INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Aplicación de redes neuronales para la identificación de objetos en tiempo real en imágenes tomadas por un quadrotor.

TESIS

Que para obtener el grado de

MAESTRIA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES.

Presenta

Ing. Gerardo Hernández Hernández

Directores

Dr. Juan Humberto Sossa Azuela

Dr. Carlos Aguilar Ibáñez

México, D.F. Noviembre 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 8 del mes de diciembre de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Aplicación de redes neuronales para la identificación de objetos en tiempo real en imágenes tomadas por un quadrotor"

Presentada por el alumno(a):

Hernández	**Hernández**	**Gerardo**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 1 3 0 2 3 1

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Carlos Fernando Aguilar Ibáñez

Dr. Juan Humberto Sossa Azuela

Dr. Adolfo Guzmán Arenas

Dr. Herón Molina Lozano

Dr. Juan Villegas Cortez

Dra. Elsa Rubio Espino

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

ESTADOS UNIDOS MEXICANOS
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D.F. el día 09del mes de Diciembre del año 2014, el (la) que suscribe Gerardo Hernánde Hernández alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con opción en Sistemas Digitales con número de registro A130231, adscrito al Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Juan Humberto Sossa Azuela y Dr. Carlos Fernando Aguilar Ibáñez y cede los derechos del trabajo intitulado Aplicación de Redes Neuronales para la identificación de objetos en tiempo real en imágenes tomadas por un Quadrotor, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección gerardohernandez.hernandez@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Gerardo Hernández Hernández

Nombre y firma

Resumen

En la presente tesis se proponen e implementan un conjunto de técnicas para la identificación de objetos (autos y maniquís a escala) en línea, con el fin de analizar la factibilidad de uso de herramientas como lo son los Quadrotores en tareas de vigilancia. Para realizar lo anterior, se realiza un estudio comparativo entre diferentes tipos de clasificadores estadísticos y redes neuronales artificiales.

Las técnicas propuestas hacen uso de le teoría de “puntos de interés”, que permiten extraer rasgos descriptores, que resultan ser robustos al ruido.

La plataforma robótica que se utiliza es un Quadrotor, que, como se sabe, es muy versátil por sus características intrínsecas, pero que resulta difícil en su manejo.

Como producto final se presenta una interfaz software, que permite visualizar la operación de las redes neuronales artificiales analizadas y presentar los resultados de la clasificación en tiempo real en forma gráfica al usuario.

Abstract

This thesis proposes and implements a set of techniques for identifying objects (like cars and human dummy at scale) in real time to analyze the feasibility of using Quadrotors in surveillance task. To do this, a comparative study is performed based on different types of statistics classifiers and artificial neural networks.

The proposed techniques use the theory of "interest points" which is a feature extraction technique, where the extracted features are robust to noise.

The robotic tool used is a Quadrotor, which is a very versatile platform due to its intrinsic characteristics, but it is difficult in handling.

As a final product, a software interface is presented, which implements the artificial neural networks algorithms analyzed and presents the classification results in real time to the user graphically.

Agradecimientos

Agradezco a mi familia y principalmente a mi madre por ser el pilar de mi vida y brindarme siempre esos consejos sabios que solo una madre puede dar.

A Abigail por su compañía y cariño incondicional.

A mis hermanos compañeros de toda mi vida y ejemplos a seguir.

A mis amigos y compañeros de estudio.

A mis profesores, cuya guía es inconmensurable.

Al Instituto Politécnico Nacional y al Centro de Investigación en Computación, por los apoyos económicos para la realización de este trabajo de investigación. Este trabajo de investigación fue realizado en el marco de los proyectos con número de registro SIP 20131505, SIP 20131182, SIP 20144538, SIP 20140776 y CONACyT 155014.

Contenido

Índice de Tablas

Índice de Figuras

Índice de Anexos

Glosario de Siglas

ANN……Artificial Neural Network.

API……..Application Programming Interface.

ASCII……..American Standard Code for Information Interchange.

UAV……Unmanned Aerial Vehicles.

CMOS…….Complementary metal–oxide–semiconductor.

CODEC……...Codification – De codification.

CPD…….Conditional Probability Distribution.

CSV…….Comma-separated values.

CUDA…….Compute Unified Device Architecture.

DDR……Double Data Rate memory.

DoG…….Difference of Gaussians.

DSP…….Digital Signal Processing.

FIFO……First In First Out.

FPS……..Frames Per Second.

GPU…….Graphics Processing Units.

GUI……..Graphical User Interface.

IMU…….Inertial Measurement Unit.

JNI……...Java Native Interface.

KNN……K-Nearest Neighbors Algorithm.

MLP……Multilayer Perceptron Network.

OCR……....Optical Character Recognition.

PCMD…….Progressive Command.

PCA…….Principal Component Analysis.

RBFN……..Radial Basis Function Network.

RGB……Red Green Blue color model.

SDK……Standard Development Kit.

SIFT……Scale Invariant Feature Transform.

SURF……..Speed Up Robust Features.

SVM……Support Vector Machine.

SWING……...Java GUI Widget Toolkit.

UDP……User Datagram Protocol.

UML……...Unified Modeling Language.

WEKA……Waikato Environment for Knowledge Analysis.

Capítulo 1
Introducción

Hoy en día la inclusión de herramientas robóticas en la vida diaria ya es una realidad; con robots diseñados para realizar una gran variedad de tareas, robots pedagógicos los cuales se encuentran diseñados para que interactúen con niños en pleno desarrollo, robots construidos con la finalidad de ayudar en la rehabilitación de personas que han sufrido algún accidente incapacitante o que padecen de alguna discapacidad congénita. En general la aplicación de la robótica en el quehacer diario no tiene límites y en gran medida estos se encuentran diseñados para mejorar o ayudarnos a tener una mejor calidad de vida.

Y es que el área de la robótica comprende el diseño, construcción, operación y aplicación de los robots, así como también los sistemas computacionales para su control, lectura de sensores y procesamiento de la información. Estas tecnologías en conjunción con los componentes mecánicos pueden tomar parte en situaciones y lugares en donde le resulta difícil o casi imposible al humano hacer presencia. Así mismo, en ciertas circunstancias, las herramientas robóticas poseen un mejor desempeño que cualquier ser humano, como ejemplo tenemos las líneas de producción en donde los brazos robóticos tienen una gran aceptación debido a los resultados obtenidos en las últimas décadas [27, 28].

En los últimos años, han surgido los robots aéreos no tripulados o Unmanned Aerial Vehicles (UAV's), plataformas aéreas como aviones de ala fija, helicópteros y Quadrotores, los cuales poseen la característica de ser piloteados de forma remota, cuyas últimas investigaciones apuntan en la dirección de la navegación completamente autónoma [20]. Un ejemplo de estos vehículos se puede observar en la Figura 1.1.

Figura 1.1 Ejemplos de UAVs.

Dentro de los UAV's, los modelos de los Quadrotores han llamado la atención de la comunidad científica debido a su versatilidad y bajo costo. Los Quadrotores pueden ser definidos como helicópteros de varios motores, los cuales son propulsados por cuatro rotores y cuyas hélices se encuentran orientadas en forma vertical. Estos usan dos pares de hélices, de las cuales, un par gira en sentido de las manecillas del reloj y el par restante gira en el sentido contrario de las manecillas del reloj. Estos utilizan las variaciones en las revoluciones por minuto (RPM) para controlar la altura y torque del mismo. El control del movimiento del vehículo se logra al alterar el porcentaje de rotación o velocidad angular de los pares de hélices, lo anterior se explica a detalle en el capítulo de marco teórico [30].

Estos modelos de robots en particular son la plataforma ideal para poder censar información a distancia sin importar las características del terreno o que tan difícil sea su acceso, debido a las características de vuelo que poseen pueden censar información en lagos, montañas y ciudades a baja y gran altura y deforma estacionaria, característica que los aviones de ala fija no poseen.

Dado el preámbulo anterior procederemos a describir con mayor detalle los antecedentes que motivan el presente trabajo.

1.1 Antecedentes

El termino dron o UAV se remonta al año de 1936 cuando un grupo de investigación de la marina de los Estados Unidos de América utilizó este término para referirse a los ya desarrollados vehículos aéreos controlados por radio frecuencia. Estos primeros dispositivos eran aeronaves a escala cuyos primeros modelos fueron construidos en 1934 por la compañía Reginald Denny Industries en Hollywood, California. Desde un inicio estas aeronaves estaban orientadas a cuestiones bélicas, como lo era el entrenamiento de los soldados que manejaban la artillería anti aérea. El desarrollo de los diversos prototipos principalmente fue impulsado por la primera y segunda guerra mundial y posterior mente por la guerra fría, en cuyo periodo los modelos pasaron de ser simples modelos a escala de entrenamiento hasta convertirse en aeronaves capaces de cruzar el océano atlántico en un vuelo de 26 horas continuas [22].

Sin embargo haciendo a un lado sus aplicaciones militares, los primeros UAV's utilizados para cuestiones de investigación fueron las aeronaves utilizadas para el monitoreo atmosférico cuya alternativa era más barata que los globos meteorológicos utilizados en esa época (1950). También en esta época se desarrollaron los modelos conceptuales de aeronaves impulsadas por energía solar.

Dentro de los UAV's existen principalmente dos géneros los UAV's de ala fija capaces de realizar vuelos a largas distancias propulsados principalmente por turbinas y los UAV con al menos dos rotores como lo son los helicópteros, los tri-rotores como su nombre lo indica con tres rotores, los tetra rotores, hasta los UAV's con ocho rotores, estos últimos capaces de levantar varias decenas de kilogramos de carga.

En nuestro trabajo nos enfocamos en los tetra rotores o Quadrotores, los cuales han sido ampliamente aceptados por la comunidad científica debido a su bajo costo, tamaño reducido y gran versatilidad. Debido a lo anterior las principales áreas de investigación donde actualmente se utilizan los Quadrotores son:

- Tareas de búsqueda y rescate.
- Inspección de interiores de edificios.
- Inspección de techos de edificios.

- Inspección de puentes.
- Agricultura de precisión.
- Mensajería.
- Tareas cooperativas.

Como podemos imaginar las tareas anteriormente mencionadas están estrechamente relacionadas con el área de procesamiento y clasificación de la información y en especial con el área de visión por computadora, es en este punto donde las dos áreas se juntan para lograr un objetivo en común.

Dentro del área de procesamiento y clasificación de información existen técnicas como lo son las redes neuronales artificiales, cuyo objetivo es simular un comportamiento inteligente tratando de imitar la forma en la que trabajan las neuronas biológicas. Las redes neuronales artificiales o ANN por sus siglas en inglés (Artificial Neural Networks) son algoritmos de propósito general, las cuales son utilizadas en su mayoría para tareas de análisis de datos y reconocimiento de patrones [10].

La principal característica de las ANN es su habilidad de aprender. Dado un conjunto reducido de datos la red neuronal puede descubrir las relaciones intrínsecas de la información mediante un proceso de entrenamiento. También son capaces de aprender comportamientos complejos y son altamente adaptables, ya que han sido utilizadas desde procesos para la identificación de huellas digitales hasta para el reconocimiento de objetos en líneas de producción [54].

Las redes neuronales, en resumen, son un método de aprendizaje de máquina que fue evolucionado de la idea de cómo se podría simular el comportamiento de cerebro humano. Y esta evolución resultó en la capacidad de las redes neuronales de modelar problemas complejos y altamente no lineales, además de su alta tolerancia a fallos.

Como primeros modelos de redes neuronales se tienen los perceptrones de Rosenblatt, aunque este modelo no es propiamente una red neuronal, es el primer algoritmo que describe el funcionamiento de una red neuronal, cuyo modelo sigue vigente, el cual fué publicado en 1958. También cabe mencionar que dentro de los primeros modelos de redes neuronales están los propuestos por McCulloch y Pitts en 1943, los cuales introducen la idea de las redes neuronales como máquinas

de cómputo. Hebb, contribuyendo por otra parte, postula la primera regla de aprendizaje auto organizado [58].

Como se demuestra en la literatura, los modelos de redes neuronales de una sola capa de perceptrones poseen limitaciones en cuanto a la clasificación de patrones linealmente separables (problema del XOR). Debido a lo anterior se desarrolló lo que hoy conocemos como red neuronal de perceptrones multicapa (MLP por sus siglas en inglés, la cual se estudia a mayor detalle en el capítulo de marco teórico), modelo que demuestra que el uso de redes neuronales, el cual puede clasificar conjuntos de patrones complejos y linealmente separables.

Debido al auge que tuvieron las redes neuronales y al teorema de separabilidad de patrones, estos dieron paso a la creación de nuevas técnicas de clasificación, como lo son las técnicas de aproximación estocásticas, la implementación de aproximaciones hibridas de clasificación para la clasificación de patrones, como las redes de funciones de base radial (RBFN por sus siglas en inglés), cuya estructura de la red neuronal se compone de tres capas. Y como redes de última generación, las redes o máquinas de vector suporte (SVM), las cuales en su definición básica se describen como máquinas de aprendizaje binario con algunas propiedades elegantes, cuya finalidad es, dado un conjunto de entrenamiento, la SVM deberá de construir un híper plano el cual servirá como superficie de decisión y este híper plano maximizará la distancia entre las muestras representativas de dos clases [58].

Dada la anterior introducción procedemos a describir a detalle el planteamiento del problema que se pretende resolver en el presente trabajo de tesis. En el capítulo 3 se explican a detalle cada una de las técnicas de clasificación anteriormente mencionadas.

1.2 Planteamiento del problema

Como ya se mencionó, el reciente auge de los Quadrotor ofrece una gran variedad de alternativas de investigación, debido a la complejidad que plantea el manejo y adquisición de datos por medio de esta herramienta y por otro lado su versatilidad y bajo costo, nos surge la cuestión ¿Es posible utilizar un robot como lo es el Quadrotor como herramienta de adquisición de información? y aún más en específico, ¿Es posible realizar la detección de objetos mediante una cámara montada en el Quadrotor?, ¿A qué altura es posible realizar la detección de objetos?, ¿Qué características debe

de tener la cámara para poder procesar las imágenes?, ¿Cómo se realiza la clasificación y que técnicas se recomienda utilizar?

Es debido a las preguntas anteriores que se motiva el desarrollo de la presente tesis, ya que como se comentará en la sección del estado del arte, existen diversos estudios y simulaciones que utilizan estas herramientas, la mayoría con un procesamiento de video post vuelo, pero hasta el momento del desarrollo de esta tesis no se encontró un estudio comparativo formal que nos sirviera en un futuro como guía para el desarrollo de plataformas de investigación más robustas.

1.3 Justificación

Como parte del proyecto de investigación para la creación de un escuadrón de Quadrotores, cuya principal finalidad es la de realizar vigilancia aérea, éstos deberán de ser capaces de analizar las situaciones en piso mediante imágenes tomadas por una cámara montada en dichos dispositivos.

De esta forma en un lapso de 10 a 15 años, el objetivo es que un Quadrotor pueda identificar los incidentes como lo son incendios, accidentes automovilísticos, entre otros, estos incidentes podrán ser monitoreados y atendidos con mayor eficacia. Proporcionando una mejora en la calidad de vida en lo general de la ciudadanía.

Sin embargo, el estado actual del arte aún se encuentra lejos de lograr lo anterior, por lo que una primera aproximación para resolver la problemática es analizar la factibilidad en el uso de estas herramientas, así como realizar un estudio formal sobre las técnicas las cuales nos ayuden a analizar las situaciones en piso e identificación de objetos en tiempo real.

1.4 Objetivo general

Se trata de que un Quadrotor sea un supervisor en el aire; el cual seguirá una ruta pre establecida de vuelo y dentro de la ruta de vuelo existirán diferentes escenarios los cuales se deberán de analizar. Cada escenario contendrá diferentes objetos, los cuales deberán ser identificados y analizados en tiempo real.

1.5 Objetivos específicos

- Diseñar e implementar una interface que permita controlar al Ar. Drone Parrot.
- Implementar una interface que permita realizar el análisis de las imágenes tomadas por el Quadrotor en tiempo real.
- Realizar un estudio comparativo entre diferentes redes neuronales y algunos clasificadores de tipo estadístico.
- Del análisis anterior, implementar el clasificador que mejores resultados arroje.
- Los algoritmos desarrollados deberán de ser probados en una ruta de vuelo y los resultados de la identificación en tiempo real de los objetos por escenario deberán de ser reportados al usuario mediante una GUI.

1.6 Consideraciones iniciales

Las consideraciones iniciales que se tomaron en cuenta para el desarrollo de la presente tesis son:

- El área que comprende cada uno de los escenarios es de 120 cm. de ancho por 90 cm. de largo.
- El color de fondo de los escenarios propuestos deberá de ser contrastado, de color negro, lo cual no es una limitación, pero el contraste nos ayuda sopesar las limitaciones de la cámara de video.
- Los objetos a identificar poseen varias formas y colores, estos se especifican en el capítulo 4.
- El área de vuelo que se contempla para las maniobras de Quadrotor es de 3 metros cuadrados.
- La disposición de los escenarios contemplados para su análisis es de un cuadrado, sin embargo no es una restricción, ya que el Quadrotor puede ser manipulado en vuelo libre.

1.7 Organización de la tesis

La presente tesis se encuentra organizada como se describe a continuación:

Capítulo 2: Se presentan el estado del arte, el cual contiene los trabajos actuales más representativos del área a la cual pertenece la presente tesis.

Capítulo 3: En este capítulo se presenta el marco teórico, es decir, las técnicas utilizadas para desarrollar el presente trabajo y, en caso de ser necesario, la teoría de fondo para poder entender las técnicas y algoritmos utilizados.

Capítulo 4: Se explica a detalle cómo se utilizan las técnicas y algoritmos presentados en el capítulo 3 para lograr los objetivos de esta tesis.

Capítulo 5: Se presentan los resultados obtenidos en la etapa de análisis y experimentación.

Capítulo 6: Se discuten los resultados, se enuncian las conclusiones y se recomiendan trabajos futuros.

Apéndice A: En esta sección se discute y se muestra que las técnicas utilizadas también pueden ser utilizadas para la clasificación de objetos en fondos complejos.

Apéndice B: En esta sección se describe un pequeño estudio realizado para la reducción de las dimensiones de los descriptores de los puntos de interés.

Apéndice C: Se enuncian y explican los métodos clave de la presente tesis.

Capítulo 2
Estado del arte

En las siguientes secciones se exponen algunos trabajos de investigación en las áreas de redes neuronales, así como en el uso de UAV como herramientas para el desarrollo de plataformas de investigación.

2.1 Uso de UAVs como plataformas de investigación

En las últimas décadas, las potencias mundiales se han dado a la tarea de crear vehículos autónomos, robots de todo tipo, androides, humanoides, robots móviles, marítimos y aéreos. En nuestro estudio nos enfocaremos principalmente en los vehículos aéreos no tripulados. Estos son utilizados en una gran variedad de aplicaciones, los cuales poseen características de vuelo autónomo de varias decenas de kilómetros como lo son los UAV de ala fija, pero que carecen de agilidad en su maniobrabilidad, hasta los Quadrotor, los cuales han tenido un gran auge en los últimos años, esto debido a su versatilidad y bajo costo, estas herramientas que pueden ser manipuladas tanto en interiores como en exteriores y son ampliamente utilizados para el procesamiento de imágenes en tiempo real, rastreo de vehículos, y transporte de paquetería de bajo peso, como lo intentan implementar las grandes compañías Amazon y DHL. La mayoría de estos vehículos son tele operados desde una estación de control terrestre.

Las investigaciones en este ramo, se centran en poder implementar una navegación completamente autónoma a los UAVs, como lo es la navegación basada en visión artificial, reconocimiento de

objetos, como lo son carreteras, vehículos, o marcas geoespaciales, así mismo para que realicen tareas coordinadas entre varios Quadrotor, dado este preámbulo, mencionaremos algunos de los estudios más recientes en esta área.

2.1.1 Método mejorado de rastreo de objetos en videos tomados por un UAV

El objetivo de este trabajo, fue el implementar una modificación al algoritmo de reconocimiento de objetos Mean Shift, ya que este último es capaz de detectar objetos por su forma, y realizar un seguimiento de los mismos en una secuencia sucesiva de imágenes. Las pruebas se realizaron con el video tomado desde un Quadrotor, en cuya toma aparece una autopista en donde el algoritmo detecta y sigue de forma correcta y eficiente los automóviles que en esta aparecen, como lo muestran sus resultados [51]:

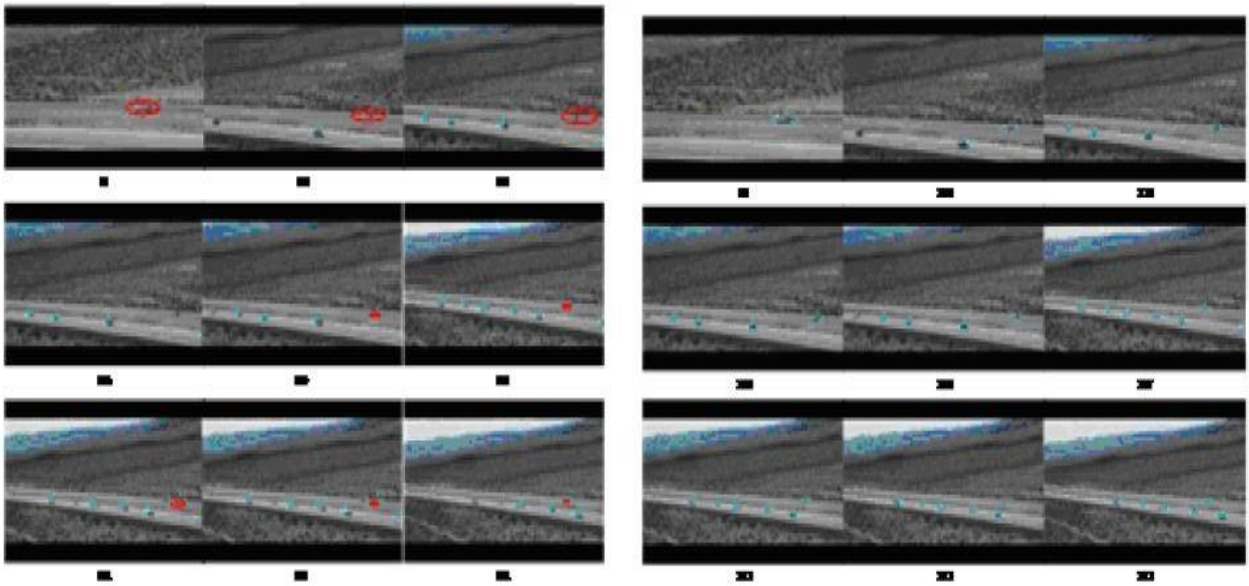

Figura 2.1.1.1 Ejemplo de rastreo de objetos mediante Mean Shift.

2.1.2 Evasión de obstáculos en vuelo simulado con base en visión artificial

Este trabajo presenta una conjunción entre detección automática de obstáculos y detección de velocidad mediante flujo óptico, junto con el entorno virtual de Google Earth, se simula la navegación autónoma de un Quadrotor, aplicando los algoritmos antes mencionados a las imágenes proporcionadas por el entorno virtual. Los resultados muestran como el umbralado de

la varianza del gradiente de la diferencia del flujo óptico posee un efecto crítico en la detección de caminos con diferentes anchos [6].

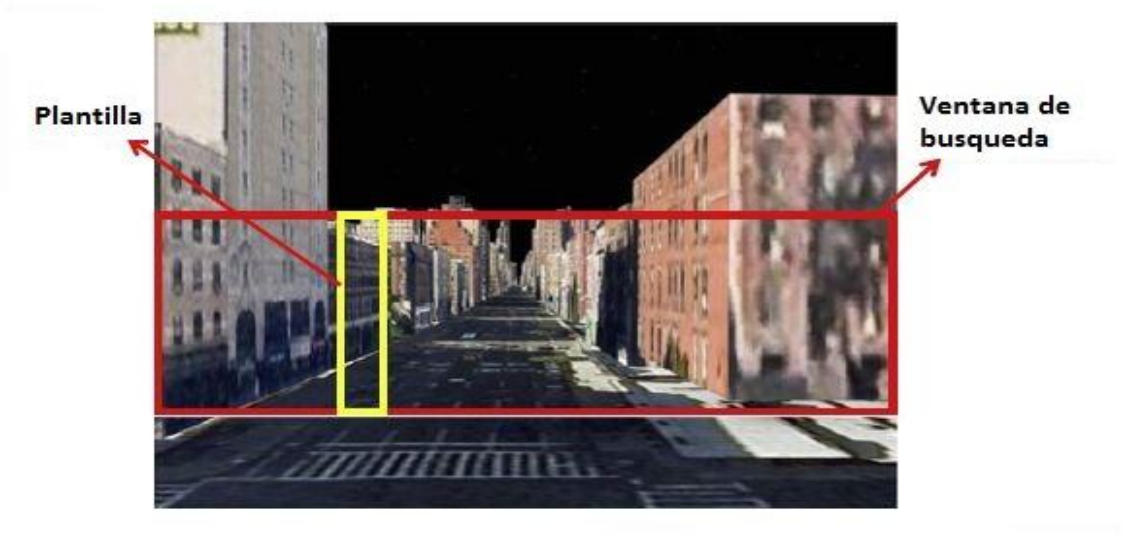

Figura 2.1.2.1 Ejemplo de navegación virtual por visión por computadora.

Utilizando la herramienta de navegación del entorno virtual de Google, se establecen rutas de traslado de varias cuadras cada una, como se muestra en la siguiente imagen:

Figura 2.1.2.2 Ejemplo de navegación virtual autónoma.

Los resultados de este trabajo son satisfactorios, pero como en toda simulación, existen aspectos de un ambiente real que no son tomados en cuenta.

2.1.3 Identificación de imágenes objetivo basado en SIFT

Este trabajo presenta un método para el procesamiento en tiempo real de una gran cantidad de información proveniente de videos tomados por un UAV, el cual puede incrementar la eficiencia de identificación de objetivos. Este trabajo basa su operación en el algoritmo SIFT (Scale Invariant Feature Transform), el cual puede superar algunos efectos de la deformación de las imágenes, pero su desempeño no alcanza los objetivos en tiempo de procesamiento. Con las modificaciones propuestas por este trabajo se demuestra que este algoritmo puede alcanzar dichos requerimientos.

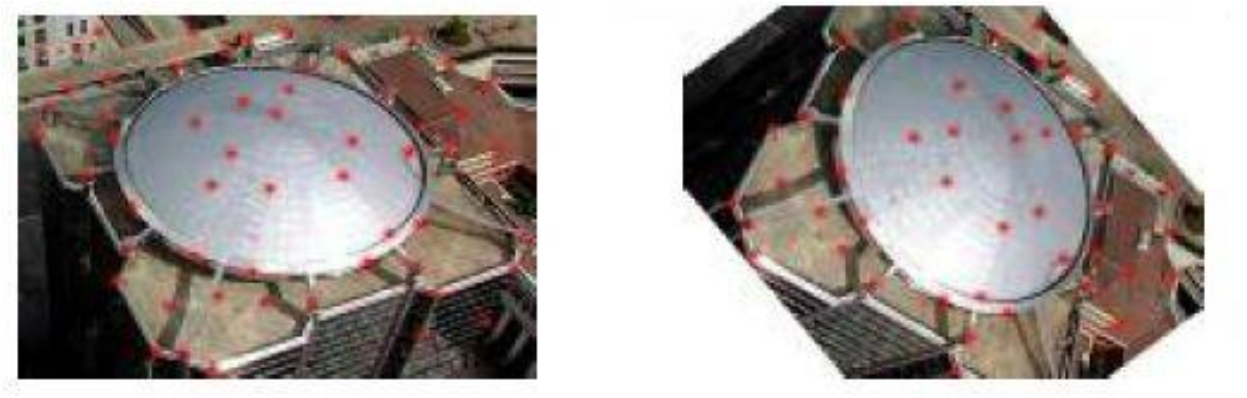

Figura 2.1.3.1 Ejemplo de identificación de puntos descriptores SIFT.

Las ventajas de este trabajo es que se realizó con imagen reales, tomadas de un Quadrotor, y nos muestran una tabla comparativa donde se observa los porcentajes de mejora de identificación exitosa [64]:

	Number of correct matched point		Number of wrong matched point		Accuracy rate		Time of matching	
	Traditional algorithm	Improved algorithm	Traditional algorithm	Improved algorithm	Traditional algorithm	Improved algorithm	Traditional algorithm	Improved algorithm
The first group	52	56	5	2	91.2%	96.5%	4.23	3.16
The second group	347	389	19	9	92.3%	97.7%	24.35	21.53

Tabla 2.1.3.1 Tabla comparativa de puntos descriptores SIFT.

2.1.4 Implementación de rastreo y aterrizaje visual para un Quadrotor

Este trabajo desarrollado en la universidad de Beihang en Beijin, China, implementa dos algoritmos de control sencillos basados en PID, uno para el rastreo y otro para el aterrizaje de un Quadrotor. En el aspecto de visión artificial, se utiliza identificación por color, cálculo de la distancia euclidiana para realizar el seguimiento de un objeto, y por forma con el cálculo de centro de masa, para identificar la zona de aterrizaje. Para llevar a cabo sus experimentos utilizaron el modelo Ar. Drone en un ambiente controlado [69].

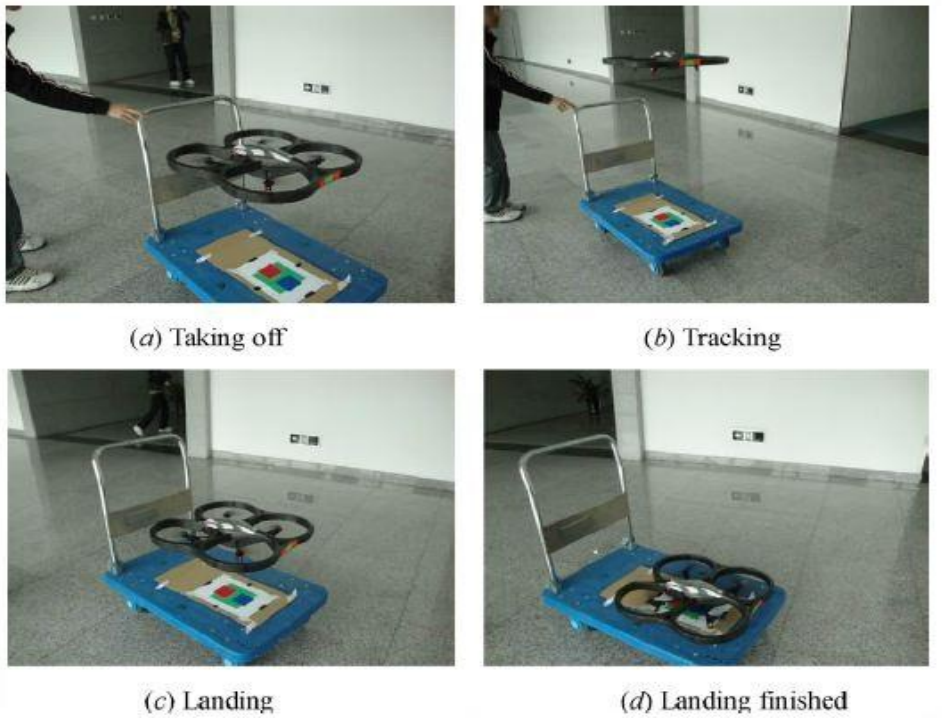

Figura 2.1.4.1 Ejemplo de rastreo y aterrizaje visual para un Quadrotor.

Con este trabajo se demuestra que en condiciones controladas, se puede controlar de forma muy confiable y acertada un Quadrotor para realizar diversas acciones.

2.1.5 Investigación en deslizamientos de tierra con procesamiento de imágenes

Este estudio realizado en Francia y Suiza, nos muestra cómo se puede analizar los deslizamientos de tierra desde un conjunto de imágenes en mosaico tomadas por un UAV, a las cuales primeramente se les somete a un proceso Fotogramétrico y se compara con una base de datos de imágenes anterior, así mediante su análisis con técnicas de procesamiento de imágenes se determinan las áreas afectadas por deslizamientos de tierra, como se muestra en la siguiente imagen donde los deslizamientos de tierra primarios se observan con una flecha alargada roja y los deslizamientos secundarios se observan con flechas amarillas y naranjas más pequeñas [61].

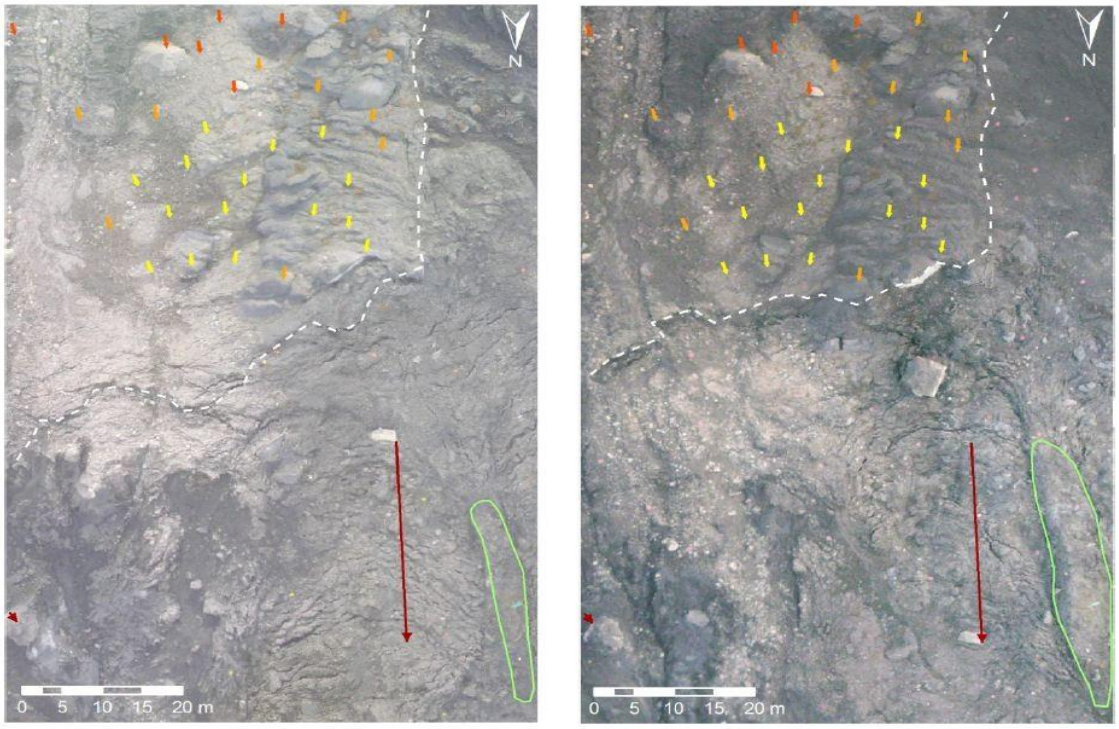

Figura 2.1.5.1 Ejemplo de análisis de deslizamientos de tierra en imágenes tomadas por un Quadrotor.

Capítulo 3
Marco teórico

En este capítulo se explican las principales técnicas de procesamiento de imágenes utilizadas en el presente trabajo y en específico se extiende el tema detección de puntos de interés con el algoritmo Speed Up Robust Features, seguido de una introducción a las técnicas de clasificación como lo son las redes neuronales y los principales clasificadores estadísticos. Y terminamos con una explicación amplia del funcionamiento y características de la plataforma que utilizamos, la cual es el Quadrotor Ar. Drone Parrot 2.0.

3.1 Visión Artificial

Como humanos, somos capaces de percibir la estructura de los objetos de nuestro entorno en tres dimensiones con aparente sencillez. Pensemos, por ejemplo, que tan vivida es una flor al observarla, como percibimos las tres dimensiones de este objeto. Con solo observarla podemos definir la forma, la translucidez de cada pétalo, los patrones sutiles de brillo y sombra sobre la superficie y sin mayor esfuerzo podemos separar la flor del fondo. Ahora pensemos en un retrato de un grupo de personas, observando el retrato, podremos contar (e identificar) con facilidad el número de personas retratadas, determinar su estado de ánimo observado la apariencia de su rostro. Los psicólogos perceptuales han invertido décadas tratando de entender cómo trabaja el sistema visual humano, en paralelo, los investigadores en visión por computadora han desarrollado técnicas matemáticas que nos permitan recuperar la forma tridimensional y la apariencia de los

objetos en una imagen. En la actualidad, ya existen técnicas para el cómputo certero de modelos parciales 3D. Así, dado un conjunto lo suficientemente largo de "vistas" de un objeto en particular o "feçade", se puede recrear una superficie 3D acertada utilizando modelos de comparación estereoscópica. Podemos seguir a una persona en movimiento en un ambiente complejo, podemos también, con un cierto grado de éxito, identificar y nombrar todas las personas en una fotografía utilizando una combinación de detección de rostros, vestimenta y cabello. Sin embargo, a pesar de todos estos avances, el sueño de tener una computadora capaz de interpretar una imagen al mismo nivel de un niño de 2 años, sigue siendo una ilusión [56].

¿Por qué la visión por computadora es tan difícil?, en parte, es porque el problema de la visión por computadora es un "problema inverso", en el cual, buscamos el recuperar aspectos no conocidos dada información insuficiente o incompleta. De tal forma que debemos reorganizar los modelos probabilísticos y físicos de tal forma que se desambigüen las soluciones potenciales. Sin embargo, el modelado del mundo en forma visual en toda su complejidad es mucho más difícil de lo que hemos platicado hasta este momento. En visión por computadora se trata de describir el mundo que nos rodea en una o más imágenes y reconstruir sus propiedades, como lo es la forma, la iluminación y las distribuciones de color. Es sorprendente que los humanos y animales realicen estas actividades sin el mayor esfuerzo, mientras los algoritmos de visión son muy propensos a errores [56]. Hoy en día la visión por computadora es utilizada para una gran variedad de tareas y aplicaciones las cuales incluyen:

- Reconocimiento Óptico de caracteres *(OCR)*: el cual se utiliza para la lectura e identificación de caracteres escritos a mano.
- Inspección Automatizada: se utiliza para la rápida inspección de objetos para el aseguramiento de la calidad de los mismos, en el cual se utilizan visión estereoscópica con iluminación especializada.
- Reconocimiento detallado: reconocimiento de objetos para la automatización de líneas de producción.
- Reconstrucción de modelos *3D (Fotogrametría)*: construcción completamente automatizada de modelos 3D.

- Aplicaciones Médicas: registro de imagines pre e intra operación, o para realizar estudios a largo plazo como la morfología cerebral.
- Seguridad en movimiento autónomo: detección de obstáculos inesperados, como lo son piedras, en condiciones donde las técnicas de visión activa como lo son el radar no funcionan o son poco confiables.
- Reconocimiento de huellas digitales: para la autentificación automática y aplicaciones forenses.
- Vigilancia: para el monitoreo de intrusos y análisis de tráfico en autopistas.

De esta forma, se describe un sistema de visión por computadora como un conjunto de actividades que nos permite realizar la identificación y clasificación de los objetos pertenecientes a una imagen, estas actividades principales son:

- Mejoramiento de la Imagen.
- Segmentación y Etiquetado.
- Representación y Descripción.
- Reconocimiento de Formas.

Según González y Woods [55], el diagrama completo de un sistema de visión por computadora se muestra en la figura 3.1.1:

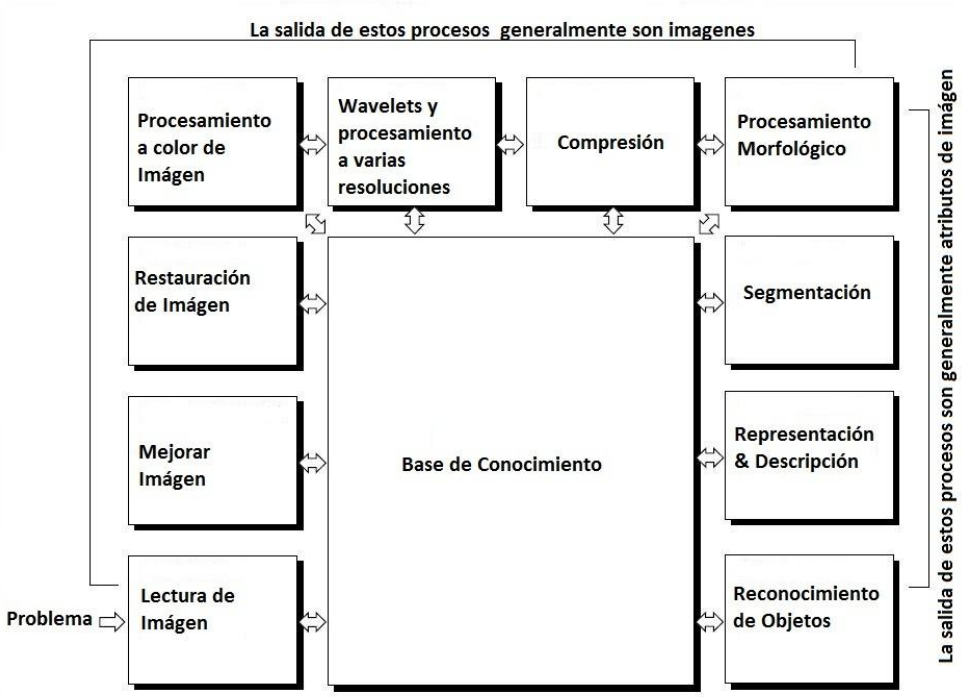

Figura 3.1.1 Sistema Completo de Visión por Computadora Según González y Woods [55].

3.1.1 Mejoramiento de la Imagen

El mejoramiento de imágenes es una de las áreas más atractivas y simples del procesamiento digital de imágenes. Básicamente la idea detrás de estas técnicas es la de mejorar los detalles oscuros, o el de simplemente resaltar ciertas características de interés en una imagen. Un ejemplo familiar de mejoramiento es cuando incrementamos el contraste de una imagen debido a que "esta luce mejor", es importante resaltar que el mejoramiento es un área muy subjetiva del procesamiento de imágenes.

3.1.2 Segmentación y Etiquetado

La segmentación es el proceso de dividir una imagen en múltiples segmentos (conjuntos de pixeles). El objetivo de la segmentación es la de simplificar o cambiar la representación de la imagen en algo con mayor significado y que al mismo tiempo sea más sencillo de analizar. Típicamente la segmentación se utiliza para la identificación de objetos y sus características (líneas, curvas, etc.) en imágenes. De forma más detallada, la segmentación es el proceso de

asignar un identificador a un conjunto de pixeles que comparten las mismas características en una imagen.

3.1.3 Representación y Descripción

La representación y descripción casi siempre es la etapa siguiente a la de segmentación y etiquetado, cuya salida es usualmente un conjunto de pixeles los cuales constituyen a un objeto en la imagen o una región de interés. La primera decisión que se debe tomar es si el conjunto de datos (pixeles) representa los límites de una región o la región completa. La representación de límites es apropiada cuando nos enfocamos en las características externas de un objeto, como lo son las esquinas o puntos de inflexión. La representación regional es apropiada cuando nos enfocamos en las propiedades internas de los objetos, como lo es la textura o la forma del esqueleto del objeto. De esta forma se debe de especificar un método para la descripción de los datos los cuales nos representan los puntos de interés, estos métodos son llamados "selección de características", los cuales se encargan de la extracción de atributos los cuales nos proporcionan información cuantitativa de los objetos de interés.

3.1.4 Reconocimiento de Formas

El reconocimiento de formas es el proceso que le asigna un nombre "con significado" a un objeto, por ejemplo "automóvil", el cual se basa en su descripción y características. El proceso de reconocimiento de objetos requiere de un conocimiento previo de lo que es un vehículo, una persona, etc. A este conocimiento previo le llamamos "Base de Conocimiento". Esta base de conocimiento puede ser tan simple como el detalle de las regiones de una imagen donde la información de interés se sabe que estará localizada.

3.1.5 Detección de Color

Una computadora solo puede trabajar con números, de tal forma que una imagen es una gran colección de números para una computadora, cada combinación de números nos representa un color en específico, dependiendo del espacio de color esta combinación nos representará un color u otro, de tal forma que la detección de color es el proceso de analizar todo es espacio de muestras que conforman la imagen y dado un umbral identificar el valor de dicho pixel como elemento o perteneciente al conjunto de valores del color buscado[28].

3.2 Scale Invariant Feature Transform (SIFT)

Este mecanismo de extracción de puntos de interés, transforma a una imagen en una colección larga de vectores de rasgos locales, cada uno de estos vectores es invariante a transformaciones de traslación, escalamiento, rotación y en un grado mínimo a luminosidad.

Los puntos de interés son identificados utilizando una etapa de filtrado. La primera etapa identifica las coordenadas de los puntos clave en un espacio de escala, buscando las regiones donde la Diferencia de Gaussianas (DoG) posee un valor máximo o mínimo. Cada punto identificado es utilizado para generar un vector de rasgos que describe a la imagen en una región local relativa al espacio de escala muestreado. Estos vectores resultantes se denominan "SIFT Keys" [15].

3.2.1 Localización de puntos de interés (SIFT Keys)

Para lograr que los rasgos identificados sean invariantes ante rotaciones se eligió la extracción de los mismos mediante la selección de máximos y mínimos de una función de Diferencia de Gaussianas aplicada a un espacio de escala, de esta forma se localizan los puntos de interés (Keys) en regiones y escalas con una gran variación de las mismas. Como una función Gaussiana 2D es separable, su convolución con una imagen de entrada se puede aplicar en dos etapas de funciones Gaussianas de 1D en la horizontal y en la vertical.

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} \qquad (3.2.1)$$

En la detección de rasgos, todas las operaciones de suavizado se realizan tomando un valor de $\sigma = \sqrt{2}$ [15].

Figura 3.2.1.1 Ejemplo de localización de puntos de interés (en color rojo) SIFT.

3.2.2 Estabilidad de los puntos de interés SIFT

Para caracterizar la imagen en cada coordenada de un punto de interés, la imagen suavizada A en cada nivel de la pirámide es procesada para extraer los gradientes y su orientación. Así en cada pixel A_{ij}, la magnitud del gradiente de la imagen M_{ij}, y su orientación, R_{ij} se calculan mediante diferencias a nivel de pixel [15]:

$$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i,j+1})^2} \tag{3.2.2.1}$$

$$R_{ij} = a\tan 2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij}) \tag{3.2.2.2}$$

3.3 Speed Up Robust Features (SURF)

En 2006, tres personas, H. Bay, T. Tuytelaars, y L. Van Gool, publicaron un artículo llamado "SURF: Speeded Up Robust Features", el cual introduce un nuevo algoritmo llamado SURF, como el nombre lo sugiere, es una versión más veloz del algoritmo SIFT. El cual es un algoritmo de detección y descripción invariante a transformaciones de escala, rotación y luminosidad [48].

Esto se logra apoyándose en imágenes integrales para el proceso de convolución de imágenes, basándose en la robustez de los vectores principales de la Matriz de Hessian para los vectores detectores y descriptores, simplificando estos métodos al máximo.

Una de las características importantes a mencionar de SURF es que ni el detector ni los descriptores usan la información del color. La imagen de entrada es analizada a varias escalas para garantizar la invariancia a cambios de escala.

3.3.1 Detección de puntos de interés

La detección de puntos de interés utiliza una aproximación muy básica de la matriz de Hessian, la cual tiende al uso de imágenes integrales y así se reduce el tiempo de cómputo drásticamente.

3.3.2 Imágenes Integrales

Este tipo de cálculo es utilizado para un cómputo rápido de filtros de convolución de tipo caja. De tal forma que la entrada de una imagen integral $I_{\sum}(x)$ en el punto $x=(x,y)$, representa la suma de todos los pixeles de la imagen de entrada I en una región rectangular formada por el punto de origen y el punto x.

$$I_{\sum}(x)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i,j) \tag{3.3.2}$$

Una vez que la imagen integral ha sido calculada, solo toma tres sumas el calcular la suma de intensidades de cualquier punto vertical [48].

Imagen Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

5 + 4 +2 +2 +1 3 = 17

Imagen Integral

5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

34 – 14 -8 + 5 = 17

Figura 3.3.2.1 Ejemplo del cálculo de una imagen integral.

3.3.3 Matriz de Hessian en base a los puntos de interés

El detector que se utiliza para la detección de puntos se basa en la Matriz de Hessian, esto debido a su buen comportamiento en precisión y repetitividad. De forma precisa con éste se detectan estructuras de tipo "manchón" en regiones donde el determinante es máximo. SURF también se apoya en la Matriz de Hessian para la selección de la escala.

Así, dado un punto $x=(x,y)$ en una imagen I, la Matriz de Hessian $H(x,\sigma)$ en un punto x a una escala σ se define como:

$$H(x,\sigma)=\begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{3.3.3}$$

Donde $L_{xx}(x,\sigma)$ es la convolución discreta de la derivada parcial de segundo orden de la Gaussiana $\frac{\partial^2}{\partial x^2}g(\sigma)$ de la imagen I en el punto x, análogamente para $L_{xy}(x,\sigma)$ y $L_{yy}(x,\sigma)$ [29]. Debido a que los filtros reales son no ideales, se aproxima la Matriz de Hessian con filtros de caja como se observa en la figura 3.3.3.1.

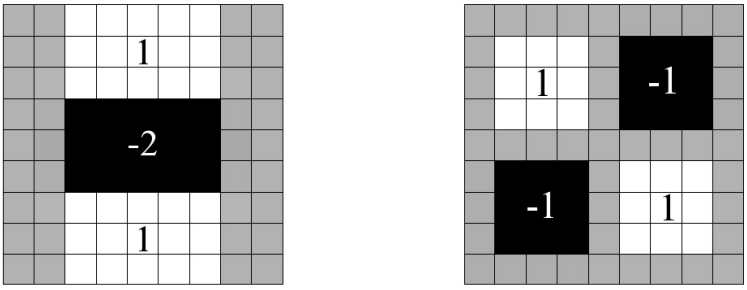

Figura 3.3.3.1 Aproximación de la derivada parcial de segundo orden de la Gaussiana para ventanas de dos dimensiones.

3.3.4 Representación en espacio de escala

Los puntos de interés que se encuentren en la imagen deben de ser consistentes a diferentes escalas. Los espacios de escala son usualmente implementados como una pirámide de imágenes. Estas

imágenes son suavizadas repetidamente con Gaussianas y sub muestreadas para obtener las imágenes en el nivel más alto de la pirámide. Lo anterior se ejemplifica en la imagen 3.3.4.1.

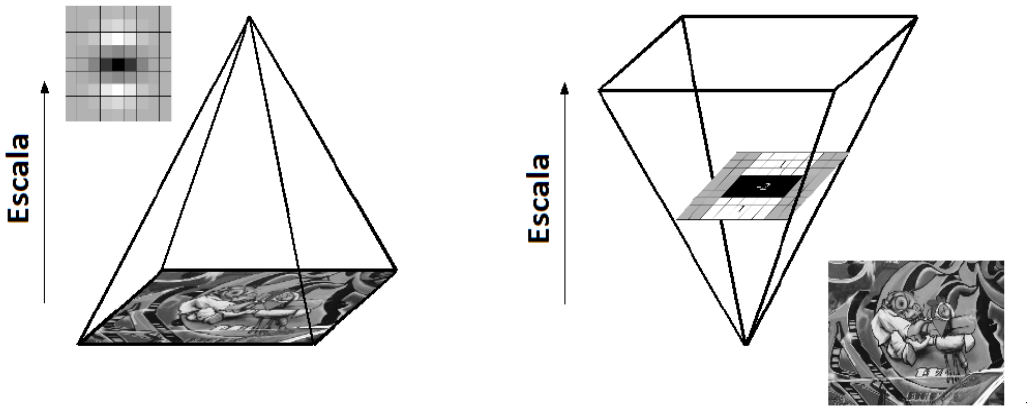

Figura 3.3.4.1 Representación de Espacio de Escala de una imagen.

Lowe en su trabajo [15] resta las capas de la pirámide de espacio a escala para obtener las imágenes de Diferencia de Gaussianas (DoG) en las cuales se pueden encontrar los bordes y los manchones ("blobs").

3.3.5 Localización de los puntos de interés

Para poder localizar los puntos de interés en la imagen y en las imágenes a escala, se aplica una supresión no-máxima en una vecindad de 3x3x3. La máxima del determinante de la matriz de Hessian es entonces interpolada en escala y en espacio de imagen.

3.3.6 Descripción y apareo de los puntos de interés

El descriptor del punto de interés describe la distribución de la intensidad en la vecindad de dicho punto de interés, similar a la información del gradiente extraído por SIFT. Para lograr esto SURF se basa en la distribución de la respuesta del Wavelet de Haar de primer orden en dirección de x y y, explota la velocidad de las imágenes integrales y usa una dimensión de solo 64 elementos. Para la extracción del primer descriptor, el primer paso es construir una región cuadrada centrada en el punto de interés. El tamaño de esta ventana es de 20s (donde s es la unidad de escalamiento). Un ejemplo de la construcción de las ventanas se muestra a continuación en la figura 3.3.6.1:

Figura 3.3.6.1 Ejemplo de las ventanas de descriptores a diferentes escalas.

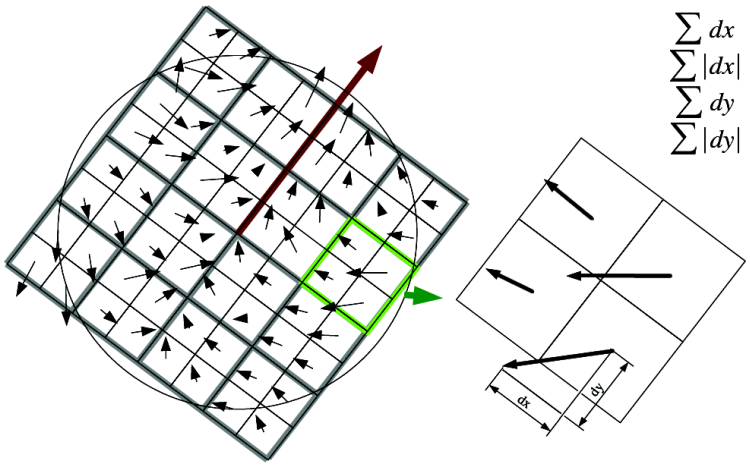

Figura 3.3.6.2 Ejemplo de la sumatoria de las repuestas de los Wavelet de Haar.

Cada región es dividida en subregiones más pequeñas de 4 x 4 (figura 3.3.6.2). Para cada subregión se calcula la respuesta del Wavelet de Haar. Por razones de simplicidad llamamos a la respuesta en la dirección horizontal del Wavelet de Haar como d_x, y a la respuesta en la dirección vertical del Wavelet de Haar como d_y. Entonces las respuestas del Wavelet d_x y d_y se suman sobre cada una de las subregiones y estas forman el primer conjunto de entradas en el vector descriptor. Para brindar información acerca de la polaridad y cambios de intensidad se extrae también la suma de

los valores absolutos de las respuestas del Wavelet, $|d_x|$ y $|d_y|$. Así cada subregión posee un vector descriptor de cuatro dimisiones $\mathbf{v} = (\sum d_x, \sum d_y, |\sum d_x|, |\sum d_y|)$. Concatenando esto por todas las sub regiones de 4 x 4, resulta en el vector descriptor de longitud 1x64 [48].

SURF es hasta cierto punto similar al concepto de SIFT en el sentido de que ambos se enfocan en la distribución espacial de la información del gradiente.

3.4 Redes neuronales artificiales

Las redes neuronales artificiales se componen principalmente de unidades simples de procesamiento, las cuales son llamadas "neuronas", desde el punto de vista biológico una neurona es un interruptor entre varias señales de entrada y una de salida. Este interruptor se activa si las entradas a la neurona alcanzan un determinado umbral de activación, entonces un pulso de salida será emitido por la neurona como resultado de su activación, los componentes de una neurona biológica se muestran a continuación [43]:

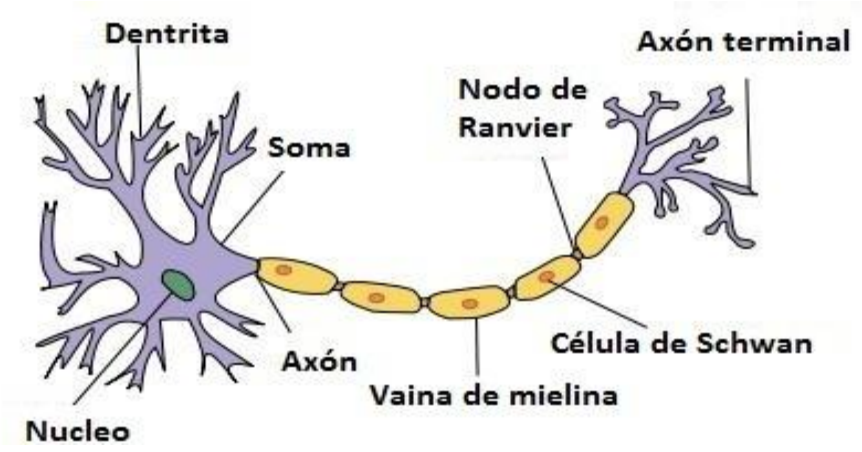

Figura 3.4.1 Elementos principales de una neuronal.

Como punto de entrada de las neuronas, se encuentran las dendritas, las cuales reciben la información mediante conexiones especiales llamadas sinapsis. Posteriormente la señal se

transmite a través del axón, desde el cual la señal es transformada mediante un proceso químico, para después ser transmitida a las neuronas conectadas a esta.

Mediante un proceso de simplificación, el modelo biológico de las neuronas puede ser trasladado a un proceso técnico, donde las consideraciones más importantes se mencionan a continuación:

Entrada Vectorial: La entrada de una neurona artificial consiste de varios elementos, de tal forma que esta es definida como un vector.

$$\vec{\mathbf{x}}, x \subseteq \vec{\mathbf{x}}, x \in \Re \tag{3.4.1}$$

La Sinapsis cambia las señales de entrada: En las redes neuronales artificiales, las entradas son multiplicadas por un número (“peso sináptico”), este conjunto de pesos representa la información almacenada de una red neuronal artificial.

$$\sum_i w_i x_i \tag{3.4.2}$$

Salida Escalar: La salida de una neurona artificial es un escalar, lo cual significa que una neurona está compuesta de un solo componente, el cual suma las señales de entrada a la neurona y produce solo una señal de salida.

$$y = f\left(\sum_i w_i x_i \right) \tag{3.4.3}$$

Características no lineales: Las entradas de una neurona artificial no son proporcionales a la salida de la misma.

Pesos ajustables: los pesos que ponderan las entradas de la neurona son variables. Esto añade una gran dinamicidad a las redes neuronales artificiales, debido a la gran cantidad de “conocimiento” que se encuentra almacenado en sus pesos [17].

De forma gráfica podemos ver a una red neuronal artificial como se muestra en la siguiente figura:

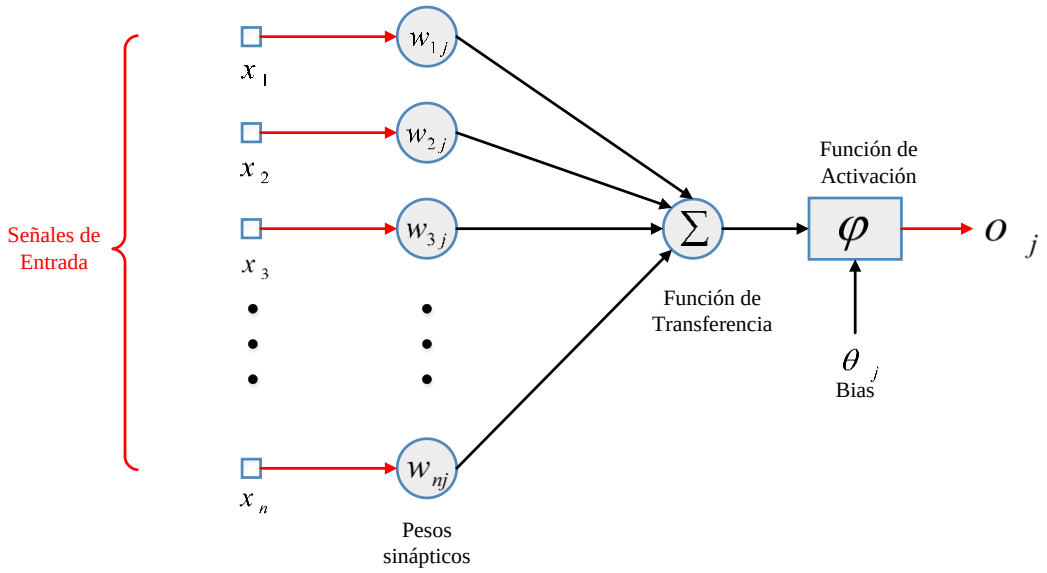

Figura 3.4.2 Esquema general de una Red Neuronal Artificial.

Uno de los aspectos más interesantes de las redes neuronales es su capacidad de familiarizarse con los problemas como resultado de un entrenamiento y una vez finalizado el entrenamiento, la red neuronal será capaz de resolver nuevos problemas pertenecientes a la misma clase de entrenamiento. Este mecanismo es conocido como generalización.

Hoy en día existe una gran cantidad de variaciones de redes neuronales, desde el primer modelo de redes propuesto por McCulloch and Pitts en 1943, y las siguientes variaciones, las cuales pueden ser divididas en:

Redes Neuronales de Primera Generación, también conocidas como perceptrones [43], las cuales se componen de dos secciones, una sumatoria y un nivel de umbral. La parte de la sumatoria recibe la información de entrada ponderada por los pesos sinápticos y realiza la función de umbral sobre el resultado de la suma. Los datos de entrada y salida pueden ser 0 y 1's.

Redes Neuronales de Segunda Generación se componen de neuronas que realizan el cómputo en dos etapas, la suma de los valores recibidos a través de la sinapsis, y una función Sigmoide, cuya

entrada es la suma de los parámetros de entrada ponderados por los pesos sinápticos, esta función se denomina como función de transferencia [57]. Un ejemplo común de este tipo de redes neuronales artificiales es la Red Neuronal Artificial de Base Radial, de la cual se dará una mayor descripción en las siguientes secciones.

Redes Neuronales de Tercera Generación o redes neuronales pulsantes, como el modelo de Eugene M. Izhikevich [26] o Alnajjar, F. con su modelo de redes neuronales pulsantes para la generación de comportamiento adaptativo en robots autónomos [2], las cuales quedan fuera del alcance del presente trabajo.

3.5 Red neuronal de perceptrones multicapa (MPL)

Como una mejora de las redes neuronales de un solo perceptrón, surge lo que ahora es conocido como Red Neuronal de Perceptrones Multicapa o MLP por sus siglas en inglés (figura 3.5.1). Estas redes neuronales se caracterizan por:

- El modelo de cada una de las neuronas en la red neuronal incluye una función de activación no linear la cual es diferenciable.
- Estas redes contienen uno o más capas ocultas, intermedias a las capas de entrada y salida.
- La red neuronal posee un alto grado de conectividad.

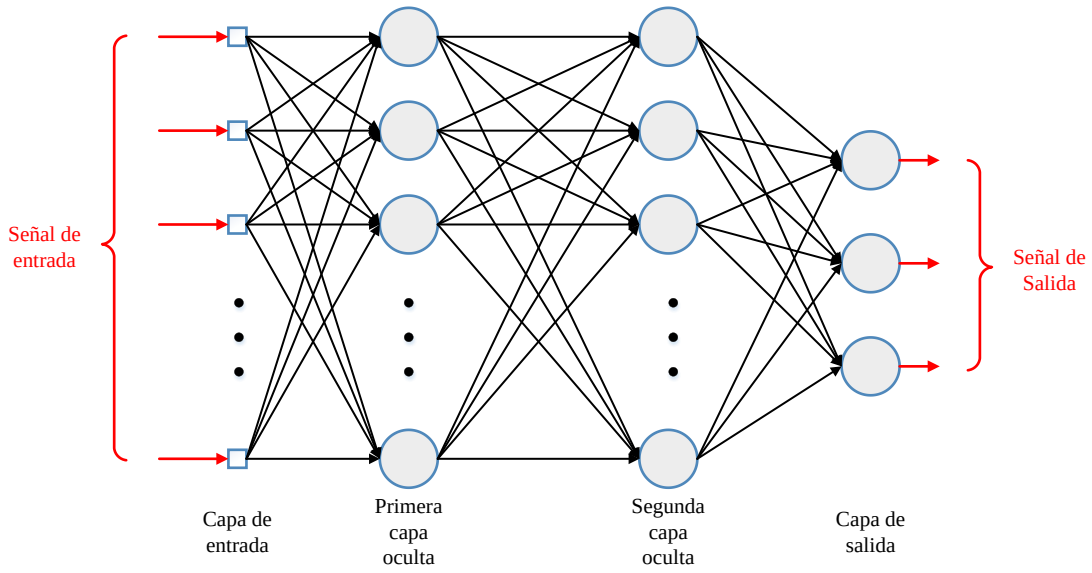

Figura 3.5.1 Esquema general de una Red Neuronal de Perceptrones Multicapa.

Las características anteriores, son las responsables del conocimiento y comportamiento que muestre la red. De forma resumida, el proceso de aprendizaje debe decidir cuales características del patrón de entrada deben de ser representadas por las neuronas de la capa intermedia. Por lo que el proceso de aprendizaje se hace más difícil, ya que la búsqueda se realiza en un espacio más amplio de posibles funciones [58].

Este tipo de redes neuronales poseen un tipo de entrenamiento llamado algoritmo Back-Propagation, el cual se realiza en dos fases:

- *La etapa Forward*, en donde los pesos sinápticos de la red neuronal se calculan dada una señal de entrada, la cual se propaga por todas capas hacia adelante de la red neuronal hasta la capa de salida.
- *La etapa Backforward*, en la cual la señal de error es producto de la comparación de la señal de salida de la red contra la señal deseada, esta señal de error se propaga de la capa de salida hacia la capa de entrada, y en cada capa intermedia los pesos sinápticos son ajustados en proporción directa a la señal de error producida [58].

3.6 Red Neuronal Artificial de Base Radial

Las redes neuronales de base radial (RBNN) en su forma estructural se ejemplifican como se muestra en la figura 3.6.1. Como se pude apreciar de esta figura, una red neuronal de base radial consta básicamente tres capas:

- *Capa de entrada*, la cual consiste de m nodos de entrada, donde m es la dimensionalidad del vector de entrada $\mathbf{x}$.
- *Capa oculta*, la cual consiste del mismo número N de unidades computacionales que poseen los ejemplos de entrenamiento, y cada unidad computacional es descrita por una función de base radial.

$$\varphi_j(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{x}_j\|), j = 1,2,...,N \tag{3.6.1}$$

  Donde $\mathbf{x}_j$ en (3.6.1) define el centro de la función de base radial. Y $\mathbf{x}$ es el patrón de entrada.

- *Capa de salida*, la cual en esta estructura, consiste de una sola unidad computacional. En este tipo de arquitectura no existe restricción en el tamaño de la capa de salida, solo teniendo en claro que esta capa es mucho más pequeña que la capa intermedia.

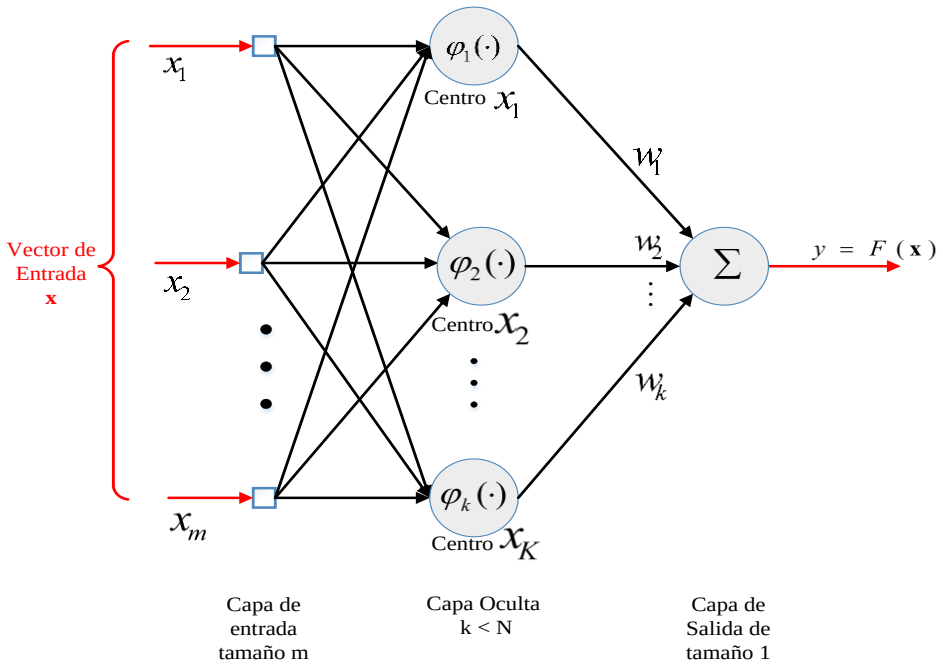

Figura 3.6.1 Estructura de una Red Neuronal de Base Radial.

Una de las funciones kernel más utilizadas en este tipo de arquitecturas es la función Gaussiana como función de base radial. En cuyo caso cada unidad computacional de la capa intermedia de la red es definida por:

$$\varphi_j(\mathbf{x}) = \varphi(\mathbf{x} - \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma_j^2}\left\|\mathbf{x} - \mathbf{x}_j\right\|^2\right), j = 1,2,...,N \qquad (3.6.2)$$

Donde σ_j es la medida del ancho de la j-ésima función Gaussiana con centro en $\mathbf{x}_j$ [58].

3.7 Máquina de soporte vectorial

Básicamente una Máquina de soporte vectorial (SVM por sus siglas en inglés), es una máquina de aprendizaje binario con algunas propiedades muy elegantes. La principal idea detrás de una SVM se resume como: “Dado un conjunto de entrenamiento, la máquina de vector soporte construye un híperplano como superficie de decisión, en el cual, los márgenes de separación tanto positivos como negativos se han maximizado [13].

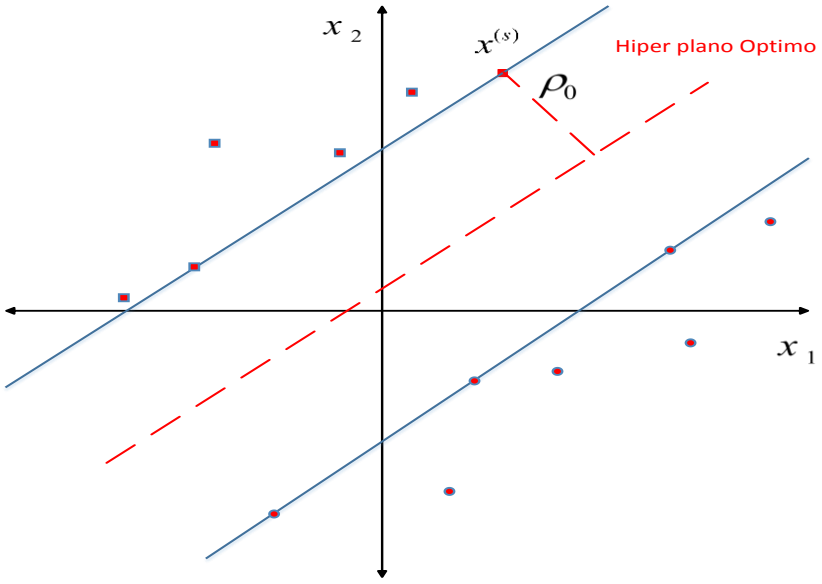

Figura 3.7.1 Ejemplo de híper plano para una Maquina de Vector Soporte.

Así dado un conjunto de entrenamiento $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, donde $\mathbf{x}_i$ es el patrón de entrada para la i-ésima muestra y d_i su respuesta deseada, la ecuación del híper plano que realiza la separación es:

$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + b = 0 \tag{3.7.1}$$

Donde $\mathbf{x}$ es el vector de entrada, $\mathbf{w}$ es el vector de pesos ajustable y b es el bias. Podemos escribir: [58].

$$\begin{aligned} \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq 0 \quad &\text{para} \quad d_i = +1 \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b < 0 \quad &\text{para} \quad d_i = -1 \end{aligned} \tag{3.7.2}$$

Lo anterior se ejemplifica en la figura 3.7.1 donde se muestra el híper plano óptimo y los puntos en color rojo sobre las líneas son los puntos de soporte del híper plano.

3.7.1 SVM Kernel

Dado el teorema de Mercer's, el cual nos especifica que un kernel candidato para una SVM, es aquel núcleo continuo y simétrico que esté definido en un intervalo cerrado $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$, con

coeficientes positivos $\lambda_i > 0, \forall i$. La definición formal del teorema de Mercer's se presenta a continuación:

"Suponga que K es un núcleo continuo simétrico y no negativo. Entonces existe una base ortonormal $\{e_j\}_j$ de $L^2[a,b]$ que consiste de eigen funciones de T_k, tal que la secuencia correspondiente de eigen valores $\{\lambda_j\}_j$ es no negativa. Las eigenfunciones correspondientes a los eigenvalores no cero, son continuas en el intervalo $[a,b]$ y K se representa como:"

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j e_j(s) e_j(t) \tag{3.7.1}$$

"donde la convergencia es absoluta y uniforme" [44].

Sin embargo, el teorema de Mercer's no nos especifica de qué forma se deben de construir estos kernel, de tal forma que los tres tipos de kernel más utilizados son:

El Kernel Lineal el cual está dado por la fórmula (3.7.1). El Kernel Polinomial dado por:

$$(\mathbf{x}^{\mathbf{T}}\mathbf{x}_i + 1)^p \tag{3.7.3}$$

Y el Kernel de base radial se define como:

$$\exp\left(-\frac{1}{2\sigma^2}\left\|\mathrm{x}-\mathrm{x}_\mathrm{i}\right\|^2\right) \tag{3.7.4}$$

3.8 Clasificadores Estadísticos

En el presente trabajo se busca determinar y utilizar de forma experimental el mejor clasificador para nuestro problema en particular, por lo que para un mejor análisis se tomarán en cuenta para nuestro comparativo tres de los clasificadores estadísticos más utilizados de hoy en día, los cuales describimos a continuación.

3.8.1 Naive Bayes

Uno de los clasificadores más efectivos, en el sentido de su comportamiento predictivo es el clasificador llamado Naive Bayes, descrito en [23]. Este clasificador "aprende" de la probabilidad condicional de cada atributo i del conjunto de entrenamiento A. Dada una clase C, la clasificación de una instancia en particular del conjunto $A_1,...,A_n$, es la predicción de la probabilidad más alta a la que pertenezca determinado atributo. Estos cálculos se basan en una asunción de independencia, es decir todos los atributos de A_i son condicionalmente independientes, por independencia se debe de entender independencia probabilística [46]. El modelo del predictor Naive Bayes se puede expresar matemáticamente en base a las probabilidades condicionales de la siguiente forma:

$$p(C \mid F_1,...,F_n) = \frac{p(C)\,p(F_1,...F_n \mid C)}{p(F_1,...,F_n)} \qquad (3.8.1)$$

Donde $p(C)$ es la probabilidad de ocurrencia de la clase C. $p(F_1,...,F_n \mid C)$ es la probabilidad condicional de ocurrencia de la clase F_i dada la clase C. $p(F_1,...,F_n)$ es la probabilidad de ocurrencia de la clase $F_i, 1 \le i \le n$.

3.8.2 Bayes Net

Las redes de Bayes son representaciones gráficas de las relaciones probabilísticas de un conjunto de variables aleatorias y estas se definen de la siguiente forma.

Dado un conjunto $X = \{X_1,...,X_n\}$ de variables aleatorias discretas, donde cada variable X_i toma valores de un conjunto finido de datos, denotado por $Val(X_i)$. Una Red Bayesiana es una grafo a cíclico dirigido G, que codifica la distribución de probabilidad sobre el conjunto X, los nodos del grafo corresponden a las variables aleatorias $X_1,...,X_n$. Los lazos del grafo corresponden a la influencia directa de una variable sobre otra, es decir, si existe un lazo directo del nodo X_i al nodo X_j, la variable X_i será el nodo padre de la variable X_j. Cada nodo posee una probabilidad condicional que se representa por $\mathrm{CPD} = p(X_i \mid Pa(X_i))$, donde $Pa(X_i)$ denota el padre de X_i en G. El par $(\mathrm{G},\mathrm{CPD})$ codifica el conjunto de disrupción $p(X_1,...,X_n)$. Así un único conjunto

de distribución de probabilidades sobre X del grafo G se puede expresar de la siguiente forma [34]:

$$p(X_1,...,X_n) = \prod_i (p(X_i \mid Pa(X_i))) \qquad (3.8.2)$$

Y la representación gráfica de una Red Bayesiana se muestra en la siguiente figura:

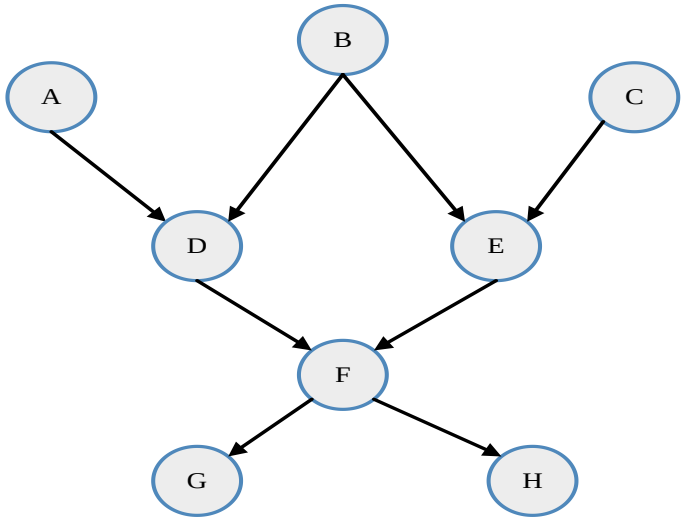

Figura 3.8.2.1 Ejemplo del grafo de una Red Bayesiana.

3.8.3 K Nearest Neighbor (KNN)

El algoritmo KNN es uno de los algoritmos más simples y fundamentales de clasificación el cual debe de ser tomado en cuenta para cualquier estudio serio de clasificación. En 1951 Fix y Hodges introdujeron por primera vez este algoritmo.

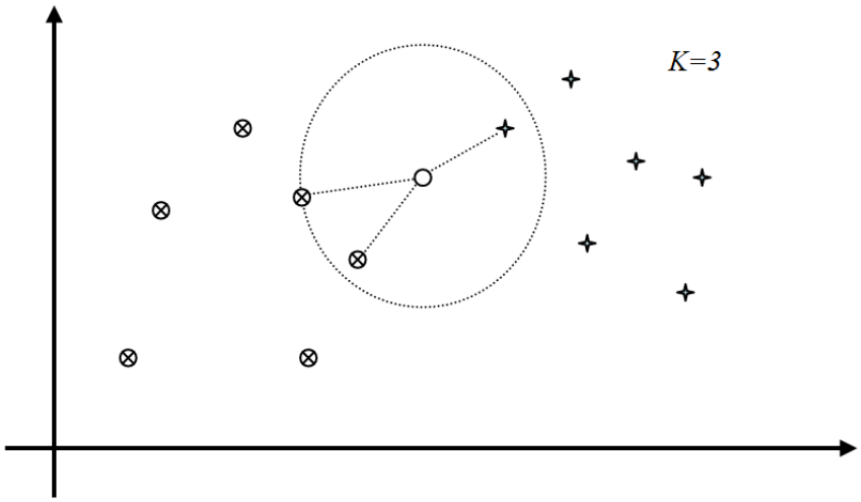

Figura 3.8.3.1 Ejemplo de clasificación por KNN, Aquí, 'x', el punto blanco, es clasificado con la clase ⊗, dado que de sus k (3) próximo vecinos, (1) pertenece a la clase +, y (2) a la clase ⊗. [36].

El algoritmo clasificador KNN se basa comúnmente en la distancia euclidiana del conjunto de muestras de entrenamiento. Así sea $\mathbf{x}_i$ un conjunto de entrada con p rasgos $(x_{i1}, x_{i2},..., x_{ip})$ y sean n el número total de muestras de entrada $(i = 1,2,..., n)$ y p en número total de rasgos $(j = 1,2,..., p)$. La distancia euclidiana entre la muestra $\mathbf{x}_i$ y la muestra $\mathbf{x}_j$ se define como: [27].

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + ... + (x_{ip} - x_{lp})^2} \tag{3.8.3}$$

Una descripción gráfica del concepto del vecino más cercano es dada por primera vez por [62], la cual se muestra en la figura 3.8.3.1.

3.9 AR Drone 2.0 Parrot

El AR. Drone es un Quadrotor UAV (Unmanned Aerial Vehicle) que recibe comandos mediante una conexión inalámbrica. Este puede realizar acciones pre programadas, como lo es el despegar "Take-Off", aterrizar "Land", o permanecer en la misma posición de vuelo "Hover". Este puede ser utilizado tanto en interiores como en exteriores, con y sin estructura de protección "Indoor Hull". El AR. Drone utiliza una batería de litio recargable; Este es capaz de enviar información

en tiempo real acerca de su estado actual, posición y orientación, también puede transmitir video en tiempo real de una de las cámaras seleccionadas. El AR. Drone posee un sistema operativo Linux embebido que puede ser accedido mediante el protocolo Telnet.

Figura 3.9.1 Imagen del UAV Ar. Drone Parrot 2.0.

3.9.1 Características y Componentes

El AR. Drone posee varios dispositivos de censado del medio ambiente, los cuales le permiten obtener información de la altura y posición, los cuales le ayudan a tener una mejor estabilidad, a continuación se listan estos componentes.

3.9.1.1 Especificaciones Técnicas

El AR. Drone cuenta con las siguientes especificaciones técnicas de acuerdo al fabricante [5]:

Sistema de cómputo embebido:

- Procesador a 1GHz 32 bit ARM Cortex A8.
- DSP a 800MHz TMS320DMC64xDDR 128 MB.
- 1Gbit DDR2 RAM a 200MHz.
- Wifi b/g/n.
- Puerto USB 2.0.
- Sistema Operativo Linux 2.6.32 de 32 Bits.

Sistema integral de guía:

- Acelerómetro de precisión de 3 ejes.

- Giróscopo de 3 ejes.
- Magnetómetro de 6 grados de precisión.
- Sensor ultrasónico.

Características físicas:

- Velocidad máxima 5 m/s; 18 km/h.
- Peso:
  - 380g sin caparazón de protección.
  - 420 g con caparazón de protección.

Sistemas de Seguridad:

- Caparazón de protección para interiores.
- Bloqueo automático de hélices al contacto.
- Control de emergencia para el paro de motores.

Estructura Aeronáutica:

- Hélices de alta eficiencia.
- Estructura tubular de fibra de carbón.

Motores y consumo de energía:

- 4 motores Brushless de 35 000 rpm a 15W.
- Batería de polímeros de litio a 1000 mA/h.
- Capacidad de descarga de 10C.
- Tiempo de máximo de vuelo de 12 min.

Cámaras:

- Cámara Horizontal gran angular de 93 grados, CMOS.
- Cámara vertical QVGA a 60 fps.

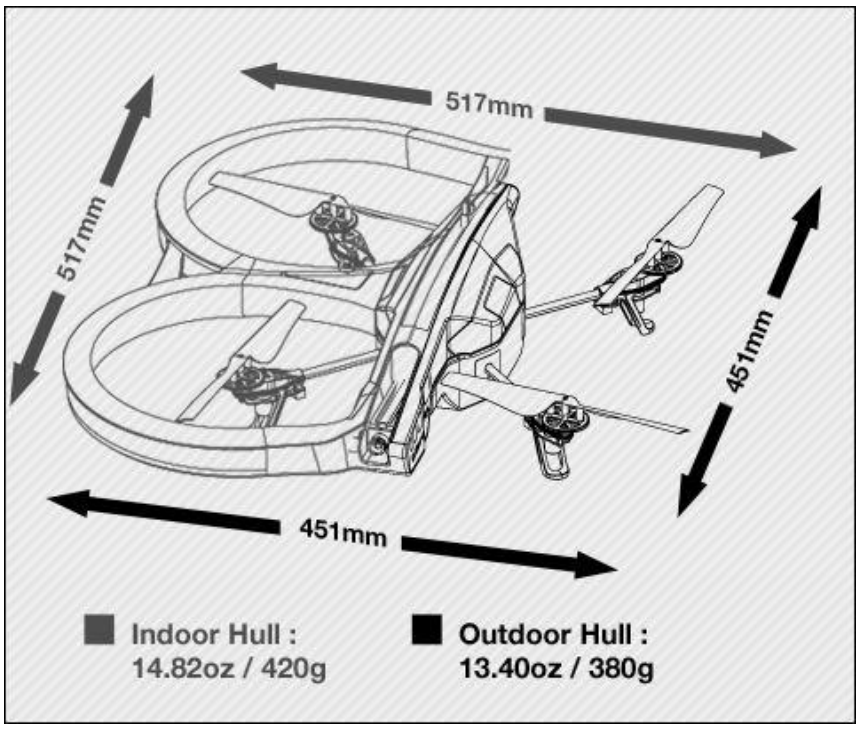

Figura 3.9.1.1.1 Esquema y dimensiones del Quadrotor Ar. Drone.

3.9.1.2 Sensores y Cámaras

El AR. Drone posee varios sensores. Estos sensores de movimiento proveen al software de medidas de deslice, inclinación y empuje. Estas medidas son utilizadas para la estabilización automática del deslice, inclinación y empuje, así como para el control de inclinación asistida.

Un sensor de telemetría ultrasónico provee de información sobre la altitud, que es utilizado para la estabilización de la altitud y control asistido de velocidad vertical.

El AR. Drone posee dos cámaras: una cámara frontal en posición horizontal, y una cámara en posición vertical con vista hacia el suelo. La cámara vertical es utilizada por el Ar. Drone para medir la velocidad relativa al suelo, esta es utilizada para realizar flotamiento automático. El AR. Drone es capaz de enviar video codificado de cualquiera de las 2 cámaras.

3.9.1.3 Conexión Inalámbrica

La conexión inalámbrica es utilizada por el cliente para controlar el Ar. Drone. Cualquier dispositivo con soporte Wifi en modo Ad-hoc puede conectarse al Ar. Drone. Cuando se prende en dron, este automáticamente crea una red Wi-Fi con un ID de la forma adrone_XXX. Si el dron detecta que una red con el ID especificado ya existe, este intentara conectarse a la red ya existente.

3.9.1.4 Input / Output Streams

El dron puede enviar dos flujos de salida de datos UDP (User Datagram Protocol), uno es el flujo de datos de navegación y el otro es el flujo de datos de video. El flujo de datos de navegación contiene información acerca del estado actual del dron. Y el flujo de datos de video, contiene los datos codificados de una de las cámaras. Además de los flujos de salida, el dron es capaz de recibir un flujo de entrada UDP que contiene los comandos del mismo.

3.9.2 Operación

Para utilizar el dron en interiores, este debe de estar protegido con su "Indoor Hull", para el uso del Quadrotor en exteriores, este posee otro armazón denominado "Outdoor Hull". A continuación se muestran las imágenes de los dos armazones, a la izquierda, el armazón para interiores y a la derecha el armazón para exteriores:

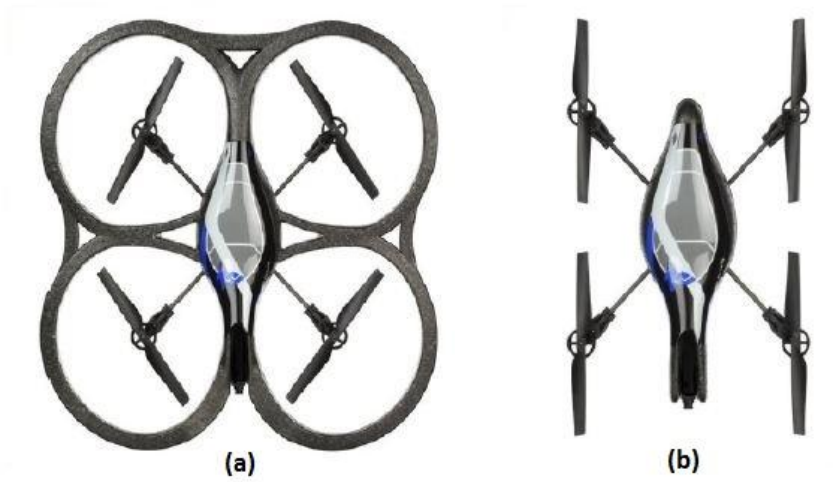

Figura 3.9.2.1 Ejemplo de los cascarones de protección de Ar. Drone. (a) cascarón para interiores, (b) cascarón para exteriores.

Para entender como vuela el AR. Drone observe la siguiente imagen, que muestra la rotación de los motores:

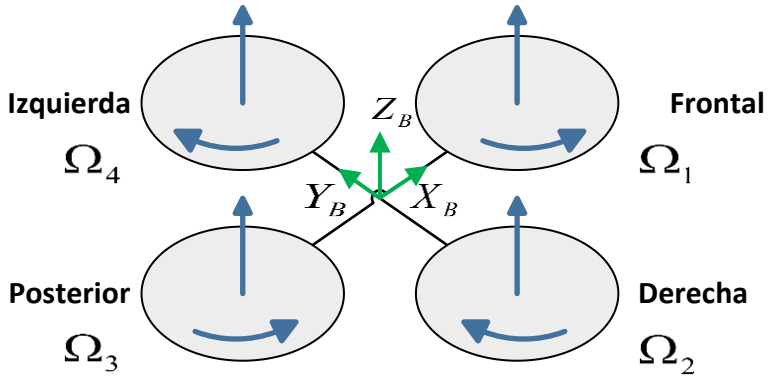

Figura 3.9.2.2 Ejemplo de funcionamiento de los motores del Ar. Drone.

Cada par de motores opuestos gira en la misma dirección, es decir, un par gira en el sentido de las manecillas del reloj, mientras el otro par gira en sentido contrario de las manecillas del reloj. Considere $\Omega_{1,}, \Omega_2, \Omega_3, \Omega_4$ como las velocidades angulares de cada motor. Para que el dron permanezca volando en la misma posición, este debe de mantener todas las velocidades angulares de los motores iguales. Es decir $\Omega_{1,} = \Omega_2 = \Omega_3 = \Omega_4$. Para moverse en la dirección de los axis XB, YB, ZB, el dron, cambia la velocidad angular de los motores, para entender cómo cambian las velocidades considere la siguiente figura:

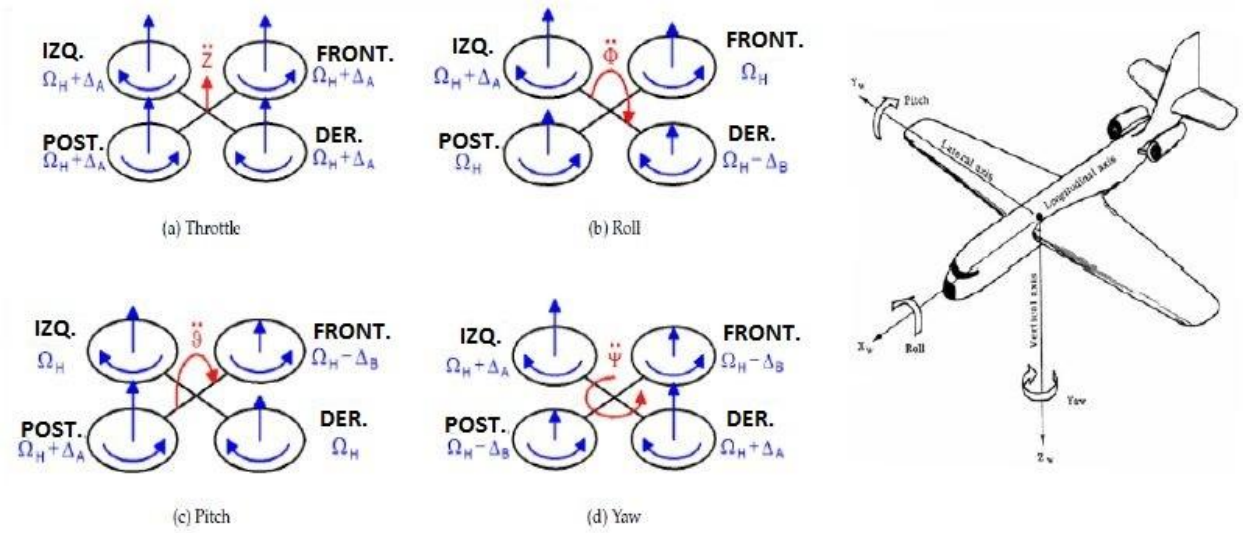

Figura 3.9.2.3 Ejemplo de cómo varía la velocidad angular de los motores para producir el desplazamiento del Ar. Drone.

Considere ΔA y ΔB como valores de velocidad angulares. Para realizar un desplazamiento hacia delante, el dron debe de cambian el ángulo de inclinación con respecto a su horizontal, para esto, es necesario cambiar las velocidades angulares de los motores delanteros y traseros, como se observa en la parte (c) de la figura anterior (3.7.2.3).

Para desplazarse hacia la izquierda o derecha, el dron debe de cambiar el ángulo de empuje, para realizar esto, es necesario cambiar las velocidades angulares de los motores derecho e izquierdo, como se observa en la parte (b) de la imagen anterior.

Para girar hacia la izquierda o hacia la derecha, el drone debe de cambiar el ángulo de deslice, para esto es necesario cambiar las velocidades angulares de todos los motores, el caso (d) de la imagen anterior, cambiando la velocidad de cada par de motores opuestos en la misma proporción.

Finalmente para ganar o perder altura, el dron debe de cambiar la velocidad angular de todos los motores en la misma proporción, caso (a) de la imagen anterior.

3.9.3 Control

Para controlar el dron, el cliente debe de enviar los comandos en forma de paquetes UDP utilizando la conexión Wifi. Los comandos se encuentran formados por cadenas de caracteres de 8 bits en formato ASCII, estas cadenas siempre inician con el prefijo "AT*", seguido del nombre del comando, un signo de igual, una secuencia de números y una lista de parámetros opcionales. Un comando AT debe de ser separado de los demás por un retorno de carro, si estos se encuentran en el mismo paquete UDP. Un ejemplo de comando es:

"AT*PCMD=21625,1,0,0,0,0".

Los comandos PCMD ("Progressive Commands") son utilizados para que se traslade y rote. La sintaxis para este comando es:

"AT*PCMD=<sequence_number>,<flag>,<roll_p>,<pitch_p>,<gaz_p>,<rot_p>".

- *sequence_number*, como su nombre lo indica, es el número de secuencia del comando y depende del número de comandos que se hayan enviado con anterioridad.

- *flag*, define si el dron considerará los argumentos siguientes. Si el valor es 1, el dron considerará los argumentos *phi, theta, gaz, rotation.* Si el valor es 0, el dron ejecutara el "hover".
- *roll_p*, es un valor en el rango [-1:1] que representa el porcentaje de empuje que el dron podrá alcanzar. Este posee un valor por omisión de 12 grados. Por ejemplo si este valor se especifica como 0.5, entonces el dron se inclinará 6 grados.
- *pitch_p*, es un valor en el rango de [-1:1] que representa el porcentaje de inclinación que el dron puede adquirir.

La interface PCMD facilita el control del dron, si el cliente desea que el dron permanezca en el aire sin hacer nada, este solo necesita enviar un comando con los 4 argumentos iguales a cero y el argumento flag igual a 1. De esta forma el dron permanecerá en su posición, pero se deslizará un poco debido a la inercia. Si el cliente desea que el dron permanezca en su posición y que además se cancele la inercia, este solo necesita enviar un comando con la bandera igual a 1.

El dron envía información acerca de su posición relativa, posición y velocidad. La información de la posición está dada en las coordenadas (X,Y,Z), del dron en el espacio, considerando la posición de origen del sistema como $(X,Y,Z)=(0,0,0)$ donde el sistema despego.

Después de que el dron ha despegado, este empieza a enviar información válida acerca de su posición en el espacio y su velocidad [12]. Lo anterior se ejemplifica en la Figura 3.9.3.1.

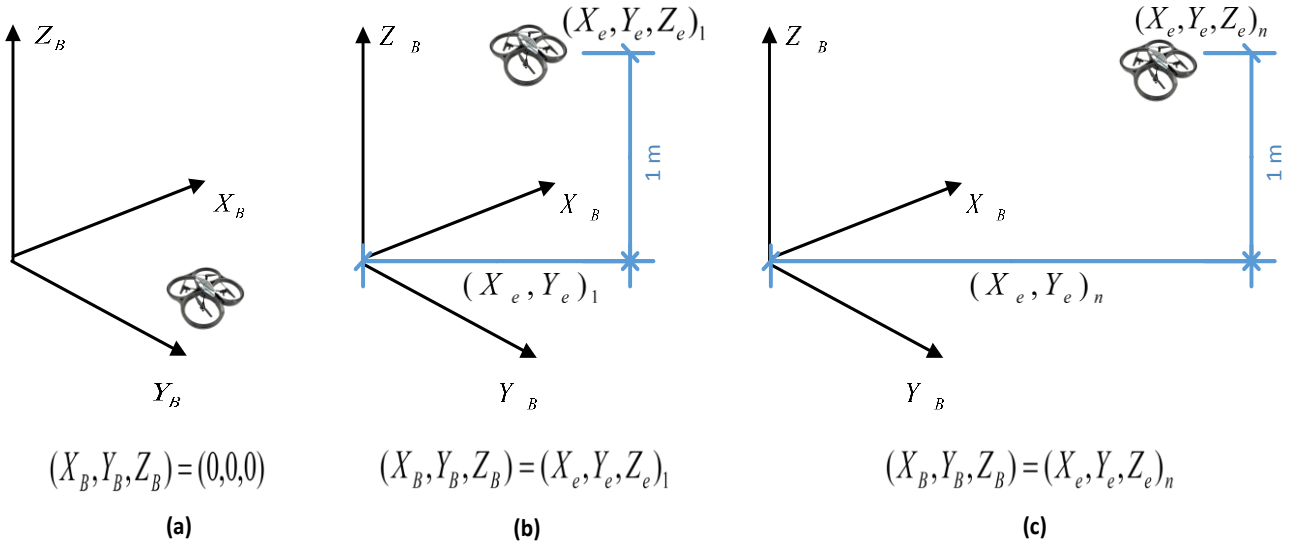

Figura 3.9.3.1 Ejemplo de la posición relativa del dron en el espacio. (a) El dron en posición inicial $(0,0,0)$. (b) El dron en la primera posición de vuelo (X_1, Y_1, X_1) . (c) El dron en la posición n de vuelo (X_n, Y_n, Z_n).

Capítulo 4
Metodología

En el presente capítulo se presentan los procedimientos, técnicas y metodologías empleados para realizar el control de forma adecuada del Quadrotor, la extracción y procesamiento del video en tiempo real, la detección de puntos de interés, su filtrado, clasificación y la puesta en operación de todo el conjunto de técnicas en una interface gráfica de usuario (GUI) como producto final.

4.1 Control del Quadrotor

Como se mencionó en la sección 3.9.3 el Quadrotor y en específico el Ar. Drone Parrot 2.0, es controlado por medio de tramas UDP, las cuales son enviadas de forma asíncrona desde el programa cliente. Estas tramas cuyo formato se describió en el capítulo 3, poseen un orden y una secuencia específica, dependiendo del módulo del Quadrotor al que se desee acceder se deberá de enviar un formato de trama en particular. Lo anterior es debido a que el Quadrotor cuenta con varios módulos de censado y uno de control, por lo que si se desea activar la transmisión del video, se deberá de enviar la trama que especifique la inicialización de captura de video.

Para realizar lo anterior se debe de contar con la implementación de este protocolo para los diferentes módulos de censado y control del Quadrotor, así como con el manejo de prioridades de los datagramas UDP. Para esto se evaluaron diferentes frameworks existentes en el mercado, estos se muestran en la siguiente tabla comparativa:

API	Lenguaje	Código fuente	Control Altura	Lectura de todos los sensores	Interface Gráfica	Multiplataforma	Documentación
Ar. Drone 2.0	C	Si	Si	Si	No	No	Si
OpenDrone Control	SCALA	No	Si	No	Si	No	Si
JavaDrone	Java	Si	Si	Si	Si	Si	Si
YAdrone	Java	Si	Si	Si	Si	Si	Si

Tabla 4.1.1 Tabla comparativa de frameworks evaluados para controlar el Ar. Drone Parrot.

Como resultado del análisis de los frameworks anterior, se optó por utilizar la API YAdrone, la cual nos proporciona una interface amigable y fácil de implementar en nuestro proyecto Java. Las interfaces gráficas y los módulos de inicialización de los sensores del Ar. Drone nos permiten un rápido acceso al estado en general del Quadrotor. En las siguientes secciones se muestran las interfaces gráficas y módulos de control que nos proporciona dicha API.

4.1.1 Ambiente de pruebas para las API de control del Ar. Drone

Para cada una de las librerías mencionadas en la tabla 4.1.1, se realizaron un total de 15 pruebas por cada framework involucrado, en cada una de las pruebas por framework se midió el tiempo de conexión al Quadrotor, perdidas de conexión, soporte para lectura asíncrona de los datos de navegación y evaluación de la interface gráfica.

API	Tiempo de Conexión (ms)	Perdida de conexión	Datos de Navegación	Interface Gráfica
Ar. Drone 2.0	154	0%	Si	Si
OpenDrone Control	430	2%	Si	No
JavaDrone	352	4%	Si	Si
YAdrone	290	1%	Si	Si

Tabla 4.1.1.1 Tabla comparativa de las características de cada framework evaluado.

Como se puede observar de las tablas 4.1.1 y 4.1.1.1 la API Ar. Drone 2.0 distribución propietaria de Parrot es una de las mejores opciones, sin embargo, como ya se mencionó se eligió el API YAdrone, debido a esta posee la característica de ser multiplataforma ya que se encuentra escrita en el lenguaje de programación Java, aunado a lo anterior, el buen rendimiento que posee en cuanto a tiempos de conexión, porcentaje bajo de desconexión, lectura de datos de navegación en multi hilo e interface gráfica amigable.

El procedimiento para la evaluación de los parámetros descritos en las tablas 4.1.1 y 4.1.1.1 es el siguiente:

1) Eliminar todos los procesos extraños ejecutándose como procesos de fondo.
2) Ejecutar el programa principal que hace uso de la API a evaluar.
3) Iniciar la conexión WiFi con el Ar. Drone.
4) Tomar el tiempo que le toma al cliente conectarse con el Ar. Drone.
5) Monitorear el estado de la conexión por 5 minutos.
6) Lectura de los datos de navegación por el periodo de tiempo especificado en (5).
7) Análisis empírico de la interface gráfica proporcionada por la API.

El procedimiento anterior se ejemplifica en el siguiente diagrama de flujo:

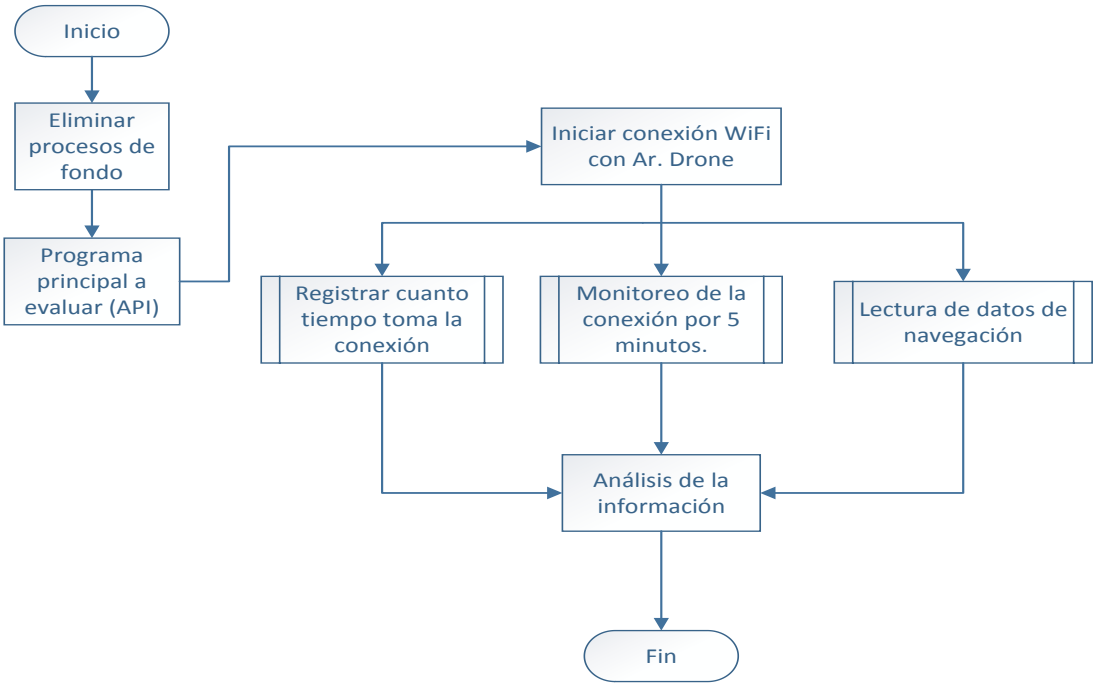

Figura 4.1.1.1 Diagrama de flujo que muestra el criterio de evaluación aplicado para evaluar las API mencionadas en las tablas 4.1.1 y 4.1.1.1.

Las características del ordenador donde se realizaron las pruebas son:

- Sistema Operativo Windows 7 a 64 bits.
- Microprocesador Intel Core i5 Quadcore @ 2.4 GHz.
- 4 GB de Memoria RAM.
- Java SDK Enviroment 1.6.0_13.
- Disco duro hibrido con 8GB de SSD.

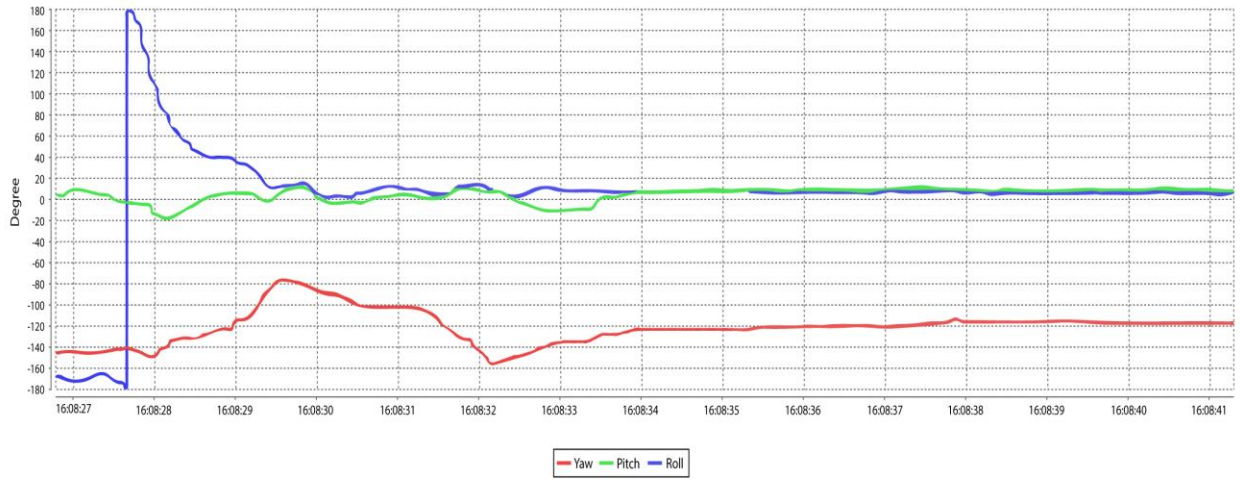

Figura 4.1.1.2 Figura que muestra el comportamiento de los ángulos de inclinación en vuelo del Quadrotor. De color rojo el ángulo de deslice, de verde el ángulo de inclinación vertical y de color azul el ángulo de inclinación horizontal.

4.1.2 Información de la posición de vuelo

El Quadrotor una vez que despega, requiere conocer su estado en vuelo estacionario y en movimiento, como lo son los ángulos de desplazamiento e inclinación, para esto el Ar. Drone utiliza una Unidad de Medida Inercial (IMU), la cual proporciona el valor de los ángulos de deslice (Yaw), inclinación (Pitch) y de empuje (Roll).

Estos ángulos se controlan mediante un controlador PID que posee integrado el Quadrotor y se varían como respuesta a los comandos de control enviados por la interface cliente. El comportamiento en vuelo de una trayectoria cerrada (como se muestra en la sección 4.1.1), a través de una ventana de tiempo dado se muestra en la figura 4.1.2.

4.1.3 Configuración de parámetros de vuelo

Dentro de los parámetros de configuración que se deben de especificar para un vuelo óptimo del Quadrotor se encuentra el ángulo de inclinación y la velocidad de los motores. Estos dos parámetros trabajan de forma conjunta. Por un lado el ángulo de inclinación o ángulo de Euler especifica el grado de inclinación que se generará cuando se requiera que el Quadrotor se desplace en cualquiera de sus ejes de libertad. Es decir si se requiere que se desplace hacia adelante, como se explicó en la sección 3.9, una diferencia de velocidad angular se produce en los pares de motores delanteros y traseros. La velocidad angular del par de motores delanteros se disminuye mientras que al mismo tiempo la velocidad angular de los motores traseros se aumenta, esto produce una variación del ángulo del Quadrotor con respecto al plano horizontal, y a este ángulo se le denomina ángulo de Euler. La anterior se ejemplifica en la siguiente figura:

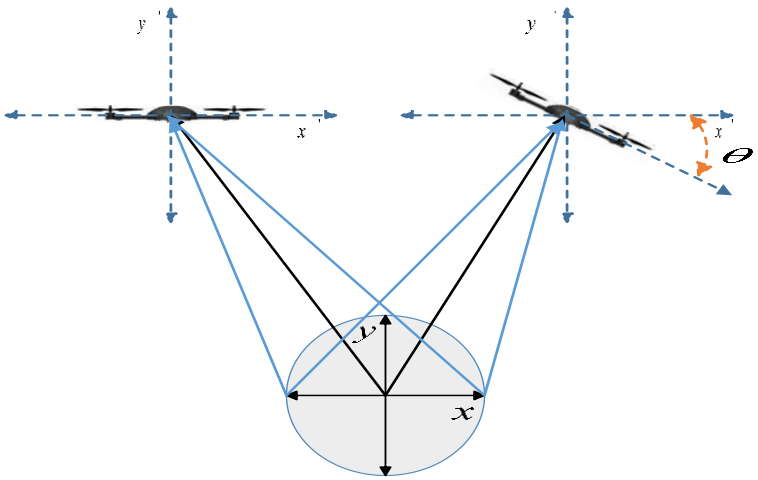

Figura 4.1.3.1 Ejemplo de variación del Angulo de Euler (θ) para lograr un desplazamiento del Quadrotor.

Por razones de seguridad y estabilidad este ángulo no debe de exceder los $\pm 30^{\circ}$ con respecto a su horizontal, de lo contrario un ángulo mayor a $\pm 30^{\circ}$ producirá que el Quadrotor pierda el control. Este ángulo es directamente responsable de la velocidad de desplazamiento, es decir, a mayor grado de inclinación con respecto al plano horizontal del Quadrotor (menor a ± 30 grados), la velocidad de desplazamiento será mayor.

La velocidad de desplazamiento y tiempos de respuesta ante comandos de movimiento también están directamente ligados a la velocidad angular de los motores de Quadrotor, es decir, a mayor

velocidad de giro angular, más rápida será la respuesta y el desplazamiento del Quadrotor en cualquiera de sus ejes será también más rápido.

Los parámetros de configuración con respecto a la velocidad angular y ángulo de inclinación, se dan en porcentajes de acuerdo a la velocidad angular máxima de los motores y al ángulo máximo de inclinación. Así para el Ar. Drone 2.0 la velocidad angular máxima a la que los motores pueden girar (según su hoja de especificaciones) es de 2000 mm/s y un ángulo máximo de inclinación es de $\pm 30^{\circ}$.

La configuración que se utiliza en el presente trabajo es:

Velocidad Angular: $\omega = \omega_{\max} * 0.5 = 1000 mm/s$.

Angulo de inclinación: $\theta = \theta_{\max} * 0.1 = \pm 3^{\circ}$ grados de inclinación.

Altura Máxima de Vuelo = 1000 mm.

Con los parámetros anteriores se logra un buen control del Quadrotor en vuelo, es decir, con la proporción de la mitad de la máxima velocidad de giro de los motores, se logra que el Ar. Drone responda lo suficientemente rápido, sin sacrificar demasiado el tiempo de vuelo. Y junto con el ángulo de desplazamiento de $\pm 3^{\circ}$ se logra que el desplazamiento sobre la horizontal sea lento y estable, lo cual es necesario para que se puedan analizar las imágenes captadas por la cámara vertical de Quadrotor (el ambiente de pruebas se describe a detalle en la sección 4.14).

4.1.4 Configuración de video

Como se ha mencionado con anterioridad, todos los sensores del Quadrotor que se deseen utilizar deben de ser primeramente configurados. Para la extracción del video, se deben de tomar en cuenta dos aspectos principalmente:

1) La frecuencia de muestreo de imágenes.
2) El tamaño de la imagen.

El Quadrotor es capaz de enviar diferentes tamaños de imágenes a diferentes frecuencias de muestreo, en este punto del diseño se debe de hacer un balance entre calidad de imagen, tiempos

de muestreo y tiempos de procesamiento de la imagen, es decir, entre mayor tamaño sea la imagen, mayor será el tiempo de procesamiento. Los formatos de imagen y frecuencias que el Quadrotor puede procesar se muestran en la Tabla 4.1.4.1 [5].

No.	Video Códec	Tamaño Imagen	FPS	Codificador Hardware	Cámara Horizontal
1	MP4_360P_CODEC	360 x 240	30	No	Si
2	H264_360P_CODEC	360 x 240	30	Si	Si
3	MP4_360P_H264_720P_CODEC	720 x 680	10	No	No
4	H_264_720_CODEC	720 x 680	10	Si	No
5	MP4_360_H264_320P_CODEC	360 x 240	30	Si	Si

Tabla 4.1.4.1 Tabla comparativa de los formatos de imagen y FPS que procesa el Quadrotor.

Para decidir que formato es el más apto a utilizar se configuró el Quadrotor con cada uno de los formatos de video mencionados en la tabla 4.1.4.1. Posteriormente se construyó un programa el cual almacenaba las imágenes enviadas y se contabilizó el número de imágenes por segundo, así también como el número de imágenes (frames) perdidas debido a problemas de sincronización, obteniendo la siguiente tabla de resultados.

No.	Video Códec	Numero Imágenes x Segundo	Numero de Imágenes Perdidas (%)
1	MP4_360P_CODEC	30	2%
2	H264_360P_CODEC	30	1%
3	MP4_360P_H264_720P_CODEC	12	3%
4	H_264_720_CODEC	10	1%
5	MP4_360_H264_320P_CODEC	30	2%

Tabla 4.1.4.2 Tabla comparativa de desempeño de los diferentes formatos de video.

Como se observa en la tabla anterior, el formato de extracción de video que se comportó de forma más estable es el formato número 2 de la tabla 4.1.4.1, H264_360P_CODEC, esto debido a que este códec no implementa la codificación de video MP4. El tamaño de imagen que otorga es de 360x240, él cual es suficiente para el procesamiento y extracción de los puntos de interés, así como también el tiempo de muestreo es lo suficiente mente alto para el análisis temporal que será descrito posteriormente en este capítulo.

Las características del ambiente de pruebas son idénticas a las de la sección 4.1.1 en cuanto a la sección de características de equipo de cómputo.

En cuanto a los porcentajes presentados en la tabla 4.1.4.2, estos se obtuvieron de la siguiente forma:

1) Eliminar todos los procesos extraños ejecutándose como procesos de fondo.
2) Ejecutar el programa principal que hace uso de la API a evaluar.
3) Iniciar la conexión WiFi con el Ar. Drone.
4) Iniciar la transmisión de video mediante los comandos AT*PCMD.
5) Contabilizar el número de imágenes por segundo recibidas en el cliente.
6) Contabilizar el número de imágenes perdidas por errores de sincronización.

Tecla	Acción
Enter	Despegar
Space	Aterrizar
1	Cámara Frontal
3	Cámara Vertical
P	Inicia / Detiene proceso de análisis
Up	Desplazamiento hacia adelante
Down	Desplazamiento hacia atrás
Left	Desplazamiento hacia la izquierda
Right	Desplazamiento hacia la derecha
Shift + Up	Incrementa Altitud de vuelo
Shift + Down	Decrementa Altitud de vuelo
Shidt + Left	Gira a la izquierda sobre el mismo eje
Shift + Right	Gira a la derecha sobre el mismo eje

Tabla 4.1.4.3 Tabla de los principales comandos del módulo de control de la API YAdrone.

4.1.5 Control mediante teclado

Como toda interface hacia un robot, el cual requiere ser manipulado de forma remota, la API YAdrone nos proporciona una interface gráfica la cual implementa un protocolo de comunicación mediante UDP y la estructura de comandos AT*PCMD descrita en la sección 3.9.3. Con esta interface gráfica se ligan las interrupciones (manejadas por componentes SWING de Java) que se producen al presionar una tecla del teclado a los comandos AT*PCMD que se envían al Quadrotor para su control. Así la configuración que se maneja para el desarrollo de la presente tesis se muestra en la tabla 4.1.5.1.

La interface de control anterior se muestra al usuario mediante una GUI, cuya imagen se muestra en la figura 4.1.5 [66]:

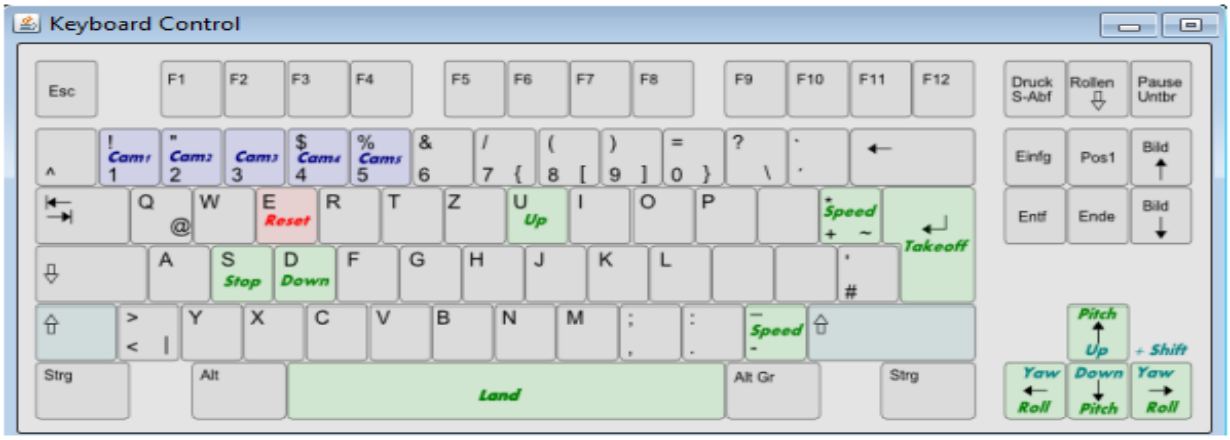

Figura 4.1.5.1 Figura de la interface de control para el Ar. Drone.

4.1.6 Transmisión y extracción de video

Como se mencionó en la sección 4.1.4, una vez realizada la configuración del video. Y establecido el canal de comunicación del mismo, el Ar. Drone es responsable del enviar las imágenes que componen el video capturado a una razón de 30 FPS. Para esto la API Yadrone nos proporciona la siguiente interface de adquisición de imágenes, la cual se apoya en la implementación de interfaces "Adapters" y "Listeners" de Java Swing, los cuales son patrones de diseño para el manejo asíncrono de eventos.

En específico la interface ImageListener de la API YAdrone debe de ser registrada en la clase en la cual se desee realizar el manejo de las imágenes. Así una vez que el Ar. Drone transmita una imagen, esta será captada de forma asíncrona por la clase que implemente dicha interface y el elemento que recibirá será un elemento del tipo BufferedImage. El diagrama UML de clases de este módulo se presenta en la figura 4.1.6.

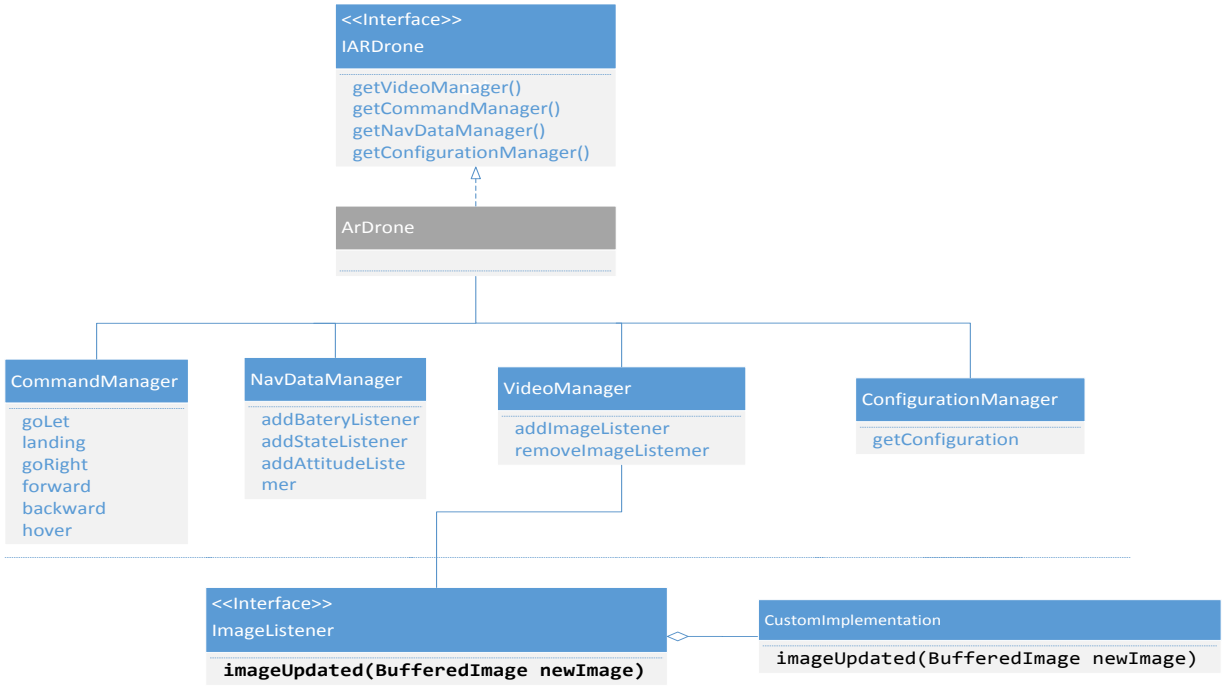

Figura 4.1.6.1 Diagrama a bloques del módulo de extracción de imágenes de Ar. Drone mediante la API YAdrone.

4.2 Extracción de puntos de interés

Una vez que identificamos la forma de extraer las imágenes del video que trasmite el Quadrotor, requerimos realizar el procesamiento de las imágenes que éste nos entrega. El procesamiento de las imágenes consiste en la detección de puntos de interés y extracción de los descriptores de los puntos de interés sobre cada una de las imágenes que se reciben del Quadrotor. En este punto como en el anterior 4.1. Se realiza un análisis sobre diferentes implementaciones que proporcionen una implementación confiable sobre el algoritmo SURF. Para esto se evalúan los siguientes frameworks, tomando en cuenta los siguientes puntos:

1) Lenguaje de programación.
2) Repetitividad de los puntos de interés detectados.
3) Tiempo de procesamiento.

El lenguaje de programación es importante, en específico este debe de ser Java, ya que la integración de la implementación de este algoritmo con el API de control del Quadrotor sería

trasparente (ver sección 4.1), sin embargo no es un requerimiento indispensable, ya que algún otra API con un buen rendimiento y en otro lenguaje de programación puede ser integrada en el proyecto mediante el uso de Java Native Interface (JNI).

La repetitividad de los puntos de interés es deseable e importante en el sentido de que un punto de interés detectado para un objeto en específico debe de presentarse en diferentes imágenes del mismo objeto con una variación mínima en cuanto a las transformaciones más comunes de una imagen (ver sección 3.1) .

El tiempo de procesamiento es otro de los factores importantes a considerar, ya que uno de los objetivos de la presente tesis, es el de poder identificar en tiempo real situaciones en suelo mediante las imágenes tomadas por el Quadrotor. Por lo que el tiempo de procesamiento en la detección de los puntos de interés y la extracción de los descriptores de puntos de interés, se vuelve un factor crítico en el rendimiento del programa.

A continuación se muestra una tabla comparativa de las implementaciones analizadas:

No.	API	Lenguaje	Versión	Multihilo
1	BoofCV	Java	0.5	No
2	OpenSURF	C++	27/05/2010	No
3	JavaSURF	Java	SVN r4	No
4	JOpenSURF	Java	SVN r24	No
5	OpenCV	C++/Java	2.3.1	No

Tabla 4.2.1 Tabla comparativa de las diferentes API's que implementan el algoritmo SURF.

Cabe mencionar que las implementaciones involucradas en la tabla comparativa 4.2 poseen una gran cantidad de algoritmos y características implementadas, las cuales no se tomaron en cuenta, el único algoritmo que se tomó en cuenta para realizar la comparación de estabilidad de los puntos de interés detectados y tiempos de procesamiento fue la implementación del algoritmo SURF para cada una de las APIs involucradas [11].

La evaluación de cada uno de los frameworks se realizó de la siguiente forma:

1) Eliminar todos los procesos extraños ejecutándose como procesos de fondo.
2) Ejecutar el programa de la API en evaluación, con un conjunto de 10 imágenes de la misma escena de un tamaño de 720×680 y de 360×240.
3) Medir el tiempo de cálculo que le toma en procesar cada imagen.

4) Contabilizar el número de puntos de interés detectados.
5) Verificar repetitividad de los puntos de interés (Manualmente).

Las características del equipo de cómputo son las mismas que se mencionaron en la sección 4.1.1. Los resultados de la evaluación de las implementaciones anteriormente mencionadas se muestran en la tabla 4.2.2:

No.	API	Tiempo Procesamiento (ms)	Repetitividad
1	BoofCV	250	97%
2	OpenSURF	530	97%
3	JavaSURF	2100	63%
4	JOpenSURF	1800	98%
5	OpenCV	650	92%

Tabla 4.2.2 Tabla comparativa de desempeño y confiabilidad de framework SURF.

Como se observa en la tabla anterior la API con la implementación para la cual (en nuestro análisis) obtuvimos los mejores resultados fue la API BoofCV, en la cual los tiempos de detección y extracción de descriptores de los puntos de interés fueron menores, así como el porcentaje de repetitividad de los mismos fue alta.

Las imágenes de prueba con las cuales se realizó el análisis se muestran a continuación en las figuras 4.2.1 a 4.2.6. Del lado izquierdo se muestra la imagen original y del lado derecho se muestra la detección de los puntos de interés representados con cruces verdes. Así también se muestra un comparativo entre la cámara frontal y la cámara vertical, en el cual se observa claramente como la resolución de las cámaras afecta en proporción directa a la detección de los puntos de interés. Y sin embargo, aun con la cámara vertical, la cual es de mucha menor resolución que la cámara horizontal, se observa que es posible identificar de forma confiable objetos mediante los descriptores de los puntos de interés.

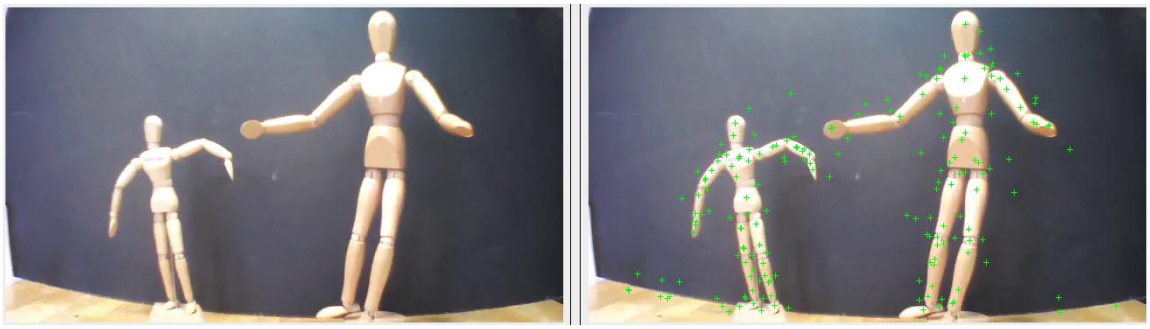

Figura 4.2.1 Ejemplo de figura de prueba 1 con detección de puntos de interés con la cámara frontal.

Figura 4.2.2 Ejemplo de figura de prueba 2 con detección de puntos de interés con la cámara frontal.

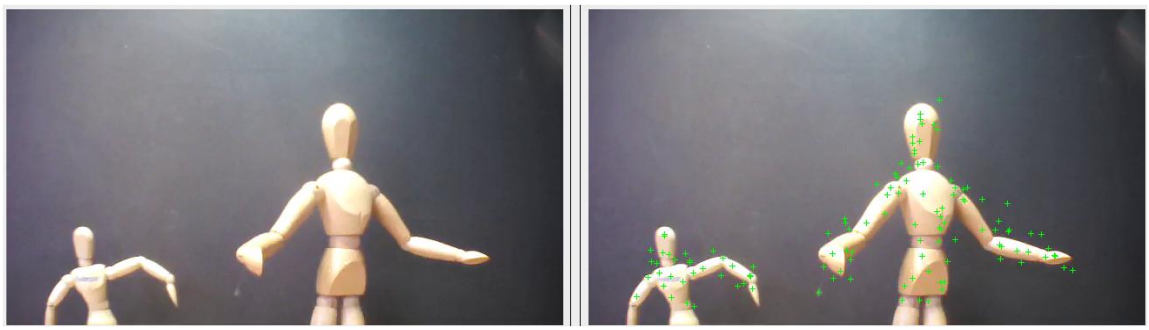

Figura 4.2.3 Ejemplo de figura de prueba 3 con detección de puntos de interés con la cámara frontal.

Figura 4.2.4 Ejemplo de figura de prueba 4 con detección de puntos de interés con la cámara vertical (menor resolución).

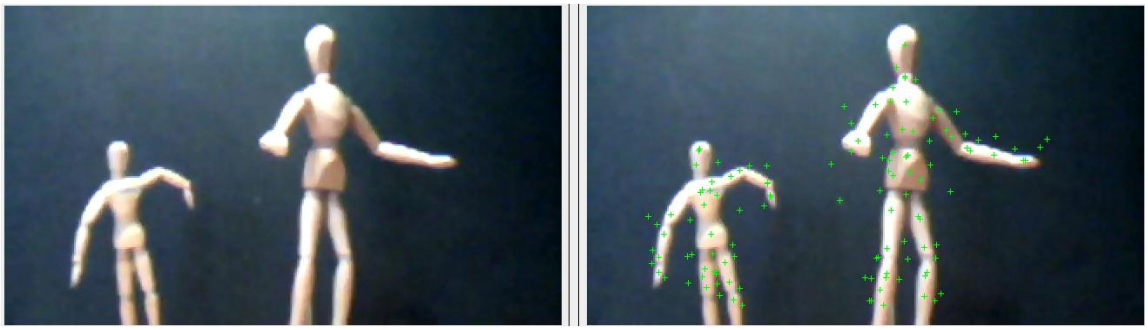

Figura 4.2.5 Ejemplo de figura de prueba 5 con detección de puntos de interés con la cámara vertical (menor resolución).

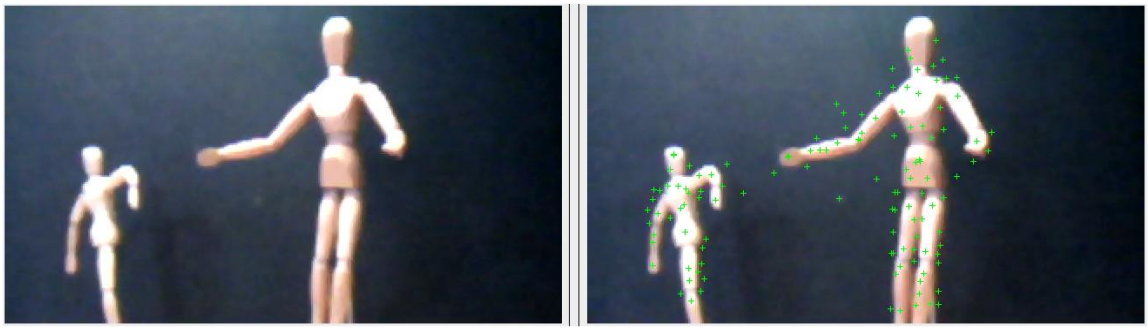

Figura 4.2.6 Ejemplo de figura de prueba 6 con detección de puntos de interés con la cámara vertical (menor resolución).

En las imágenes anteriores observamos como de la Figura 4.2.1 a la Figura 4.2.3 los detalles de los objetos enfocados poseen una mayor nitidez, esto es debido a la resolución de la imagen que produce la cámara horizontal. De la Figura 4.2.4 a la Figura 4.2.6 se observan que dichos detalles no son tan nítidos, esto también debido a que la resolución de las imágenes que produce la cámara horizontal son de menor nitidez.

4.3 Afinación del detector de puntos de interés

Una vez que se eligió el Framework a utilizar y se implementó el procedimiento para la detección de puntos de interés. Observamos que en las imágenes de prueba, el detector de puntos de interés nos proporcionaba una gran cantidad de puntos de interés detectados (de varios cientos por imagen), dependiendo de la imagen que se procese. Un gran porcentaje de los puntos de interés detectados nos proporcionaba información repetida, por lo que realizar un proceso de filtrado de dichos puntos nos ayudaría a optimizar los tiempos de procesamiento, así como el reducir la redundancia de la información a clasificar y el porcentaje de sobre entrenamiento de las redes neuronales también se verá reducido (procedimiento que más adelante se explicará).

Para afinar el detector de puntos de interés, nos apoyamos fuertemente en la implementación del algoritmo SURF de la API BoofCV. En específico y como se comentó en la sección 3.3, el detector de puntos de interés, se basa en la supresión de no máxima del resultado de realizar la diferencia de Gaussianas sobre un espacio de escala de la imagen en cuestión. Este valor de supresión de no máxima, puede ser manipulado mediante un umbral dado en porcentaje del valor máximo de energía detectado por el algoritmo. El cual al modificarlo filtrará los puntos de interés detectados en base a su valor escalar. Es decir, si al valor del umbral le damos un valor del 10%, los valores de los puntos de interés que caigan por debajo del umbral del 10% del máximo valor detectado, no serán tomados en cuenta y solo los puntos de interés con mayor energía serán procesados. Esto está dado por la siguiente formula:

$$U_T = V_{\max} - V_{\max} * (P_T / 100) \tag{4.3}$$

Donde U_T es el nuevo nivel de umbral al aplicar en la detección de puntos de interés. $V_{\max}$ es el valor máximo (escalar) de los puntos de interés y P_T es el valor de porcentaje que se especifica como umbral.

En la selección del valor del umbral, se tuvo cuidado en seleccionar un valor en el cual, los puntos de interés se filtraran para una región de interés en específico, pero que al mismo tiempo los puntos de interés restantes identificaran de forma precisa al objeto en cuestión. Así a continuación se muestran las imágenes de prueba con las cuales se realizó la selección del umbral. En estas, las

regiones de interés son las palabras sobre los objetos en primer plano y los puntos de interés espurios son los puntos de interés detectados sobre las regiones como lo es suelo y los objeto de fondo.

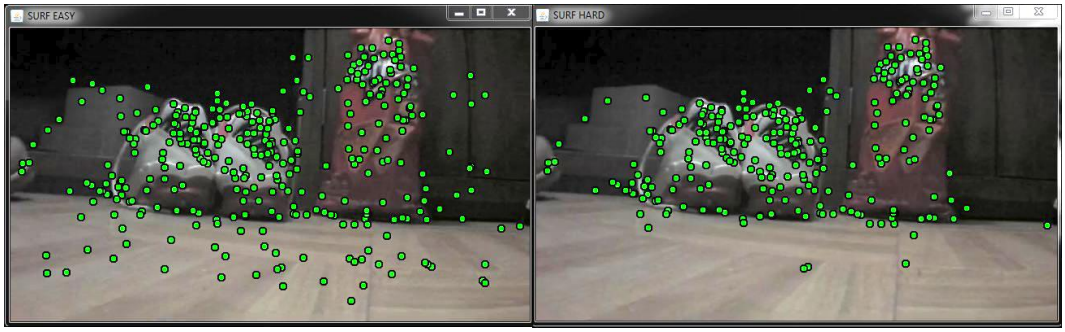

Figura 4.3.1 Comparativo entre SURF puntos de interés con umbral al 0% (izquierda) y umbral 5% (derecha).

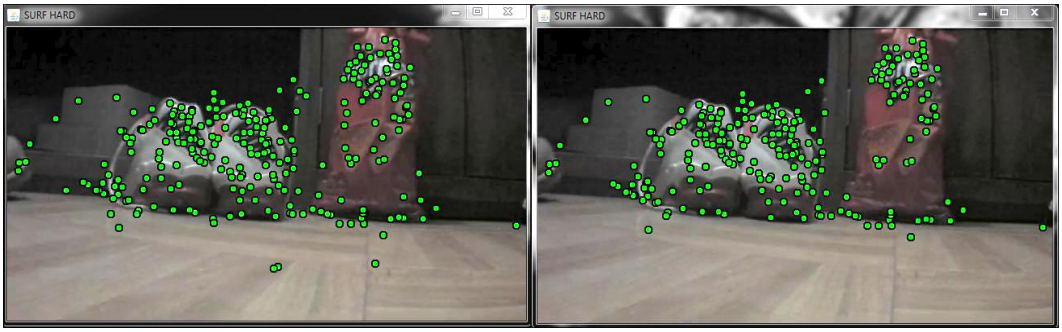

Figura 4.3.2 Comparativo entre SURF puntos de interés con umbral al 5% (izquierda) y umbral 10% (derecha).

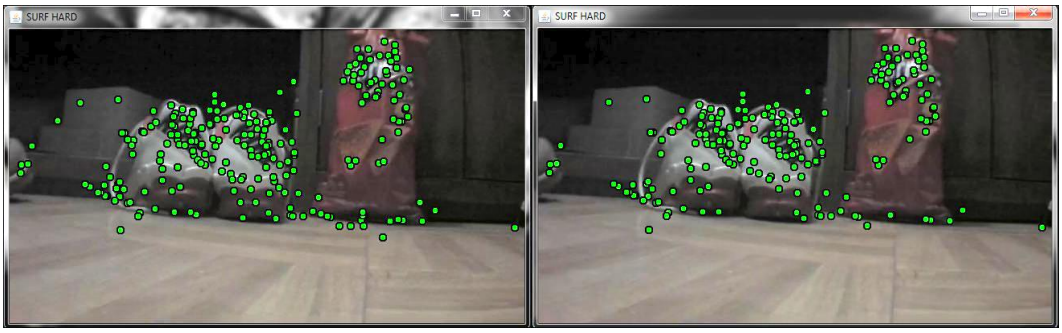

Figura 4.3.3 Comparativo entre SURF puntos de interés con umbral al 10% (izquierda) y umbral 15% (derecha).

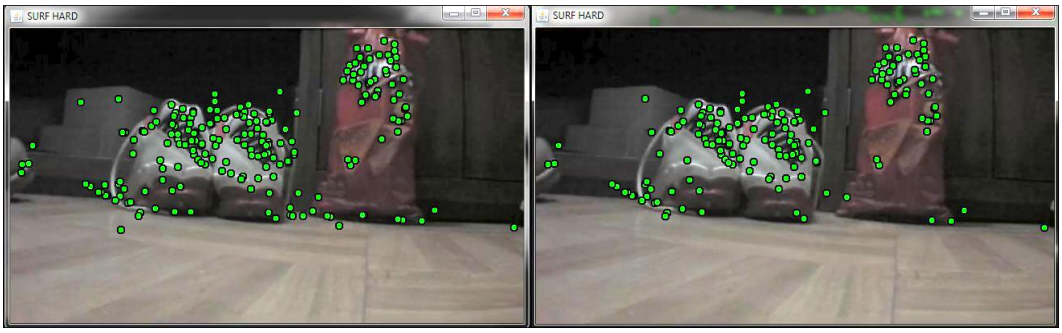

Figura 4.3.4 Comparativo entre SURF puntos de interés con umbral al 15% y umbral 20%.

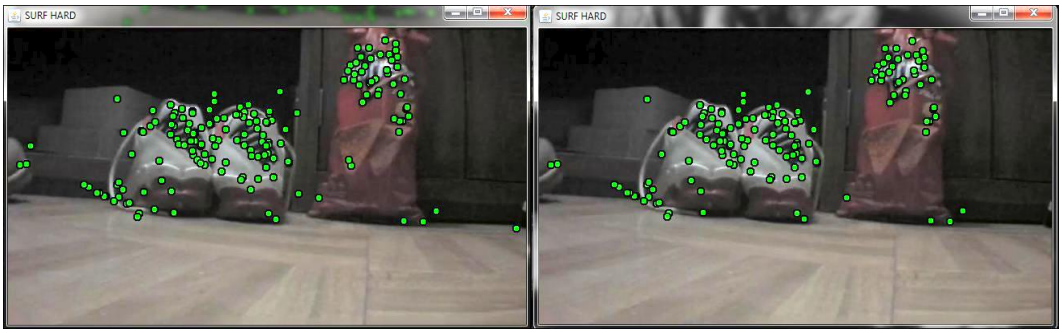

Figura 4.3.5 Comparativo entre SURF puntos de interés con umbral al 20% y umbral 25%.

Como se observa en el comparativo de las imágenes anterior, entre mayor es el valor del umbral, menor es la cantidad de puntos presentado, debido a que los puntos de interés con menor energía son eliminados y solo se preservan aquellos con mayor energía. De este comparativo podemos observar que con un valor de umbral del 10%, se filtran de manera efectiva los puntos de interés espurios (fondo y suelo) y los puntos de interés de las regiones de interés permanecen casi intactos.

4.4 Extracción de descriptores de puntos de interés

En los incisos anteriores se describió la forma en la que los puntos de interés son extraídos y filtrados en base a la implementación del algoritmo SURF. Una vez que se han detectado los puntos de interés, el proceso de extracción de los descriptores de puntos de interés (como se describió en la sección 3.3), se realiza mediante la API BoofCV, por cada punto de interés detectado y filtrado como se describió en la sección 4.3, se obtiene un vector de 64 elementos. Los elementos de este vector son valores reales los cuales son el resultado de aplicar el Wavelet de

Haar a los pixeles en la vecindad del punto de interés detectado, estos valores se encuentran normalizados en el rango de (1,-1). En la siguiente figura se muestra un ejemplo de los vectores descriptores obtenidos:

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18
0.009422821	0.013722781	-0.003134684	0.008443106	0.061836689	0.068143515	0.07261138	0.116531302	-0.047527815	0.070995142	0.078484521	0.109149113	-0.010855733	0.013987086	0.001793931	0.010334143	-0.00488792	0.031769382
-2.87E-05	7.28E-05	2.29E-05	4.96E-05	0.001828519	0.002006216	3.94E-04	8.11E-04	-0.001845195	0.006674823	0.002097799	0.004652203	-0.002992663	0.004155054	0.001285059	0.004220636	0.002774335	0.006643432
-5.44E-04	0.005915111	-1.87E-05	0.00571189	0.07302858	0.104210936	0.010872895	0.076203169	-0.017289545	0.19193292	0.067638706	0.172900495	-0.031031767	0.034262365	0.002189486	0.020449743	-0.001064123	0.006623695
0.018062454	0.047682753	-0.005318265	0.021339668	0.101537453	0.352473024	-0.020478706	0.12727438	-0.117565706	0.194788953	0.052464563	0.088849254	-0.013769826	0.018821278	0.006558501	0.01751635	-3.13E-04	0.013625725
0.006775077	0.010699349	-0.002175546	0.008191793	0.092035174	0.215621529	0.042488032	0.095359204	-0.070151137	0.237980629	0.057513293	0.102676062	-0.006868713	0.012879516	0.005277649	0.012373399	0.011843583	0.016434854
0.181287371	0.183999441	0.032730402	0.132764077	0.017262778	0.032152298	0.012265581	0.109861808	0.00570417	0.017594471	-0.008011155	0.042290215	-0.100888854	0.101404361	-0.032396037	0.058853902	0.122032343	0.137440959
-0.023526225	0.02404192	0.003999038	0.007871217	0.153129127	0.199073303	0.008608569	0.047962628	0.161048034	0.246371968	3.18E-04	0.055576439	-0.108290422	0.17541752	0.018922038	0.056389501	-0.02115778	0.021921181
-0.018330495	0.020260442	-0.001784432	0.003799873	0.187567217	0.234228859	0.067024542	0.07133748	0.071743602	0.258464199	-0.00930242	0.108536794	-0.01220136	0.169631492	-0.027244268	0.039907117	-0.014699642	0.023966388
-0.12364485	0.238667765	0.01403886	0.035609842	-0.077740142	0.086395253	0.001764693	0.015113666	0.002541061	0.006709072	0.001716025	0.008054599	0.001036209	0.005776212	-4.02E-04	0.007566679	-0.174187032	0.239008415
0.002361715	0.009070385	-0.004863943	0.00826111	0.133919572	0.140647553	-0.044383857	0.107887851	0.139681333	0.151432455	0.004154332	0.100847283	-0.047359619	0.063443774	0.026812257	0.04441389	0.003977641	0.008908191
0.003033963	0.005203646	-5.36E-04	0.004710308	0.070207515	0.075660839	0.038634746	0.04152559	0.18880043	0.218243766	0.048527832	0.117391599	0.005788756	0.039002609	-0.050231893	0.060889151	0.004019682	0.007351035
0.056497773	0.079959952	-0.00924637	0.043642467	0.14877079	0.156713612	-0.012367748	0.211697238	-0.035025032	0.102953004	0.036608586	0.126525835	-0.131539957	0.132329779	-0.005126773	0.072155593	-0.012545776	0.038596986
-0.006837834	0.068661414	-0.059199018	0.064319713	0.165153209	0.224229247	-0.002183007	0.0314962	0.021068776	0.192978747	-0.040209275	0.065803238	-0.159818478	0.235754966	0.029262079	0.067301447	3.05E-04	0.007664718
-0.061792908	0.251124547	-0.010910426	0.100414283	0.146577151	0.245154707	0.012318099	0.048280808	-0.00491115	0.137206322	0.011566172	0.039342367	-0.160317445	0.174517463	0.046451952	0.056731236	-0.006808732	0.103407032
-0.012669599	0.014870234	-0.002562414	0.003730106	0.168541651	0.22909896	0.059021084	0.069288	0.081972729	0.247224968	-0.015419538	0.09951223	-0.109241154	0.200840496	-0.010249188	0.029971006	-0.011275678	0.014090296
8.91E-04	0.003599158	0.006603159	0.008626531	0.045101654	0.052928167	0.033736893	0.036056366	0.11431161	0.20192297	0.089428988	0.216439838	0.071081634	0.129488998	0.002482797	0.140014831	-0.002964504	0.007987666

Figura 4.4.1 Ejemplo de vectores descriptores para los puntos de interés detectados.

4.5 Análisis de los puntos de interés y sus descriptores

Como hasta el momento se ha explicado, una de las razones por las cuales se eligió trabajar con la teoría de rasgos descriptores, es que estos son altamente robustos en cuanto a las transformaciones más comunes en imágenes, las cuales son, rotación, iluminación y cambios de escala. Para poder validar lo anterior nos dimos a la tarea de realizar un proceso de prueba y validación de los descriptores de puntos de interés y verificar su confiabilidad. Para esto se realizó un plan de pruebas el cual consiste en:

1) Validar la confiabilidad de los descriptores de puntos de interés.
2) Validar la invariancia a iluminación, rotación y traslación.

4.5.1 Pruebas de confiabilidad de los descriptores de puntos de interés

Para verificar que tan confiables son y según la teoría explicada en la sección 3.2 y 3.3, un conjunto de rasgos descriptores debe de identificar a los objetos pertenecientes a una imagen en diferentes imágenes de la misma escena. Para validar este punto nos dimos a la tarea de construir una interface gráfica Java SWING en la cual el procedimiento de validación de puntos de interés es el siguiente:

1) Como punto inicial se proporciona una imagen de entrada y se calculan tanto los puntos de interés, así como sus descriptores.
2) Como segundo paso, se elige otra imagen de la misma escena, se calculan los puntos de interés así como sus descriptores.
3) Los descriptores de puntos de interés de las dos imágenes se comparan mediante un algoritmo Greedy (elemento por elemento) y se muestran aquellos en los cuales hubo una coincidencia más un rango ε de error.

Los resultados de aplicar los pasos anteriores se muestran en la siguiente secuencia de imágenes, en las cuales se observa que realmente existen coincidencias entre los descriptores de los puntos de interés para diferentes imágenes de la misma escena.

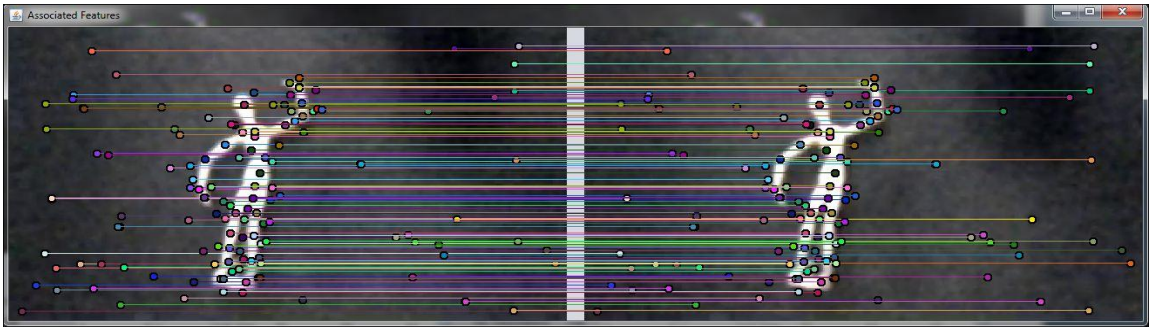

Figura 4.5.1.1 Comparación de descriptores de puntos de interés para la escena de prueba 1.

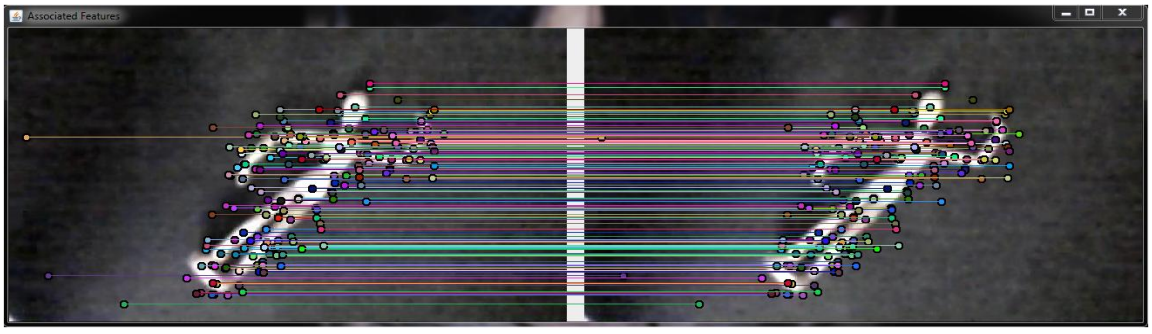

Figura 4.5.1.2 Comparación de descriptores de puntos de interés para la escena de prueba 2.

4.5.2 Pruebas con transformaciones de rotación, escala e iluminación

Con las imágenes de prueba de la sección anterior se observa claramente como existe una correspondencia directa entre los descriptores de dos imágenes similares, sin embargo es necesario verificar que la misma correspondencia se mantenga (en cierto grado) con dos o más imágenes de la misma escena pero con variaciones diversas, para comprobar lo anterior, se utilizó el mismo procedimiento de la sección 4.5.1. Obteniendo los resultados que se muestran en la siguiente secuencia de imágenes:

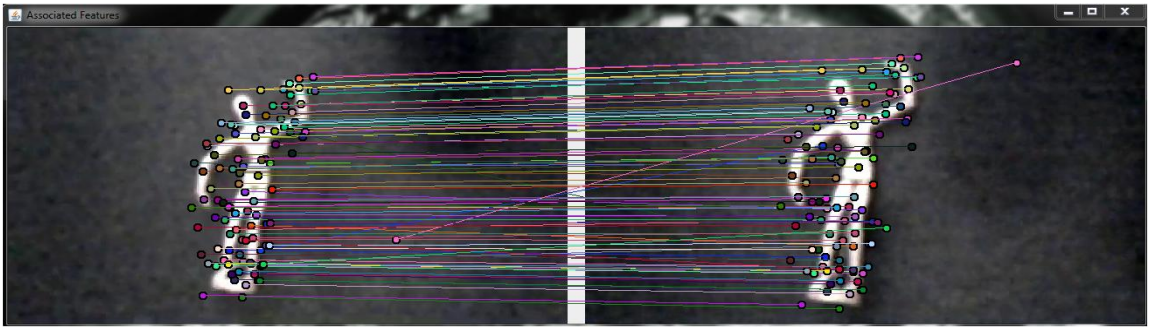

Figura 4.5.2.1 Comparación de descriptores de puntos de interés para la escena de prueba 3 (variación en escala).

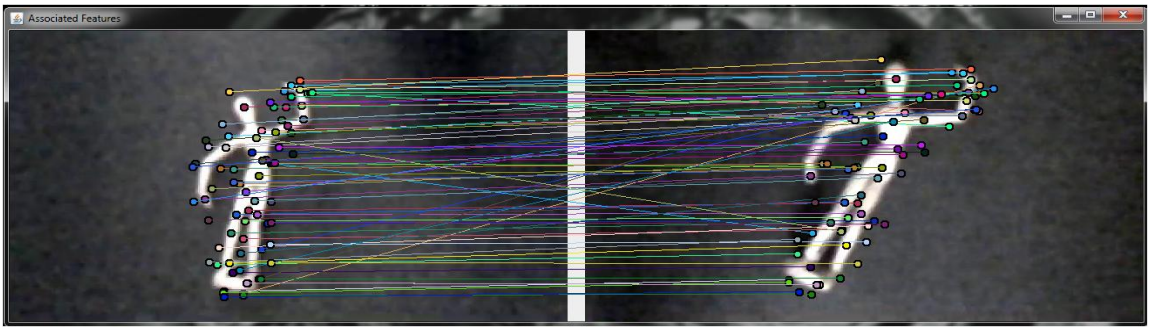

Figura 4.5.2.2 Comparación de descriptores de puntos de interés para la escena de prueba 4 (variación en escala y rotación).

Como se observa del conjunto de imágenes anterior el porcentaje de acierto del algoritmo Greedy se reduce en proporción directa al grado de variación en cuanto a luminosidad, rotación y escala se refiere. Sin embargo, un alto grado de correlación se preserva entre los puntos de interés detectados.

4.6 Clasificación de descriptores de puntos de interés

De la sección 4.1 a la sección 4.4 se explicó el procedimiento que se siguió para la extracción de los puntos de interés y sus descriptores. En la Sección 4.5 se verifico que efectivamente los descriptores son invariantes ante las transformaciones más comunes en imágenes. El conjunto de análisis anterior nos sugiere que estos descriptores pueden ser clasificados de alguna forma. En las siguientes secciones se describe el proceso de clasificación y el estudio comparativo que se realizó para determinar cuáles de los procesos de clasificación (ANN o clasificadores estadísticos) son los más aptos para resolver nuestra problemática.

4.6.1 Adecuación de los descriptores de puntos de interés para su clasificación

Como se puede observar, del estudio de los descriptores de puntos de interés de la sección 4.5, el algoritmo de detección de puntos de interés nos entrega todos los puntos pertenecientes a la imagen, estos incluyen los puntos de interés correspondientes al objeto de estudio y los correspondientes a los objetos de fondo, los cuales no son elementos de nuestro interés, por lo que se requiere de un proceso de selección para poder obtener el conjunto de puntos de interés que realmente representen al objeto de estudio. Para esto se construyó un módulo de detección y filtrado de puntos de interés basado en las características del objeto. Por ejemplo, en la extracción de los puntos de interés presentado en la sección 4.2 y 4.5, las imágenes de los objetos se presentan con el objeto en primer plano y el fondo contrastado en color negro, aun así, con el fondo contrastado el algoritmo de detección nos entrega puntos de interés correspondientes al fondo, con la información de los descriptores puntos de interés detectados también conocemos las coordenadas de los mismos, así, conocidas las coordenadas de los puntos de interés podemos determinar también el color en el que se encuentra dicho punto y de esta forma junto con la información de color tanto del objeto como del fondo, podemos filtrar de forma eficiente y confiable dichos elementos. Con lo anterior se obtiene el conjunto de descriptores de puntos de interés correspondientes al objeto, el cual en secciones posteriores se describe como son sometidos al proceso de clasificación. La GUI construida para este proceso se muestra a continuación en la figura 4.6.1.

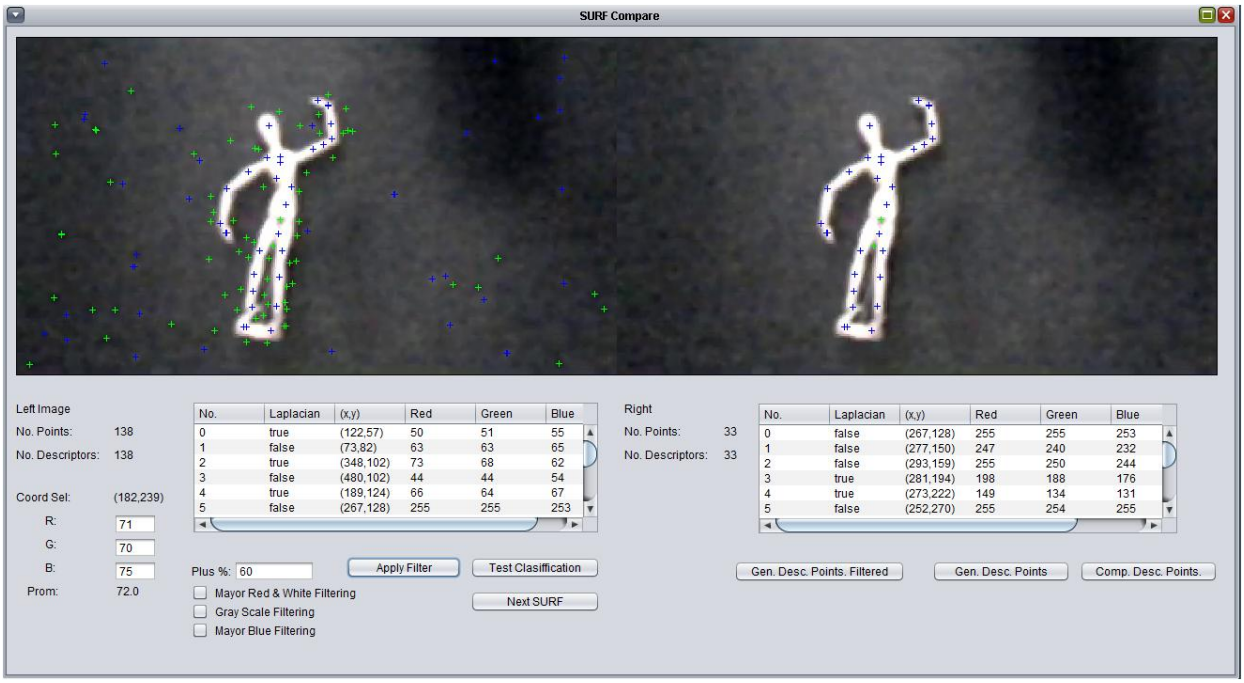

Figura 4.6.1.1 GUI para la extracción y filtrado de puntos de interés de los objetos bajo estudio.

Como se observa, en la GUI, en la parte izquierda se procesa la imagen de entrada y se extraen todos los puntos de interés, los cuales se contabilizan y se muestran en la interface gráfica. La tabla, en la parte inferior de cada imagen, contiene la información de cada punto de interés detectado, como lo es, el número de secuencia en el cual fue detectado, su atributo Laplaciano, el cual si es true, especifica que ese punto de interés es un "manchón" de color blanco o de lo contrario, si es false es un "manchón" de color negro, así como los valores RGB pertenecientes al pixel.

A la derecha de la GUI se muestra el resultado de filtrar los puntos de interés de la imagen con los filtros de la parte inferior izquierda de la interface. Como se observa en la parte inferior derecha el número de puntos filtrados es menor al original y estos puntos en la imagen de la derecha son los puntos pertenecientes al objeto de estudio. Así una vez extraídos y filtrados los puntos de interés por objeto, estos se exportan a un archivo CSV mediante el botón "Gen. Desc. Points" para su posterior clasificación.

4.6.2 Clasificación de conjunto reducido de puntos de interés

Una vez que obtuvimos los puntos de interés por objeto, procedimos a realizar una primera clasificación con un conjunto reducido de elementos, para esto, se seleccionaron los descriptores de puntos de interés correspondientes al maniquí presentado en las secciones 4.4 y 4.5 y se entrenó un conjunto de redes neuronales, así como también un conjunto de clasificadores estadísticos los cuales son:

1) Red Neuronal de Perceptrones Multicapa (MLP).
2) Red Neuronal de Base Radial (RBFN).
3) Máquina de Vector Soporte (SVM).
    a. Kernel de Base Radial.
    b. Kernel Polinomial.
    c. Kernel Lineal.
4) Clasificador Naive Bayes.
5) KNN.
6) Bayes Net.

El primer conjunto de entrenamiento para verificar la separabilidad entre los puntos de interés correspondientes al objeto de estudio (maniquí) como Clase 1 y al fondo como Clase 2, se muestra en la tabla 4.6.2. Para estas pruebas se tomaron muestras a diferentes alturas, esto debido a que la altura del Quadrotor en vuelo no es constante y tiene una variación de ± 10 cm. aproximadamente.

Primer Conjunto de Entrenamiento puntos de interés Maniquí vs Fondo				
Altura cm.	Entrenamiento Clase 1	Entrenamiento Clase 2	Prueba Clase 1	Prueba Clase 2
95	77	70	25	20
100	71	60	39	30
110	60	50	45	52
Todos	208	180	109	102

Tabla 4.6.2.1 Primer conjunto de entrenamiento para verificar la separabilidad de los puntos de interés.

En el capítulo 5 se muestra la tabla y gráfica comparativa de resultados de la clasificación con el conjunto de muestras de la tabla 4.6.2, en el cual se observa que los clasificadores que mejor resultado dan, para este conjunto de entrenamiento reducido, son los clasificadores MLP y KNN.

El clasificador MLP nos otorga un porcentaje de clasificación del 97.1831%, mientras que el clasificador KNN nos da un porcentaje de clasificación del 97.6526%.

Sin embargo, como se muestra en la sección de resultados, estos porcentajes de clasificación se obtienen de juntar todas las muestras tomadas a diferentes alturas, esto nos sugirió, que al tener un conjunto más amplio de muestras se lograría una clasificación más confiable. Por lo que se generó un segundo conjunto de entrenamiento de puntos de interés para la Clase 1 y la Clase 2 anteriormente mencionadas con un mayor número de muestras (ver tabla 4.6.2.2), el cual consiste en ampliar el número de muestras tomado para cada altura en la tabla 4.6.2 y añadir muestras de imágenes tomadas en lo que llamaremos Vuelo en Tiempo Real (VTR), el cual consiste en poner al Quadrotor en vuelo estacionario a una altura aproximada de 100 cm. del suelo y realizar la captura de imágenes del objeto de estudio con el fondo contrastado, así obtenemos el siguiente conjunto de entrenamiento.

Segundo Conjunto de Entrenamiento puntos de interés Maniquí vs Fondo				
Altura cm.	Entrenamiento Clase 1	Entrenamiento Clase 2	Prueba Clase 1	Prueba Clase 2
95	1648	1352	150	212
100	3694	2052	200	232
110	2630	2335	160	152
VTP	8125	4407	685	346
Todos	16097	10146	1195	942

Tabla 4.6.2.2 Segundo conjunto de entrenamiento para verificar la separabilidad de los puntos de interés.

La tabla comparativa de la clasificación con diferentes redes neuronales y clasificadores estadísticos se muestra en el capítulo de resultados, donde se observa que con este último conjunto de entrenamiento los mejores porcentajes de clasificación corresponden de nuevo a la red neuronal MLP y al clasificador KNN. Debido a que uno de los objetivos de la presente tesis es utilizar redes neuronales como herramienta de clasificación, de este punto en adelante, se utilizarán las redes neuronales MLP como elemento base de clasificación y los demás clasificadores (redes neuronales y clasificadores estadísticos), serán mencionados como elementos de comparación en cuanto a eficiencia de clasificación y rendimiento.

4.6.3 Validación grafica de la clasificación mediante MLP

Como en toda red neuronal, una vez que esta ha sido entrenada mediante un conjunto de datos de entrenamiento y validado su porcentaje de clasificación mediante datos de prueba, se requiere realizar una etapa de validación de porcentaje de clasificación con datos reales, debido a que estos últimos por naturaleza suelen ser datos con mayor ruido que los datos de prueba con los cuales se entrenó y valido la red neuronal. Para esto se diseñó una etapa de prueba, la cual consiste en un conjunto de 50 imágenes del objeto bajo estudio, las cuales fueron tomadas con la cámara vertical del Quadrotor en vuelo estacionario a una altura aproximada de 100 cm del suelo (vuelo en tiempo real). De esta forma se contabilizaron (manualmente) los puntos de interés clasificados por la red neuronal MLP como elementos pertenecientes al objeto y los puntos de interés como elementos pertenecientes al fondo. Obteniendo una clara diferencia (de aproximadamente 5 puntos porcentuales) entre el porcentaje de clasificación del conjunto de entrenamiento y el conjunto de muestras tomado de las imágenes en vuelo. Las gráficas, tablas de porcentajes e imágenes que muestran el porcentaje de clasificación real se presentan en la sección de resultados.

4.7 Extensión del conjunto de objetos a clasificar

En la sección 4.6.2 y 4.6.3 se explicó la metodología para la clasificación del primer objeto de estudio, el cual por sus características se logró obtener un 95.6036% de clasificación real. Para ampliar este estudio y validar de una forma más robusta estos resultados, se eligió un conjunto de objetos mayor, cuyas características variaran de un objeto a otro tanto en color, forma y tamaño, así, a estos objetos los llamaremos conjunto extendido de objetos, los cuales se muestran a continuación:

El Objeto 1 como ya se ha mostrado en secciones anteriores, es un objeto de tipo humanoide, de color café claro uniforme y sin variaciones en el color ni textura, este se muestra en las imágenes de la figura 4.7.

El Objeto 2 se muestra en las siguientes imágenes, el cual como se observa posee un color predominante rojo, con algunos detalles en color plateado y cuyas medidas se muestran en la tabla 4.7.1.

Figura 4.7.1 Conjunto de imágenes del objeto de estudio “Objeto 1”.

Figura 4.7.2 Conjunto de imágenes del objeto de estudio “Objeto 2”.

El Objeto 3 se muestra en las imágenes de la figura 4.7.2, el cual se observa posee un color predominante blanco, con algunos detalles en color negro.

Figura 4.7.3 Conjunto de imágenes del objeto de estudio “Objeto 3”.

El Objeto 4 se muestra en las imágenes de la figura 4.7.3, el cual se observa posee un color predominante verde oscuro, con algunos detalles en color negro y color plateado, se observa que tanto las texturas como contornos son completamente deferentes.

Figura 4.7.4 Conjunto de imágenes del objeto de estudio "Objeto 4".

Tabla comparativa de los objetos de estudio			
Objeto	Ancho (cm)	Largo (cm)	Color Principal
Objeto 1	7	22	Beige
Objeto 2	9	18	Rojo
Objeto 3	9	22	Blanco
Objeto 4	8	14	Verde Oscuro

Tabla 4.7.1 Tabla comparativa por tamaño y color de los objetos de estudio.

De igual forma que en el caso del Objeto 1, se debe de implementar una metodología para la extracción de los descriptores de puntos de interés para los objetos de nuestra base de datos extendida, de tal forma que se obtenga un conjunto de descriptores para cada objeto y dicho conjunto de descriptores identifique de manera única a un objeto. Lo anterior se explica en la siguiente sección.

4.8 Adecuación de descriptores puntos de interés del conjunto extendido

Una vez que se ha definido el conjunto de objetos a clasificar y como se mencionó en la sección anterior, el siguiente paso es la extracción de descriptores de puntos de interés por objeto, para esto se modificó la interface gráfica presentada en la figura 4.6.1, con la cual se obtuvieron los siguientes resultados en cuanto al filtrado de los descriptores de puntos de interés.

De igual forma que como se explicó en la sección 4.6.1, los puntos de interés filtrados aparecen en la imagen de la derecha de la figura 4.8, en la cual se observa claramente como solo se muestran los puntos de interés que se encuentran sobre el Objeto 2, cuyos puntos son lo que se utilizarán para realizar la clasificación de dicho objeto.

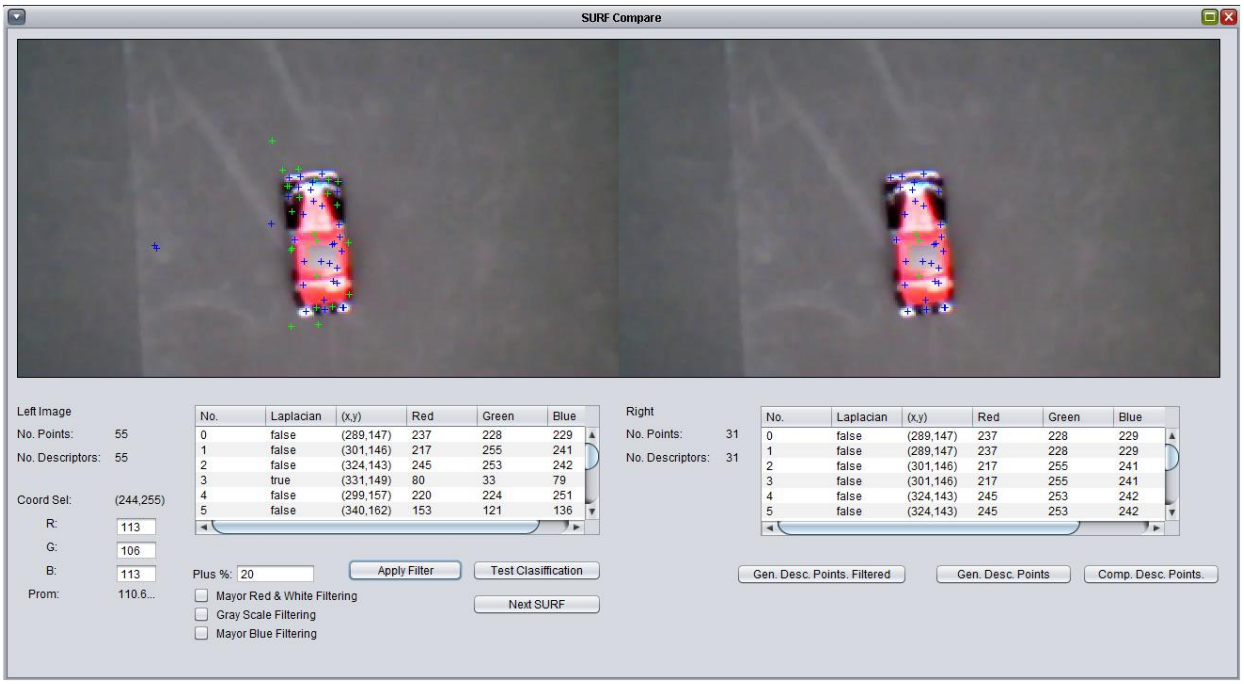

Figura 4.8.1 Filtrado de descriptores de puntos de interés para el "Objeto 2".

Para el Objeto 3 y 4 se procede de la misma forma que para los objetos anteriores, cuyo resultado final del filtrado por objeto se muestra en las siguientes figuras 4.8.2 y 4.8.3.

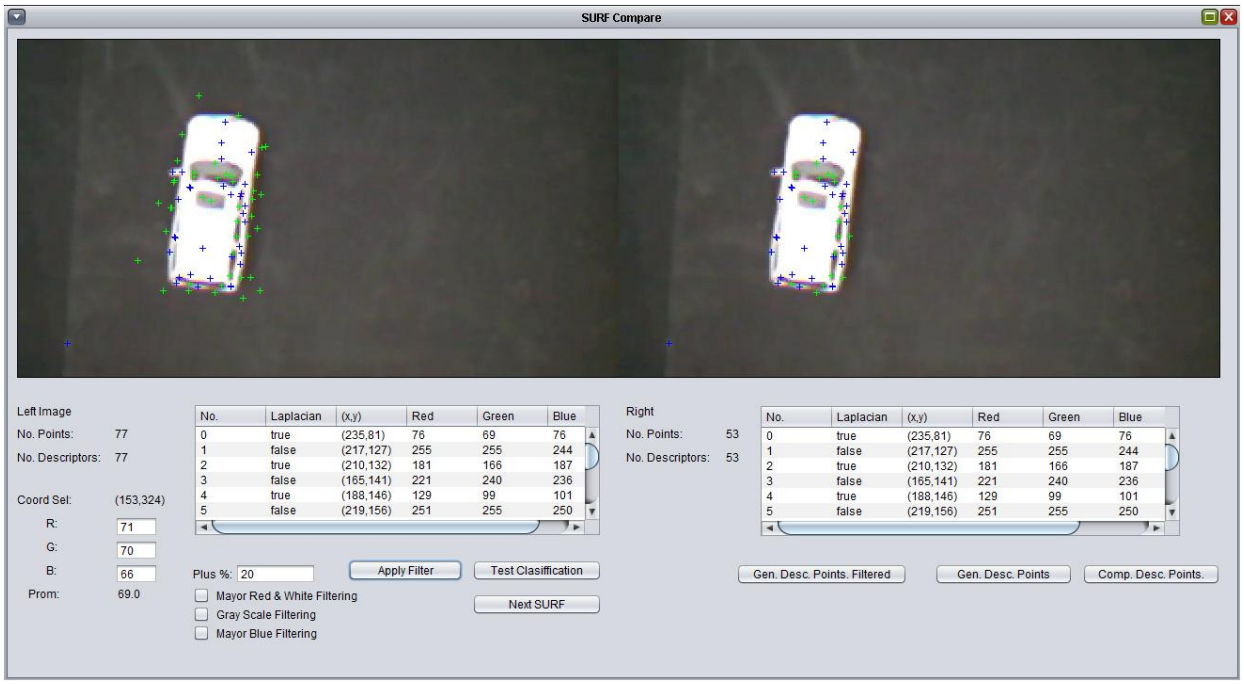

Figura 4.8.2 Filtrado de descriptores de puntos de interés para el "Objeto 3".

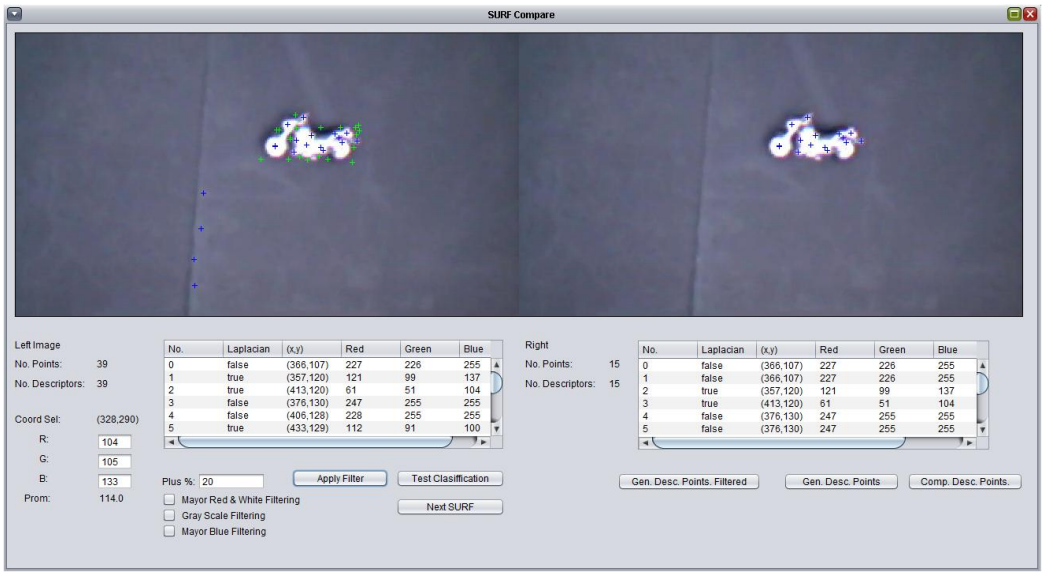

Figura 4.8.3 Filtrado de descriptores de puntos de interés para el "Objeto 2".

Una vez que se obtuvieron los puntos de interés filtrados por objetos, el siguiente paso es realizar la clasificación de dichos elementos y verificar la separación espacial de los descriptores de puntos de interés, cuya característica influye directamente en el porcentaje de clasificación de las redes neuronales.

4.9 Clasificación del conjunto extendido de objetos

Una vez que se extrajo el conjunto representativo de descriptores de puntos de interés por objeto, procedemos a clasificar los objetos mediante el conjunto de redes neuronales y clasificadores estadísticos con el fin de verificar hasta qué punto los datos son lineal mente separables y en qué grado los porcentajes de clasificación se ven afectados al introducir nuevas clases para su clasificación a la red neuronal.

Pero antes de realizar el comparativo, verificamos hasta qué punto los descriptores de puntos de interés son clasificables, es decir, para cada conjunto de puntos de interés por objeto se entrena una red neuronal MLP, el conjunto de descriptores de puntos de interés generado se muestra en la siguiente tabla.

Objeto	No. de puntos de interés de Entrenamiento	No. de puntos de interés de Prueba	No. de imágenes de prueba en vuelo estacionario
Objeto 1	11908	1117	50
Objeto 2	10908	738	50
Objeto 3	13077	1132	50
Objeto 4	4599	761	50

Tabla 4.9.1 Tabla de número de descriptores de puntos de interés generados por objeto.

La tabla de porcentajes de clasificación para la red neuronal MLP correspondiente a la tabla de datos de entrenamiento 4.9 se muestra en la sección de resultados.

El procedimiento de pruebas para validar la separabilidad de los descriptores de puntos de interés por objeto es el siguiente, entrenar el conjunto de redes neuronales para los objetos como se muestra en la siguiente matriz de pruebas:

Matriz de prueba de combinación de puntos de interés objeto					
No. Prueba	Objeto Referencia	Objeto 1	Objeto 2	Objeto 3	Objeto 4
1	Fondo	X			
2	Fondo	X	X		
3	Fondo	X	X	X	
4	Fondo	X	X	X	X

Tabla 4.9.2 Matriz de prueba para clasificación de objetos.

El conjunto de puntos de interés de entrenamiento es el mismo que se muestra en la tabla 4.9, pero con la variación que en cada prueba se añaden los descriptores de los objetos extra al conjunto de entrenamiento y al conjunto de pruebas. Los resultados de las clasificaciones de las redes neuronales MLP para cada elemento de la matriz de pruebas se muestran en la sección de resultados.

4.10 Filtrado de puntos de interés por desviación estándar post clasificación

Una vez que los puntos de interés que representan a los objetos han sido clasificados mediante la red neuronal MLP, se observó, que existían puntos de interés los cuales habían sido clasificados de forma errónea, estos puntos de interés a los cuales llamaremos puntos de interés espurios, aparecen como puntos aislados en la periferia de los objetos, como elementos del fondo y en ocasiones como el resultado de una clasificación errónea por parte de la red neuronal debido a la naturaleza ruidosa de la imagen. Por lo que analizando la naturaleza de dichos elementos, los cuales como se observan en las secciones 4.6.1 y 4.8, los puntos de interés detectados por objeto

tienen a generar cúmulos de puntos los cuales se encuentran en una distribución espacial cercana, y obviamente estos puntos de interés se encuentran sobre el objeto, no sobre la periferia del mismo o sobre el fondo.

Lo anterior nos sugirió que un procesamiento espacial de los puntos mediante un algoritmo que midiera que tan cercanos o separados se encuentran los elementos del conjunto de estudio, como lo es la desviación estándar la cual podría ayudarnos a filtrar los puntos de interés espurios. De tal forma que se aplicó el algoritmo de desviación estándar para cada uno de los objetos. Cabe mencionar que los valores de la desviación estándar para cada uno de los objetos no es constante, esto debido a que al número de puntos de interés detectado para cada objeto varía de imagen en imagen, aun cuando estas sean consecutivas, por lo que el cálculo de los mismos y su ponderación se realizan en tiempo de ejecución, por lo que de nuevo en la implementación de dicho algoritmo se tuvo el cuidado de que fuera lo más óptima posible.

Los resultados y figuras comparativas del proceso de clasificación con y sin filtrado por desviación estándar se muestran en el capítulo de resultados.

4.11 Análisis temporal de puntos de interés post clasificación

Una vez realizado el proceso de filtrado por desviación estándar como se mencionó en la sección anterior, se observa que realmente los puntos de interés espurios se eliminan en un 80% (ver el capítulo 5 de resultados), sin embargo, el 20% restante de los elementos espurios que no son eliminados poseen un comportamiento peculiar, en el sentido de que estos suelen presentarse de forma esporádica en las imágenes, por lo que se implementó una metodología de análisis histórico temporal. La cual consiste en analizar la presencia de un conjunto de puntos de interés previamente identificados como pertenecientes a una clase y filtrados mediante un filtrado por desviación estándar, los puntos resultantes de este proceso son contabilizados en una secuencia de imágenes en el cual se analiza si dicho conjunto de puntos de interés se presenta en al menos un 25% de las imágenes capturadas en un segundo, esto resulta en el análisis histórico de 3 a 4 imágenes consecutivas, en las cuales si el conjunto de puntos de interés que identifican a un objeto en particular se encuentran presentes entonces se podrá afirmar con cierto grado de certidumbre que

el objeto realmente se encuentra presente en la escena representada por las imágenes tomadas por el Quadrotor.

4.12 Flujo de Datos

Una vez presentados todos los elementos que comprenden la lógica del procesamiento de la información. En el siguiente diagrama de flujo, mostrado en la figura 4.12.1, se presenta su interacción, desde el momento en que se captura la imagen hasta que se emite un resultado:

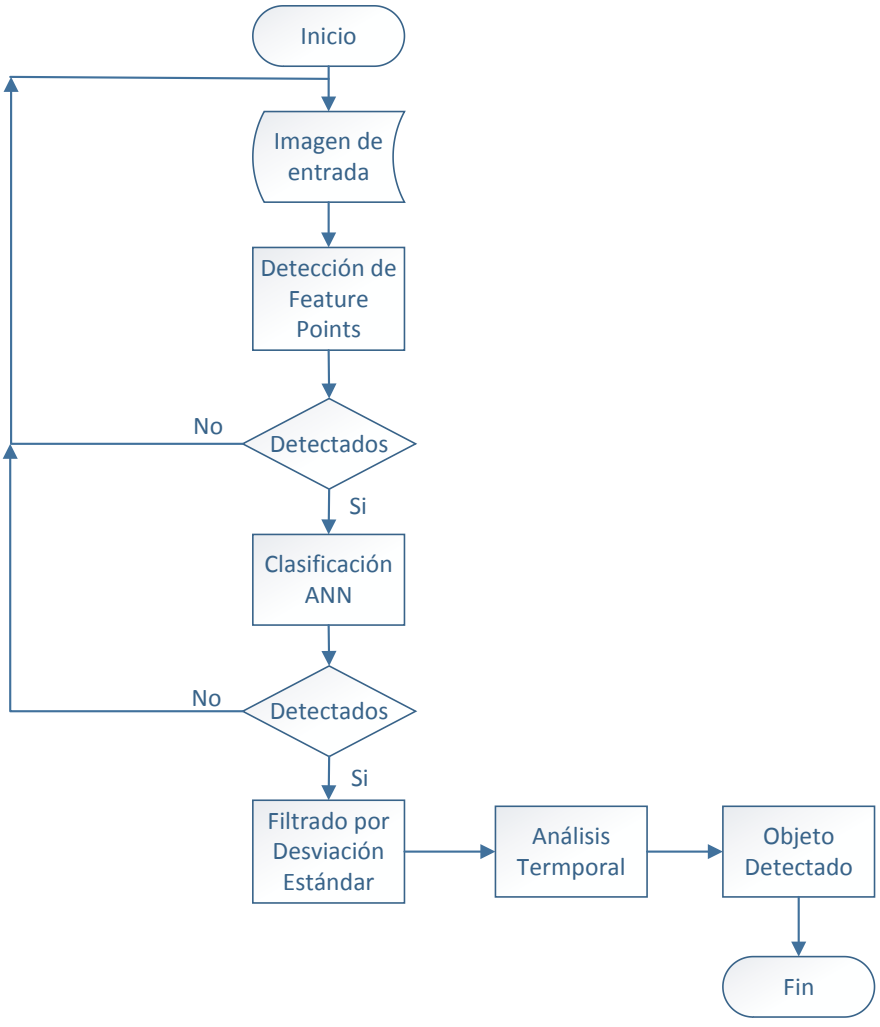

Figura 4.12.1 Diagrama de flujo del procesamiento y clasificación de puntos de interés.

El diagrama anterior como cualquier diagrama de flujo de datos es una abstracción del proceso general, en el cual solo se muestran los elementos fundamentales del procesamiento de

información, sin profundizar en los detalles de la implementación, para proporcionar un panorama mayor acerca de los detalles de la implementación se presentan en la siguiente sección los diagramas de arquitectura de la aplicación tanto del Front End como del Back End.

4.13 Arquitectura propuesta de la aplicación

Una vez identificados todos los elementos necesarios para realizar tanto la extracción de puntos de interés, su clasificación mediante una red neuronal MLP previamente entrenada, el control del Quadrotor mediante tramas UDP y el procesamiento del video en tiempo real. La arquitectura de la aplicación propuesta se muestra en la figura 4.13 y 4.13.1. En la primera se observa el Front End de la aplicación, es decir, los elementos sobre los cuales el usuario interactúa de forma directa y en la figura 4.13.1 los elementos del Back End encargados de la lógica de clasificación y procesamiento de la información.

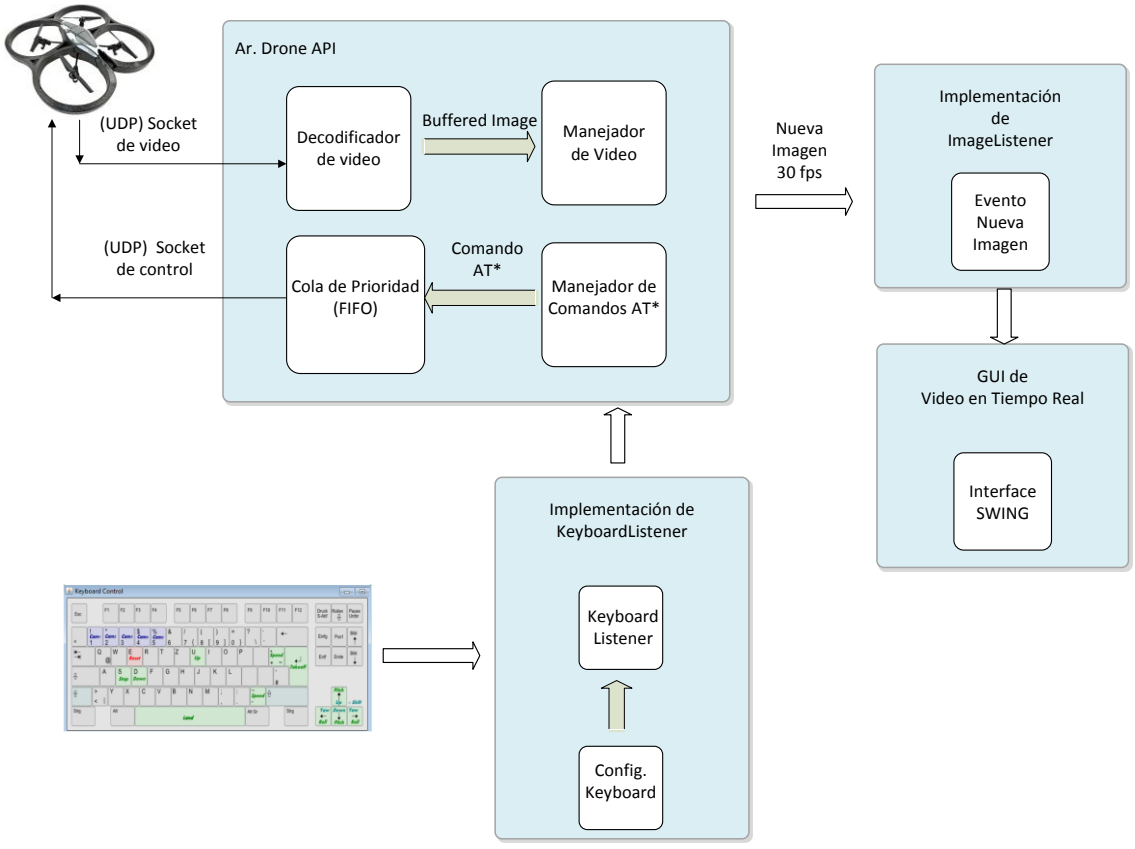

Figura 4.13.1 Arquitectura de la aplicación Front End.

Como ya se explicó en secciones anteriores, el módulo Ar. Drone API en la figura 4.13, es el módulo encargado de establecer y gestionar la comunicación mediante sockets con el Ar. Drone, la implementación en este módulo consiste en instanciar los módulos Manejador de Video y el Manejador de Comandos AT de la API, e implementar Listeners personalizados tanto para el manejo de las imágenes así como para el manejo de los comandos de control. Como se aprecia en la figura anterior, una vez que la implementación del ImageListener recibe una nueva imagen proveniente del Quadrotor, el Listener se encarga de indicar a la GUI que actualice la vista del usuario y presente en pantalla la nueva imagen recibida. Con lo anterior se presentan las imágenes adquiridas por el Quadrotor al usuario con un retraso de aproximadamente 100 a 200 ms. En cuanto a la implementación del KeyboardListener, este se presenta al usuario en forma de una imagen de un teclado, como se muestra en la figura 4.13, el cual es una representación gráfica de la interface que monitorea sí se presiona alguna tecla, una vez que se identifique que se presiona una tecla que represente un comando en particular, el KeyboardListener transmitirá este comando al Quadrotor mediante el socket de comunicación previamente establecido, el tiempo en el que se transmita el comando es directamente proporcional al tiempo en que la tecla permanezca oprimida.

En cuanto a los módulos que componen el Back End, estos se muestran en la figura 4.13.2. Como se observa en el diagrama anterior, el módulo Ar. Drone API aparece en los dos diagramas de la arquitectura, tanto en el diagrama del Front End como en el diagrama del Back End, esto es debido, a que este módulo es un punto medular en la arquitectura, es decir, este módulo es el punto de entrada para la adquisición de las imágenes provenientes del Quadrotor y de salida, ya que los comandos AT* son enviados por medio de este módulo. Una vez que una nueva imagen es obtenida por el módulo base Ar. Drone API, las imágenes en objeto Java BufferedImage son encoladas en una cola de prioridad tipo First In First Out (FIFO), esto debido a que las imágenes deben de ser procesadas en el orden de captura. Por otro lado al iniciar la aplicación se instancia un Thread con menor prioridad que los Thread que administran los módulos de comunicación con el Quadrotor, el cual en el diagrama se identifica como Thread Asíncrono Consumidor.

El módulo Thread Asíncrono Consumidor es un proceso, como su nombre lo indica, asíncrono el cual es el encargado de realizar toda la lógica de procesamiento sobre las imágenes. Como primer paso éste obtiene el primer elemento de la cola de prioridad FIFO, el cual es un objeto de tipo BufferedImage y este contiene toda la información de la imagen así como el valor de los pixeles

pertenecientes a esta. Una vez obtenida la representación discreta de la imagen en un objeto, se procede a realizar el procesamiento de detección de puntos de interés y extracción de descriptores de los puntos de interés, los cuales son representados por 2 arreglos lógicos de longitud variable, ya que por cada imagen que se procese, se obtendrá un número n diferente de puntos de interés.

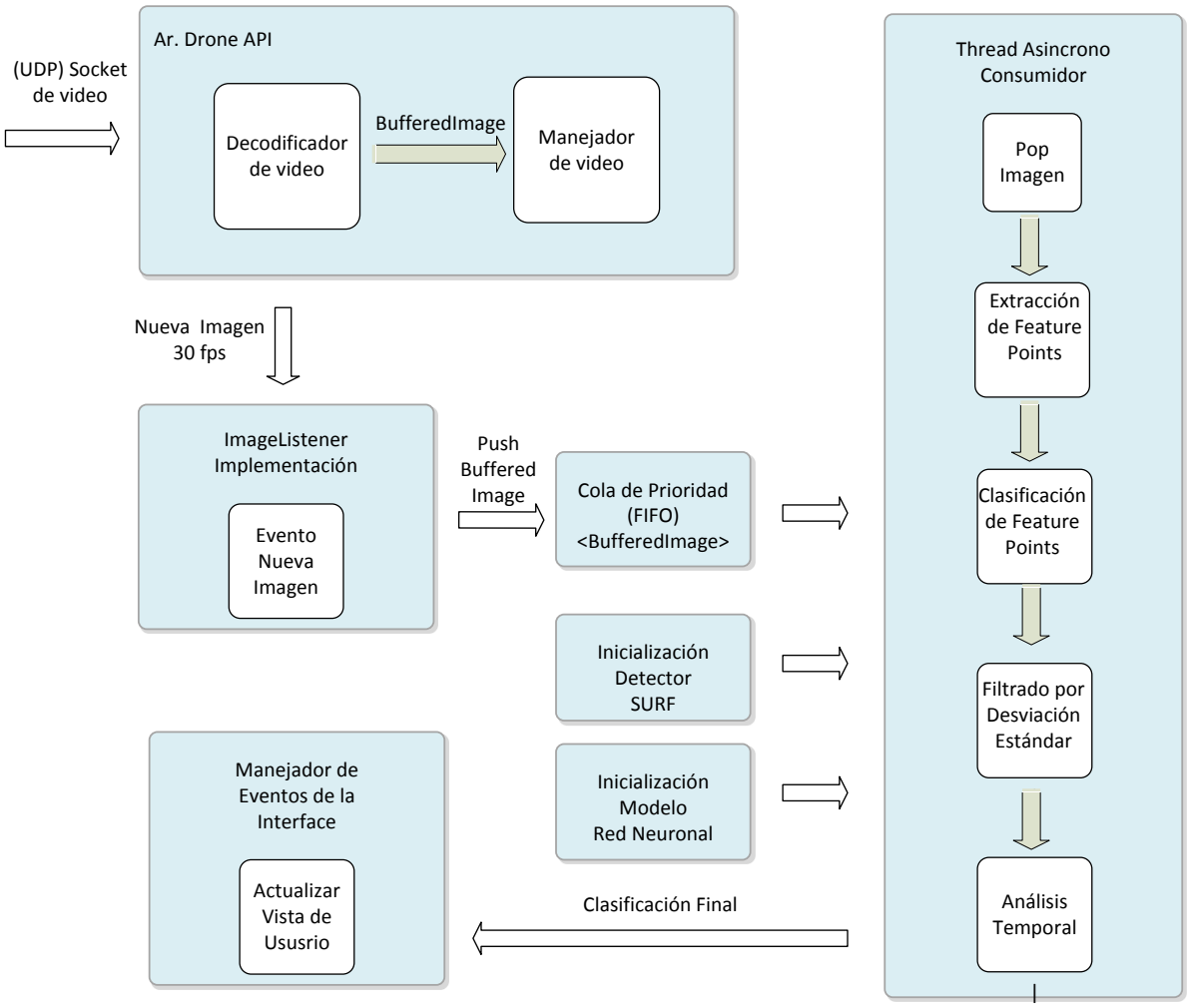

Figura 4.13.2 Arquitectura de la aplicación Back End.

Estos arreglos de puntos de interés y de descriptores poseen el mismo número de elementos, con la salvedad de que el arreglo de puntos de interés contiene solo las coordenadas x, y del punto de interés sobre la imagen, mientras que el arreglo de descriptores de puntos de interés posee en cada elemento del arreglo, otro arreglo de 64 elementos del tipo de dato double, el cual es el descriptor

del punto de interés. Estos últimos 64 elementos son los que se procesan en el módulo de Clasificación de puntos de interés, y sirven como punto de entrada para la red neuronal artificial MLP previamente entrenada, de esta manera cada uno de los descriptores de puntos de interés detectados en la imagen son clasificados por la red neuronal y el resultado de cada uno de ellos de nueva cuenta es almacenado en un arreglo de longitud variable. El arreglo con el resultado de la clasificación de cada uno de los puntos de interés detectados sobre la imagen, es procesado en la siguiente etapa de procesamiento Filtrado de puntos de interés por Desviación Estándar, debido a las razones que se explicaron en la sección 4.10.

Una vez que se completaron todos los pasos anteriores del procesamiento de la información y analizado los resultados del proceso completo, se observó que aún se presentaban errores en las clasificaciones, esto debido a la detección de puntos de interés espurios, los cuales se observó que aparecían solo en ciertas imágenes y de forma aleatoria y sin tener un precedente, por lo que se implementó un análisis histórico temporal, el cual consiste en llevar un registro de si los puntos de interés detectados para un objeto en específico se habían detectado en por lo menos una secuencia de 3 imágenes consecutivas, de ser así, se reportaba que el objeto realmente se ha detectado y no se reporta un falso positivo.

Una vez que los puntos de interés pasaron todos los filtros anteriores y se les asigno una clase de pertenencia relacionada a un objeto, se procede a reportarlos en la interface gráfica la cual se explica posteriormente en este capítulo.

4.14 Planteamiento de la ruta de vuelo

Hasta este punto, ya explicamos tanto los algoritmos aplicadas a la identificación de objetos, así como los aspectos a tomar en cuenta para el procesamiento en tiempo real de las imágenes y el control del Quadrotor. Sin embargo, no hemos especificado el cómo se hará la detección de objetos mediante la cámara del Quadrotor.

Para detallar la ruta del vuelo, primero se plantean las siguientes situaciones en piso, las cuales prueban la robustez del algoritmo desarrollado e implementado.

Como primer escenario, como se muestra en la Figura 4.14.1,al cual llamaremos E1, se plantea simplemente el Objeto Fondo el cual es el escenario más simple. En este escenario de pruebas el algoritmo de identificación no deberá de reportar la identificación de un objeto.

Como segundo escenario, al cual llamaremos E2, se plantea que el Objeto 1 se presente junto con el Objeto Fondo. Al analizar este escenario se tomaran muestras estadísticas las cuales serán presentadas en el capítulo de resultados.

Como tercer escenario, al cual llamaremos E3, se plantea que el Objeto 2 se presente junto con el Objeto Fondo. Al analizar este escenario se tomaran muestras estadísticas las cuales serán presentadas en el capítulo de resultados.

Como cuarto escenario, al cual llamaremos E4, se plantea que el Objeto 3 y Objeto 4 se presente junto con el Objeto Fondo. Al analizar este escenario se tomaran muestras estadísticas las cuales serán presentadas en el capítulo de resultados.

Como quinto escenario, al cual llamaremos E5, se plantea que los objetos Objeto 1, Objeto 2, Objeto 3 y Objeto 4 se presenten con el Objeto Fondo. Al analizar este escenario se tomaran muestras estadísticas las cuales serán presentadas en el capítulo de resultados.

En la misma ruta de vuelo se planea probar más de un escenario, esto se ejemplifica en la siguiente figura.

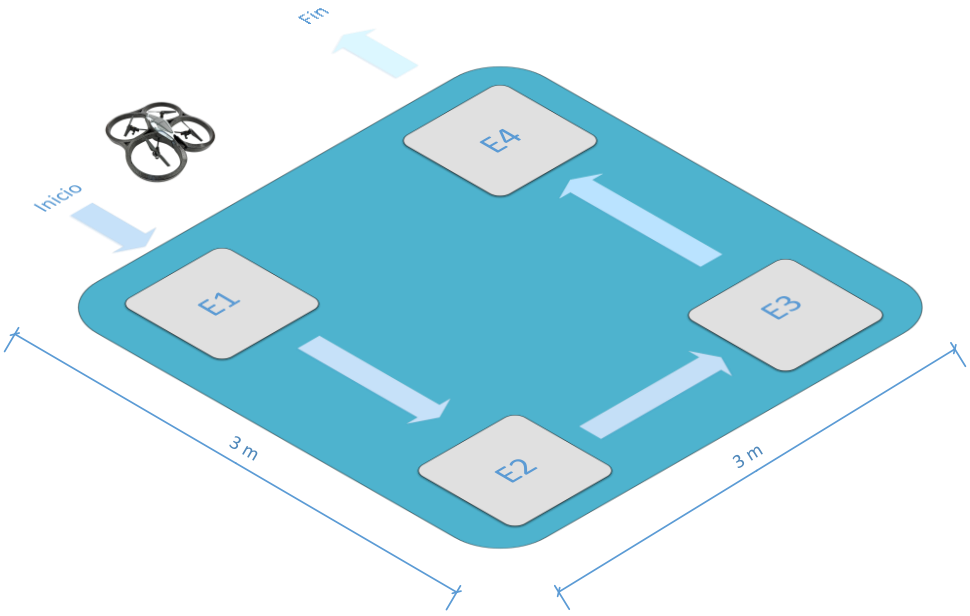

Figura 4.14.1 Ruta de vuelo propuesta para el Quadrotor.

Como se observa en la Figura 4.1.4.1 la ruta de vuelo consiste de un cuadrado con un área aproximada de 3 metros por lado. Al inicio el Quadrotor en el suelo despegará desde la posición del escenario E1, sobre la cual permanecerá un periodo de tiempo de entre 3 a 5 segundos, posteriormente se desplazará en vuelo hasta la posición del escenario E2, de igual forma en esta posición el Quadrotor permanecerá un periodo de tiempo de entre 3 a 5 segundos. Una vez que el Quadrotor se posicione sobre el escenario E2, este se desplazará hacia la posición del escenario E3 y de igual forma permanecerá suspendido un periodo de tiempo de entre 3 a 5 segundos y como etapa final el Quadrotor se desplazara en vuelo hacia la posición del escenario E4 permaneciendo un periodo de tiempo de entre 3 a 5 segundos sobre el escenario E4. En el periodo de tiempo en el cual el Quadrotor permanece sobre alguna de las escenas, este último procesa la escena con los algoritmos anteriormente descritos y los resultados se muestran en la sección final del siguiente capítulo.

Capítulo 5
Experimentos y Resultados

En el presente capítulo se presentan y discuten los resultados obtenidos en cada una de las etapas de experimentación descritas en el capítulo anterior. Cabe mencionar que los puntos que se discuten en este capítulo son progresivos, en el sentido de que los experimentos se fueron evolucionando y cada experimento depende de los resultados del experimento anterior.

5.1 Resultados de clasificación del primer conjunto de entrenamiento

Como se mencionó en la sección 4.6.2, para verificar los resultados de la clasificación del primer conjunto de entrenamiento de descriptores de puntos de interés se entrenó un conjunto de redes neuronales y de clasificadores estadísticos con base en los datos de la tabla 4.6.2, con los cuales se obtuvieron los porcentajes de clasificación que se muestran en la siguiente gráfica.

Como se observa en la gráfica comparativa de porcentajes de clasificación de la Figura 5.1.1, los porcentajes más altos de clasificación se obtienen con los clasificadores MLP, Bayes Net y KNN. Sin embargo, el número de muestras que se tomó para realizar este estudio comparativo es pequeño, apenas de algunos cientos de muestras por distancia medida (ver tabla 4.6.2). Lo anterior nos demuestra que efectivamente los descriptores de puntos de interés pueden ser clasificados de forma efectiva. Sin embargo se requiere ampliar el conjunto de entrenamiento, lo anterior se explica en la siguiente sección.

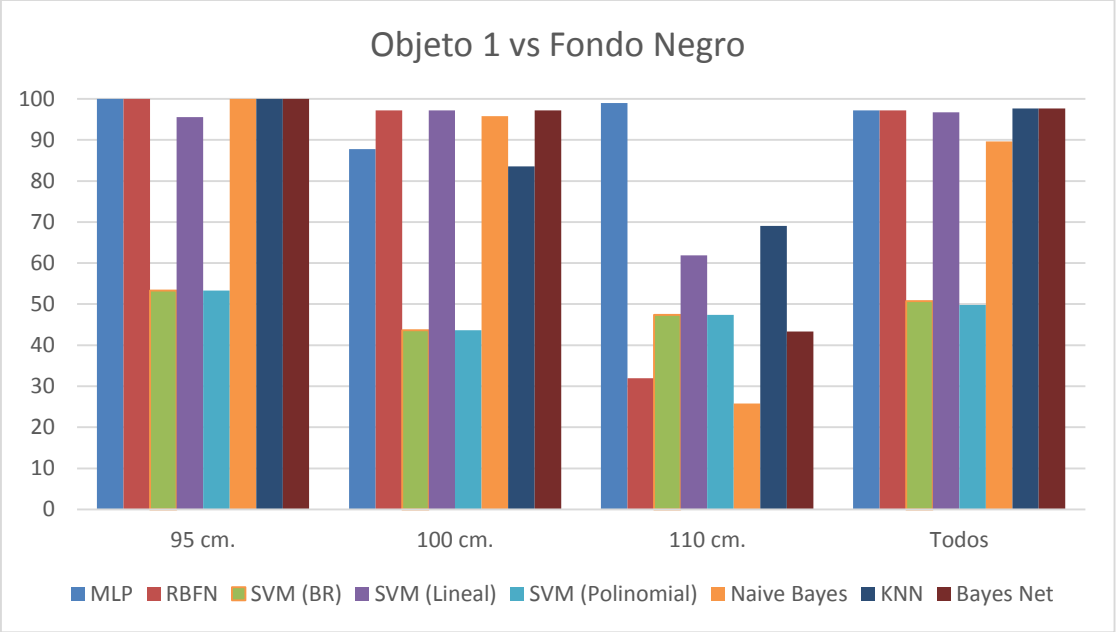

Figura 5.1.1 Grafica comparativa del porcentaje de clasificación entre diferentes técnicas.

5.2 Resultados de clasificación del conjunto extendido de entrenamiento

En la tabla 4.6.2.1 se muestra como se compone el segundo conjunto de entrenamiento, extendiendo el número de muestras por distancia de medición a varios miles como se explicó en la sección 4.6. Los resultados de la clasificación para este conjunto se muestran en la siguiente gráfica:

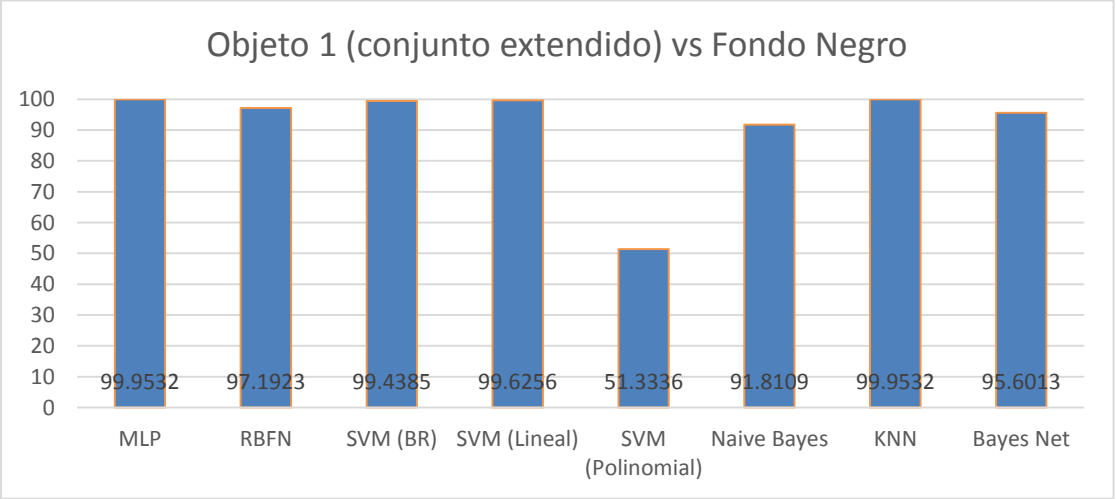

Figura 5.2.1 Grafica comparativa del porcentaje de clasificación para el segundo conjunto de datos de entrenamiento.

Como se observa en la gráfica anterior, de nueva cuenta los clasificadores que nos dan mejores resultados son la red neuronal MLP y el clasificador KNN, cuyos porcentajes de clasificación son cercanos al 100% con un número de muestras bastante amplio (ver tabla 4.6.2). Una vez obtenidos los porcentajes de clasificación con el conjunto de muestras de prueba y como se explicó en la sección 4.6.3, se procedió a validar el clasificador seleccionado el cual fuel la red neuronal MLP.

Para realizar la validación visual del clasificador seleccionado, nos apoyamos en la GUI presentada en la sección 4.6.1, la cual se modificó y su funcionamiento se explica en secciones posteriores. Como se observa en las siguientes imágenes del lado izquierdo se muestra la imagen a procesar con los descriptores de puntos de interés detectados sin clasificar, y del lado derecho se presenta la misma imagen con los descriptores de puntos de interés clasificados. Los descriptores de puntos de interés clasificados como pertenecientes al Objeto 1 se pintan en color verde, mientras que los descriptores de puntos de interés clasificados como pertenecientes al fondo se pintan en color rojo. Una vez realizado el estudio con el análisis de 50 imágenes diferentes al conjunto de entrenamiento se obtuvo lo que llamamos porcentaje de clasificación real del 95.6036% el cual, a pesar de ser menor al porcentaje reportado por el conjunto de entrenamiento sigue siendo un muy buen porcentaje de clasificación. A continuación se presenta en conjunto pequeño de imágenes de muestra que se utilizaron para calcular el porcentaje de clasificación real.

Figura 5.2.2 Primera imagen de muestra del conjunto de prueba de la clasificación con MLP para el Objeto 1.

Figura 5.2.3 Segunda imagen de muestra del conjunto de prueba de la clasificación con MLP para el Objeto 1.

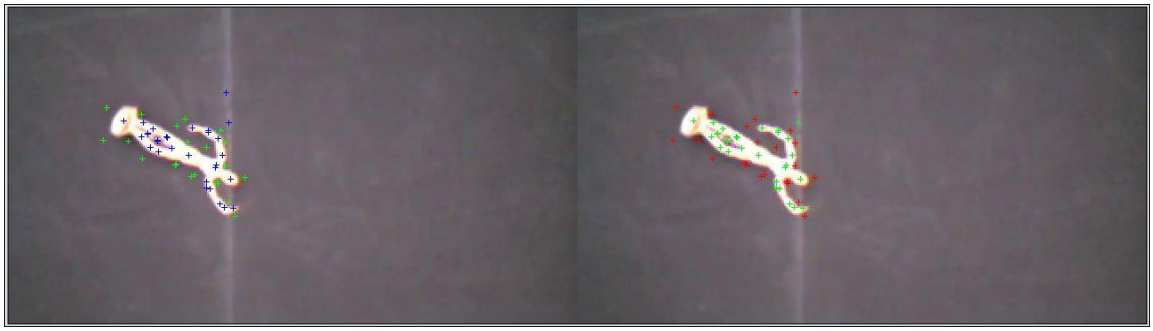

Figura 5.2.4 Tercer imagen de muestra del conjunto de prueba de la clasificación con MLP para el Objeto 1.

Las imágenes del conjunto de prueba se seleccionaron de tal forma que tuvieran variaciones diversas sobre escala, rotación y luminosidad, cabe resaltar que el conjunto de imágenes se capturo con el Quadrotor en vuelo a una altura aproximada de 100 cm sobre el suelo.

5.3 Resultados de clasificación de cada objeto

Como se menciona en las secciones 4.7, 4.8 y 4.9 una vez que se tiene una certeza en cuanto a los porcentajes de clasificación para el primer objeto de estudio “Objeto 1”, se requiere extender el conjunto de objetos que se clasifican, para esto, como se explicó en secciones anteriores, se eligió un conjunto extra de objetos presentado en la sección 4.7, a los cuales se les extrajo los descriptores de puntos de interés con el procedimiento explicado en la sección 4.8 y la clasificación se realizó con una red neuronal MLP los resultados de la clasificación para cada uno de los objetos se muestran en la siguiente gráfica.

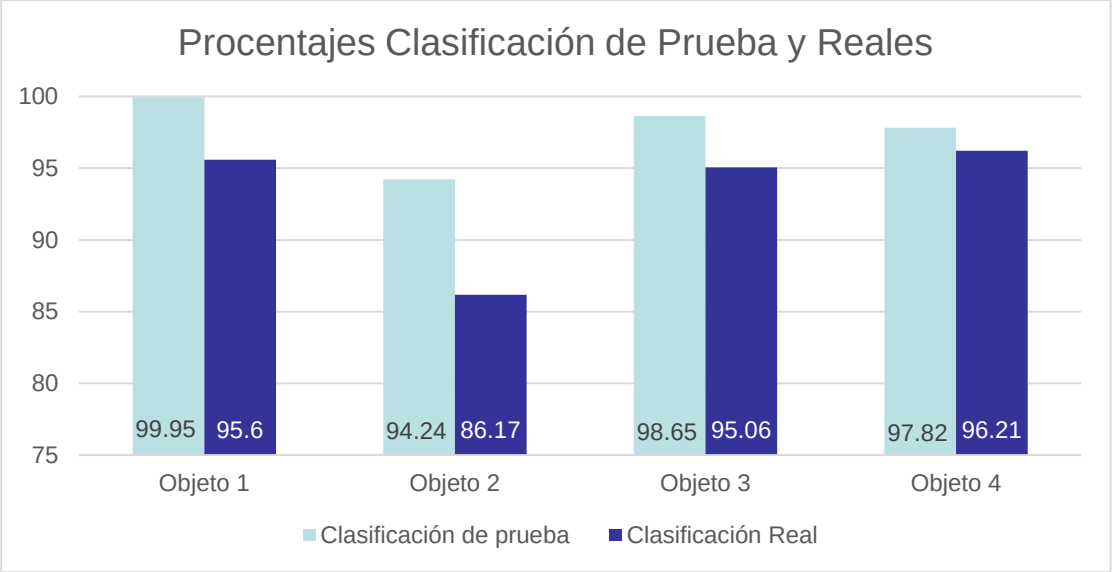

Figura 5.3.1 Gráfica comparativa entre porcentajes de clasificación de prueba y reales.

De igual forma que con el primer objeto de estudio “Objeto 1”, se realiza una validación post entrenamiento para identificar el porcentaje de falsos positivos, donde por cada objeto se tomó un conjunto de 50 imágenes y se analizó de forma manual, como se observa en la gráfica anterior, los porcentajes de clasificación real son menores a los porcentajes reportados por los datos de prueba, aun así, los porcentajes de clasificación real poseen un valor alto y son confiables como se observa en la gráfica anterior. Las imágenes de prueba que se tomaron como base para realizar las pruebas de clasificación real por objeto se muestran en la siguiente secuencia de imágenes, en donde los descriptores de puntos de interés clasificados como parte del objeto se dibujan en rojo y los descriptores de puntos de interés que no pertenecen al objeto se pintan en verde.

En la siguiente secuencia de imágenes se observa que la mayor parte de los descriptores de puntos de interés detectados son clasificados de forma correcta y solo algunos son clasificados de forma incorrecta como lo son los descriptores de puntos de interés dibujados en color rojo y que se encuentran sobre el fondo, con lo que se demuestra que los porcentajes de clasificación son confiables.

Figura 5.3.2 Primera imagen de muestra del conjunto extendido de prueba para el Objeto 2.

Figura 5.3.3 Segunda imagen de muestra del conjunto extendido de prueba para el Objeto 2.

Figura 5.3.4 Tercera imagen de muestra del conjunto extendido de prueba para el Objeto 2.

La siguiente secuencia de imágenes muestra los resultados de las pruebas de clasificación sobre imágenes reales (imágenes tomadas con el Quadrotor en vuelo) para el Objeto 3, de igual forma que para el objeto anterior del lado derecho se muestran las imágenes originales con los descriptores de puntos de interés dibujados sobre la imagen y sin clasificación, del lado derecho se muestra la misma imagen con los descriptores de puntos de interés clasificados mediante la red

neuronal MLP, con los descriptores de puntos de interés clasificados como pertenecientes al Objeto 3 dibujados en color rojo, y los no pertenecientes al Objeto 3 dibujados en color verde. Los resultados de este procedimiento de clasificación son menos ruidosos que los resultados obtenidos en la clasificación para el Objeto 2.

Figura 5.3.5 Primera imagen de muestra del conjunto extendido de prueba para el Objeto 3.

Figura 5.3.6 Segunda imagen de muestra del conjunto extendido de prueba para el Objeto 3.

Figura 5.3.7 Tercera imagen de muestra del conjunto extendido de prueba para el Objeto 3.

En la siguiente secuencia de imágenes se muestra el proceso de clasificación para el Objeto 4, hasta el momento el objeto más difícil de clasificar, tanto por el tamaño, detalles y número de puntos de interés detectados ya que el número de estos últimos es mucho menor en comparación con el número de puntos de interés detectados para los objetos anteriores. De igual forma los descriptores de puntos de interés clasificados como parte del objeto son dibujados en color rojo y los descriptores de puntos de interés clasificados como fondo son dibujados en color verde, los resultados de esta prueba de clasificación se muestran a continuación.

Figura 5.3.8 Primera imagen de muestra del conjunto extendido de prueba para el Objeto 4.

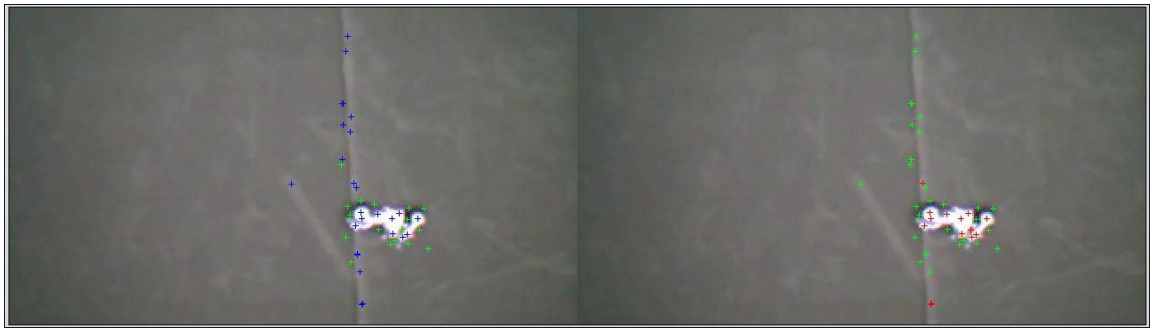

Figura 5.3.9 Segunda imagen de muestra del conjunto extendido de prueba para el Objeto 4.

Una vez que validamos que los porcentajes de clasificación mediante la red neuronal MLP son confiables para cada objeto de nuestro conjunto de objetos, el siguiente paso es verificar hasta qué punto se puede entrenar una red neuronal y que porcentajes de clasificación nos otorga la red neuronal para cada uno de los objetos. Para verificar los porcentajes de clasificación, no solo entrenamos la red neuronal MLP sino que también entrenamos el conjunto de clasificadores

mencionado en la sección 5.1. Estos procedimientos de clasificación se describen en las siguientes secciones.

Figura 5.3.10 Tercera imagen de muestra del conjunto extendido de prueba para el Objeto 4.

5.4 Resultados de clasificación de 2 objetos

En la sección anterior se demostró que es posible realizar la extracción y clasificación mediante una red neuronal MLP de cada uno de los objetos de estudio presentado en las secciones anteriores, y como ya se mencionó es necesario extender el estudio para que se realice la clasificación de más de un objeto en la misma escena. Para verificar lo anterior se seleccionaron los objetos "Objeto 1" y "Objeto 2" de nuestro conjunto de objetos de estudio y se entrenó el conjunto de clasificadores mencionado anteriormente dando como resultado los porcentajes de clasificación presentados en la siguiente gráfica (figura 5.4.1). Como se observa en la gráfica anterior de nueva cuenta los porcentajes de clasificación más altos son los obtenidos mediante los clasificadores MLP y KNN, donde MLP supera a KNN por tan solo unas décimas y procediendo de igual forma que en las secciones anteriores, se construyó un conjunto de 50 imágenes de prueba diferentes del conjunto de entrenamiento, para realizar la validación de clasificación real, mostrada en la última columna de la figura 5.4 y como era de esperarse el porcentaje de validación utilizando MLP es menor al reportado en la pruebas de entrenamiento de las redes neuronales y aunque es un valor menor al obtenido para la clasificación de un solo objeto, el valor obtenido en estas pruebas sigue siendo alto, mayor a un 90%. En la siguiente secuencia de imágenes se muestran algunas de las imágenes utilizadas para realizar la validación de la clasificación real para los objetos seleccionados.

En las figuras 5.4.2 a 5.4.4 se observa como las pruebas de clasificación incluyen transformaciones de rotación y escala para los dos objetos implicados.

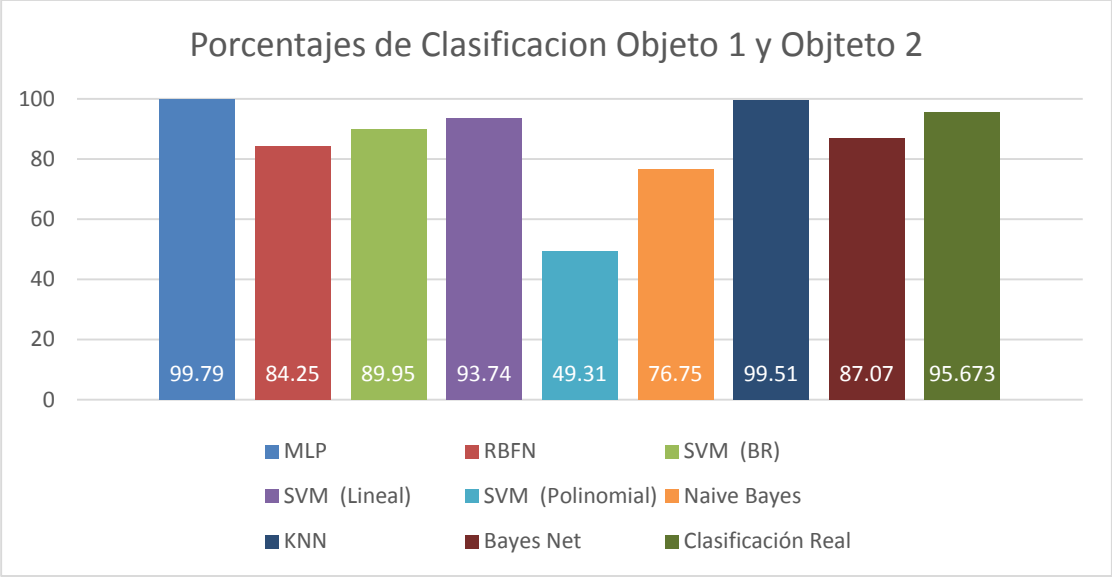

Figura 5.4.1 Grafica comparativa de porcentajes de clasificación para 2 objetos.

Figura 5.4.2 Primera imagen de prueba de clasificación para 2 objetos.

Figura 5.4.3 Segunda imagen de prueba de clasificación para 2 objetos.

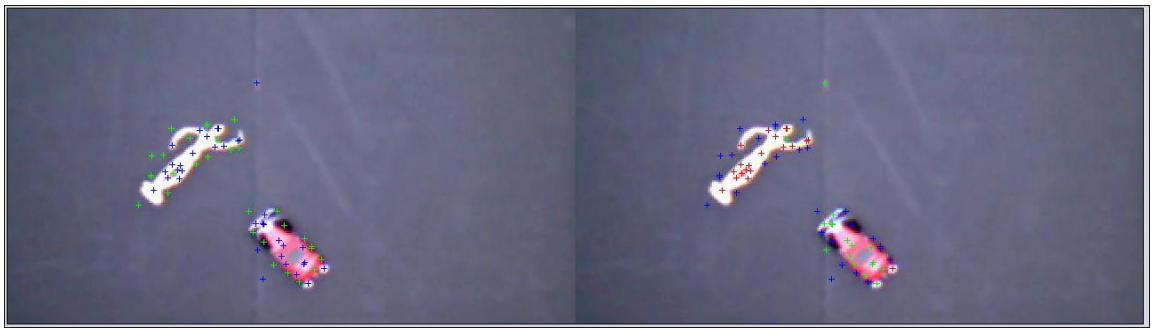

Figura 5.4.4 Tercera imagen de prueba de clasificación para 2 objetos.

5.5 Resultados de clasificación de 3 objetos

Como en la sección anterior se obtuvieron buenos resultados en cuanto a la clasificación de los descriptores de puntos de interés para 2 objetos y la finalidad es de poder clasificar los 4 objetos presentados en una misma escena, el siguiente paso lógico es el de tratar de clasificar 3 objetos en la misma escena, para esto se eligieron los objetos "Objeto 1", "Objeto 2" y "Objeto 3", se extrajeron y clasificaron los descriptores de puntos de interés pertenecientes a cada objeto, con el conjunto de clasificadores ya presentado. Los porcentajes de clasificación del proceso descrito anterior se muestran en la siguiente gráfica.

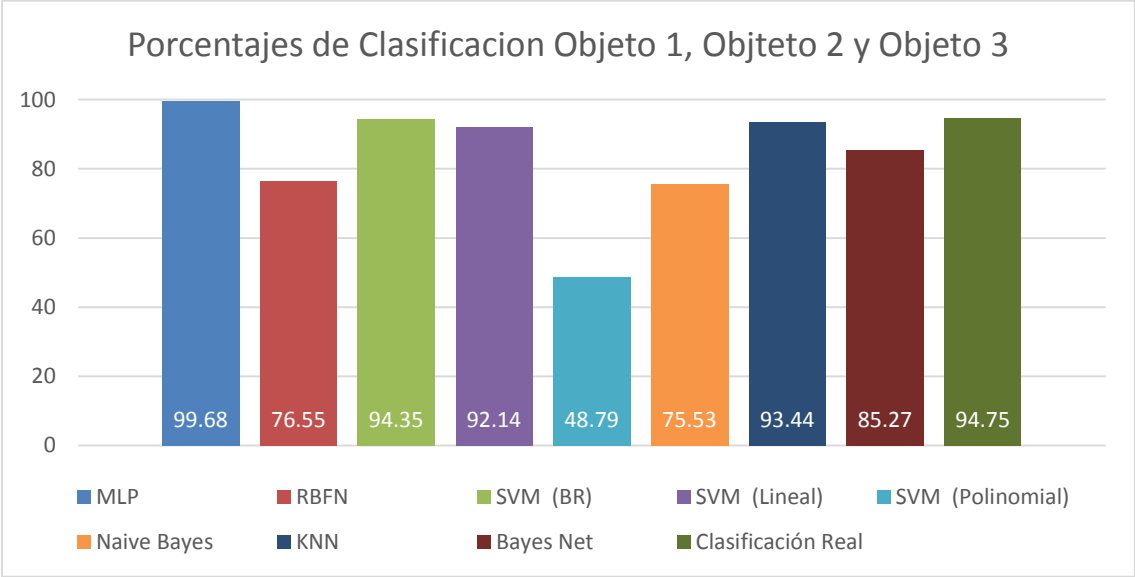

Figura 5.5.1 Grafica comparativa de porcentajes de clasificación para 3 objetos.

Como se observa en la gráfica anterior, al introducir el tercer objeto en el conjunto de clasificación, los porcentajes de clasificación se ven reducidos en solo unas décimas, e igual que en las pruebas de clasificación anteriores la última columna muestra el porcentaje de clasificación real de un conjunto de 50 imágenes, el cual es independiente al conjunto de imágenes de entrenamiento. Los resultados de clasificación para este conjunto de entrenamiento se muestran en la secuencia de imágenes mostrada en las figuras 5.3.2 a 5.3.4.

Figura 5.5.2 Primera imagen de prueba de clasificación para 3 objetos.

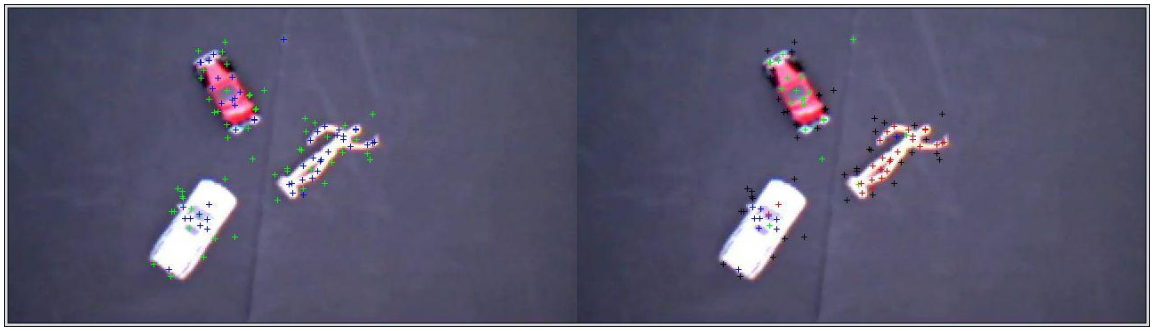

Figura 5.5.3 Segunda imagen de prueba de clasificación para 3 objetos.

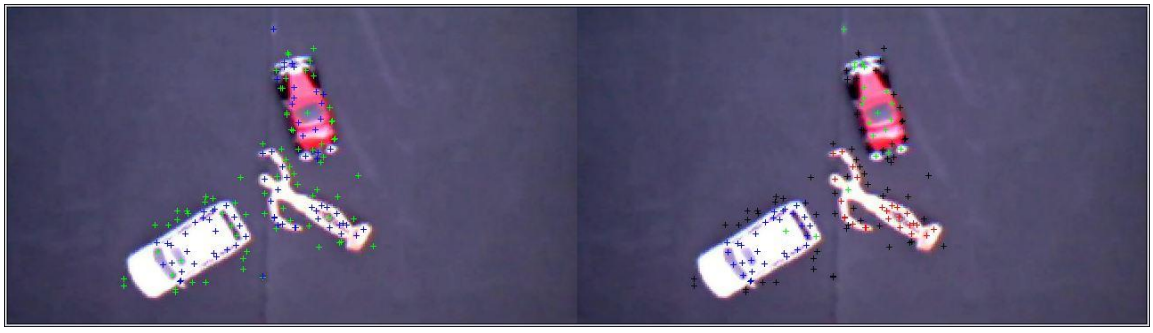

Figura 5.5.4 Tercera imagen de prueba de clasificación para 3 objetos.

En las imágenes anteriores los descriptores de puntos de interés clasificados como correspondientes al Objeto 1 se dibujan del lado derecho en color rojo, los descriptores de puntos de interés clasificados como correspondientes al Objeto 2 se dibujan en color verde, los descriptores de puntos de interés clasificados como correspondientes al Objeto 3 se dibujan en color azul marino y los descriptores de puntos de interés correspondientes al fondo se dibujan en color negro.

Resultados de clasificación de 4 objetos

Siguiendo la lógica planteada en los puntos anteriores, procedemos a realizar el proceso de clasificación de los puntos descriptores de puntos de interés para los cuatro objetos en la misma escena, los porcentajes resultantes de la clasificación se muestran en la siguiente gráfica.

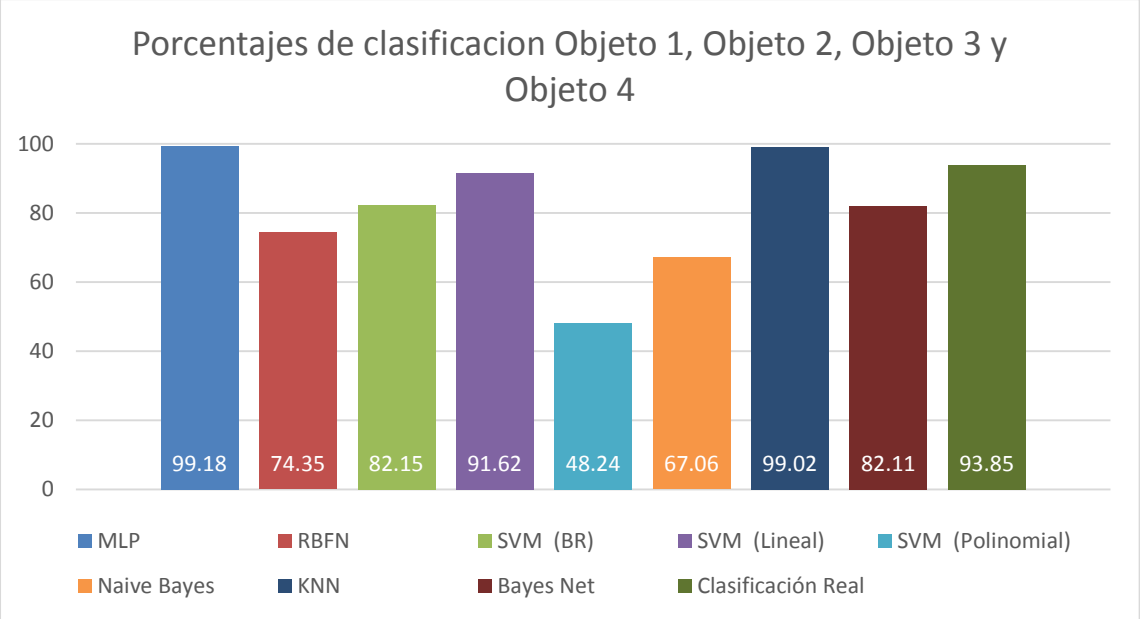

Figura 5.5.5 Grafica comparativa de porcentajes de clasificación para 4 objetos.

En la tabla anterior se observa el mismo patrón de resultados obtenidos en los experimentos anteriores, con los porcentajes más altos de clasificación obtenidos por la red neuronal y el clasificador KNN con solo una diferencia de décimas de punto porcentual. De igual forma se observa como el porcentaje de clasificación real es menor al reportado por el conjunto de

descriptores de puntos de interés de prueba, sin embargo este último porcentaje sigue siendo muy confiable. En la siguiente secuencia de imágenes se muestran los resultados de la clasificación de los 4 objetos.

Figura 5.5.6 Primera imagen de prueba de clasificación para 4 objetos.

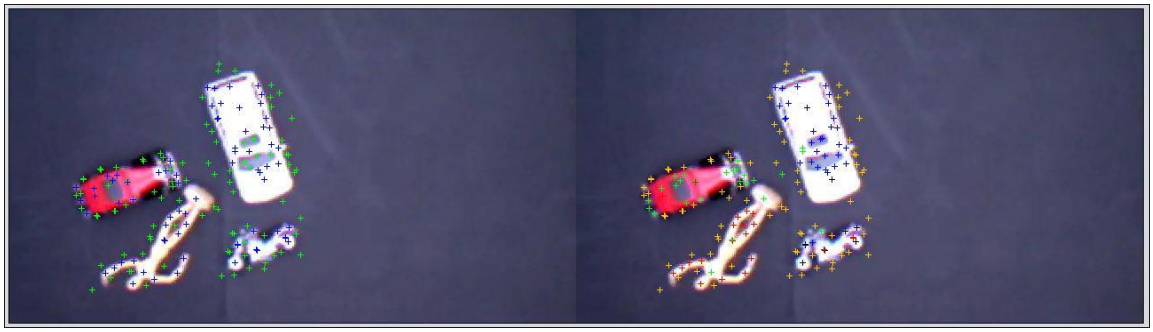

Figura 5.5.7 Segunda imagen de prueba de clasificación para 4 objetos.

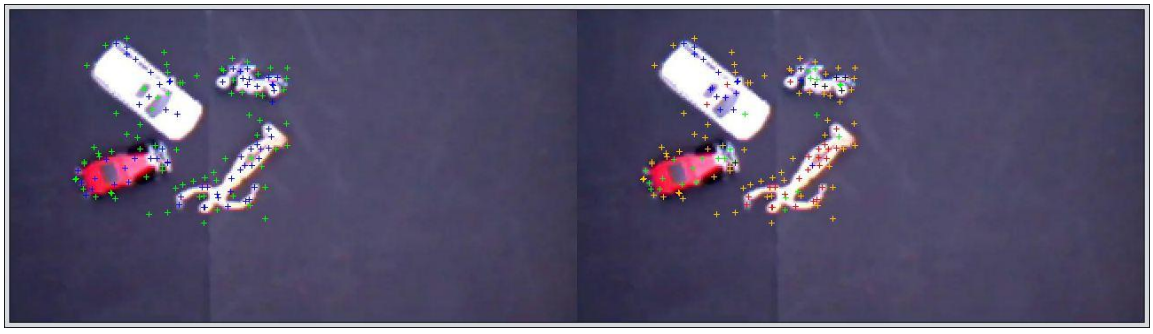

Figura 5.5.8 Tercera imagen de prueba de clasificación para 4 objetos.

En donde los puntos de los descriptores de puntos de interés clasificados como pertenecientes al Objeto 1 se dibujan en color rojo, los descriptores de puntos de interés clasificados como pertenecientes al Objeto 2 se dibujan en color verde, los descriptores de puntos de interés clasificados como pertenecientes al Objeto 3 se dibujan en color azul marino y los descriptores de puntos de interés clasificados como pertenecientes al Objeto 4 se dibujan en color negro.

En este último tópico nos dimos a la tarea de verificar la clasificación real por objeto del conjunto de imágenes tanto las de prueba como las del conjunto de clasificación real, cuyos resultados se muestran en la siguiente gráfica.

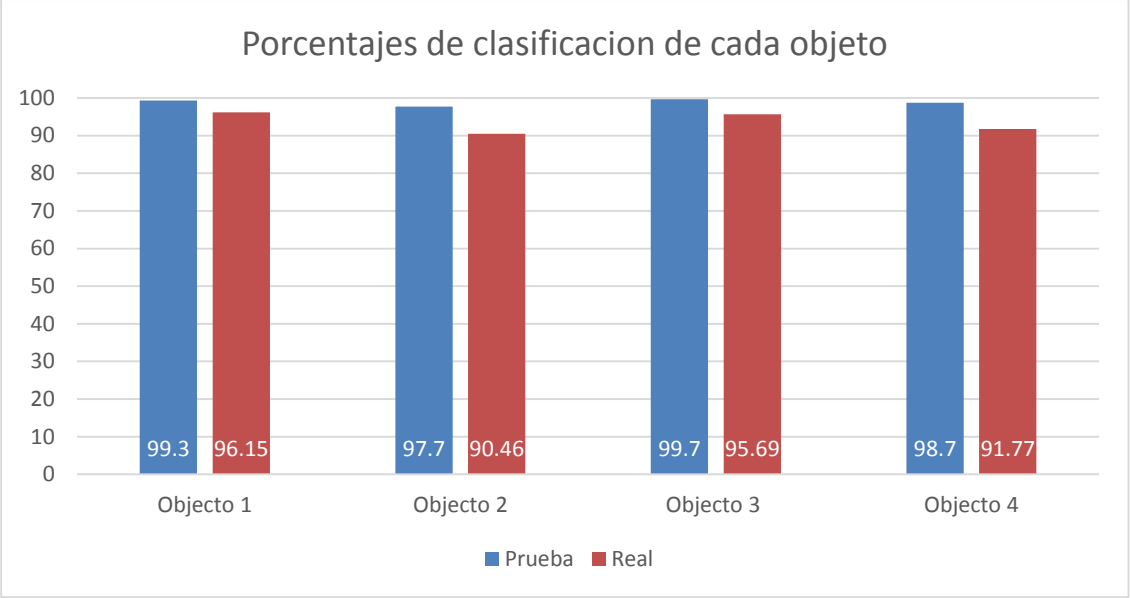

Figura 5.5.9 Grafica comparativa de porcentajes de clasificación por cada objeto.

Las imágenes anteriores se tomaron teniendo en cuenta modificaciones en rotación, cambio de escala y cambios de iluminación de la escena.

Hasta este punto de la fase de experimentación se observó que es posible realizar la clasificación de los cuatro conjuntos de descriptores de puntos de interés en la misma escena con un porcentaje aceptable en cuanto a la clasificación. Sin embargo observamos que existen errores de clasificación debido a falsos positivos. Los cuales se tratan en la siguiente sección mediante un proceso de filtrado por desviación estándar.

5.6 Resultados de filtrado espacial por desviación estándar post clasificación

Como se puede observar en las imágenes presentadas en las secciones anteriores, el proceso de clasificación, clasifica de forma correcta más del 90% de los descriptores de puntos de interés, sin embargo, aproximadamente el 10% de los descriptores mal clasificados introducen errores sobre todo al tratar de identificar al objeto sobre la imagen, como se muestra en la figura 5.7.1.

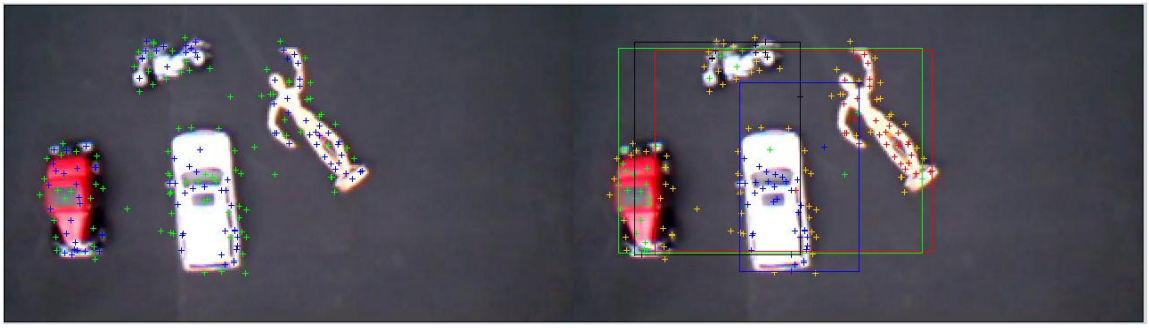

Figura 5.6.1 Identificación de objetos en base a la clasificación.

En donde el proceso de identificación consiste en encerrar en un recuadro al objeto identificado, mediante el color de clasificación asignado como se ha mencionado en secciones anteriores. Así para el Objeto 1 el recuadro dibujado es de color rojo y dentro de este se deberá de contener al Objeto 1, para el Objeto 2 el recuadro dibujado es de color verde y dentro de este se deberá de contener al Objeto 2, para el Objeto 3 el recuadro es dibujado de color azul marino y el Objeto 3 deberá de ser contenido dentro del recuadro, para el Objeto 4 el recuadro es dibujado en color negro y el Objeto 4 deberá de ser contenido dentro del este recuadro.

Como se puede observar de la figura 5.7, los recuadros dibujados sobrepasan las dimensiones de los objetos y hasta se sobre ponen unos recuadros a otros, estos resultados se obtienen de la clasificación errónea de los descriptores de puntos de interés espurios. Para poder eliminar estos puntos de interés mal clasificados se implementó un filtrado por desviación estándar, cuyos resultados comparativos se muestran a continuación.

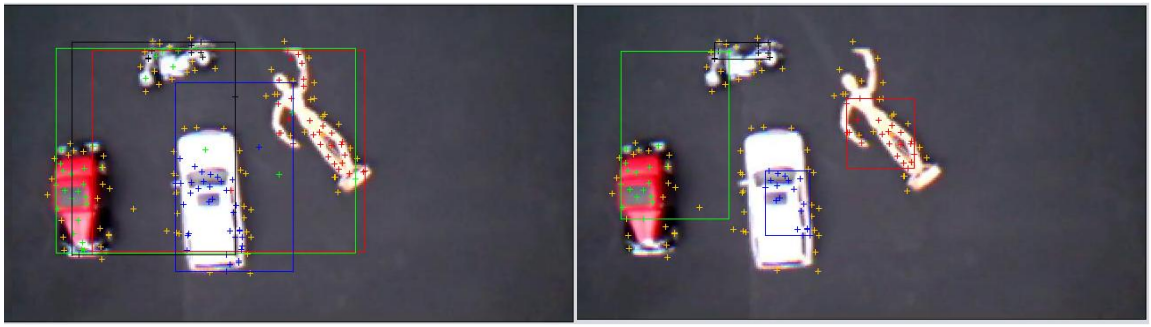

Figura 5.6.2 Identificación de objetos con y sin filtrado por desviación estándar.

En la imagen anterior del lado izquierdo se muestra la clasificación sin filtrado por desviación estándar, donde como ya se había comentado el trazo de los recuadros de identificación se sobre ponen, del lado derecho se muestra la misma imagen pero con el filtrado por desviación estándar, y como se observa en la imagen, los recuadros encierran una parte o a todo el objeto, lo cual permite una mejor identificación visual de los objetos y disminuye los errores de clasificación y presentación de resultados.

A continuación se presenta una serie de imágenes a modo de comparativo entre identificación con y sin filtrado por desviación estándar.

Como resultado del estudio comparativo podemos comentar, que el filtrado por desviación estándar efectivamente filtra de forma efectiva los puntos de interés espurios, siempre y cuando estos sobrepasen la media calculada para dicha imagen, de lo contrario, algunos puntos no serán filtrados por este proceso y la identificación aunque certera trazará recuadros de un área mucho mayor al objeto identificado.

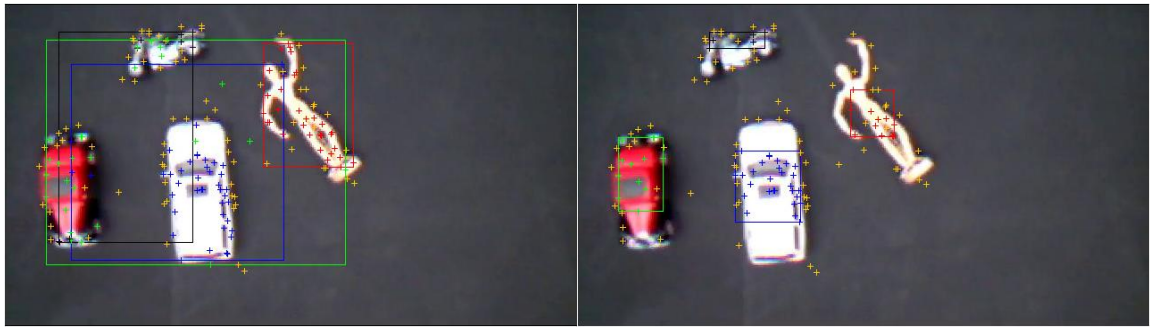

Figura 5.6.3 Primer imagen de objetos con y sin filtrado por desviación estándar.

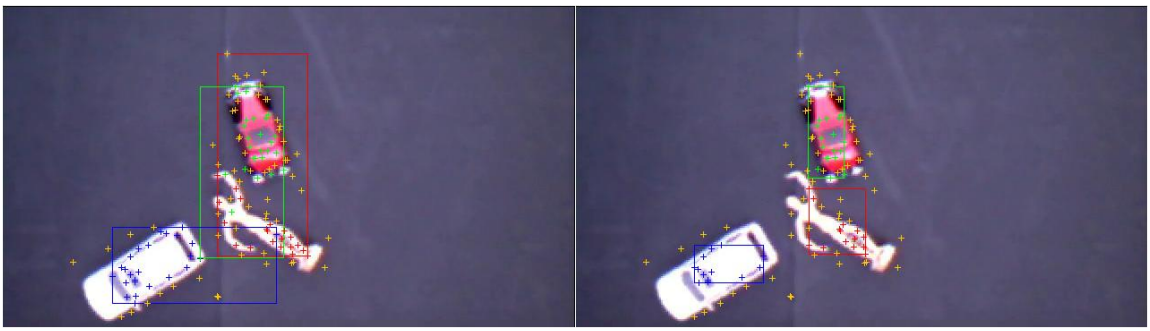

Figura 5.6.4 Segunda imagen de objetos con y sin filtrado por desviación estándar.

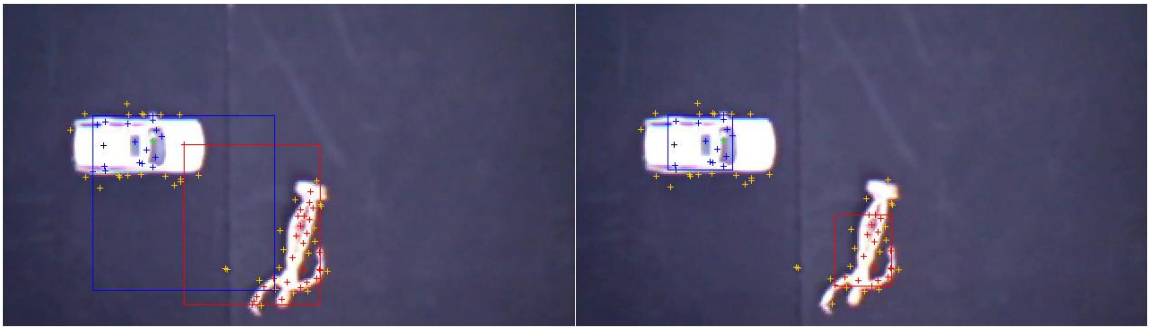

Figura 5.6.5 Segunda imagen de objetos con y sin filtrado por desviación estándar.

5.7 Arquitectura de la red neuronal MLP implementada

Durante el desarrollo de la presente tesis desde el capítulo anterior donde se tratan las diferentes técnicas de clasificación, se ha comentado que el mejor procedimiento para la clasificación de los descriptores de puntos de interés, según nuestros resultados, es la red neuronal MLP, cuya arquitectura final se muestra a continuación en la figura 5.8.1.

Como se observa de esta figura la red consta de 3 capas completamente conectadas hacia adelante, en la capa de entrada se tienen 64 nodos, en la capa intermedia se cuentan con un total de 34 nodos y en la capa final o de salida se cuentan con 5 nodos, un nodo por cada clase (contando el fondo) que se especificó en la etapa de entrenamiento.

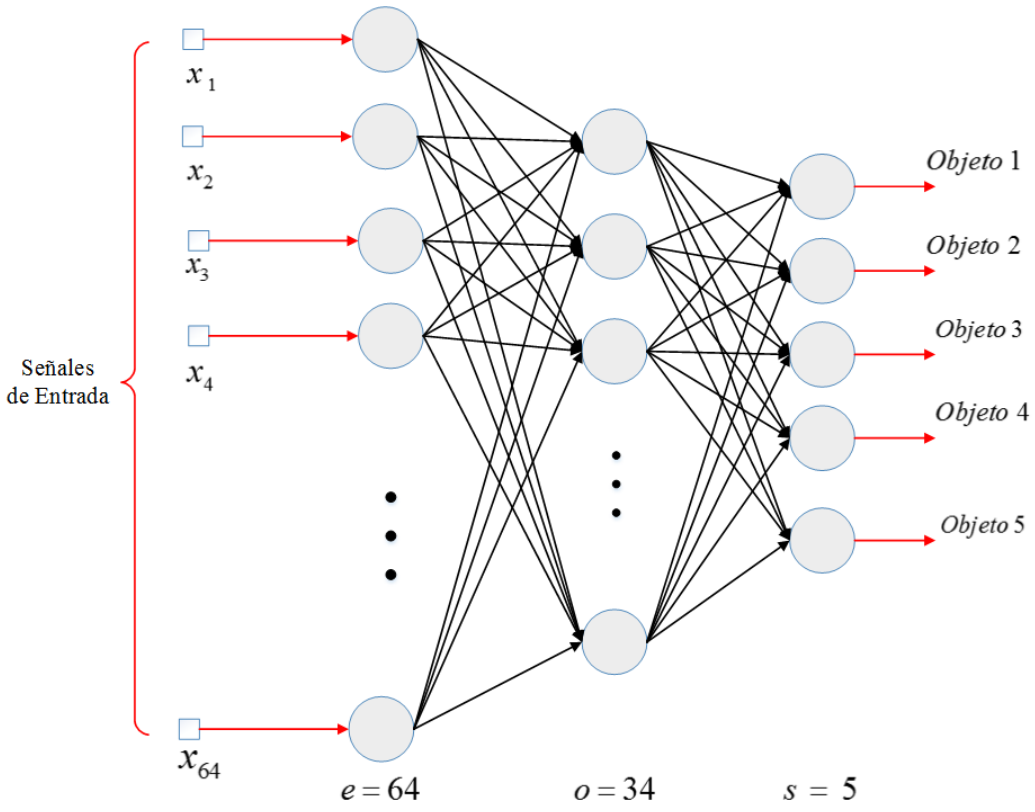

Figura 5.7.1 Arquitectura de la red neuronal MLP implementada.

5.8 Descripción de la interface gráfica

Una vez que hemos explicado cada uno de los componentes de la presente tesis, describiendo desde cómo se obtienen las imágenes provenientes del Quadrotor, pasando por la etapa de control y la etapa de procesamiento de imágenes, identificación, clasificación, filtrado y presentación de resultados, todas estas etapas se unen en un producto final, una interface gráfica amigable para el usuario, cuyo Front End y Back End ya se presentaron en secciones anteriores, por lo que en esta sección procedemos a describir los elementos que componen a dicha interface.

Como elemento principal de la interface de usuario, se presenta la pantalla que se muestra en la siguiente figura.

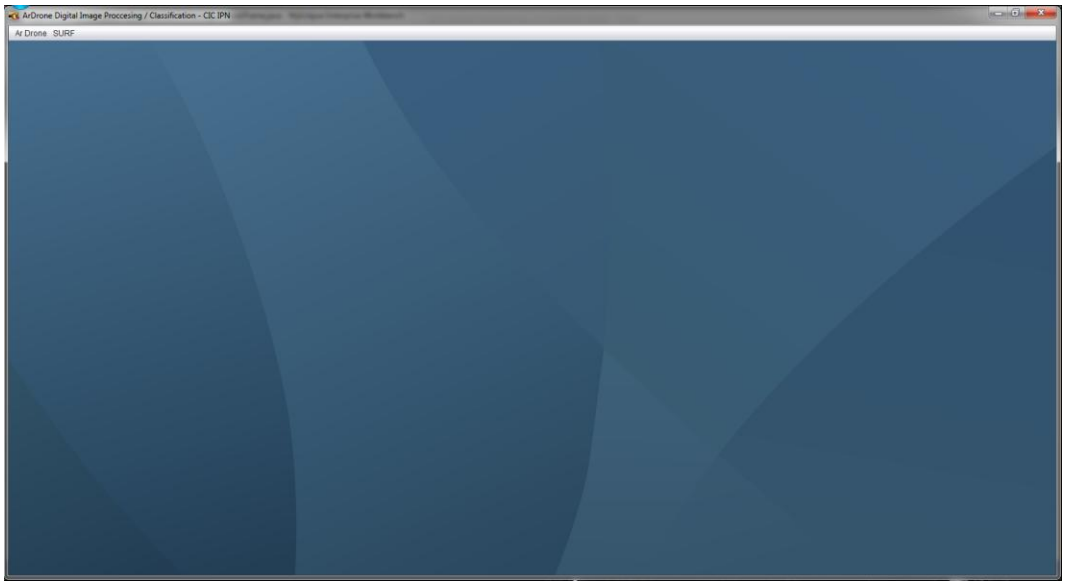

Figura 5.8.1 Ventana principal de la aplicación de control y procesamiento para el Ar. Drone.

En donde el menú de usuario Ar. Drone nos presenta el módulo de Análisis Automático el cual se explica en la siguiente sección y el menú de usuario SURF nos presenta el módulo de análisis de imágenes y extracción de descriptores de puntos de interés presentado en la sección 4.6.1.

5.8.1 Módulo de Análisis Automático

Para acceder al módulo de Análisis Automático de la aplicación, se debe de seleccionar el menú Ar. Drone y posteriormente el submenú “Auto. Analysis” el cual presentará la siguiente pantalla. Esta pantalla cuenta con tres paneles principales. El panel de video que se muestra en la parte superior, y el cual a su vez, está dividido en dos componentes JPanel, el componente JPanel del lado izquierdo presenta el video en tiempo real proveniente del Quadrotor sin ningún tipo de procesamiento, video que nos sirve como referencia visual. El panel de lado derecho el cual nos presenta los resultados de la extracción y clasificación en tiempo real.

El panel inferior izquierdo nos muestra la interface gráfica de control presentada en la sección 4.1.5. Y el panel inferior derecho nos muestra las imágenes de los objetos a identificar. Este panel consta de las cuatro imágenes de los objetos a identificar y delante de cada una de ellas un recuadro el cual nos especifica si el objeto de interés ha sido detectado e identificado por el proceso de análisis de imágenes. Si el objeto no se ha identificado, el recuadro delante de la imagen que

representa el objeto será pintado en color negro y de lo contrario, si el objeto ha sido identificado el recuadro será pintado de color verde. Lo anterior se ejemplifica en las siguientes imágenes.

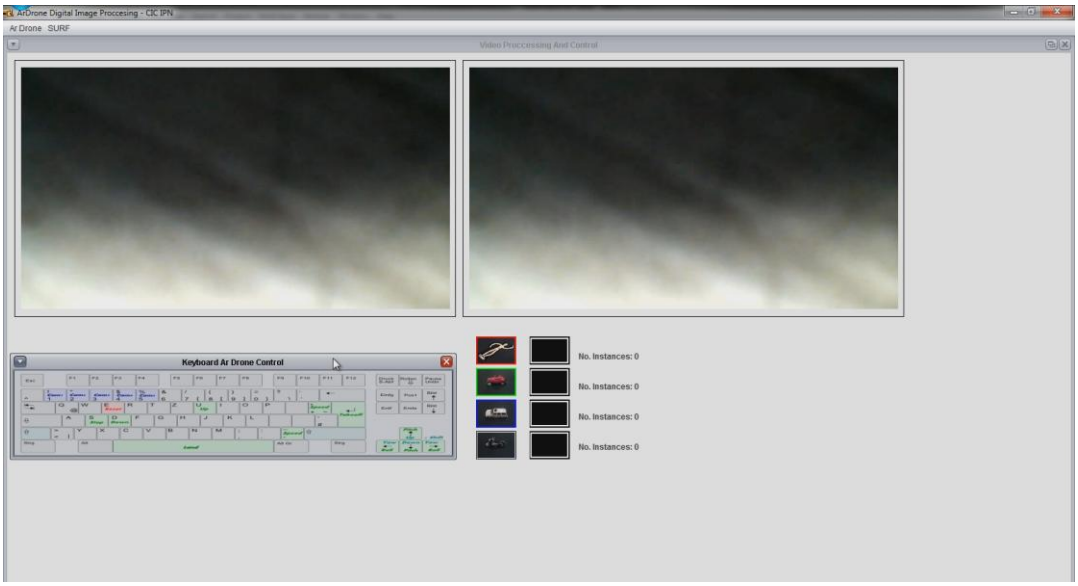

Figura 5.8.1.1 Ventana de control y análisis automático de video.

Figura 5.8.1.2 Ventada de identificación de objetos en estado inicial.

En la figura 5.9.1.2 se muestra el estado inicial del panel, donde en la primer columna se muestra cada uno de los objetos a ser identificados, en la siguiente columna se muestra un recuadro al cual llamamos recuadro de identificación, al iniciar el proceso este es pintado en color negro y en la tercer columna se muestra una leyenda “No. Instances: 0” la cual indica el número de imágenes en la cual el objeto correspondiente a la fila ha sido identificado, una vez que el objeto de interés ha

sido detectado, el recuadro de identificación se pinta en verde y la leyenda “No. Instances: 0” desplegara el número de imágenes en las cuales el objeto ha sido detectado e identificado.

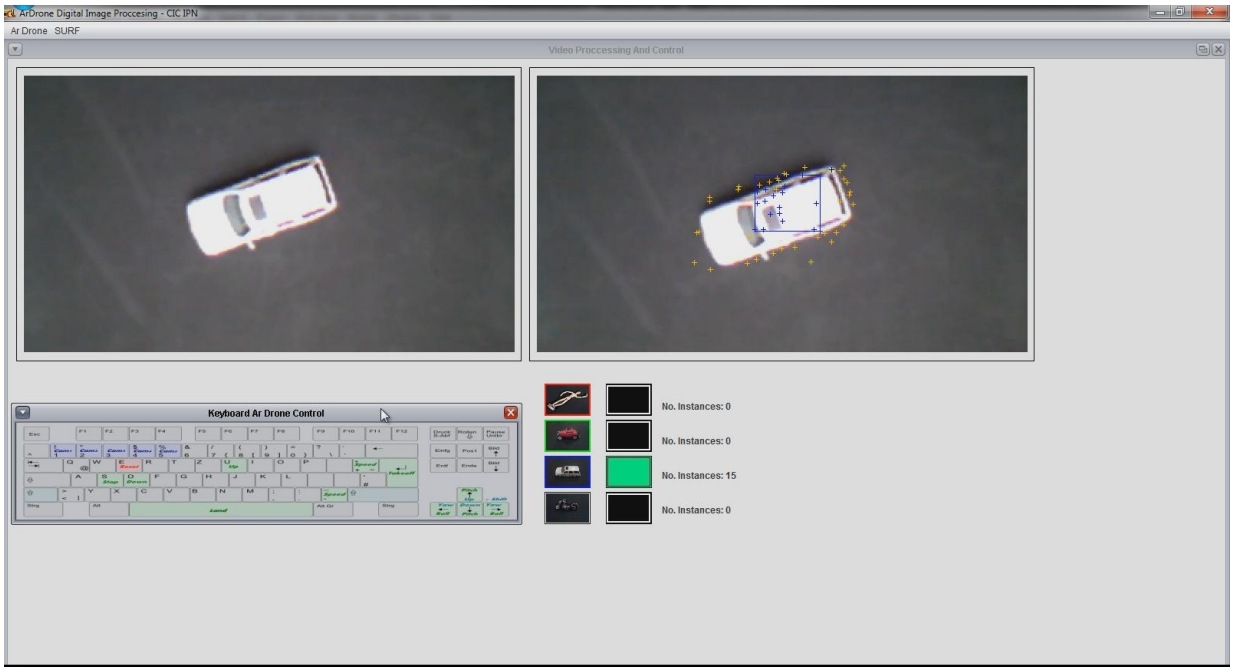

Figura 5.8.1.3 Ventada de identificación automática con un objeto identificado.

Como otro ejemplo se presenta la imagen en la figura 5.9.1.3, en la cual se observa que en el panel de video del lado izquierdo aparece el Objeto 3 como imagen de referencia. Del lado derecho aparece la misma imagen pero en esta también se presentan los puntos de interés detectados sobre la escena y también el recuadro en color azul marino, el cual representa el resultado de la clasificación e identificación del Objeto 3. Como se observa en la escena de la figura 5.9.1.3 no existe error de clasificación ya que el recuadro en azul marino aparece exactamente sobre el objeto.

Otro ejemplo es el que se muestra en la figura 5.9.1.4, la cual se muestra a continuación.

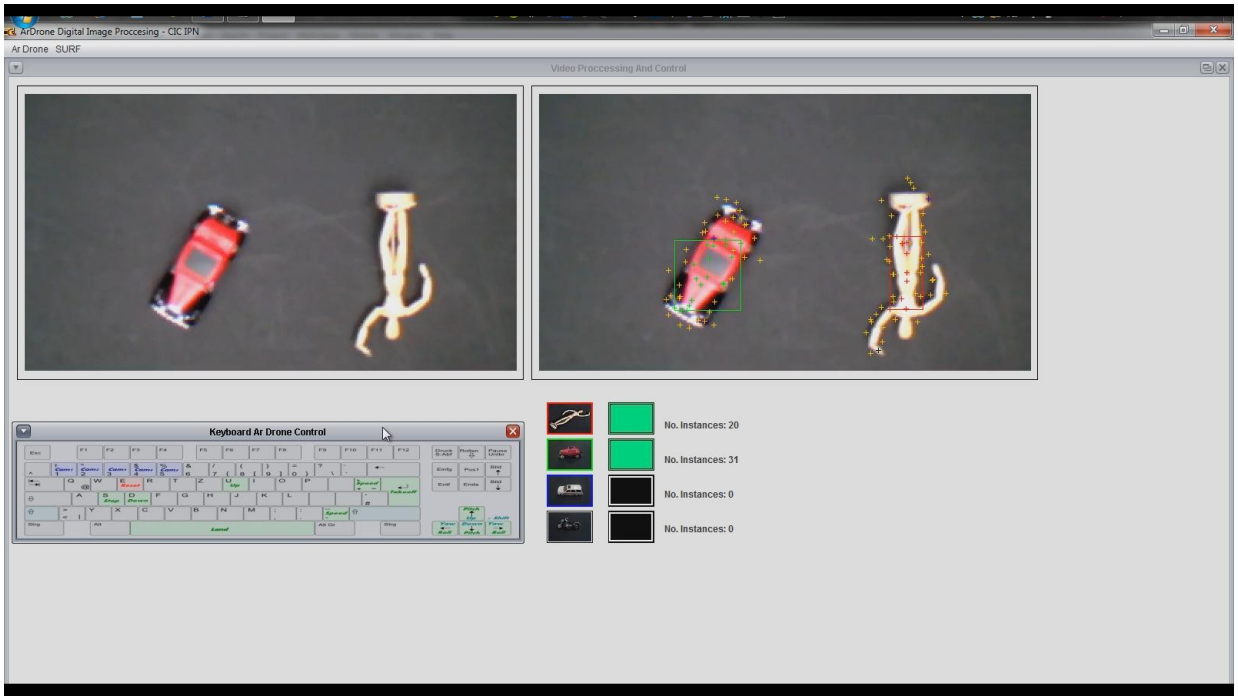

Figura 5.8.1.4 Ventada de identificación automática con dos objetos identificados.

En la figura anterior se muestra que el proceso de identificación, el cual puede identificar más de un objeto en la misma escena, en esta imagen se observa como los objetos Objeto 1 y Objeto 2, presentes en la escena, son clasificados de forma correcta, para el Objeto 1 se observa como el recuadro en rojo es dibujado directamente sobre el maniquí, y para el Objeto 2 también se observa como el recuadro en color verde es dibujado directamente sobe el Objeto 2, mostrando una clasificación correcta.

5.9 Ruta de vuelo

En el capítulo anterior se explicó la forma en la que el Quadrotor realiza la ruta de vuelo, de tal forma que pueda analizar los diferentes escenarios propuestos, para esto se construyeron los escenarios que conforman la ruta de vuelo. La ruta mencionada se muestra en el siguiente conjunto de imágenes.

Figura 5.9.1 Primer figura de los escenarios que conforman la ruta de vuelo del Quadrotor.

Figura 5.9.2 Segunda figura de los escenarios que conforman la ruta de vuelo del Quadrotor.

Como se muestra en las imágenes anteriores, el Quadrotor inicia la ruta de vuelo en el primer escenario E1 el cual simplemente contiene el objeto fondo, posteriormente el Quadrotor en vuelo procede a analizar el escenario dos E2, el cual como se observa contiene al Objeto 1, posteriormente el Quadrotor procede a analizar al escenario tres E3 el cual contiene al Objeto 2, y

como etapa final el Quadrotor en vuelo procede a analizar el escenario cuatro E4 el cual contiene a los objetos Objeto 3 y Objeto 4.

Los resultados y observaciones de los experimentos descritos en las secciones anteriores de discuten en el siguiente capítulo.

5.10 Discusión de resultados

Como en todo proceso de clasificación, este último depende en gran medida del pre procesamiento, extracción y filtrado de rasgos. Como ya se mencionó, la variación del ángulo del Quadrotor con respecto al plano horizontal produce deformaciones varias en los objetos contenidos en las imágenes que de este se extraen, por lo que un análisis por área, forma o incluso por color no resultaba factible en un pre procesamiento de rasgos (al menos no para nuestro caso). Debido a lo anterior se optó por utilizar los descriptores de puntos de interés que como ya se mencionó en el capítulo 3, estos son robustos a las transformaciones más comunes en imágenes y son independientes del color.

Sin embargo hasta el momento del desarrollo de esta tesis, la mayoría de los trabajos analizados en el estado del arte, se orientaba a realizar una comparación de los descriptores de puntos de interés mediante algoritmos Greedy o de fuerza bruta (comparación de elemento a elemento). Y algunos otros utilizaban SVM como separadores de clases de los descriptores de puntos de interés, entre la clase de interés y la clase de fondo. En ningún caso de la literatura estudiada se encontró un estudio formal comparativo entre diferentes clasificadores tanto de redes neuronales como clasificadores de tipo estadístico. Debido a lo anterior es que se realizó el estudio comparativo entre clasificadores y se presentaron los resultados obtenidos.

Aun y cuando se realizó el estudio comparativo entre las diferentes técnicas de clasificación, se observó que existían errores de clasificación por parte de la der neuronal seleccionada, ya que una red neuronal clasificará de forma correcta los patrones hasta cierto punto ruidosos que se le presenten a la entrada, pero dará muy malos resultados en cuanto a patrones completamente ajenos al conjunto de entrenamiento, por lo que la toma de muestras provenientes del Quadrotor en vuelo fue realmente complicado y costoso en términos de tiempo, sin embargo, se lograron buenos

resultados de clasificación (arriba del 90%) teniendo cuidado en la forma en que se tomaban las imágenes a procesar.

Capítulo 6
Conclusiones, trabajo a futuro y recomendaciones

En el presente y último capítulo iniciamos comentando las conclusiones obtenidas al desarrollar e implementar los algoritmos presentados en esta tesis. Las ventajas y desventajas observadas en la etapa de experimentación sobre los algoritmos seleccionados, se comentan también trabajos futuros y recomendaciones.

6.1 Conclusiones

Es posible bajo ciertas condiciones, como se demuestra en este trabajo, el realizar la detección de objetos en tiempo real desde una cámara de video montada en un Quadrotor (sin base estabilizadora). Esto por su naturaleza es bastante complicado debido a la inherente inestabilidad que poseen los Quadrotores, lo que resulta en imágenes o videos con distorsiones tales como variaciones en la distancia de foco, distorsiones en el plano de referencia y en nuestro caso particular variaciones de iluminación, ya que todas las pruebas se realizaron a luz de día.

Como se demostró, es posible realizar la extracción de descriptores de puntos de interés, clasificarlos mediante una red neuronal MLP previamente entrenada, realizar un filtrado espacial mediante desviación estándar y un filtrado temporal histórico en aproximadamente 100 a 150 ms. con imágenes de 360 x 240 pixeles en formato RGB y con las características de un equipo de

cómputo como se mencionó en la sección 4.1.1. Claro apoyándonos en técnicas de programación que mejoran los tiempos de procesamiento y garantizan el flujo ordenado de imágenes.

Con base al estudio comparativo entre las diferentes redes neuronales y clasificadores de tipo estadístico, se demuestra y sustenta el uso de la red neuronal MLP como mejor opción para nuestro caso en particular.

6.2 Trabajo a futuro

Dentro del área de visión por computadora existe una gran variedad de trabajos y en conjunción con una plataforma tan dinámica como lo es un Quadrotor, existe una gran gama de variaciones que se pueden implementar como lo son:

- Implementar otras variantes del algoritmo SURF y verificar cuál de estas es la que posee el mejor desempeño.
- Implementar otros algoritmos de extracción de puntos de interés mediante algoritmos evolutivos [74] y robustecer el estudio comparativo.
- Implementar de forma paralela el algoritmo de extracción de puntos y descriptores SURF, ya sea con Hiper Threading de JAVA o utilizar alguna plataforma altamente paralela como lo es CUDA y GPU´s.
- Implementar la navegación autónoma junto con la identificación automática de los escenarios propuestos en tiempo real.
- Probar el desempeño y rendimiento de los algoritmos propuestos con diferentes cámaras a mayor resolución, como por ejemplo las cámaras de teléfonos móviles.
- Entrenar y probar los algoritmos propuestos con escenarios a una mayor escala, por ejemplo, utilizando vehículos reales dentro de un estacionamiento y verificar su comportamiento.

- Utilizar los algoritmos de identificación implementados en la presente tesis para desarrollar un sistema de seguimiento de objetos, o dar seguimiento a cambios en los objetos previamente identificados, mediante un Quadrotor.
- Probar el mismo es quema de identificación de objetos con algoritmos de clasificación, localización e identificación Deep Learning utilizando un framework como Caffe.

6.3 Recomendaciones

Una plataforma como lo es un Quadrotor comprende varias consideraciones de manejo tanto en hardware, software genérico y de aplicación.

6.3.1 Recomendaciones Hardware

- Equipo de protección para el usuario, esto debido a que el usuario puede resultar con cortes debido a las aspas giratorias.
- Etapa de familiarización con el manejo del Quadrotor, ya que esta plataforma es sub actuada y su manejo resulta ser difícil.
- Un número suficiente de baterías de repuesto, ya que el tiempo de duración de las baterías con el Quadrotor en vuelo es de aproximadamente de 8 minutos. Para la presente tesis se utilizaron 3 baterías.
- Piezas suficientes de repuesto como lo son motores y hélices. En el desarrollo de la presente tesis se utilizaron 3 repuestos completos de hélices debido a los accidentes que se tuvieron con el Quadrotor.

6.3.2 Recomendaciones de software genérico y de aplicación

Como recomendaciones en cuanto a los algoritmos propuestos, podemos mencionar:

- Para realizar la prueba de los algoritmos de identificación, deberá de ser en un lugar cerrado o con sombra, debido a que si incide luz directa sobre el Quadrotor, la sobra producida por este impedirá el correcto procesamiento de las imágenes.

- La presente tesis se desarrolló con el lenguaje de programación JAVA versión 1.6, se recomienda poseer la herramienta de desarrollo Eclipse Galileo o mayor, así como mínimo la versión JAVA JDK 1.6.
- El equipo de cómputo deberá de tener al menos las características mencionadas en la sección 4.1.1.
- Para la prueba e implementación de las redes neuronales se utilizó la distribución del software WEKA versión 3, compilado con la versión SDK 1.6 de JAVA.
- En la parte de procesamiento de imágenes se utilizó la librería estándar para el algoritmo SURF llamada BoofCV la cual también fue compilada completamente con la versión de JAVA 1.6.
- La API utilizada para el control de Quadrotor, no es la distribución estándar, ya que se le realizaron algunas modificaciones en el módulo de CommandManager.

Referencias

[1] A. A. Boguslavskii, S. M. Sokolov, Detecting Objects in Images in Real-Time Computer Vision Systems Using Structured Geometric Models, Programming and Computer Software,Vol. 32, No. 3, pp. 177–187. 2006.

[2] Alnajjar, F.; Murase, K. A Simple Aplysia-like Spiking Neural Network to Generate Adaptive Behavior in Autonomous Robots. Adaptive Behavior 14 (5): 306–324. 2008.

[3] Ana Lameira, Rui Jesus, Nuno Correia,Local Object Detection and Recognition in Mobile Devices, IWSSIP, pp. 11-13. April 2012.

[4] Anelia Angelova, Shenghuo Zhu, Efficient Object Detection and Segmentation for Fine-Grained Recognition, IEEE Conference on Computer Vision and Pattern Recognition. 2013.

[5] Ar. Drone Parrot, Developer Guide 2.0. The Flying Video Game, [Online]. Available: http://www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf. [Accessed: 17- Dec- 2013].

[6] Aydin Eresen, Nevrez Imamoglu, Autonomous Quadrotor Flight with Vision-Based Obstacle Avoidance in Virtual Environment, Expert Systems with Applications 39. 2012.

[7] B. Abhik, Principal Component Analysis using R, Noviembre 2009.

[8] Bai Xiao, Song Yi-Zhe, Peter Hall, Learning Invariant Structure for Object Identification by Using Graph Methods, Computer Vision and Image Understanding 115 1023–1031. 2011.

[9] Beat Fasel, Luc Van Gool, Interactive Museum Guide: Accurate Retrieval of Object Descriptions, Computer Vision Laboratory (BIWI), ETH Zurich, Sternwartstr. 7, 8092 Zurich, Switzerland. 2007.

[10] Benitez J, Castro J, Requina, Are Artificial Neural Networks Black Boxes? IEEE Trans Neural Networks 8:1156–1164. 1997.

[11] Boofcv.org, Performance:SURF – BoofCV. [Online]. Available: http://boofcv.org/index.php?title=Performance:SURF. [Accessed: 24- Nov- 2013].

[12] Carlos Eduardo Pereira, A Java Autopilot for Parrot ARDrone, Universidade Federal do Rio Grande Do Sul, November 2011.

[13] Cortes, C.; Vapnik, V. Support-vector networks. Machine Learning 20 (3): 273. 1995.

[14] Cynthia Beatriz Pérez Castro, Cómputo evolutivo como enfoque en la descripción del contenido de la imagen aplicado a la segmentación y el reconocimiento de objetos, tesis de maestría, Julio 2010.

[15] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints, Cascade Filtering approach. IJCV, 60(2):91 - 110, Enero 2004.

[16] David J. Livingstone, Artificial Neural Networks, Methods and Applications, ChemQuest, Sandown, UK. 2009.

[17] Dkriesel.com, A Brief Introduction to Neural Networks · D. Kriesel. [Online]. Available: http://www.dkriesel.com/en/science/neural_networks. [Accessed: 02- Jul- 2014].
[18] David R. Parker, Model, Predict and Test: Towards a Rigorous Process for Acquisition of Object Detection and Recognition Algorithms for Un-manned Air Vehicle Applications, BAE SYSTEMS Military Air & Information, United Kingdom. Julio 2012.
[19] Debashree Mandal, Karen Panetta, Human Visual System Inspired Object Detection and Recognition, Department of Electrical and Computer Engineering Tufts University Medford, MA, USA. 2012.
[20] Department of transportation, Federal Aviation Administration. Unmanned Aircraft Operations in the National Airspace System. 2006.
[21] Ding Zuchun, An Effective Keypoint Selection Algorithm in SIFT, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 6, No. 2, April 2013.
[22] Donald, David, Encyclopedia of World Aircraft Standard Aircraft, p.854. 1997.
[23] Duda, R. O. & P. E. Hart. Pattern Classification and Scene Analysis. New York: John Wiley & Sons. 1973.
[24] Duy-Nguyen Ta, Wei-Chao Chen, SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors, Georgia Institute of Technology. 2009.
[25] Ehsan Fazl, John S. Zelek, Region Detection and Description for Object Category Recognition, Systems Design Engineering, Intelligent Human Machine Systems Lab Waterloo, ON, Canada. Mayo 2007.
[26] Eugene M. Izhikevich, Simple Model of Spiking Neurons, IEEE Transactions on neural networks, Vol. 14, No. 6, November 2003.
[27] Fix, E., Hodges, J.L. Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
[28] H.J.C. Luijten, Basics of Color Based Computer Vision Implemented in Matlab, DCT.2005.
[29] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, Speeded-Up Robust Features (SURF), ETH Zurich, BIWI Sternwartstrasse 7 CH-8092 Zurich Switzerland. 2008.
[30] Hoffmann, G.M., Rajnarayan, D.G. The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC). November 2004.
[31] Hu Shuo, Wu Na, Object Tracking Method Based on SURF, AASRI Conference on Modeling and Control, 2012.
[32] Huiyu Zhou, Yuan Yuan, Chunmei Shi, Object Tracking using SIFT Features and Mean Shift, Computer Vision and Image Understanding 113 345–352. 2009.
[33] ISO Standard 8373. Manipulating Industrial Robots. 1994.
[34] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Francisco, California, 1988.
[35] José-Juan Hernández-Lópeza, Ana-Linnet Quintanilla-Olveraa, Detecting Objects Using Color and Depth Segmentation with Kinect Sensor", Procedia Technology 3 196 – 204. 2012.
[36] Juan Villegas Cortez, Reconocimiento de tipos de letra. Tesis de maestría, Junio 2005.
[37] K.M. Faraoun, A. Rabhi, Data Dimensionality Reduction Based on Genetic Selection of Feature Subsets, UDL University SBA, 22000, Algeria. 2007.
[38] Leena Silvoster M., Govindan V.K., Convolutional Neural Network Based Segmentation, Communications in Computer and Information Science, 5th International Conference on Information Processing, ICIP 2011.

[39] Li-Chen Fu, Cheng-Yi Liu, Computer Vision Based Object Detection and Recognition for Vehicle Driving, Proceedings of the IEEE International Conference on Robotics 8 Automation Seoul, Korea. Mayo, 20011.
[40] Lior Wolf1, Tal Hassner, Similarity Scores Based on Background Samples, Lecture Notes in Computer Science 5995. 2009.
[41] LIU Xiaofeng, PENG Zhongren*, ZHANG Liye, LI Li, Unmanned Aerial Vehicle Route Planning for Traffic Information Collection, Journal of Transportation Systems Engineering and Information Technology Volume 12, Issue 1, Febrero 2012.
[42] M. Lo Brutto, A. Garraffa, P. Meli, UAV Platforms for Cultural Heritage Survey: First Results, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-5, 2014.
[43] McCulloch, Warren; Walter Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5 (4): 115–133. 1943.
[44] Mercer, J. Functions of Positive and Negative Type and their Connection with the Theory of Integral Equations, Philosophical Transactions of the Royal Society.1909.
[45] Néstor Morales, Jonay T. Toledo, Leopoldo Acosta, Rafael Arnay, Real-time Adaptive Obstacle Detection Based on an Image Database, Computer Vision and Image Understanding 115 1273–1287. 2011.
[46] Nir Friedman, Dan Geiger. Bayesian Network Classifiers, Machine Learning, 29, 131–163. 1997.
[47] Oguz Altun, Songül Albayrak, An Evaluation of Local Interest Regions for Non-Rigid Object Class Recognition, Expert Systems with Applications 39, 2335–2340. 2012.
[48] OpenCV 3.0.0-dev documentation - Docs.opencv.org, Introduction to SURF (Speeded-Up Robust Features). [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html . [Accessed: 05- Jan- 2014].
[49] Paolo Piccinini, Andrea Prati, Rita Cucchiara, Real-time Object Detection and Localization with SIFT-based Clustering, Image and Vision Computing 30 573–587. 2012.
[50] Peng Liu, Zhijun Meng, Zhe Wu, Identification of Lateral/Directional Model for a UAV Helicopter in Forward Flight, Procedia Engineering 16 137 – 143. 2011.
[51] Pengfei Fang, Jianjiang Lu, Yulong Tian, Zhuang Miao, An Improved Object Tracking Method in UAV Videos, Procedia Engineering 15 634 – 638. 2011.
[52] Qiang Zhou, Object Detection and Recognition Via Deformable Illumination and Deformable Shape, Chrontel Inc San Jose, CA 95131. 2006.
[53] Quan Miao, Guijin Wang, A New Framework for On-Line Object Tracking Based on SURF", Pattern Recognition Letters 32, 1564–1571.2011.
[54] Quinlan JR, Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA. Diciembre 1991.
[55] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing Second Edition, Prentice Hall, Upper Saddle River, New Jersey, 2002.
[56] Richard Szeliski, Computer Vision, Algorithms and Applications, Springer, Washington, USA.2010.
[57] Sergio Davies, Ph. D. Thesis Learning in Spiking Neural Networks, University of Manchester, School of Computer Science, 2012.
[58] Simon Haykin. Neural Networks and Learning Machines, McMaster University, Canada, third Edition Prentice Hall. Noviembre 2008.

[59] T. Shivanand, Shahedur Rahman, Efficient and Robust Detection and Recognition of Objects in Grayscale Images, 978-1-4244-5967-4/10 IEEE. 2010.
[60] Turek, Fred D. Machine Vision Fundamentals, How to Make Robots See. NASA Tech Briefs 35 (6): 60–62. Junio 2011.
[61] U. Niethammer, S. Rothmund, U. Schwaderer, J. Zeman, M. Joswig, Open Source Image Processing Tools for Low Cost UAV, Landslice Investigations, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-1/C22 UAV-g. 2011.
[62] Voronoi, G. Nouvelles Applications des Paramètres Continus à la Théorie des Formes Quadratiques. Journal für die Reine und Angewandte Mathematik. 133, 97-178; 1907.
[63] Wulfram Gerstner, Werner M. Kistler, Spiking Neuron Models Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.
[64] Xi Chao-jian,Guo San-xue, Image Target Identification of UAV Based on SIFT, Procedia Engineering 15, 3205 – 3209. 2011.
[65] Xi Zhang, Chew Lim Tan, Handwritten Word Image Matching Based on Heat Kernel Signature, Computer Analysis of Images and Patterns, 15th International Conference, CAIP 2013 York, UK, August 2013.
[66] YADrone - Yet another AR.Drone framework, [Online]. Available: http://vsis-www.informatik.uni-hamburg.de/oldServer/teaching//projects/yadrone/. [Accessed: 02-Nov- 2013].
[67] Yanwei Pang, WeiLi, Fully Affine Invariant SURF for Image Matching, Neurocomputing 85 6–10. 2012.
[68] Yasir Mohd Mustafah, Amelia Wong Azman, Fajril Akbar, Indoor UAV Positioning Using Stereo Vision Sensor, Procedia Engineering 41 575 – 579. 2012.
[69] Yingcai Bi, Haibin Duan, Implementation of Autonomous Visual Tracking and Landing for a Low-Cost Quadrotor, Optik 124, 3296– 3300. 2013.
[70] Yuexing Han, Recognize Objects with Three Kinds of Information in Landmarks, Pattern Recognition 46 2860–2873.2013.
[71] Yves Dufournaud, Cordelia Schmid, and Radu Horaud, Image Matching with Scale Adjustment, Computer Vision and Image Understanding 93 175–194. 2004.

Apéndice A
Estudio de reducción de dimensionalidad de rasgos

Como se ha mencionado a lo largo del trabajo descrito en los capítulos anteriores, uno de los objetivos principales es el de poder realizar una correcta clasificación e identificación de los objetos propuestos en el menor tiempo posible. Para lograr lo anterior se propuso hacer un estudio para poder reducir la dimensionalidad de los descriptores de puntos de interés, los cuales poseen una dimensionalidad de 64 rasgos. En este punto se propuso utilizar un algoritmo estadístico de análisis de la información como lo es el algoritmo PCA (Principal Component Analysis), el cual determina los rasgos de mayor peso mediante una matriz de covarianza. La explicación del funcionamiento de este método queda fuera del alcance de este anexo, sin embargo, para mayor información puede consultar las referencias mencionadas [7].

A.1 Resultados del análisis con PCA

El análisis de las contribuciones de cada uno de los rasgos se realizó mediante la interface gráfica que proporciona la herramienta WEKA, en la cual se somete el conjunto de datos de entrenamiento presentado en la sección 4.9 al análisis mediante el algoritmo PCA. Dando como resultado una reducción de la dimensionalidad de 64 rasgos a un mínimo de 35 rasgos compuestos. Estos 35 nuevos rasgos compuestos son el resultado del análisis de la matriz de covarianza, en donde los rasgos se presentan en orden de la contribución que realizan para la correcta clasificación de las clases dadas, así los primeros 35 nuevos rasgos resultantes de este análisis se presentan a continuación.

Resultado de análisis de componentes mediante PCA		
No. Rasgo	Contribución por rasgo	Composición
1	0.8849	0.306c34+0.292c18+0.269c36+0.264c52+0.261c20+0.255c50+0.25 c4+…
2	0.7823	0.304c37+0.295c38+0.29 c21+0.289c53+0.266c22+0.265c5+0.261c54+…
3	0.6937	-0.25c10+0.25 c27+0.244c58-0.244c12+0.239c60-0.232c14-0.213c6-…
4	0.626	-0.332c55-0.32c59+0.293c11+0.286c7-0.266c51-0.256c63+0.244c15+…
5	0.5628	0.358c25+0.333c41-0.283c24-0.267c28+0.266c9+0.241c57-0.24c40-…
6	0.5018	-0.354c45-0.352c29+0.324c46-0.303c61-0.277c13+0.27 c30+0.213c62+…
7	0.4568	0.283c11+0.273c7+0.243c15+0.236c3+0.217c59-0.215c60-0.212c64+…
8	0.4161	-0.406c17-0.401c33-0.339c1-0.332c49-0.203c41-0.199c13-0.185c25-…
9	0.3804	0.31 c31-0.306c39+0.284c57-0.27c35+0.214c41-0.191c11-0.189c55+…
10	0.3498	0.259c43+0.239c42-0.237c48+0.233c58-0.215c19-0.212c44-0.211c40+…
11	0.3236	0.305c35+0.241c4+0.235c39+0.226c8-0.215c10-0.21c50-0.185c33-…
12	0.2994	0.361c35+0.335c19+0.304c47+0.266c39-0.221c53-0.211c48-0.206c29-…
13	0.2781	0.341c2-0.312c12-0.282c56+0.278c23+0.277c18-0.249c16-0.208c40-…
14	0.2578	0.445c19+0.421c23-0.28c7-0.25c51-0.239c3-0.199c64-0.192c11-…
15	0.2397	-0.377c31+0.293c9+0.244c63+0.239c25-0.21c61-0.202c30+0.199c15-…
16	0.2227	0.322c19+0.293c3-0.25c27+0.247c48-0.232c14+0.22 c59-0.218c51-…
17	0.2064	0.361c31-0.301c47-0.283c9+0.256c58-0.242c43-0.209c7-0.201c32-…
18	0.1907	0.308c15-0.292c23-0.253c64-0.233c8-0.231c62-0.21c51-0.21c24+…
19	0.1753	-0.319c57-0.305c62+0.3 c39-0.262c47-0.247c64-0.236c21-0.197c45-…
20	0.1623	-0.349c63-0.275c35+0.273c55+0.255c19+0.208c51+0.202c26+…
21	0.1504	0.38 c61-0.341c13+0.28 c47-0.27c35+0.232c45-0.222c1+0.218c49+…
22	0.1398	-0.279c49-0.262c22-0.262c13-0.254c15+0.249c1+0.232c14+0.21c17+…
23	0.1293	0.328c36-0.319c50+0.258c40+0.248c20+0.231c38+0.221c39+0.212c62-…
24	0.1189	0.314c19-0.278c52-0.24c4-0.232c57+0.214c38+0.201c18+0.2 c61+…
25	0.1102	0.436c51-0.362c59+0.236c32-0.235c4+0.229c60+0.183c5-0.183c47+…
26	0.1021	-0.308c62+0.29 c44-0.245c24-0.235c28-0.23c9+0.22 c29+0.219c15+…
27	0.0943	0.288c56+0.229c24-0.207c34+0.201c10+0.2 c2-0.199c16+0.188c11+…
28	0.087	0.387c64-0.28c46-0.253c56-0.243c61-0.237c13+0.19 c43+0.185c26-…
29	0.0802	0.326c20-0.283c15+0.271c11-0.271c3+0.226c13-0.221c29+0.184c12-…
30	0.0738	0.453c63+0.314c61+0.281c1+0.233c27-0.228c16-0.214c33+0.196c37+…
31	0.0674	0.295c49-0.264c43+0.253c56+0.23 c55+0.222c1+0.217c60-0.213c50-…
32	0.0621	0.333c38+0.303c8-0.259c5+0.258c48+0.245c46+0.236c22+0.221c30+…
33	0.057	-0.353c49-0.273c14+0.233c26-0.223c58+0.221c63+0.217c22+0.204c23+…
34	0.0521	0.378c40-0.297c24+0.283c51-0.236c55+0.207c62+0.197c12-0.189c6+…
35	0.0475	-0.322c32+0.249c60+0.215c18-0.213c1-0.206c10+0.205c31-0.204c47+…

Tabla A.1 Matriz de reducción de dimensionalidad de rasgos SURF.

Como se puede observar de la tabla anterior, los 35 rasgos compuestos garantizan una correcta clasificación de hasta un 95.37%, obviamente entre mayor cantidad de rasgos se elijan, mayor será el porcentaje de clasificación.

Dadas las características de los rasgos anteriores, procedimos a implementar un conjunto de pruebas con el banco de imágenes presentadas en la sección 4.9. Corroborando que efectivamente entre menor número de rasgos se tome para la composición de los 35 nuevos rasgos, menor será el porcentaje de clasificación y de lo contrario entre mayor sea el número de rasgos originales implicados en el cálculo de los nuevos rasgos el porcentaje de clasificación será mayor. A

continuación se presenta la tabla comparativa entre número de rasgos y el porcentaje correcto de clasificación.

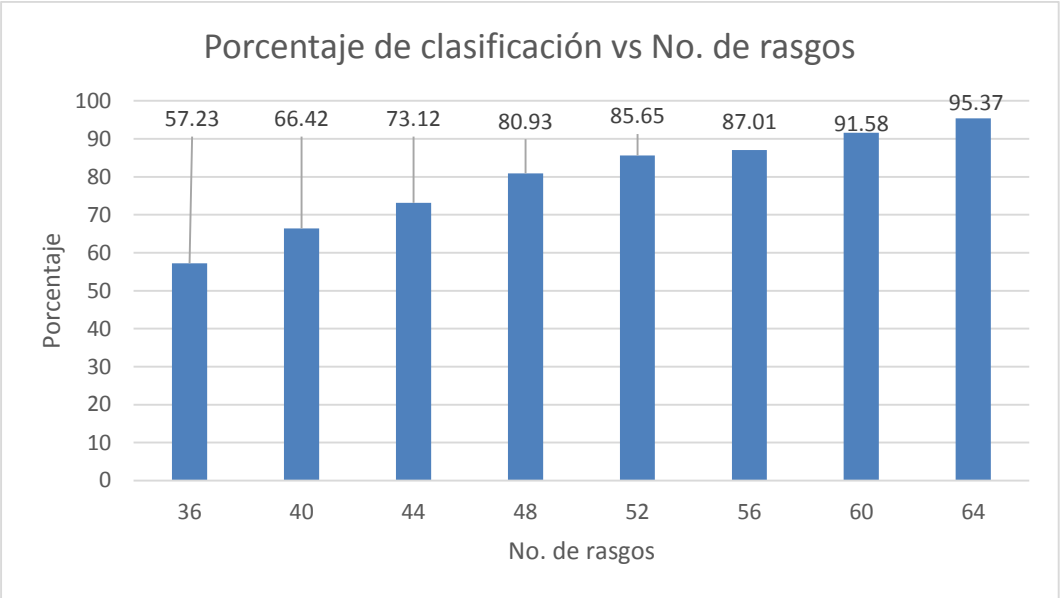

Figura A.1 Grafica de porcentajes de clasificación de acuerdo al número de rasgos seleccionados.

La tabla A.1 se obtuvo analizando un conjunto de 50 imágenes de prueba.

La inclusión de este proceso sobre el algoritmo de procesamiento de información presentado en el capítulo 4 sección 12 se dejó de lado debido a que la inclusión de este procedimiento requería de un bloque extra de procesamiento de información pre clasificación, así como la perdida de varios puntos porcentuales en el porcentaje de clasificación sin mencionar el incremento en el tiempo de respuesta del proceso en general. Sopesando lo puntos anteriores, se presenta este análisis como anexo para futuras referencias y consideraciones.

Apéndice B
Estudio de clasificación del conjunto de objetos para fondos complejos

Como parte del estudio realizado en cuanto a la robustez de los descriptores de puntos de interés, no solo se realizaron pruebas de clasificación con fondos contrastados de color negro, también se tomó otro tipo de fondo más complejo, el cual ofrecía una mayor complejidad de tratamiento. Y realizando el mismo proceso de extracción de descriptores de puntos de interés, entrenamiento de redes neuronales y clasificación de los mismos, se obtuvieron los resultados de clasificación que se muestran en la siguiente tabla.

Tabla comparativa de porcentajes de clasificación para fondos complejos				
Clasificador	Objeto 1	Objeto 2	Objeto 3	Objeto 4
MLP	91.50	88.31	89.54	88.70
SVM (Lineal)	86.89	80.43	81.53	77.91

Tabla B.1 Porcentajes de clasificación con diferentes clasificadores para fondos complejos.

Los dos clasificadores que se presentan en la tabla anterior fueron seleccionados debido a que estos últimos fueron los clasificadores que mejores resultados arrojaron (consultar capítulo 5 de resultados).

La validación gráfica de la clasificación de los descriptores de puntos de interés, se realizó de igual forma que en los experimentos anteriores, con un conjunto de 50 imágenes por objeto. Como se

muestra en la tabla anterior y se ratifica en el siguiente conjunto de imágenes que los porcentajes de clasificación son menores que los obtenidos utilizando un fondo contrastado, esto debido a que el volumen de puntos detectado es mayor en este conjunto de imágenes de fondos complejo, y su separabilidad lineal es menor. El promedio de puntos de interés detectados en las imágenes con fondo complejo asciende a 450 puntos de interés por imagen, mientras que el número promedio de puntos de interés detectado en las imágenes de fondo negro contrastado es aproximadamente de 120 puntos de interés por imagen, lo cual influye directamente en los tiempos de respuesta, es decir, a mayor cantidad de puntos a clasificar y filtrar, mayor será la cantidad de tiempo consumido por el proceso. Debido a lo anterior es que se decidió solamente presentar este estudio como sustento de que es posible realizar la clasificación de los objetos sobre fondos complejos, sin llegar a implementar esta clasificación en el programa final, ya que con la gran cantidad de puntos detectados, aunando las características del sistema de cómputo no se lograba realizar el procesamiento en tiempo real, y el procesamiento se ejecutaba con un retardo de aproximadamente de 2 a 4 (dependiendo la imagen de entrada) segundos con respecto al video original.

A continuación se presenta una muestra del conjunto de imágenes sobre la cual se validaron los porcentajes de clasificación de los descriptores de puntos de interés sobre fondos complejos.

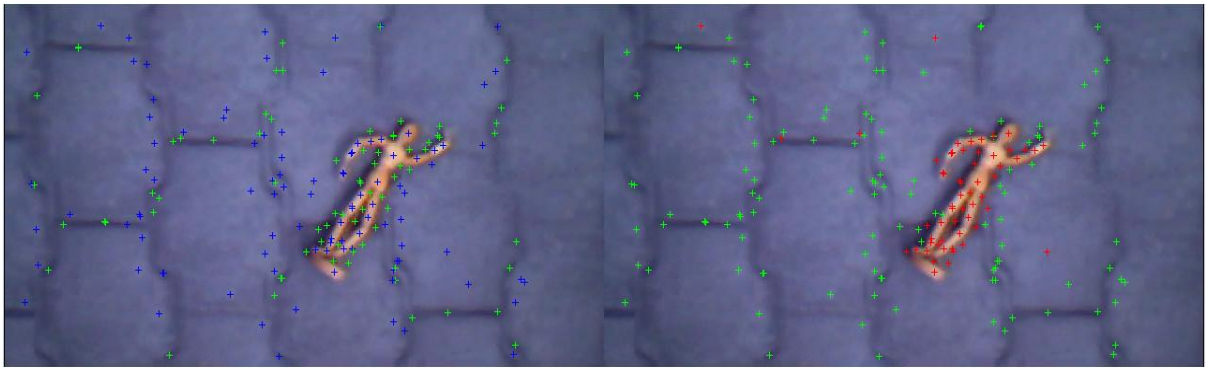

Figura B.1 Primer imagen de clasificación del Objeto 1 con fondo complejo.

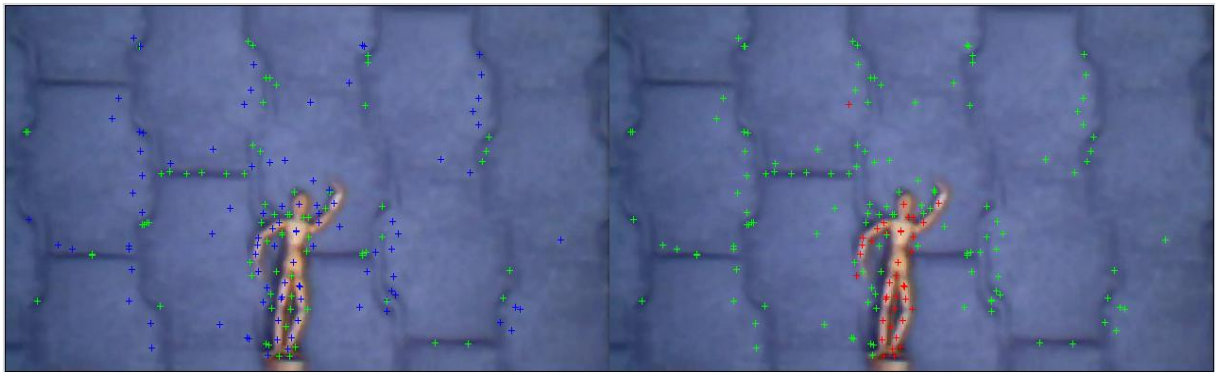

Figura B.2 Segunda imagen de clasificación del Objeto 1 con fondo complejo.

Como se observa en el par de imágenes anteriores, el porcentaje de clasificación de los puntos de interés (más de 400 puntos de interés) es mayor al 90%, pero como ya se mencionó, la clasificación de todos los puntos de interés detectados para imágenes con fondos complejos toma entre 1 y 2 segundo en ser completada.

A continuación, en la figura B.3, se muestran los resultados de la clasificación con fondo complejo para el Objeto 2:

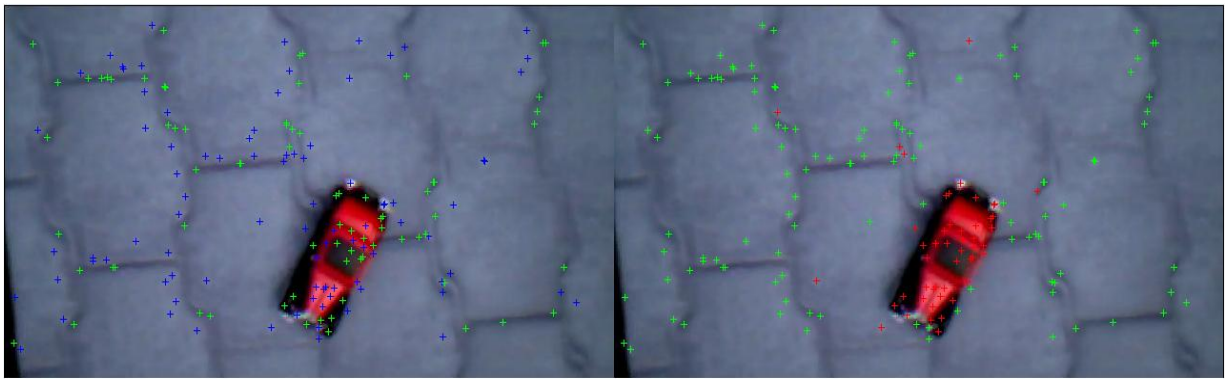

Figura B.3 Primer imagen de clasificación del Objeto 2 con fondo complejo.

En la figura B.4 se muestran resultados interesantes, en el sentido de que el porcentaje de clasificación se ve mínimamente afectado por los cambios de luminosidad que se observan entre la imagen B.3 y B.4. De igual forma que para el Objeto 1, los porcentajes de clasificación son muy confiables.

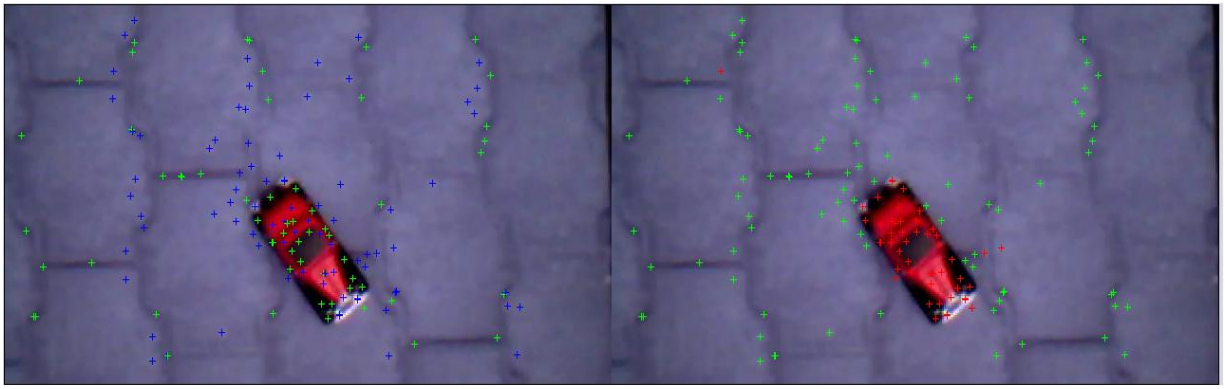

Figura B.4 Segunda imagen de clasificación del Objeto 2 con fondo complejo.

A continuación, en las figuras B.5 y B.6, se muestran los resultados de la clasificación con fondo complejo para el Objeto 3.

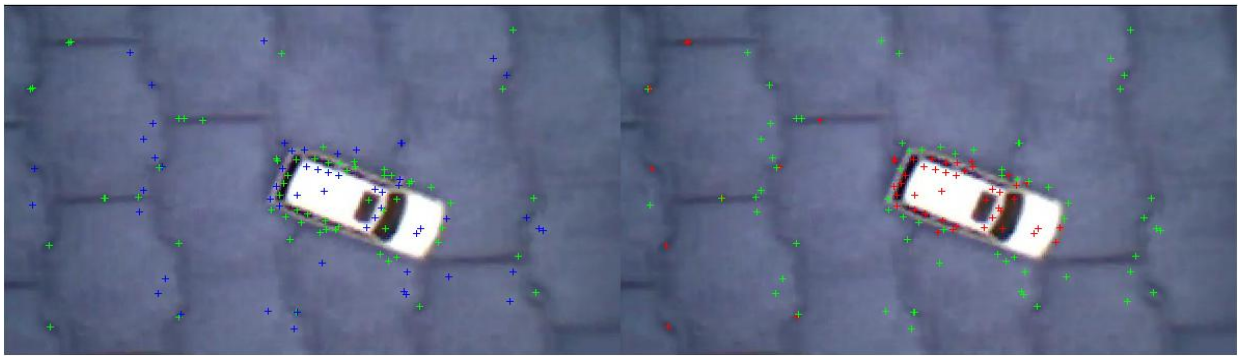

Figura B.5 Primer imagen de clasificación del Objeto 3 con fondo complejo.

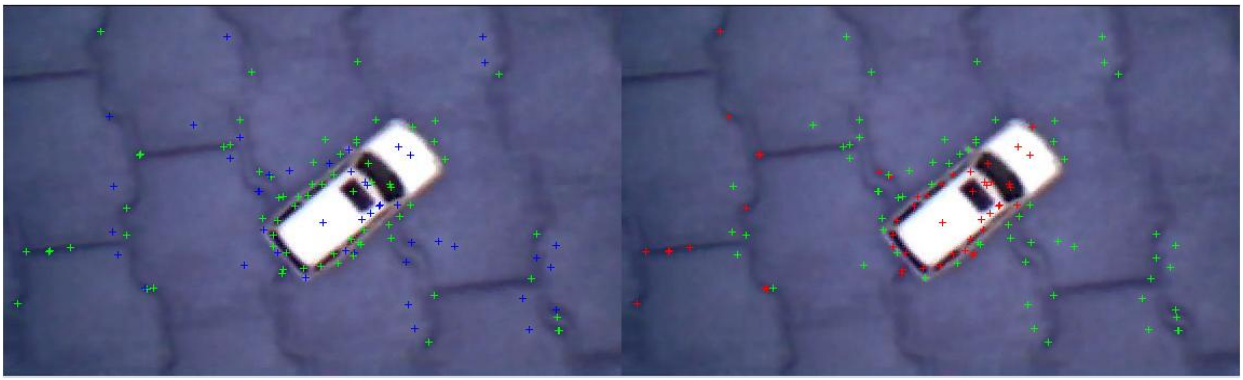

Figura B.6 Segunda imagen de clasificación del Objeto 3 con fondo complejo.

Como se observa en las imágenes anteriores para el Objeto 3, el porcentaje de errores de clasificación se incrementa, aun así, es posible clasificar los descriptores de puntos de interés en un alto porcentaje.

Como último objeto de prueba, tenemos al Objeto 4, cuyas imágenes de prueba de clasificación con fondos complejos se muestran a continuación.

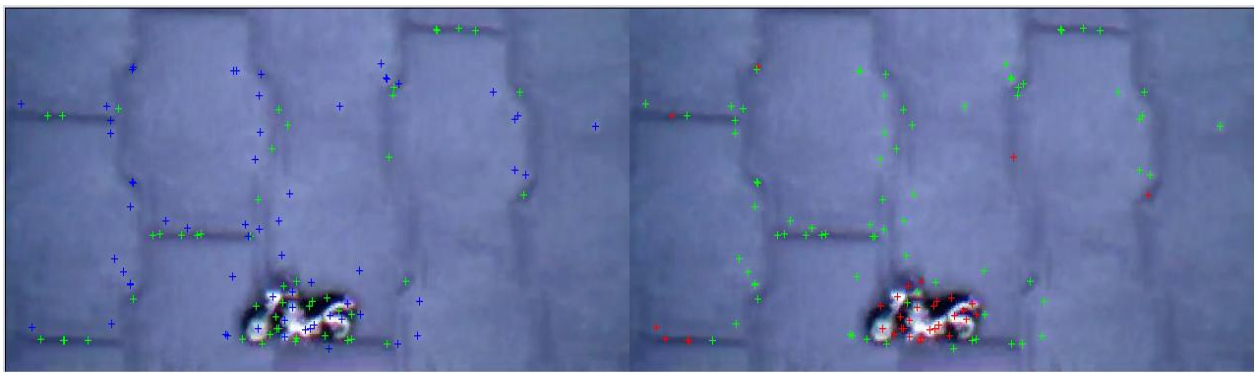

Figura B.7 Primer imagen de clasificación del Objeto 4 con fondo complejo.

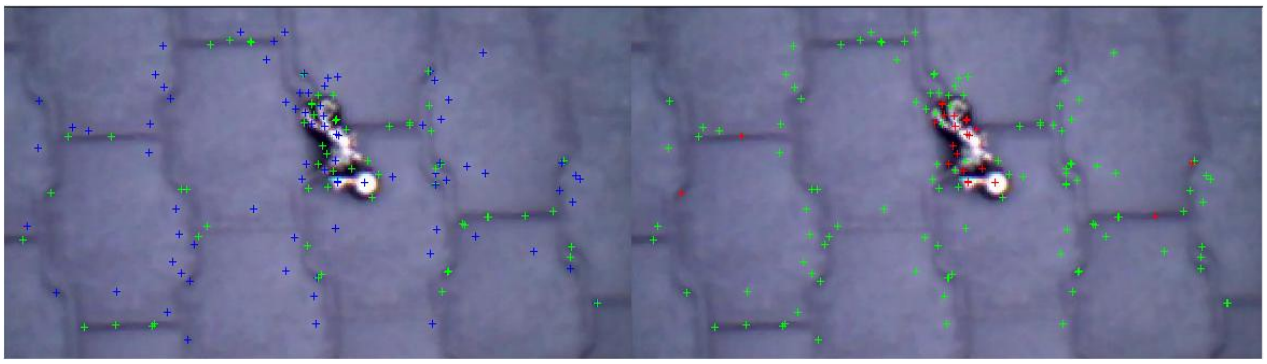

Figura B.8 Segunda imagen de clasificación del Objeto 4 con fondo complejo.

Como resultado del análisis anterior podemos concluir y afirmar que la clasificación de objetos con fondos complejos utilizando técnicas como lo es la detección de puntos de interés puede realizarse de forma altamente confiable, con las correctas etapas de pre procesamiento y filtrado. Sin embargo, esto también requerirá de un mayor poder de cómputo si lo que se desea es realizar la clasificación en tiempo real como hoy en día lo requieren varias aplicaciones en robótica.

Apéndice C
Análisis por histogramas y su clasificación lineal

En este capítulo se realiza un análisis de la factibilidad en cuanto a clasificación se refiere, por medio del análisis de histogramas a niveles de gris y en tres canales (RGB), así como su aplicación para realizar la clasificación lineal de los escenarios que se describieron en el capítulo 4 (Metodología). Para realizar lo anterior se analizan los histogramas de las imágenes de los escenarios propuestos en la sección 4.14, los cuales contienen desde 1 hasta 4 objetos.

C.1 Análisis de histogramas a niveles de gris para el conjunto de objetos a clasificar

En esta sección se extraen y analizan los histogramas de las imágenes de los escenarios propuestos en la sección 4.14. Iniciando el análisis con el escenario más simple, el cual solo contiene un objeto, hasta llegar al escenario más complejo el cual contiene 4 objetos en la misma imagen.

De tal forma que en la siguiente secuencia de imágenes se muestra del lado izquierdo la imagen original a analizar, junto con el resultado de clasificación mediante un umbralado a niveles de gris, y del lado derecho se muestra su respectivo histograma en niveles de gris.

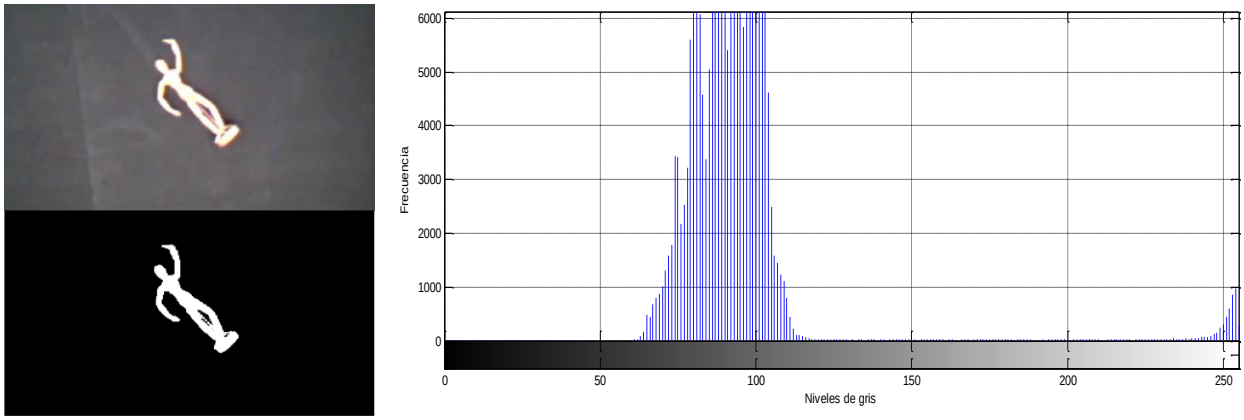

Figura C.1 Clasificación del Objeto 1 por medio de histograma a niveles de gris.

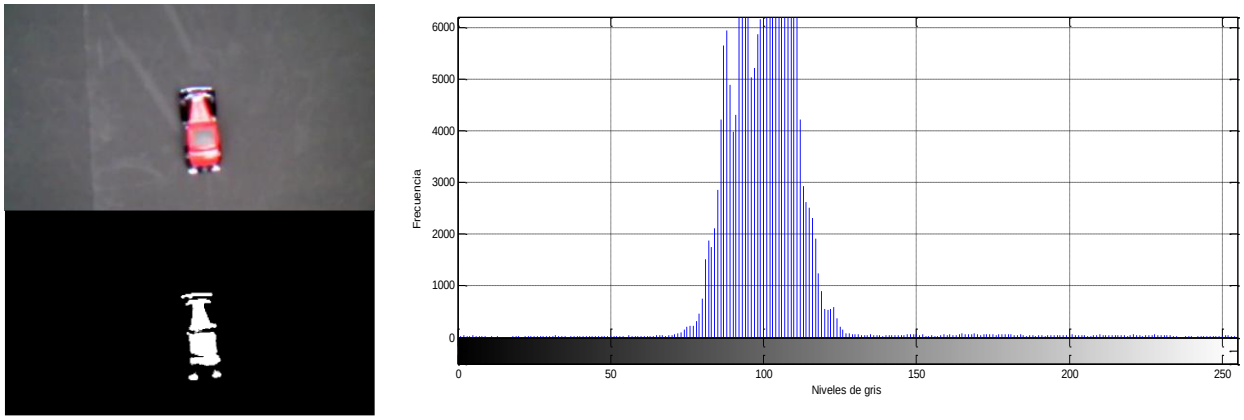

Figura C.2 Clasificación del Objeto 2 por medio de histograma a niveles de gris.

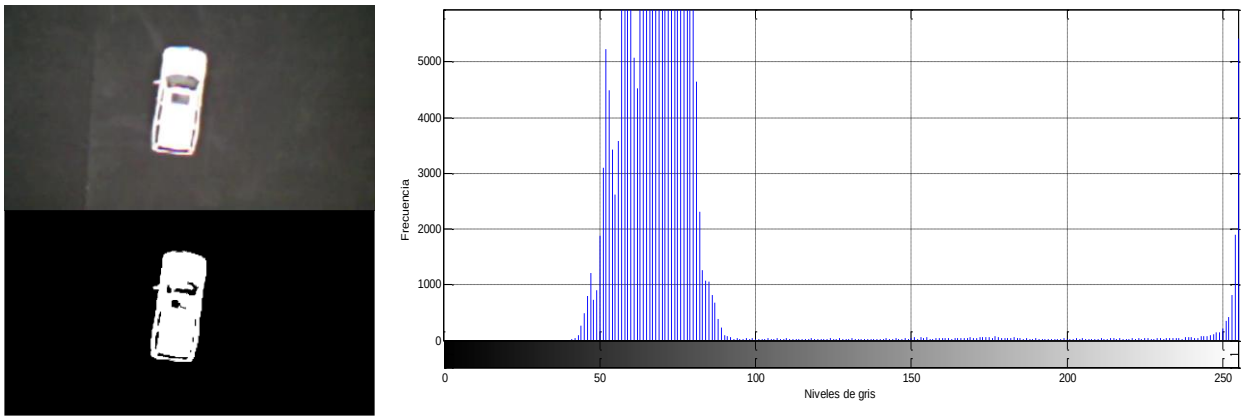

Figura C.3 Clasificación del Objeto 3 por medio de histograma a niveles de gris.

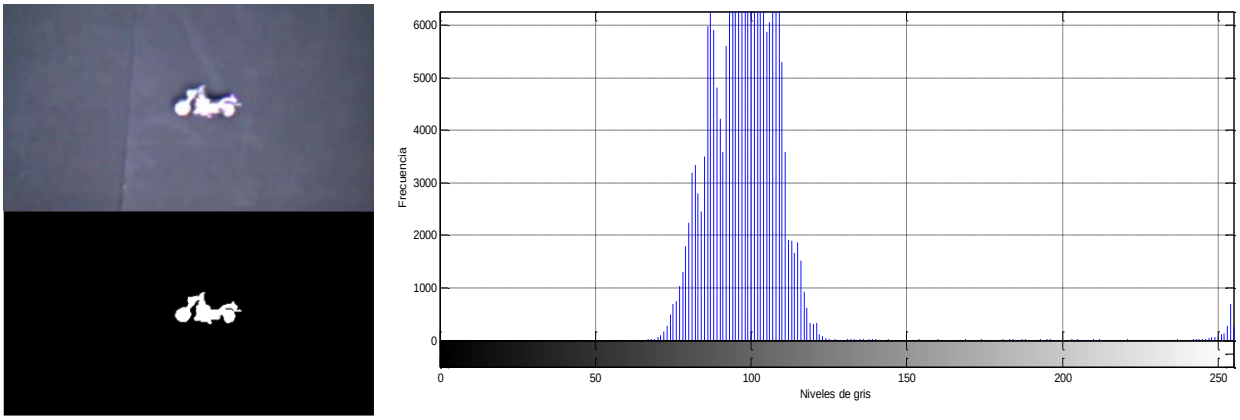

Figura C.4 Clasificación del Objeto 4 por medio de histograma a niveles de gris.

Del análisis de las imágenes anteriores (C.1 – C.4) se observa que para los objetos 1,3 y 4 la clasificación mediante histogramas a niveles de gris es posible, sin embargo para el Objeto 2 este tipo de clasificación resulta un poco deficiente. Cabe mencionar que las imágenes no son sujetas a ningún pre procesamiento de filtrado que ayude a resaltar las características de los objetos. Las imágenes anteriores son los escenarios más simples. En el siguiente conjunto de imágenes se analizan escenarios más complejos con 2,3 y 4 objetos en la misma escena.

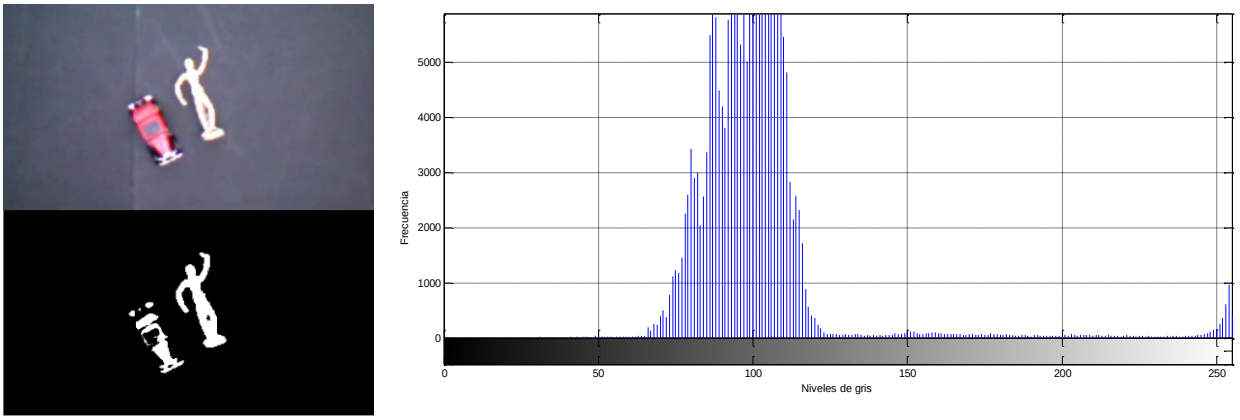

Figura C.5 Clasificación de 2 objetos en la misma escena por medio de histograma a niveles de gris.

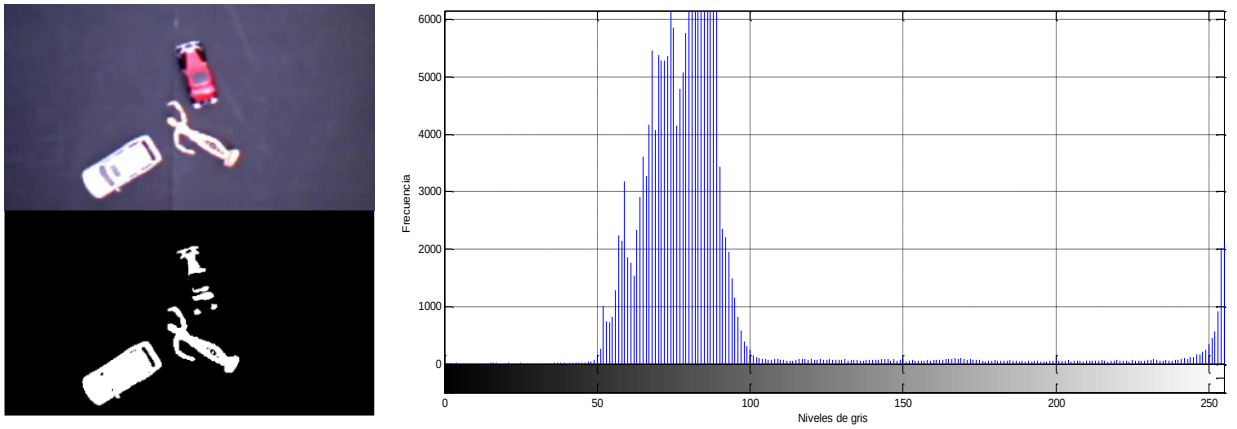

Figura C.6 Clasificación de 3 objetos en la misma escena por medio de histograma a niveles de gris.

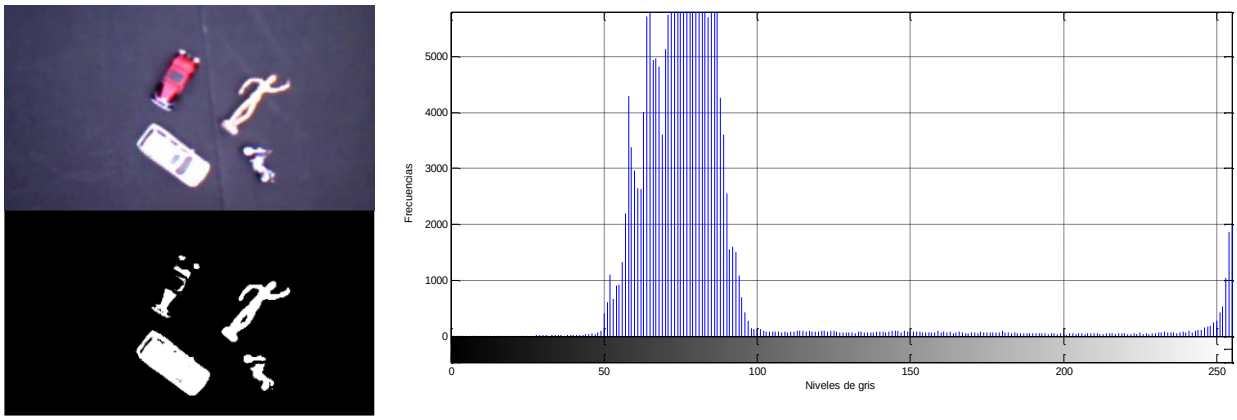

Figura C.7 Clasificación de 4 objetos en la misma escena por medio de histograma a niveles de gris.

Del análisis de las imágenes anteriores (C.5 – C.7) se observa que la metodología de clasificación por histogramas a niveles de gris, para las imágenes de los escenarios con más de 2 objetos, no arroja buenos resultados en cuanto a clasificación e identificación de objetos se refiere, por lo que esto nos sugiere adoptar una metodología de extracción de rasgos más compleja, como lo es la aplicación de histogramas en canales RGB, como se muestra en la siguiente sección.

C.2 Análisis de histogramas RGB para el conjunto de objetos a clasificar

En esta sección se realiza el mismo análisis que en la sección anterior con la diferencia de que los histogramas se calculan con base a los tres diferentes canales que componen la imagen. En la siguiente secuencia de imágenes se muestra dicho análisis.

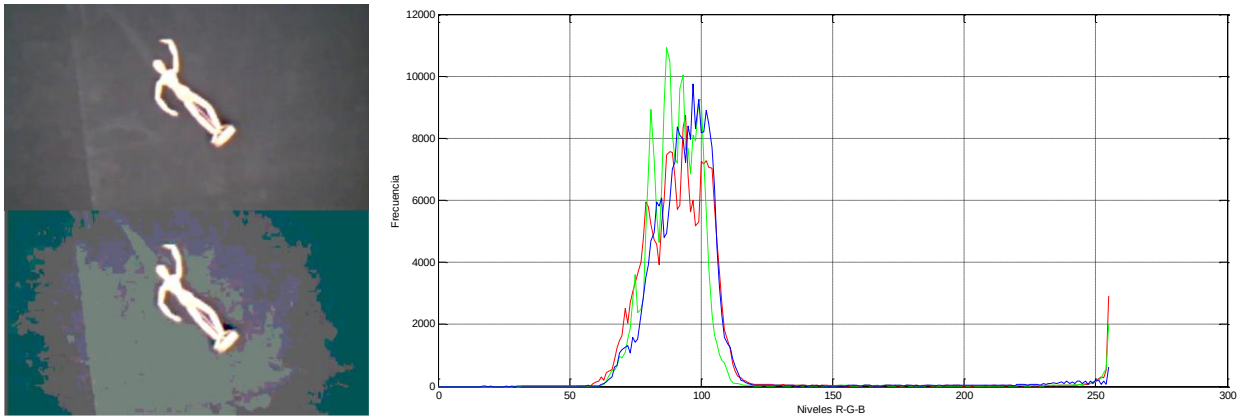

Figura C.8 Clasificación del Objecto1 por medio de histograma en canales RGB.

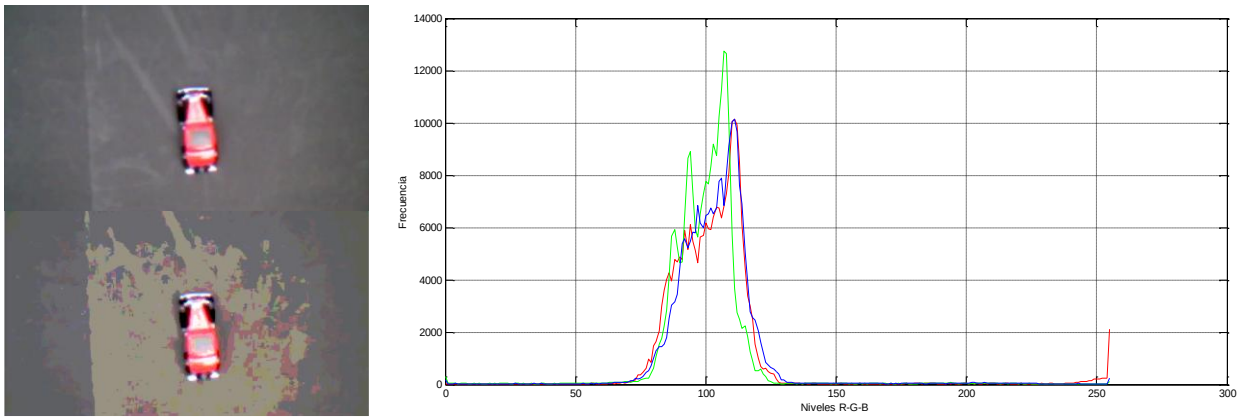

Figura C.9 Clasificación del Objeto 2 por medio de histograma en canales RGB.

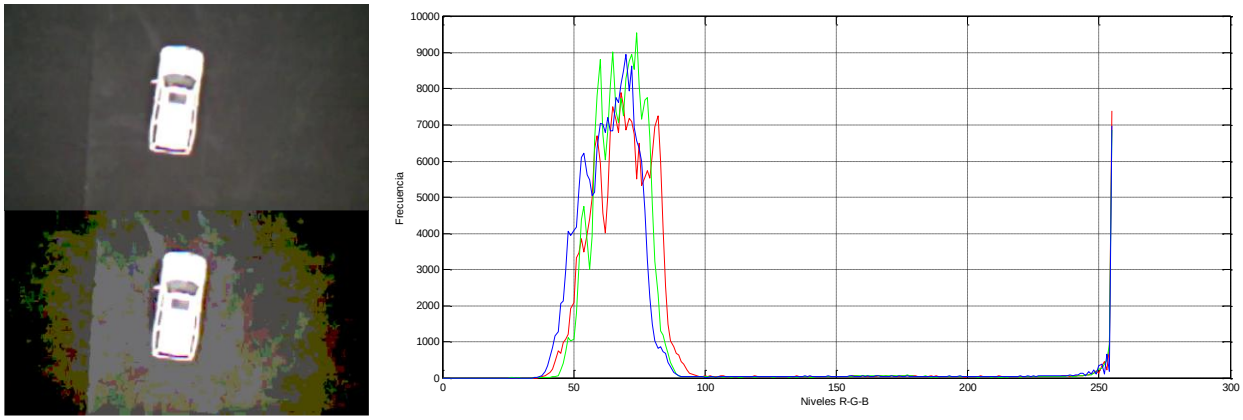

Figura C.10 Clasificación del Objeto 3 por medio de histograma en canales RGB.

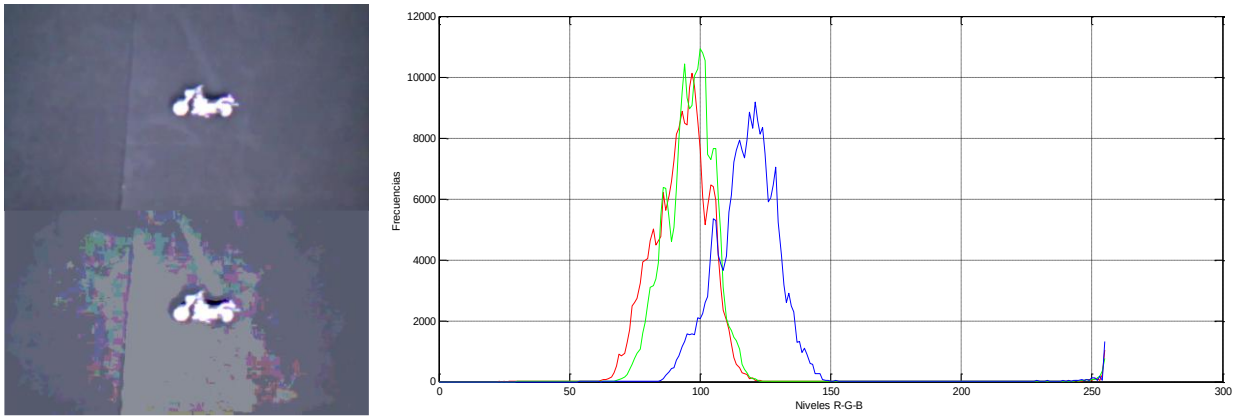

Figura C.11 Clasificación del Objeto 4 por medio de histograma en canales RGB.

De nueva cuenta cabe mencionar, que las imágenes no son sometidas a algún tipo de pre procesamiento o filtrado que resalte a los objetos contenidos en cada imagen. Como se observa en la secuencia de imágenes anterior (C.8-C.11), los niveles de umbralado en los canales RGB para todos los objetos pueden ser establecidos de tal forma que se aísle el objeto para escenarios simples (un objeto por escena). Por lo que aplicando la misma metodología que en la sección anterior se analizan imágenes de escenarios más complejos con 2,3 y 4 objetos por escena.

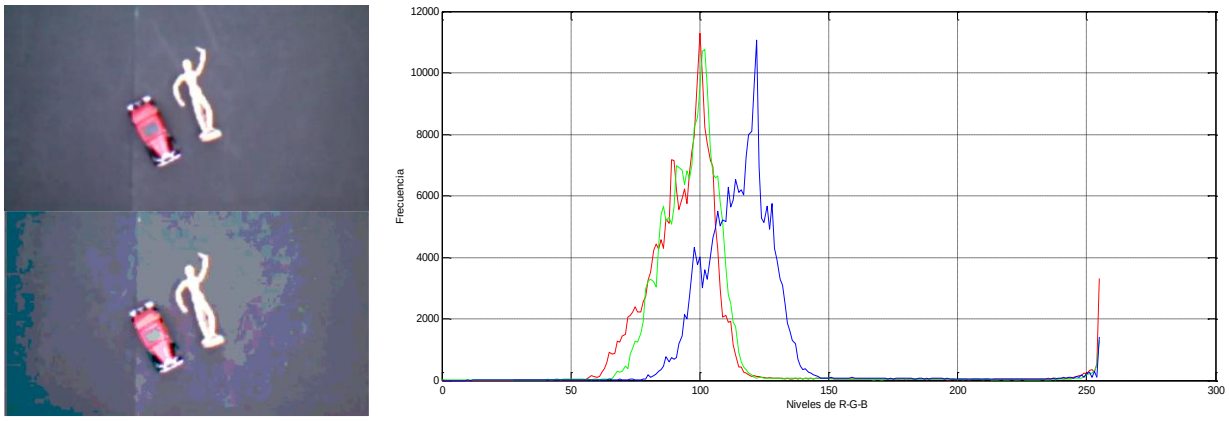

Figura C.12 Clasificación de 2 objetos en la misma escena mediante histograma en canales RGB.

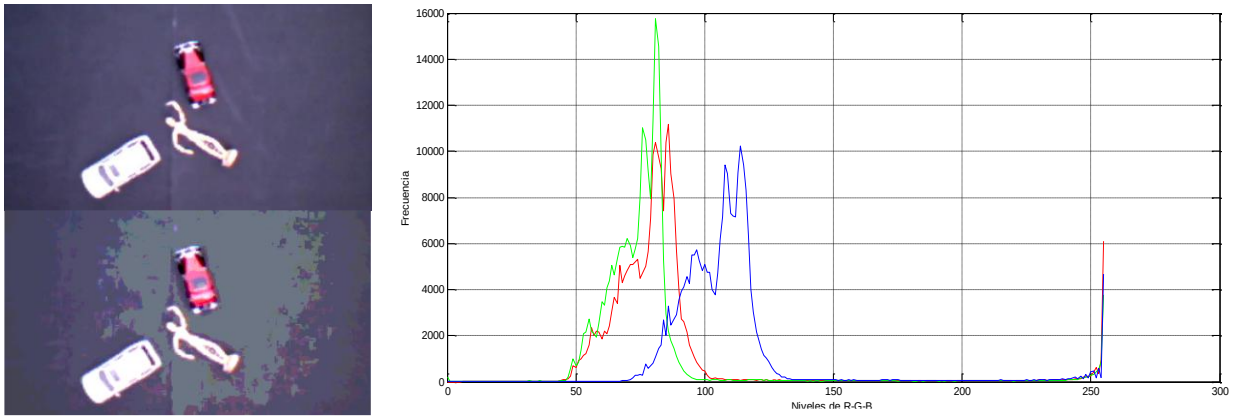

Figura C.13 Clasificación de 3 objetos en la misma escena mediante histograma en canales RGB.

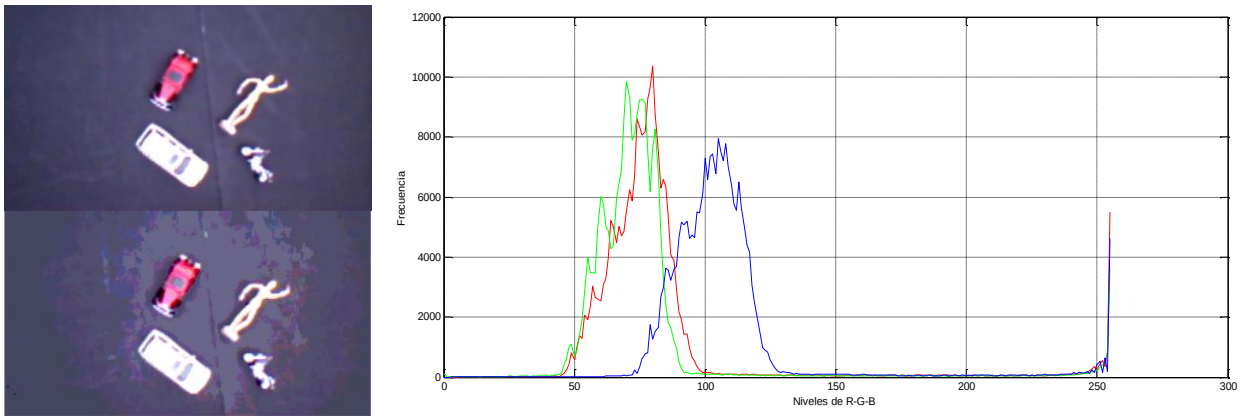

Figura C.14 Clasificación de 4 objetos en la misma escena mediante histograma en canales RGB.

De igual forma que en el análisis de la sección C.1, de la secuencia de imágenes C.8-C.14 se puede determinar que la clasificación de objetos mediante el uso de histogramas en varios canales es factible para escenarios complejos, aunque para escenarios con 3 y 4 objetos esta técnica resulta un tanto deficiente.

Por lo que en conclusión, podemos afirmar que la clasificación de objetos realizando un análisis por histogramas en los canales RGB, es factible, sin embargo cabe mencionar que este proceso de clasificación se tendría que complementar con otras técnicas, como lo es, el análisis de regiones conexas, calibración de la cámara de video y un ambiente controlado de iluminación, esto debido a que los objetos 1,3 y 4 comparten el mismo espectro de colores.

Apéndice D
Código fuente de los principales módulos

En el presente y último capítulo se presentan y describen los principales métodos en código fuente, los cuales son el punto medular para poder entender y replicar el presente trabajo. Cabe mencionar que el código fuente se encuentra en el lenguaje de programación JAVA y se requiere como mínimo la versión 5 del SDK para poder ser ejecutada de forma correcta. El orden de presentación de los métodos es el orden en el cual se ejecutan dentro del proceso de clasificación.

El primer método es el método de extracción de puntos de interés y sus descriptores dada una imagen en forma de objeto *BufferedImage* y su previa inicialización. El método de inicialización del detector SURF se muestra a continuación:

```
1     public static void initializeFeatureSURfDetector(){
2           if ( Constants.detDesc == null)
3                 Constants.detDesc = FactoryDetectDescribe.surfStable(
                  new ConfigFastHessian(5, 2, 200, 2, 9, 4, 4), null,null,
                  ImageFloat32.class);
4           if ( Constants.scorer == null )
5                 Constants.scorer =
                  FactoryAssociation.defaultScore(Constants.detDesc.getDescriptionT
                  ype());

6           if ( Constants.associate == null)
7                 Constants.associate = FactoryAssociation.greedy(Constants.scorer,
                  Double.MAX_VALUE, true);
8     }
```

En el método anterior se observa cómo se inicializa el detector de puntos de interés mediante la línea:

```
1      FactoryDetectDescribe.surfStable(
```

```
2            new ConfigFastHessian(5, 2, 200, 2, 9, 4, 4), null,null,
             ImageFloat32.class);
```

En la cual se observan los parámetros de configuración para la matriz de Hessian. Para obtener una mayor referencia de estos parámetros consulte la referencia [11].

Una vez inicializado el detector de puntos de interés, se procede a inicializar el clasificador, que en nuestro caso es una red neuronal de tipo MLP, para realizar lo anterior nos basamos en la implementación de los modelos serializados que nos proporciona la interface WEKA, de tal forma que el método de inicialización del clasificador es el siguiente:

```
1      public static void  initialize( String pathModel){
2            try{
3                  Constants.generalClassifyer = (Classifier)
                   SerializationHelper.read( pathModel );

4                  CSVLoader csvLoaderTest = new CSVLoader();
5                  csvLoaderTest.setSource( new File(
                   Constants.ANN_MODEL_PATH_INSTANCE ));

6                  Constants.testInstances = csvLoaderTest.getDataSet();

7                  Constants.testInstances.setClassIndex(
                   Constants.testInstances.numAttributes() -1 );

8            }catch( Exception e ){
9                  e.printStackTrace();
10           }
11     }
```

En el método anterior de la clase *MyClassifier*, el método *initialize*, lee la configuración de la red neuronal pre entrenada del archivo especificado por la variable *`pathModel`*. También inicializa la variable *`CSVLoader csvLoaderTest`*, la cual es necesaria para el proceso de clasificación.

A continuación se instancia el proceso asíncrono el cual consumirá la cola FIFO de objetos BufferedImage, la ejecución del proceso asíncrono se realiza mediante las siguientes líneas:

```
1            AnalasysThread at = new AnalasysThread();
2            at.start();
```

La implementación del proceso asíncrono (que es el centro de nuestra aplicación) se muestra a detalle a continuación:

```
1     @Override
2     public void run() {
3           // se inicializa el detector e Feture Points
4           DIP.initializeFeatureSURfDetector();
5     while ( Constants.isThreadRunning ){
6           // se hace pop de la cola de imagenes
7           BufferedImage buffImg = Constants.getSetBufferedImageQueue(null, false);
8           if (buffImg != null){
9                 List<Point2D_F64> points = new ArrayList<Point2D_F64>();
10                FastQueue<SurfFeature> descriptions = null;
11                List<Double> listClasses = new ArrayList<Double>();

12                // se calculan los Feature Points y sus descriptores
13                descriptions = DIP.harderComparator( buffImg, points);

14                // se clasifica cada uno de los descriptores
15                for (int i =0; i < descriptions.size(); i++) {
16                      SurfFeature sf = descriptions.get(i);
17                      double klass = MyClasiffier.clasify(  sf.getValue() );
18                      listClasses.add( klass );
19                }

20                //se filtra por desviación estandar cada uno de los descroptores
21                // preclasificados
22                Utils.filterPointByStdDeviation ( points, listClasses);
23                List<List<Point2D_F64>> listOfList= Utils.getListByClasses(
                  points, listClasses );


24                Graphics2D g2D =  buffImg.createGraphics();

25                // se dibuja su localizacion y clasificacion
26                if (Constants.saveImage ){
27                      GUIUtils.calcularSquareObject(g2D, listOfList.get(0), 0.0
                        , Color.RED);
28                      GUIUtils.calcularSquareObject(g2D, listOfList.get(1), 1.0
                        , Color.GREEN);
29                      GUIUtils.calcularSquareObject(g2D, listOfList.get(2), 2.0
                        , Color.BLUE);
30                      GUIUtils.calcularSquareObject(g2D, listOfList.get(3), 3.0
                        , Color.BLACK);

31                      GUIUtils.drawPointsClassified_Point2D(g2D, points,
                        listClasses);

32                }else{
33                      List<Point2D_F64> listKlassTemp = new
                        ArrayList<Point2D_F64>();

34                      GUIUtils.calcularSquareObject(g2D, listKlassTemp, 0.0,
                        Color.RED);

35                      GUIUtils.calcularSquareObject(g2D, listKlassTemp, 1.0,
                        Color.GREEN);
```

```
36                          GUIUtils.calcularSquareObject(g2D, listKlassTemp, 2.0,
                            Color.BLUE);

37                          GUIUtils.calcularSquareObject(g2D, listKlassTemp, 3.0,
                            Color.BLACK);
38                  }

39              // se actuaziza la interface grafica
40              GUIUtils.redrawImageIcondAndCounters();
41              Constants.setImageArDroneProcess( buffImg);

42              Constants.jVideoControlPanel.videoProcessJPanel.repaint();
43              }
44      }
45 }
```

Como se observa en el método anterior, en la ejecución de la siguiente línea:

```
1       BufferedImage buffImg = Constants.getSetBufferedImageQueue(null, false);
```

Que es de donde se obtiene la imagen de la cola FIFO, en la secuencia de llegada, posteriormente se extraen los puntos de interés y sus descriptores en la siguiente línea de código:

```
1       descriptions = DIP.harderComparator( buffImg, points);
```

Donde los descriptores estarán contenidos en la variable *descriptions* y los puntos de interés en la variable *Points*.

Ya que obtuvimos los puntos de interés y sus respectivos descriptores, procedemos a realizar la clasificación de cada uno de estos mediante la red neuronal pre entrenada con el siguiente código:

```
1               for (int i =0; i < descriptions.size(); i++) {
2                       SurfFeature sf = descriptions.get(i);
3                       //clasifica cada descriptor
4                       double klass = MyClasiffier.clasify(  sf.getValue() );
5                       listClasses.add( klass );
6               }
```

En el código anterior la línea ***double*** *klass = MyClasiffier.clasify(sf.getValue())*, clasifica el descriptor obtenido por *sf.getValue()* y nos regresa el número de índice de clase a la cual pertenece dicho descriptor.

Una vez que se obtienen los índices de clases para cada descriptor de puntos de interés, estos se filtran mediante el filtro espacial por desviación estándar, con la siguiente linera de código:

```
1            Utils.filterPointByStdDeviation (points, listClasses);
```

El cual nos regresa una sub lista de los puntos filtrados en la misma variable *points*. Una vez filtrados los puntos de interés, se procede a calcular el área del objeto y dibujar su clasificación mediante recuadros de color. El código es el siguiente:

```
1            Graphics2D g2D =  buffImg.createGraphics();

2            if (Constants.saveImage ){
3                  GUIUtils.calcularSquareObject(g2D,   listOfList.get(0),   0.0   ,
                   Color.RED);
4                  GUIUtils.calcularSquareObject(g2D, listOfList.get(1), 1.0 ,
                   Color.GREEN);
5                  GUIUtils.calcularSquareObject(g2D, listOfList.get(2), 2.0 ,
                   Color.BLUE);
6                  GUIUtils.calcularSquareObject(g2D, listOfList.get(3), 3.0 ,
                   Color.BLACK);

7                  GUIUtils.drawPointsClassified_Point2D(g2D, points, listClasses);
8            }
```

Los métodos de filtrado especial por desviación estándar se muestran a continuación:

```
1      public static void filterPointByStdDeviation ( List<Point2D_F64> pointsRef,
       List<Double> listClasses){

2      // se obtien una sublista por klase de clasificación identificada
3      List<List<Point2D_F64>> listOfListKlasses = getListByClasses( pointsRef,
       listClasses );

4      List<Point2D_F64> listKlass0 = listOfListKlasses.get(0);
5      List<Point2D_F64> listKlass1 = listOfListKlasses.get(1);
6      List<Point2D_F64> listKlass2 = listOfListKlasses.get(2);
7      List<Point2D_F64> listKlass3 = listOfListKlasses.get(3);

8      // se obtiene el promedio de la clase
9      double meanKlass0[] = Utils.getMean( listKlass0);
```

```
10    // se obtiene la desviación estándar de la clase
11    double standDeviationK0[] = Utils.getStandardDeviation( meanKlass0,
      listKlass0);

12    double meanKlass1[] = Utils.getMean( listKlass1);
13    double standDeviationK1[] = Utils.getStandardDeviation( meanKlass1,
      listKlass1);

14    double meanKlass2[] = Utils.getMean( listKlass2);
15    double standDeviationK2[] = Utils.getStandardDeviation( meanKlass2,
      listKlass2);

16    double meanKlass3[] = Utils.getMean( listKlass3);
17    double standDeviationK3[] = Utils.getStandardDeviation( meanKlass3,
      listKlass3);

18    ArrayList<Integer> idx = new ArrayList<Integer>();


19    for ( int i = 0; i  < pointsRef.size(); i++){

20          Point2D_F64 point =  pointsRef.get(i);

21          // se analiza si el punto cae dentro del rango de la desviación estándar
22          // por cada clase identificada

23          switch ( listClasses.get(i).intValue() ) {
24                case 0:
25                      if( !(meanKlass0[0] - standDeviationK0[0] <= point.getX()
                        &&  point.getX() <= meanKlass0[0] + standDeviationK0[0] )
                        || !(meanKlass0[1] - standDeviationK0[1] <= point.getY()
                        && point.getY() <= meanKlass0[1] + standDeviationK0[1] ) )

26                      idx.add( i);

27                      break;

28                case 1:
29                      if( !(meanKlass1[0] - standDeviationK1[0] <= point.getX()
                        && point.getX() <= meanKlass1[0] + standDeviationK1[0] )
                        || !(meanKlass1[1] - standDeviationK1[1] <= point.getY()
                        && point.getY() <= meanKlass1[1] + standDeviationK1[1]))

30                      idx.add( i);

31                      break;

32                case 2:
33                      if( !(meanKlass2[0] - standDeviationK2[0] <= point.getX()
                        && point.getX() <= meanKlass2[0] + standDeviationK2[0] )
                        || !(meanKlass2[1] - standDeviationK2[1] <= point.getY()
                        && point.getY() <= meanKlass2[1] + standDeviationK2[1] ))

34                      idx.add( i);
```

```
35                      break;

36                case 3:
37                      if( !(meanKlass3[0] - standDeviationK3[0] <= point.getX()
                        && point.getX() <= meanKlass3[0] + standDeviationK3[0] )
                        || !(meanKlass3[1] - standDeviationK3[1] <= point.getY()
                        && point.getY() <= meanKlass3[1] + standDeviationK3[1] ))

38                      idx.add( i);

39                      break;

40                default:
41                      break;
42                }
43          }

44          for(int i = idx.size()-1 ; i >  0; i--){
45                if(  idx.get(i) <  pointsRef.size()){
46                      pointsRef.remove( idx.get(i).intValue() );
47                      listClasses.remove(  idx.get(i).intValue() );
48                }
49          }
50    }

51    // calculo de desviación estandar por lista de coordenadar de los Feature
      Points
52    public static double[] getStandardDeviation( double mean[], List<Point2D_F64>
      coordinatesList ){
53    double variance[] = new double[2];

54    for ( Point2D_F64 coord: coordinatesList){
55          variance[0] =  variance[0] +  Math.pow((  coord.getX() - mean[0]),2);
56          variance[1] =  variance[1] +  Math.pow((  coord.getY() - mean[1]), 2);
57          }

58    variance[0] =  variance[0] / coordinatesList.size();
59    variance[1] =  variance[1] / coordinatesList.size();

60    variance[0] =  Math.sqrt( variance[0]);
61    variance[1] =  Math.sqrt( variance[1]);

62    return variance;
63     }

64    // calculo del promedio por coordenadas de los Feature Points
65    public static double[] getMean ( List<Point2D_F64> coordinatesList ){
66    double mean[] = new double[2];

67    for( Point2D_F64 coord: coordinatesList ){
68          mean[0] += coord.getX();
69          mean[1] += coord.getY();
70    }

71    mean[0] = mean[0] / coordinatesList.size();
```

```
72      mean[1] = mean[1] / coordinatesList.size();

73      return mean;
74      }
```

Los tres métodos anteriores constituyen la implementación del método más común del algoritmo de desviación estándar.