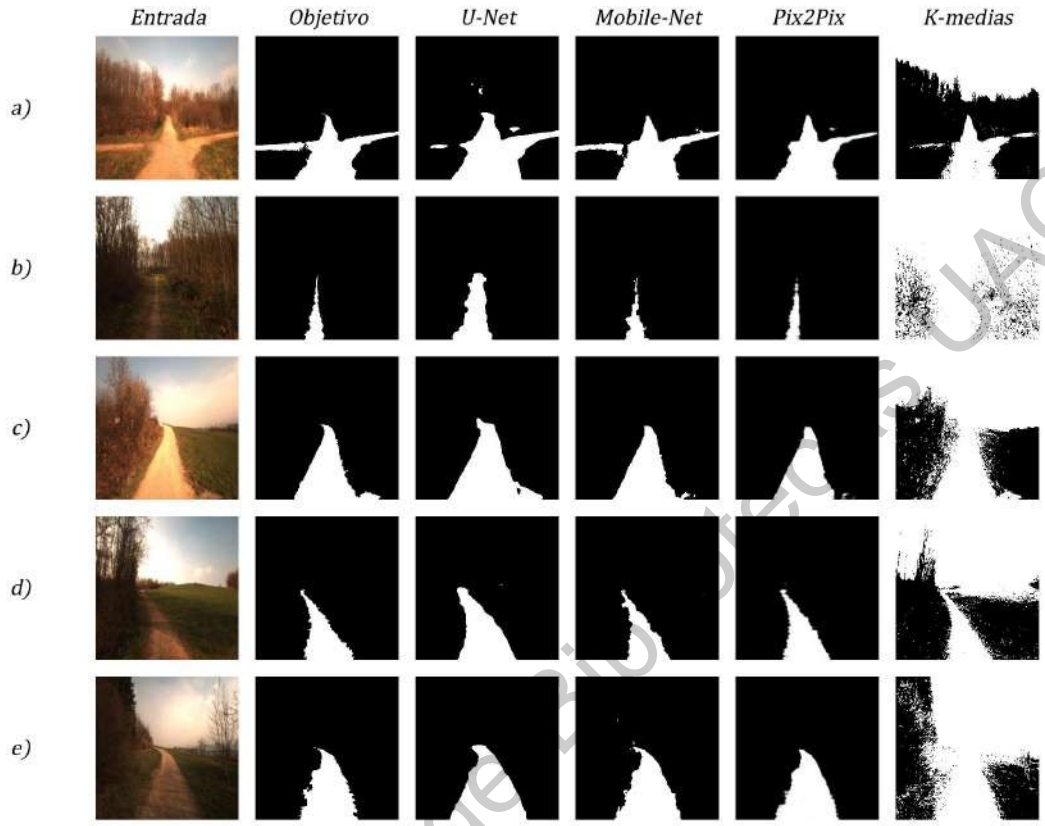


Figura 4.2: *Comparación de imágenes de entrada y generadas por cada arquitectura empleando base de datos de* (Valada y cols., 2016).

En el método de agrupación por k-medias se buscaron 4 clases diferentes tomando en consideración que la imagen de entrada podría tener: camino, vegetación, cielo y pasto. Para determinar que las imágenes elegidas fueron las correctas se calculó la intersección entre la unión para cada una de las clases siendo la elegida aquella el valor más alto. Las imágenes generadas por los cuatro modelos son de dimensión 256×256 para todos los modelos de segmentación.

Las imágenes de prueba en la Figura 4.2 fueron elegidas al azar, sin embargo se puede apreciar que no todas presentan un buen contraste entre el camino y la vegetación. El modelo k-medias es ineficiente ante este tipo de imágenes, como se aprecia en 4.30 *b)* no se logró una buena segmentación del camino. Si se inspecciona a mayor detalle cada una de las imágenes

generadas podr´ıamos notar que la imagen generada por U-Net presenta bordes cuadrados mientras que Pix2Pix y M-Net parecen más lisos.

4.1.3. Comparación de IoU empleando conjunto de pruebas

En los resultados mostrados se empleo el 30 % de la base de datos de (Valada y cols., 2016). En la Figura 4.3 se muestra el valor de IoU sobre todas las imágenes en el conjunto de prueba.

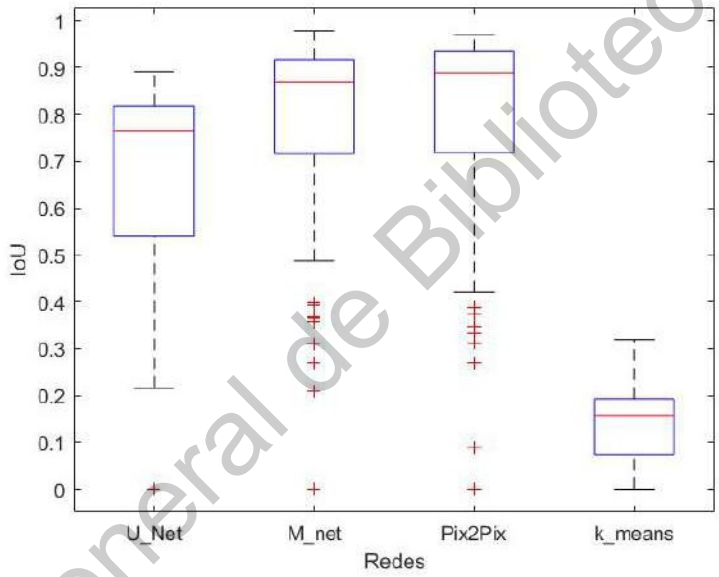


Figura 4.3: *Comparación de desempeño empleando IoU empleando el conjunto de pruebas de la base de datos real.*

En total se emplearon cerca de 100 imágenes para realizar la prueba de IoU. En la Figura 4.3 se que los trea modelos basados en CNN tienen un mejor desempeño que el modelo k-means, siendo Pix2Pix el que mejor resultado tiene. Las dos arquitecturas basadas en U-Net tienen un desempeño diferente siendo mejor el que emplea redes adversarias.

4.2. Resultados con la base de datos creada

Los siguientes resultados se obtuvieron al re-entrenar las previamente entrenadas con la base de datos de imágenes reales. Del mismo modo se mostrará el resultado del valor de pérdida para cada época. La comparación de imágenes se realizo con cinco imágenes de fuera de los conjuntos de entrenamiento y prueba. Las cinco imágenes fueron elegidas al azar.

4.2.1. Valor de pérdida

Se empleo el 100 % de la base de datos de prueba para re-entrenar los modelos. Los valores de pérdida para cada red se muestra en la Figura 4.4. Se emplearon 150 épocas como máximo pero un criterio de paro para cada red, siendo M-Net la única red en alcanzarlo.

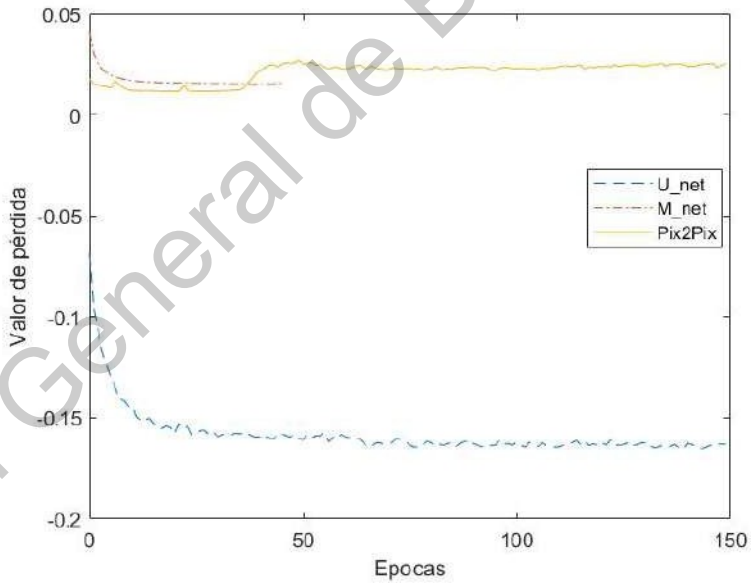


Figura 4.4: *Valor de perdida de cada modelo empleando base de datos propia.*

Los comportamientos en el entrenamiento de las redes al re-entrenar es muy diferente al mostrado en la Figura 4.1 iniciando con un valor de pérdida menor y mejorando en menor medida. En Mobile-Net si fue efectivo el pre-entrenamiento con la base de datos real ya que

se redujo el tiempo necesario para entrenar la red.

4.2.2. Comparación de imágenes

Se eligieron cinco imágenes de manera aleatoria y se separaron de los conjuntos de entrenamiento y prueba. Las cinco imágenes incorporan componentes muy parecidos en tanto a contraste entre el camino y la vegetación.

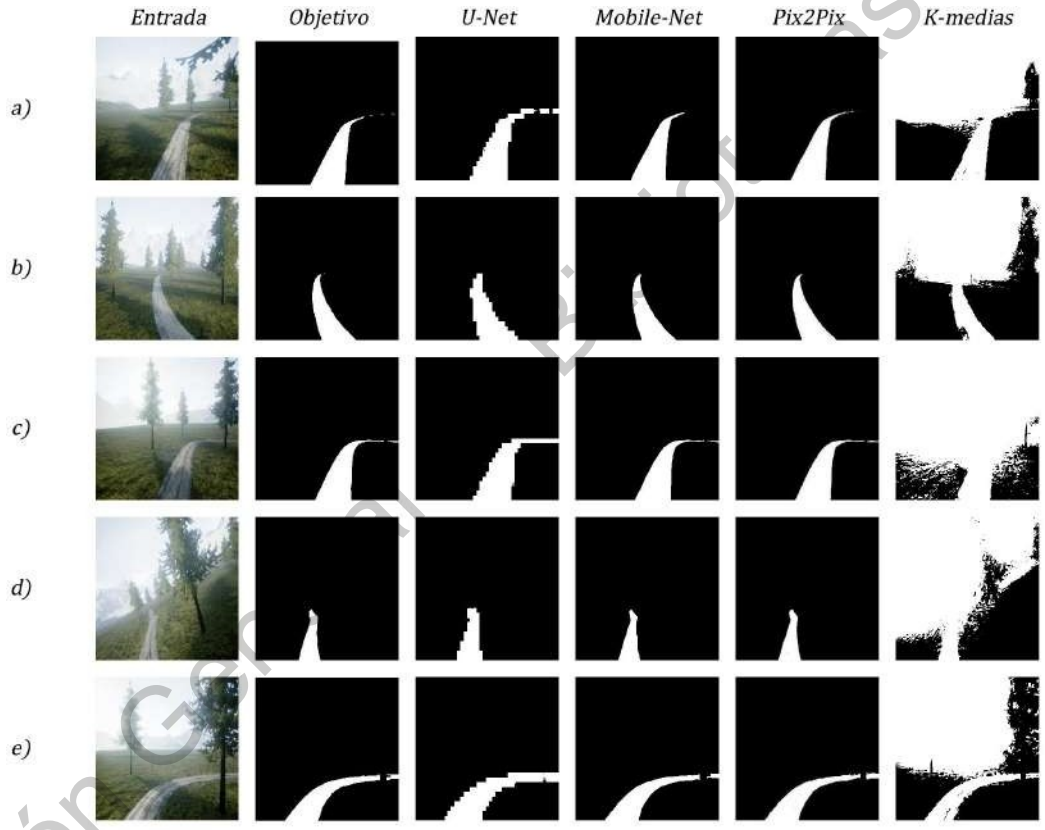


Figura 4.5: *Comparación de imágenes de entrada y generadas por cada modelo empleando base de datos propia.*

El modelo k-means tiene 4 clases y se determina que centro arroja los mejores resultados, las imágenes mostradas en la columna `K-means` son los mejores resultados. En estas imágenes se puede apreciar que el cielo y el camino pueden tener una tonalidad semejante por ello mismo es que las imágenes generadas con k-means no hacen diferencia entre el

camino y el cielo.

Las imágenes generadas por la arquitectura U-Net presentan bordes aun más pixelados que las demás arquitecturas, siendo más notorio en 4.33 *a)* y 4.33 *e)*. También se puede apreciar que en algunas ocasiones la imagen generada tiene componentes que no aparecen en la imagen objetivo como en 4.33 *c)*. Las imágenes generadas por los modelos Pix2Pix y Mobile-Net tienen resultados igualmente buenos y superiores a k-means.

4.2.3. Comparación

La comparación de desempeño de IoU se realizo con una base de datos generada a partir de un camino diferente. La base de datos de prueba consta de 1500 imágenes y su segmentación. Se aplicaron los mismos métodos mostrados de pre-procesamiento que a la base de datos de entrenamiento. En la Figura 4.6 se muestran los valores de IoU de cada uno de los modelos de segmentación.

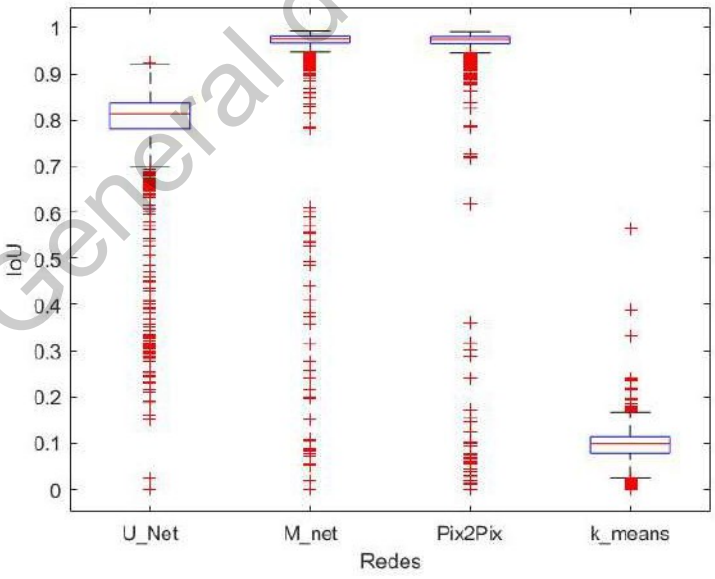


Figura 4.6: *Comparación de desempeño empleando IoU empleando el conjunto de pruebas de la base de datos propia.*

El desempeño del método k-means es muy inferior a los métodos basados en CNN. La ar-

quitectura Pix2Pix presenta mejores resultados que la arquitectura U-Net pero la arquitectura con mejor desempeño en Mobile-Net. Podemos atribuir la gran cantidad de valores atípicos al parecido que tienen muchas imágenes sin embargo no afectan en gran medida al valor medio de cada red.

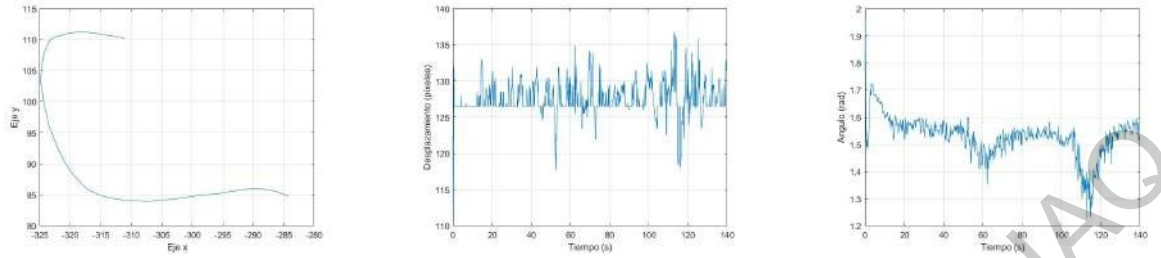
4.3. Resultados del sistema completo

Con los resultados previos se valida el uso de modelos de segmentación basados en CNN para este proyecto, por ello se empleo la arquitectura Mobile-Net ya que mostró mejores resultados de IoU con los datos de prueba de la simulación. El sistema funciona de la siguen te manera:

- Se obtiene imagen de la simulación.
- Se segmenta la imagen encontrando el camino.
- Se obtiene la desviación lateral y angular.
- Se Aplica la acción correctiva que mantiene al dron en el centro del camino.
- Se almacenan los datos de posición del drone y de los elementos externos.

4.3.1. Acción correctiva en curvas

Las acciones correctivas son más evidentes en las curvas y por ello se consideran coma las perturbaciones en el sistema. En la Figura 4.7 se muestran las acciones de control ejercidas por los controladores PID de desv´ıo lateral y de derivo angular. Dichas acciones de control se llevaron a cavo en una de las curvas por ello se muestra la proporción en el camino,
La imagen 4.7a muestra una parte del recorrido en la que se tiene la curva, 4.7b muestra la acción correctiva lateral y 4.7c muestra la acción correctiva lateral. El controlador PID lateral mantiene al sistema oscilando ± 9 p´ıxeles en la curva y el controlador PID angular lo mantiene ± 25 grados. Se le atribuye el ruido del sistema a las pequeñas irregularidades del



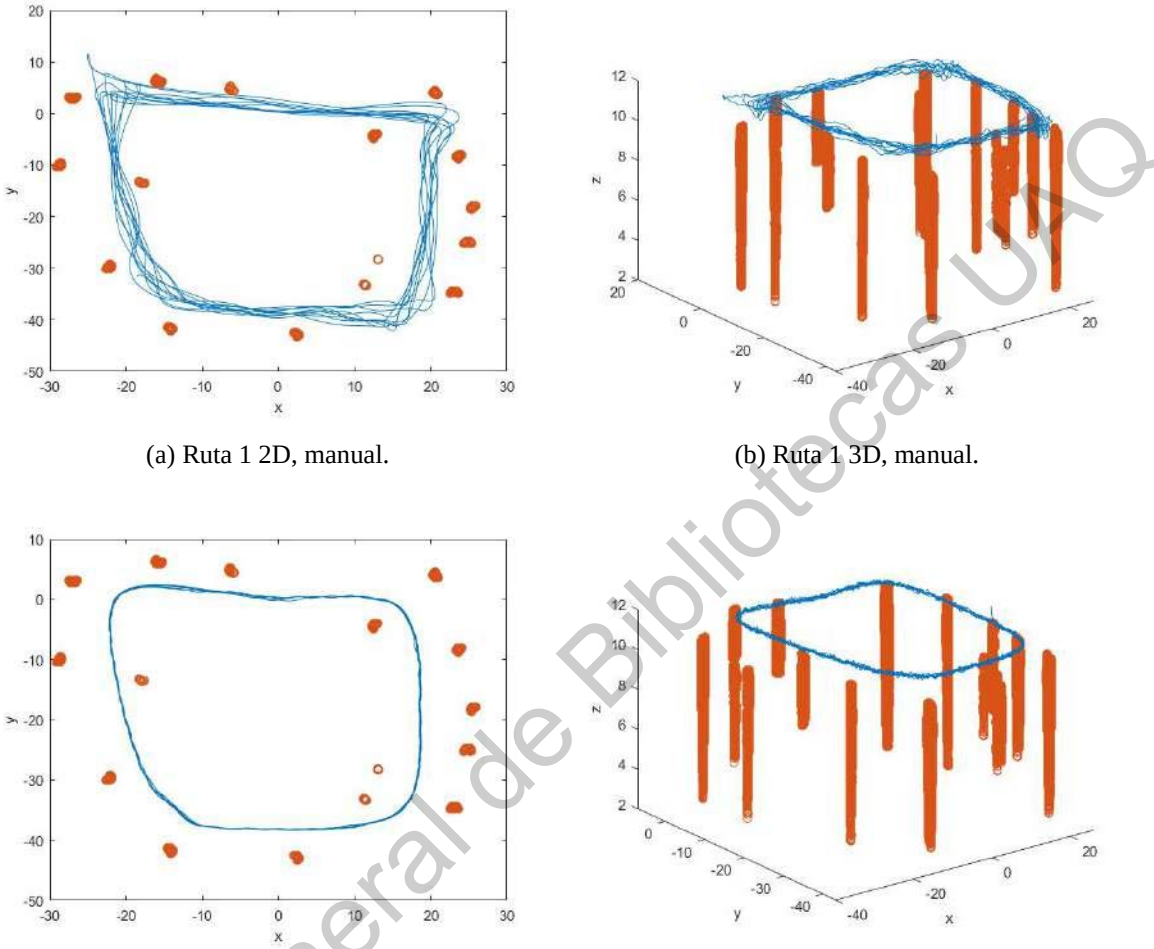
(a) Curva. (b) PID lateral. (c) PID angular.

Figura 4.7: *Acción correctiva de los controladores.*

camino en las que el centro del camino se mueve del centro de la imagen. Se decidió que los sistemas de control no ejercieran una acción demasiado grande para mantener la estabilidad del sistema.

4.3.2. Obtención del mapa

Se almacenaron los recorridos generados de manera manual y de manera autónoma así como los elementos en el trayecto. Los resultados mostrados en 4.8 se obtuvieron a partir de un camino generado de manera arbitraria, se muestran los recorridos en dos y tres ejes para tener la perspectiva de los elementos en el trayecto.



(a) Ruta 1 2D, manual.

(b) Ruta 1 3D, manual.

(c) Ruta1 2D, autónomo.

(d) Ruta1 3D, autónomo.

Figura 4.8: *Comparación de ruta 1 generada de manera manual y autónoma.*

En las Figura 4.8 se muestra el recorrido generado (azul) as´ı como los diferentes elementos en el entorno (naranja). Las figuras 4.8a y 4.8b corresponde a recorridos generados de manera manual, es decir, a partir de la experiencia del piloto de drones, Las figuras 4.8c y 4.8d se generaron de manera automática con el sistema funcionando.

Ya que no es posible determinar los valores exactos del recorrido real, se empleo una métrica de semejanza entre las rutas. Con el uso de la métrica distancia de Fréchet se puede cuantificar la semejanza entre recorridos sin la necesidad de que los recorridos tengan la misma dirección y dimensión. Si la semejanza entre recorridos es baja, cercana a cero, podemos

asegurar que el sistema tiene alta repetibilad. Los resultados de dicha evaluación se muestran en 4.9.

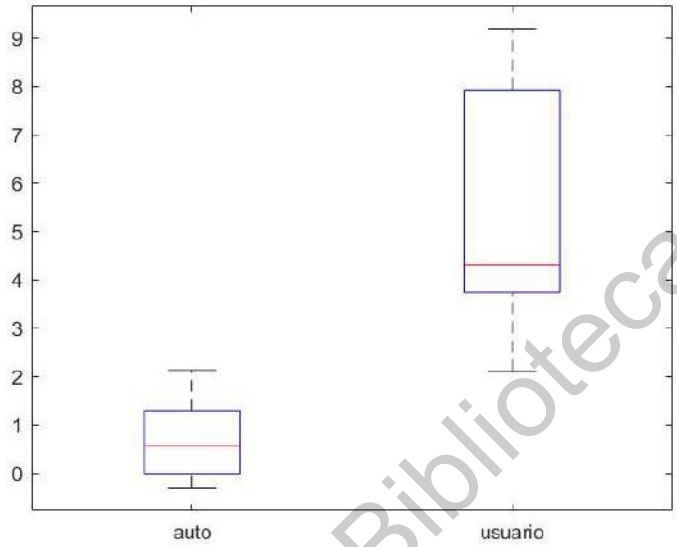


Figura 4.9: *Comparación de distancia de Fréchet empleando la ruta 1.*

Los valores en la Figura 4.9 muestran el correcto desempeño del sistema al obtener el trayecto del camino. Los resultados mostrados se obtuvieron a partir de 8 rutas diferentes por método (autonomo y manual) dando en total 27 combinaciones.

5. Conclusiones y trabajo futuro

Las diferentes pruebas realizadas con cada arquitectura mostró que no es necesario emplear un modelo con una cantidad excesiva de parámetros entrenables ya que la arquitectura Mobile-Net y U-Net mostraron un desempeño satisfactorio, sin embargo se demostró que el método de entrenamiento si influye en gran medida al desempeño de la red.

En contraste, la arquitectura U-Net tuvo un desempeño inferior a la arquitectura Pix2Pix a pesar de que ambas se basan en U-Net. Esto se debe a que la arquitectura Pix2Pix busca entrenar el modelo generador de manera apresurada y precisa mediante las redes adversarias. La imagen del renglón dos de la Figura 4.2 nos da una mejor idea del desempeño de las arquitecturas ya que se considera que las condiciones de luz pueden estar en constante cambio. La imagen muestra un bajo contraste entre el camino y la vegetación por ello resulta en una imagen difícil de segmentar, por otro lado la segmentación realizada por el modelo k-means no se desempeña bien con imágenes de estas condiciones.

Las imágenes mostradas en 4.5 tienen gran similitud dado que en la simulación las condiciones climatológicas y de luminosidad no cambian, sin embargo el desempeño de k-means no es muy sobresaliente ya que segmenta el camino y el cielo en la misma clase. Por otro lado la arquitectura U-Net presenta bordes pixelados en la imagen generada mismos que no se observan en Pix2Pix ni en Mobile-Net.

Se evaluaron las tres arquitecturas sobre una simulación funcionando en tiempo real, las tres arquitecturas tienen caracter´ısticas similares U-Net ((2015)) emplea bloques residuales, Mobile-Net ((2018)) también emplea bloques residuales y Pix2Pix ((2017)) emplea una U-Net como generador. La arquitectura con mayor desempeño empleando como comparación IoU es Mobile-Net por ello los resultados mostrados con el sistema completo emplea la red Mobile-Net.

También se mostró una comparación de las tres arquitecturas contra una de las técnicas clásicas de segmentación por agrupamiento. En las diferentes pruebas pudimos notar que este algoritmo es susceptible a los elementos fuera del camino como lo son sombras o árboles por ello creemos que el uso de algoritmos basados en CNN en segmentación esta justifi-

cado.

Uno de los factores que más resaltan es la cantidad de imágenes que pueden ser procesadas por segundo ya que este factor es fundamental para el correcto funcionamiento de un vehículo autónomo por ello se propone evaluar las tres arquitecturas al recorrer un trayecto empleando las mismas condiciones.

Las rutas generadas de manera autónoma muestran tener mayor similitud entre ellas ya que el usuario común no tiene noción de la ubicación del centro del camino. El drone simulado mantiene mucha inercia en las curvas lo que ocasiona que el vuelo sea irregular en estos puntos. El sistema puede implementar un controlador de velocidad en el eje x ya que los puntos del centro del camino permiten anticipar acciones. El control lateral tiene una oscilación máxima de 9 píxeles mismos que podemos atribuir a las irregularidades en el camino. También podemos apreciar estas irregularidades en las gráficas del camino ya que en el eje z se aprecian saltos constantes.

Este trabajo es un paso crucial en un proyecto de vehículos autónomos cuyo avance representa la validación y desarrollo de vehículos autónomos en entornos fuera de riesgo ya que combina el funcionamiento convencional de un drone con una simulación en tiempo real propiciando la integración de técnicas de inteligencia artificial. Finalmente la base de datos se encuentra disponible para el uso de la comunidad en general.

5.1. Trabajo futuro

- Generar una base de datos propia con imágenes reales.
- Implementar el sistema en un drone real.
- Probar arquitecturas en sistemas embebidos.
- Probar el sistema en un entorno real.

Referencias

Ahmad, N., Ghazilla, R. A. R., Khairi, N. M., y Kasi, V. (2013). Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, *1*(2), 256–262.

Al-Amri, S. S., Kalyankar, N., y Khamitkar, S. (2010). Image segmentation by using edge detection. *International journal on computer science and engineering*, *2*(3), 804–807.

Alt, H., y Godau, M. (1995). Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, *5*(01n02), 75–91.

Angelina, S., Suresh, L. P., y Veni, S. K. (2012). Image segmentation based on genetic algorithm for region growth and region merging. En *2012 international conference on computing, electronics and electrical technologies (icceet)* (pp. 970–974).

Aqel, M. O., Marhaban, M. H., Saripan, M. I., y Ismail, N. B. (2016). Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, *5*(1), 1897.

Badrinarayanan, V., Kendall, A., y Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481–2495.

Bhabatosh, C., y cols. (1977). *Digital image processing and analysis*. PHI Learning Pvt. Ltd.

Bloesch, M., Omari, S., Hutter, M., y Siegwart, R. (2015). Robust visual inertial odometry using a direct ekf-based approach. En *2015 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 298–304).

Brand, C., Schuster, M. J., Hirschmüller, H., y Suppa, M. (2014). Stereo-vision based obstacle mapping for indoor/outdoor slam. En *2014 ieee/rsj international conference on intelligent robots and systems* (pp. 1846–1853).

Bravo, V. A. O., Arias, M. A. N., y Cardenas, J. A. C. (2013). Análisis y aplicación del filtro de kalman a una senal con ruido aleatorio. *Scientia et technica*, *18*(1), 267–274.

Bresciani, T. (2008). Modelling, identification and control of a quadrotor helicopter. *MSc theses*.

Chen, S. W., Nardari, G. V., Lee, E. S., Qu, C., Liu, X., Romero, R. A. F., y Kumar, V. (2020). Sloam: Semantic lidar odometry and mapping for forest inventory. *IEEE Robotics and Automation Letters*, *5*(2), 612–619.

Cheng, Z., Li, S., Zhi, Y., Gerhard, N., Tom, D., y Rustam, S. (2019). Learning kalman network: A deep monocular visual odometry for on-road driving. *Robotics and Autonomous Systems*, *121*, 103234.

Craig, J. J. (1989). Robotics. *digital Encyclopedia of Applied Physics*.

Csurka, G. (2017). *Domain adaptation in computer vision applications*. Springer.

Dai, B., He, Y., Yang, L., Su, Y., Yue, Y., y Xu, W. (2020). Simsf: A scale insensitive multisensor fusion framework for unmanned aerial vehicles based on graph optimization. *IEEE Access*.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. En *Iccv* (Vol. 3, pp. 1403–1410).

Dayoub, M., Birech, R. J., Haghbayan, M.-H., Angombe, S., y Sutinen, E. (2020). Co-design in bird scaring drone systems: potentials and challenges in agriculture. En *International conference on advanced intelligent systems and informatics* (pp. 598–607).

de Aeronáutica Civil, D. G. (2017). Co av-23/10 r4. *Circular Obligatoria, SCT*.

Dehariya, V. K., Shrivastava, S. K., y Jain, R. (2010). Clustering of image data set using k-means and fuzzy k-means algorithms. En *2010 international conference on computational intelligence and communication networks* (pp. 386–391).

Dou, X., Huo, Y., Liu, Y., y Wang, X. (2019). An unmanned aerial vehicle pose estimation system based on slam. En *Iop conference series: Materials science and engineering* (Vol. 569, p. 042036).

Dubé, R., Cramariuc, A., Dugas, D., Sommer, H., Dymczyk, M., Nieto, J., … Cadena, C. (2019). Segmap: Segment-based mapping and localization using data-driven descrip-

tors. *The International Journal of Robotics Research*, 0278364919863090.

Dubé, R., Cramariuc, A., Dugas, D., Sommer, H., Dymczyk, M., Nieto, J., … Cadena, C. (2020). Segmap: Segment-based mapping and localization using data-driven descriptors. *The International Journal of Robotics Research*, *39*(2-3), 339–355.

Dumoulin, V., y Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Durrant-Whyte, H., y Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, *13*(2), 99–110.

Eiter, T., y Mannila, H. (1994). *Computing discrete fréchet distance* (Inf. Téc.). Citeseer.

Epic Games. (s.f.). *Unreal engine.* Descargado de `https://www.unrealengine.com`

Fernandez, D., y Price, A. (2004, Dec). Visual odometry for an outdoor mobile robot. En *Ieee conference on robotics, automation and mechatronics, 2004.* (Vol. 2, p. 816-821 vol.2). doi: 10.1109/RAMECH.2004.1438023

Fréchet, M. M. (1906). Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, *22*(1), 1–72.

Fukushima, K., y Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. En *Competition and cooperation in neural nets* (pp. 267–285). Springer.

Garcia, R., Rubio, F., y Ortega, M. (2012). Robust pid control of the quadrotor helicopter. *IFAC Proceedings Volumes*, *45*(3), 229–234.

Geiger, A., Lenz, P., Stiller, C., y Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR).*

Geiger, A., Lenz, P., y Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. En *Conference on computer vision and pattern recognition (cvpr).*

Gonzalez, R. C., y Woods, R. E. (2003). *Steven l. eddins digital image processing using matlab.* Prentice Hall, 1ra Edicion.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems, 27*.

Grewal, M., y Andrews, A. (2001, 01). Kalman filtering: theory and practice using matlab. *New York: John Wiley and Sons, 14*. doi: 10.1002/9780470377819

Gurusamy, V., Kannan, S., y Nalini, G. (2013). Review on image segmentation techniques. *J Pharm Res, 20125*, 4548–4553.

He, K., Gkioxari, G., Dollár, P., y Girshick, R. (2017). Mask r-cnn. En *Proceedings of the ieee international conference on computer vision* (pp. 2961–2969).

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hong, S., y Kim, J. (2016a). Efficient visual slam using selective image registration for autonomous inspection of underwater structures. En *2016 ieee/oes autonomous underwater vehicles (auv)* (pp. 189–194).

Hong, S., y Kim, J. (2016b). Efficient visual slam using selective image registration for autonomous inspection of underwater structures. En *2016 ieee/oes autonomous underwater vehicles (auv)* (pp. 189–194).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hwang, J., Kim, J. J., y Lee, K.-W. (2021). Investigating consumer innovativeness in the context of drone food delivery services: Its impact on attitude and behavioral intentions. *Technological Forecasting and Social Change, 163*, 120433.

Isola, P., Zhu, J.-Y., Zhou, T., y Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1125–1134).

Kang, W.-X., Yang, Q.-Q., y Liang, R.-P. (2009). The comparative research on image seg-

mentation algorithms. En *2009 first international workshop on education technology and computer science* (Vol. 2, pp. 703–707).

Katsuhiko, O. (2010). *Modern control engineering*.

Kayalibay, B., Jensen, G., y van der Smagt, P. (2017). Cnn-based segmentation of medical imaging data. *arXiv preprint arXiv:1701.03056*.

Khan, M. W. (2014). A survey: Image segmentation techniques. *International Journal of Future Computer and Communication*, *3*(2), 89.

Kim, J. U., Kim, H. G., y Ro, Y. M. (2017). Iterative deep convolutional encoder-decoder network for medical image segmentation. En *2017 39th annual international conference of the ieee engineering in medicine and biology society (embc)* (pp. 685–688).

Labbé, M., y Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, *36*(2), 416–446.

Lee, Y.-J., Park, J.-T., y Song, J.-B. (2011). Three-dimensional outdoor slam using rotation invariant descriptors of salient regions. En *2011 11th international conference on control, automation and systems* (pp. 1174–1177).

Lim, E. H., y Suter, D. (2009). 3d terrestrial lidar classifications with super-voxels and multi-scale conditional random fields. *Computer-Aided Design*, *41*(10), 701–710.

Lindeberg, T., y Li, M.-X. (1997). Segmentation and classification of edges using minimum description length approximation and complementary junction cues. *Computer Vision and Image Understanding*, *67*(1), 88–98.

Lu, Y., Chen, Y., Zhao, D., y Chen, J. (2019). Graph-fcn for image semantic segmentation. En *International symposium on neural networks* (pp. 97–105).

Luukkonen, T. (2011). Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, *22*, 22.

Mishra, B., Garg, D., Narang, P., y Mishra, V. (2020). Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, *156*, 1–10.

Mortazi, A., y Bagci, U. (2018). Automatically designing cnn architectures for medical

image segmentation. En *International workshop on machine learning in medical imaging* (pp. 98–106).

Muller, P., y Savakis, A. (2017). Flowdometry: An optical flow and deep learning based approach to visual odometry. En *2017 ieee winter conference on applications of computer vision (wacv)* (pp. 624–631).

Mur-Artal, R., y Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, *33*(5), 1255–1262.

Naidoo, Y., Stopforth, R., y Bright, G. (2011). Quad-rotor unmanned aerial vehicle helicopter modelling & control. *International Journal of Advanced Robotic Systems*, *8*(4), 45.

Naz, S., Majeed, H., y Irshad, H. (2010). Image segmentation using fuzzy clustering: A survey. En *2010 6th international conference on emerging technologies (icet)* (pp. 181–186).

Nister, D., Naroditsky, O., y Bergen, J. (2004, June). Visual odometry. En *Proceedings of the 2004 ieee computer society conference on computer vision and pattern recognition, 2004. cvpr 2004.* (Vol. 1, p. I-I). doi: 10.1109/CVPR.2004.1315094

Nüchter, A., Lingemann, K., Hertzberg, J., y Surmann, H. (2007). 6d slam—3d mapping outdoor environments. *Journal of Field Robotics*, *24*(8-9), 699–722.

O'Shea, K., y Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Özgen, A. C., y Ekenel, H. K. (2020). Words as art materials: Generating paintings with sequential gans. *arXiv preprint arXiv:2007.04383*.

Pierzchała, M., Giguère, P., y Astrup, R. (2018). Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam. *Computers and Electronics in Agriculture*, *145*, 217–225.

Quintana, A. G., Hassanalian, M., y Abdelkefi, A. (2018). Aerodynamic analysis of a morphing drone with spanning and sweeping in transition modes. En *2018 aiaa/ahs adaptive structures conference* (p. 0796).

Ragot, N., Khemmar, R., Pokala, A., Rossi, R., y Ertaud, J.-Y. (2019). Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map. En *2019 eighth international conference on emerging security technologies (est)* (pp. 1–6).

Rogowska, J. (2000). Overview and fundamentals of medical image segmentation. *Handbook of medical imaging, processing and analysis*, 69–85.

Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. En *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Rothman, D. (2018). *Artificial intelligence by example: Develop machine intelligence from scratch using real artificial intelligence use cases*. Packt Publishing Ltd.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., y Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4510–4520).

Senthilkumaran, N., y Rajesh, R. (2009). Image segmentation-a survey of soft computing approaches. En *2009 international conference on advances in recent technologies in communication and computing* (pp. 844–846).

Shah, S., Dey, D., Lovett, C., y Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. En *Field and service robotics*. Descargado de https://arxiv.org/abs/1705.05065

Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., ... Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, *7*, 48572–48634.

Shan, P. (2018). Image segmentation method based on k-mean algorithm. *EURASIP Journal on Image and Video Processing*, *2018*(1), 81.

Slabaugh, G. G. (1999). Computing euler angles from a rotation matrix. *Retrieved on August*, *6*(2000), 39–63.

Smolyanskiy, N., Kamenev, A., Smith, J., y Birchfield, S. (2017). Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awa-

reness. En *2017 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 4241–4247).

Snyder, W. E., y Qi, H. (2017). *Fundamentals of computer vision*. Cambridge University Press.

Songhui, M., Mingming, S., y Chufeng, H. (2019). Objects detection and location based on mask rcnn and stereo vision. En *2019 14th ieee international conference on electronic measurement & instruments (icemi)* (pp. 369–373).

Sturm, J., Magnenat, S., Engelhard, N., Pomerleau, F., Colas, F., Burgard, W., ... Siegwart, R. (2011, June). Towards a benchmark for rgb-d slam evaluation. En *Proc. of the rgb-d workshop on advanced reasoning with depth cameras at robotics: Science and systems conf. (rss).* Los Angeles, USA.

Subash, K., Srinu, M. V., Siddhartha, M., Harsha, N. S., y Akkala, P. (2020). Object detection using ryze tello drone with help of mask-rcnn. En *2020 2nd international conference on innovative mechanisms for industry applications (icimia)* (pp. 484–490).

Sultana, F., Sufian, A., y Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: a survey. *Knowledge-Based Systems*, *201*, 106062.

Szklarski, J., Ziemiecki, C., Szałtys, J., y Ostrowski, M. (2019). Real-time 3d mapping with visual-inertial odometry pose coupled with localization in an occupancy map. En *Conference on automation* (pp. 388–397).

Tustison, N., y Gee, J. (2009). Introducing dice, jaccard, and other label overlap measures to itk. *Insight J*, *2*.

Valada, A., Oliveira, G., Brox, T., y Burgard, W. (2016). Deep multispectral semantic scene understanding of forested environments using multimodal fusion. En *International symposium on experimental robotics (iser).*

Wang, Y., Peng, M., Di, K., Wan, W., Liu, Z., Yue, Z., ... Teng, B. (2019). Vision based obstacle detection using rover stereo images. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Yang, X., Li, X., Ye, Y., Lau, R. Y., Zhang, X., y Huang, X. (2019). Road detection and

centerline extraction via deep recurrent convolutional neural network u-net. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(9), 7209–7220.

Yuheng, S., y Hao, Y. (2017). Image segmentation algorithms overview. *arXiv preprint arXiv:1707.02051*.

Zhang, B. (2010). Computer vision vs. human vision. En *9th ieee international conference on cognitive informatics (icci'10)* (pp. 3–3).

Zhang, G., Lee, J. H., Lim, J., y Suh, I. H. (2015). Building a 3-d line-based map using stereo slam. *IEEE Transactions on Robotics*, *31*(6), 1364–1377.

Zhang, Q., Chang, X., y Bian, S. B. (2020). Vehicle-damage-detection segmentation algorithm based on improved mask rcnn. *IEEE Access*, *8*, 6997–7004.

Zhang, X., Zhou, X., Lin, M., y Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6848–6856).

Zhang, Z., y Wan, W. (2018). Dovo: Mixed visual odometry based on direct method and orb feature. En *2018 international conference on audio, language and image processing (icalip)* (pp. 344–348).

Zhao, J., Fang, Y., Hong, Q., Pan, Z., Huang, L., y Zhang, D. (2019). Uav-based identification of achnatherum splendens community combining k-means and artificial fish swarm algorithm. En *Igarss 2019-2019 ieee international geoscience and remote sensing symposium* (pp. 2439–2442).

Zhou, H., Kong, H., Wei, L., Creighton, D., y Nahavandi, S. (2016). On detecting road regions in a single uav image. *IEEE transactions on intelligent transportation systems*, *18*(7), 1713–1722.

Zoph, B., Vasudevan, V., Shlens, J., y Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 8697–8710).

6. Anexos

6.1. Consideraciones éticas

El proyecto se realizara en el campus Aeropuerto de la Universidad Autónoma de Querétaro sin ningún financiamiento externo. El proyecto no involucra pruebas con seres vivos ni es necesaria información de terceros.

A pesar de que en este documento no se específica una aplicación final, se mostraran las consideraciones necesarias aplicables en México para cada una de las aplicaciones mostradas tanto en antecedentes como en resultados esperados.

- Vehículo aéreo no tripulado. La norma CO AV-23/10 tiene vigencia desde el 25 de julio del 2017, (de Aeronáutica Civil, 2017). En ella se establecen los lineamientos para operar un aeronave tripulada a distancia (RPAS). Entre las consideraciones más importantes que se abordan en este circulatorio se resaltan las siguientes:
  - Registro del Drone. Siguiendo el siguiente procedimiento:
    - Llenar el formulario REGISTRO DE RPAS COMERCIALIZADOS EN MÉXICO o REGISTROS DE RPAS POR PROPIETARIOS según corresponda. Las ligas las encontrarás dentro del documento CO AV-23/10 R4.
    - Incluir documentos digitalizados que avalen la propiedad legal del equipo, en este caso puede ser la factura emitida por el vendedor, los formatos admitidos son .pdf, .docx, .jpg o .png.
    - Debes enviar un correo electrónico a la dirección rpas@sct.gob.mx solicitando registrar tu Drone, adjuntando el formulario completo en el formato proporcionado más adelante (Formato de Excel, Apéndice ”K.º Apéndice ”J”según sea el caso, junto el documento impreso firmado por el solicitante y agregando tu Registro Federal de Contribuyentes con homoclave, as´ı como las copias digitalizadas de la documentación que acredite la propiedad legal de tu Drone.

- Licencia de vuelo. La licencia es necesaria siempre y cuando el uso del Drone tenga fines comerciales (no aplicaria para el proyecto). En caso de que el Drone pese menos de 2 kg. y que su uso de recreativo no es necesario contar con licencia. En caso de desarrollar una aplicación empleando un Drone, se buscaría que este no pese más de 2 kg.
- Consideraciones de carácter común.
    - No volar sobre propiedades privadas sin permiso previamente solicitado
    - No volar cerca de aeropuertos y helipuertos ya que puedes provocar accidentes con pérdidas humanas
    - No invadir espacio aéreo asignado a otras aeronaves (no elevarse más de 122 metros sobre el piso del área de despegue)
    - No volar directamente sobre multitudes de personas, ya que el equipo puede caer accidentalmente.
    - Si en algún momento es necesario realizar algún vuelo excediendo los límites de altitud, siempre y cuando tengas justificación para hacerlo, puedes pedir permiso especial en la oficina de dirección general de aeronáutica civil más cercana a la locación de vuelo, usualmente en los aeropuertos existen oficinas de ésta dependencia con quien podrías asesorarte para lograrlo.

- Vehículo terrestre no tripulado. No existe una norma gubernamental que lo regule, sin embargo deberá solicitarse un permiso en caso de ser usado en áreas publicas como parques o dentro de la universidad. En ambos casos se buscara al encargado del espacio a usar y el encargado de seguridad para dar fe de que no se haga mal uso de los espacio y que no se dañe a terceros.

- Simulación. Si la aplicación es una simulación desarrollada en PC, no aplicaría ningún tipo de regulación ética o gubernamental.