POSGRADO INTERINSTITUCIONAL EN CIENCIA Y TECNOLOGÍA

CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

Algoritmo para la detección de objetos en entornos abiertos basado en espacios binarios y conceptos asociativos

Tesis

QUE PARA OBTENER EL GRADO DE

DOCTORADO EN CIENCIA Y TECNOLOGÍA CON ESPECIALIDAD EN MECATRÓNICA

PRESENTA:

Alberto Vázquez Cervantes

ASESORES

Dr. Hugo Jiménez Hernández

Dr. Jorge Alberto Soto Cajiga

Dedicatoria

Le dedico esta tesis a mis padres y a mi novia Yuriko que han sido mi faro y mi camino hacia al éxito

Agradecimientos

Le agradezco a CIDESI, por su espacio y material prestado para el desarrollo de esta tesis.
A Conacyt, quien me ofreció su beca de doctorado para mi manutención a lo largo del proyecto de investigación.
A el Dr. Carlos Rubio quien cumplió mi sueño de pertenecer a la comunidad de CIDESI y en especial a el área de energías.
A mi asesor de tesis, Hugo Jiménez H. por su esfuerzo y dedicación, quien con sus conocimientos, experiencia, paciencia y motivación ha logrado en mí que pueda terminar mis estudios con éxito.
A mi codirector de tesis, Jorge Alberto Soto por su confianza y apoyo durante el desarrollo de mi formación.
A mi padre Floriberto, que me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.
A mi madre Carmen, por su confianza y el apoyo, que sin duda alguna en el trayecto de mi vida me ha demostrado su amor, corrigiendo mis faltas y celebrando mis triunfos.
A mi hermana Ana, que con sus consejos me ha ayudado a afrontar los retos que se me han presentado a lo largo de mi vida.
Agradezco a todas las personas que de una u otra forma estuvieron conmigo, porque cada una aportó con un granito de arena; y es por ello que a todos y cada uno de ustedes les dedico todo el esfuerzo, sacrificio y tiempo que entregué a esta tesis.

Índice general

Índice de figuras

Índice de tablas

Introducción.

Día a día, los sistemas de vigilancia se han hecho más común en nuestros días, en cuestiones de seguridad y monitoreo bajo ciertos escenarios estos son considerados una manera económica para tareas que benefician a la sociedad [1]. Esta situación hace confiable el uso de cámaras en dispositivos aéreos no tripulados como son los drones []. Estos son versátiles en el desarrollo de tareas automáticas debido al nivel de integración y la interfaz de comunicación inalámbrica. En los sistemas de monitoreo es necesario saber la ubicación de los objetos por tal motivo se requieren técnicas de localización. Este no es un problema trivial por las diversas situaciones no controladas tales como el movimiento de la cámara, saturación por exceso de luz, oclusiones o distorsiones de geometría debido a la proyección [2]. Con respecto a estas situaciones, varios enfoques en el estado del arte resultan no ser robustas ante todas estas circunstancias, existen algunos algoritmos de seguimiento y detección que asumen varias restricciones de escenarios, otros trabajos asumen cámara fija o conocimiento previo del escenario [3]. Por otro lado, existen algunas propuestas que utilizan algoritmos robustos que cubren estas dificultades; sin embargo, su complejidad computacional incrementa haciéndolos inviables para ser implementados en un sistema embebido [4]. En vehículos aéreos, ambas situaciones no son adecuadas, si el algoritmo es demasiado simple, no ubicará objetos bajo varias afectaciones; pero al contrario, si se consideran elementos para robustecer el algoritmo la complejidad computacional se eleva. Entonces, si el algoritmo es robusto el procesamiento en línea se hace más complicado, lo que afecta directamente a la autonomía aviones no tripulados [5]. Las situaciones comentadas anteriormente muestran la necesidad de desarrollar enfoques de baja complejidad, pero de alta confiabilidad.

Por estas razones esta tesis propone un algoritmo para detectar por ejemplar,

proporcionando un objetivo para ubicarlo, después de un itinerario de muestreo realizado por un vehículo no tripulado con una cámara frontal. Se adquiere evidencia visual, analizando secuencias de cuadros adquirida por la cámara. El algoritmo de detección se basa en un proceso paralelo dinámico, en el que la mayoría de las veces se comporta como un algoritmo de baja complejidad; es decir, el algoritmo realiza un proceso jerárquico de análisis. La propuesta está pensada de manera que los niveles superiores de la jerarquía tienen una complejidad baja, y en niveles inferiores la complejidad crece. Este esquema logra que en promedio se tenga una complejidad baja. El proceso superior usa un análisis general y selecciona áreas candidatas como posibles objetos de destino. El nivel jerárquico inferior usa más información para discriminar al candidato objetivo. El análisis jerárquico de profundidad representa con alta probabilidad los objetivos candidatos relacionados con el objetivo de búsqueda. Un proceso de arriba hacia abajo se realiza dinámicamente, lo que resulta en una complejidad variable, evitando el uso excesivo de recursos. Además que ofrece una búsqueda en el espacio de los datos de un orden $\frac{1}{c}\log_p(A)$, donde c es el número de procesos paralelos, p particiones en los datos para la búsqueda y A el total de los datos utilizados para medir la similitud.

El enfoque jerárquico usa un conjunto de características binarias extraídas del objetivo, esta característica aleatoria es el resultado de codificar como una cadena binaria un conjunto de los cambios intensidad del objetivo expresados en escala de grises. El proceso de codificación representa zonas distintivas como cadenas binarias, estas zonas son pequeñas ventanas de bajo una transformación F, de modo que es invariante para diversas afectaciones de imagen; para nuestros propósitos, se usa una matriz de tensor a partir de gradiente de imagen. La representación binaria tiene varias propiedades para desarrollar clasificadores de alta dimensionalidad; por ejemplo, conceptos ortogonales, mezcla de cadenas o superposición de cadenas, bajo la métrica L_1 [6] y operadores XOR [7], permiten desarrollar un anillo asociativo disperso.

El anillo asociativo, a partir de sus dos operadores, puede definir un proceso para hacer asociaciones entre las características de codificación con el objeto de destino en una estructura jerárquica. La estructura jerárquica se implementa como una relación de orden de similitud entre cadenas binarias relacionadas con las características del objetivo. El proceso de elección cuando la evidencia pertenece a un objetivo similar se aprecia como una ruta hacia el árbol, la posición del árbol después de la evidencia candidata de prueba define el nivel de similitud entre el objetivo y el candidato. Se utiliza un criterio de aceptación para definir como será una similitud adecuada entre los candidatos y el objetivo.

Se utiliza un criterio de aceptación para definir qué tan precisa será una similitud adecuada entre los candidatos y el objetivo. En términos de aplicación, la complejidad del algoritmo se expresa como una optimización de la arquitectura del hardware con el objetivo de una eficiencia energética. El anillo asociativo se implementa mediante la optimización del uso de operadores binarios en la arquitectura física.

Objetivo general

Desarrollar un algoritmo de búsqueda por ejemplar robusto para la localización y seguimiento de objetos para vehículos autónomos.

Objetivos específicos

- Diseñar una codificación invariante a cambios lumínicos con una complejidad baja.
- Implementar clasificador binario con conceptos asociativos y conceptos de anillo.
- Realizar un esquema para mejorar la codificación con el fin de lograr una invarianza a rotación y escala.
- Realizar un modelo experimental para evaluar el modelo de seguimiento.
- Realizar un modelo experimental para validar experimentalmente el funcionamiento.

Hipótesis

Si se tiene un modelo asociativo basado en instrucciones básicas del procesador contenidas el un espacio binario y una codificación robusta como entrada del sistema, entonces es posible generar una solución del problema de implementación de un seguidor de objetos en línea, considerando las dificultades que se presentan en los procesos internos relacionadas con la administración de tareas y la limitada energía del dispositivo. Entonces se logrará así una disminución en la complejidad computacional haciendo viable su seguimiento en vehículos autónomos dentro del ambiente de hardware embebido 1.

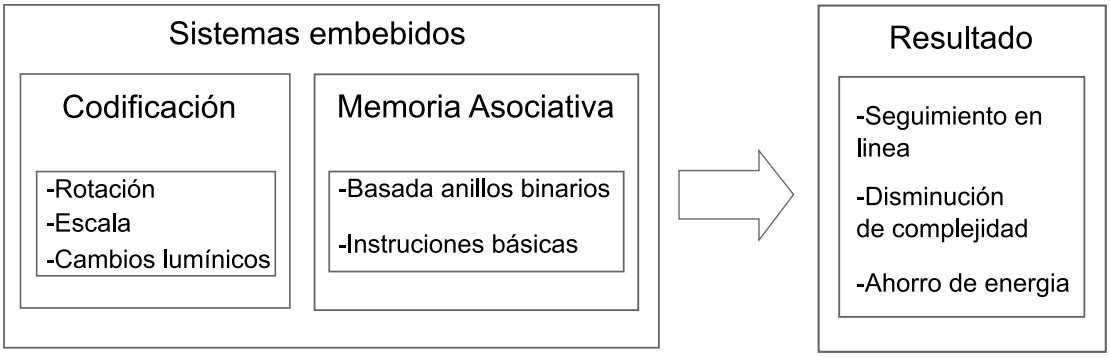

Figura 1: Esquema de la hipótesis

Justificación

Existen algunos problemas relacionados al análisis de localización y seguimiento los cuales están principalmente relacionados con el aumento en los últimos años de tecnologías que incrementan la resolución de las imágenes y la innovación en equipos de cómputo como se muestra en la Fig. 2. Estos permiten hacer mejor esta tarea, sin embargo para los sistemas embebidos este procesamiento rebasa las capacidades para alcanzar respuestas en tiempo real.

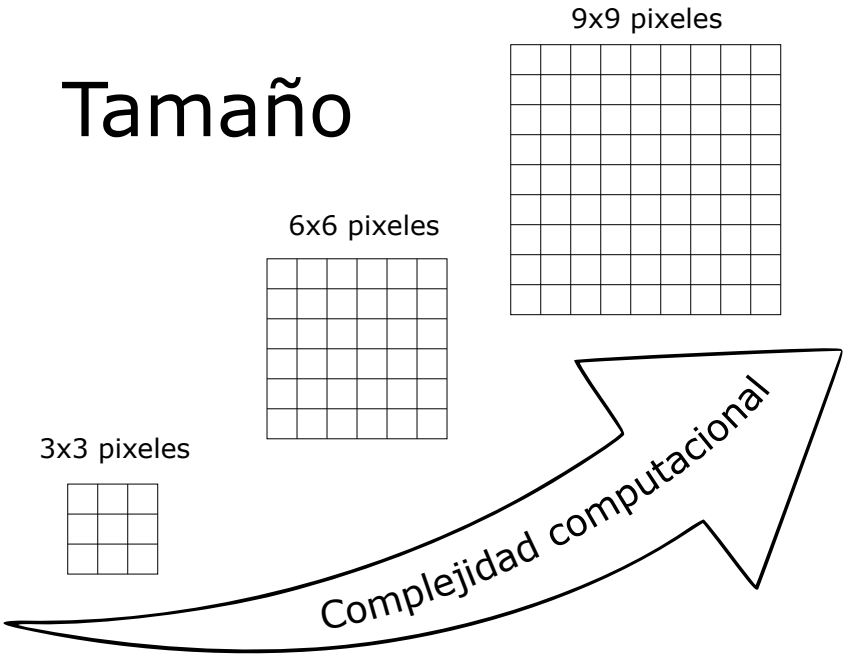

Figura 2: Crecimiento de tiempo de ejecución con respecto a tamaño de imagen

Planteamiento del problema

Actualmente los problemas de seguimiento se conjuntan en determinar la representación de los objetos ya que estos deben ser robustas ante rotación, escala, cambios lumínicos generales, así como una metodología que nos identifique el objeto de los demás, sin embargo utilizando las técnicas del estado del arte que tienen esta eficiencia suelen ejecutarse con una complejidad computacional elevada. Esto tiene como consecuencia sobrecalentamiento en el procesador y reducción en el tiempo de duración de la batería.

Organización

El documento se organiza de la siguiente manera: En §1 es un panorama general del uso de los dispositivos reconfigurables hasta el momento, además de las técnicas de clasificación y elementos que intervienen para realizar dicha tarea.
En §2 se describen los conceptos clave para comprender el objetivo del trabajo, y el análisis bibliográfico realizado para el trabajo, además de la estructuración lógica del material y el análisis crítico del mismo bajo el objetivo de esta tesis.
Después en §3 se introduce a los métodos que se seleccionaron para el desarrollo del proyecto, se presentan las fases que se llevan a cabo y el análisis de las técnicas usadas.
En §4 se evalúan los métodos descritos en el capitulo anterior mediante un proceso experimental, y se presentan los resultados obtenidos midiendo la complejidad computacional.
Finalmente se muestran las conclusiones de lo realizado, observaciones, trabajo a futuro y lineas de investigación que se desarrollan con la culminación de la tesis.

1 Fundamento teórico

En este capitulo se desarrollan los conceptos básicos necesarios para la comprensión de este proyecto de investigación. Partimos de la idea fundamental de lo que son los sistemas de vigilancia en la actualidad así como las limitantes que tienen en algunos escenarios.
Posteriormente se analiza los modelos de seguimiento que se encuentran actualmente en el estado del arte.
Después se muestran los dispositivos donde se ejecutan estos algoritmos, su nivel de robustez ante algunas situaciones y conceptos relacionados a su uso, desventajas y desventajas.
Finalmente se muestran algunos modelos de memorias y codificaciones que se han utilizado para llevar a cabo tareas de seguimiento para objetos que utilizan como base una búsqueda por ejemplar.

1.1. Visión artificial como apoyo a vigilancia

En la actualidad se buscan nuevos retos que aprovechen al máximo las capacidades de la visión artificial. Para probar la robustez de los métodos se utiliza principalmente en entornos desconocidos o que están cambiando dinámicamente, en esta área de investigación activa requiere una cantidad de recursos de computación de alto rendimiento debido al volumen de datos que se procesarán. El monitoreo tiene una amplia gama de aplicaciones tanto en público como en entornos privados, como seguridad, prevención del delito, control del tráfico, predicción y detección de accidentes, control de pacientes, vigilancia basada en vehículos aéreos autónomos, sistemas de vigilancia de aeropuertos, etc.) Existe un creciente interés en el monitoreo debido a la disponibilidad de sensores y procesadores con procesadores simples, así como la demanda de la necesidad de seguridad del público. Por lo tanto, en aplicaciones móviles y autónomas, la eficiencia energética y la precisión del sistema son requisitos cruciales.

Las operaciones se deben ser precisas y es indispensable que garanticen un funcionamiento en línea. Por lo general, los módulos de visión avanzada a menudo se basan en procesadores de vídeo integrados de alta eficiencia energética que respaldan la vigilancia y el seguimiento y otras necesidades de procesamiento para diversas aplicaciones.

1.2. Localización y seguimiento

La localización y seguimiento de objetos es un importante campo de trabajo para los sistemas de visión. La proliferación de las computadoras de alto desempeño, las cámaras de gran calidad y la creciente necesidad de automatizar el análisis de vídeo han generado un interés para la creación de algoritmos de seguimiento robustos, las tres fases principales de estos sistemas son: la detección de objetos de interés en movimiento, el seguimiento de estos objetos en cada cuadro y el análisis de la ruta que genero dicho objeto. Usualmente para realizar tareas tales como:

- Reconocimiento basado en movimiento, como identificación de personas basado en su forma de caminar, detección automática de objetos, etc;
- Sistemas de vigilancia, monitoreo de actividades anormales en escenarios conocidos o eventos inesperados;
- Indexamiento de video, anotaciones y recuperación automática en bases de datos multimedia;
- Interacciones humano-computadora, en el reconocimiento de gestos, seguimiento de ojos como sistema de entrada de computadoras;
- Monitoreo de tráfico, en estadísticas en tiempo real de congestionamiento de trafico en flujo vehicular;
- Navegación de vehículos, planificación de rutas basadas en vídeo y con la capacidad de evasión de obstáculos

En su forma más simple, el seguimiento se puede definir como el problema de estimar la trayectoria de un objeto en la imagen que se mueve alrededor de una escena. En otras palabras, el seguidor asigna etiquetas consistentes a los objetos seguidos en diferentes cuadros de un video. Adicionalmente, dependiendo del dominio de seguimiento, también puede proporcionar información, como la orientación, el área o la forma de un objeto. Las dificultades al realizar seguimiento de objetos pueden ser debido a:

- La pérdida de información causada por la proyección del mundo 3D en una imagen 2D,
- El ruido en imágenes,
- Movimiento de objeto complejo,
- Naturaleza no rígida o articulada de los objetos,
- Oclusiones parciales y completas de objetos,
- Las formas de objetos complejos
- Cambios en la iluminación
- Requisitos de procesamiento en tiempo real.

1.2.1. Detección de objetos

La detección de objetos es una tecnología informática que se ocupa de detectar instancias de objetos semánticos de cierta clase en imágenes y videos digitales. La detección de objetos se puede hacer utilizando algunos elementos básicos. algunas de las técnicas para representar a los objetos se muestran en la Figura. 1.1

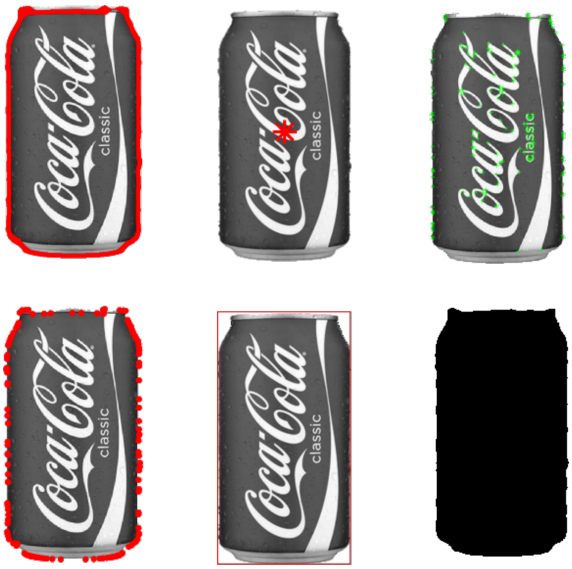

Figura 1.1: Distintas maneras de representar un objeto en procesamiento de imágenes

Sin embargo esta tarea se complica cuando el objeto se desplaza o se mueve y el sensor de que hace la captura de los datos no es perfecto como se muestra en la figura 1.2

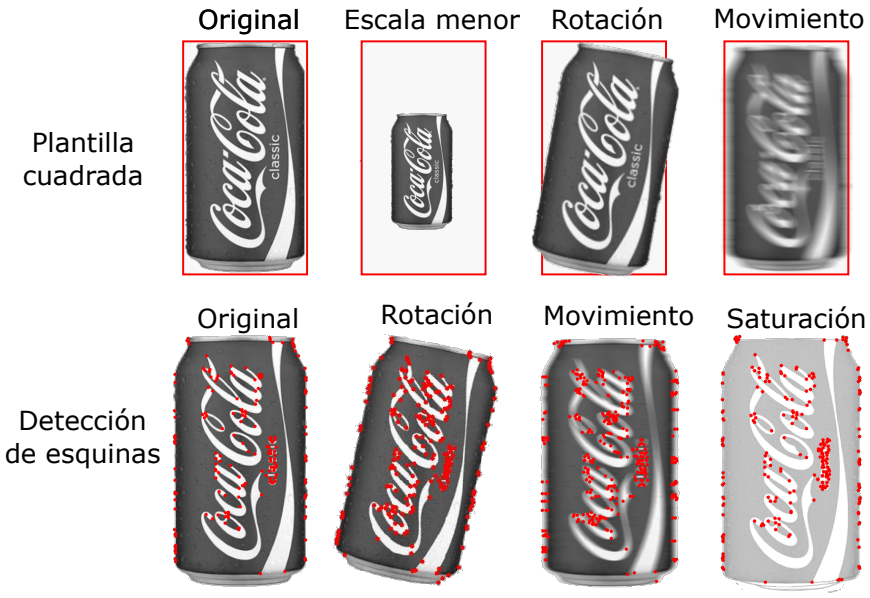

Figura 1.2: Distintas problemáticas en procesamiento de imágenes

1.3. Sistemas embebidos

Para procesar la información dentro de este ambiente de seguimiento y detección de objetos existen varios dispositivos que se analizarán desde los métodos clásicos hasta los mas avanzados y a su vez sus ventajas y desventajas.

Hace décadas, las unidades centrales de procesamiento (CPU) como se muestra en la Figura 1.3 se implementaron en transistores discretos y, más tarde, en dispositivos lógicos de circuito integrado. Todo eso cambió cuando el primer microprocesador, el 4004 de 4 bits de Intel, hizo su debut en 1971. Durante muchos años, los productos de Intel y sus competidores fueron la única opción para los ingenieros que querían una potencia de procesamiento programable en sus diseños.

1.3.1. CPU vs MPU

Ahora la CPU es un componente en un sistema más grande. Una unidad de microprocesador independiente (MPU) agrupa la CPU con interfaces periféricas como administración de memoria DDR3 y DDR4, PCIe, buses seriales como USB 2.0, USB 3.0, Ethernet y más, por lo que estos diseños son flexibles y versátiles y están diseñados para funcionar en múltiples sistemas operativos de alto nivel (OS) como Windows, iOS, Linux, etc. Para diseños más compactos, una unidad de microcontrolador (MCU) combina el núcleo de la CPU con la memoria interna, muchos periféricos en un solo circuito integrado o

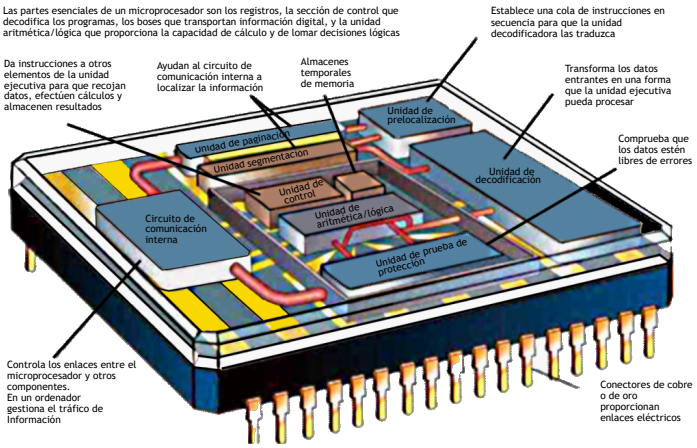

Figura 1.3: Partes del CPU

en un solo paquete, donde normalmente ejecutan un sistema operativo simplificado.
El microprocesador tiene influencia en la computadora portátil y de escritorio, y los microcontroladores están en todas partes en aplicaciones integradas, como paneles de instrumentos automotrices, controladores de motores o tarjetas inteligentes. La arquitectura de hardware de la CPU de estos dispositivos está diseñada para uso general, pero a menudo incluye bloques especializados, como unidades de punto flotante (FPU) para operaciones matemáticas. Las CPU de gama baja ejecutan operaciones de manera secuencial, pero los núcleos de procesamiento múltiples ahora son estándar en los microprocesadores y microcontroladores de gama alta, Xeon de Intel tiene hasta 22 núcleos.

1.3.2. FPGA y ASIC

Aunque estas CPU son adecuadas para la computación de uso general, en los últimos años ha surgido una gran cantidad de aplicaciones computacionalmente exigentes que requieren arquitecturas más especializadas. Los ejemplos incluyen búsqueda de alta velocidad; aprendizaje automático e inteligencia artificial (IA); informática de alto rendimiento (HPC) en centros de datos; procesamiento de gráficos en tiempo real, incluyendo realidad virtual y videojuegos; productos industriales como el control digital del motor y aplicaciones relacionadas con la automoción, como los sistemas avanzados de asistencia al conductor (ADAS) y, pronto, vehículos autónomos.
Los diseñadores en estos campos pueden recurrir a tres opciones de procesamiento adicionales: la unidad de procesamiento de gráficos (GPU), la matriz de compuerta programable en campo (FPGA) como se muestra en la Figura 1.4 y un circuito integrado de aplicación específica (ASIC).

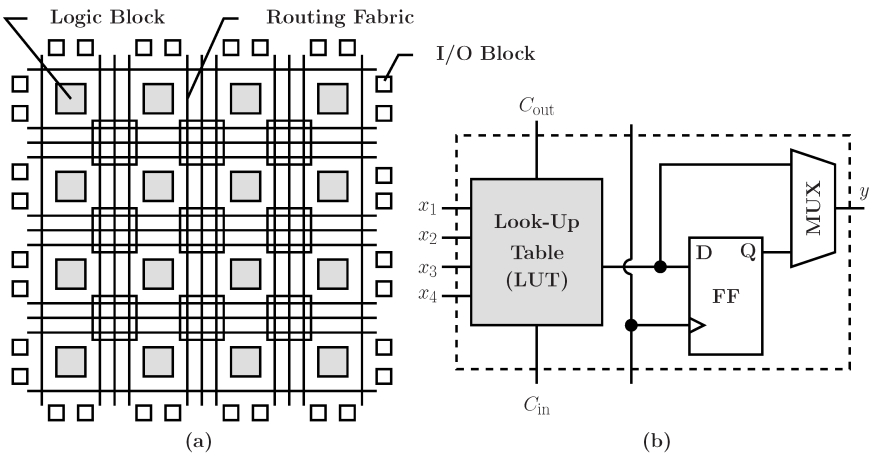

Figura 1.4: Matriz de compuertas de una FPGA

1.3.3. GPU vs FPGA

La GPU se introdujo por primera vez en la década de 1980 para descargar operaciones gráficas simples de la CPU. A medida que los gráficos se expandieron a 2D y, más tarde, a la representación 3D, las GPU se hicieron más potentes. La operación altamente paralela es muy ventajosa cuando se procesa una imagen compuesta de millones de píxeles, por lo que las GPU de la generación actual incluyen miles de núcleos diseñados para la ejecución eficiente de funciones matemáticas. El último dispositivo de Nvidia, el Tesla V100 Figura 1.5, contiene 5,120 núcleos CUDA para operaciones de acumulación múltiple de ciclo único y 640 núcleos tensoriales para multiplicación de matriz de ciclo único. Muchos algoritmos en otros campos se prestan a la ejecución paralela, por lo que las GPU se han extendido mucho más allá de su aplicación inicial. Muchas de las supercomputadoras más rápidas del mundo, por ejemplo, incluyen miles de GPU y CPU.

El FPGA consta de bloques de hardware internos con interconexiones programables por el usuario para personalizar las operaciones para una aplicación específica. A diferencia de los otros dispositivos mencionados, las conexiones entre bloques se pueden reprogramar fácilmente, cambiando la operación interna del hardware y permitiendo que un FPGA acomode los cambios a un diseño, o incluso que admita una nueva aplicación, durante la vida útil de la pieza, esta flexibilidad hace que el FPGA sea una excelente opción para aplicaciones en las que los estándares están evolucionando, como la televisión digital, la electrónica de consumo, los sistemas de ciberseguridad y las comunicaciones inalámbricas.

Figura 1.5: Tesla V100 de Nvidia

En el otro extremo del espectro, un ASIC está diseñado específicamente para su aplicación prevista, solo tiene los bloques necesarios para un funcionamiento óptimo, incluida una CPU, GPU, memoria, etc. Aunque los diseñadores pueden incorporar núcleos IP de terceros, como la CPU Arm Cortex, o bloques prediseñados para funciones estándar, como una capa física Ethernet, un ASIC es un diseño desde cero, es el más adecuado para aplicaciones de gran volumen. La Unidad de procesamiento de tensor (TPU), por ejemplo, es un acelerador desarrollado por Google específicamente para el aprendizaje automático de redes neuronales. Google también ofrece acceso TPU a empresas externas a través de su unidad de computación en la nube

1.3.4. Futuro de los sistemas embebidos

Cada vez más, vemos una convergencia de las tres categorías a medida que los proveedores buscan el conjunto de funciones óptimo para aplicaciones emergentes. Los FPGA SoC vienen con CPU, GPU y bloques DSP de IP de hardware o software. Las CPU incluyen aceleradores de hardware y ASIC para funciones criptográficas, y la GPU Tesla T4 de NVIDIA incluye elementos FPGA integrados para aplicaciones de inferencia AI.
Para ayudar a la comprensión de la importancia de los embebidos y realizar un seguimiento de todo, como se ve en la tabla 1.1 que resume las características principales de los cuatro dispositivos, junto con algunas de sus fortalezas y debilidades relativas.

Tabla 1.1: Tabla comparativa de sistemas embebidos

	CPU	FPGA	GPU	ASIC
Visión general	Procesador secuencial tradicional para aplicaciones de uso general.	Colección flexible de elementos lógicos y bloques IP que son configurables y modificables	Originalmente diseñado para gráficos; ahora se usa en una amplia gama de aplicaciones computacionalmente intensivas	Circuito integrado personalizado optimizado para la aplicación final
Procesamiento	MCU y MPU de uno o varios núcleos, más bloques especializados: FPU, etc.	Configurado para la aplicación; Los SoC incluyen hard o soft IP CORE (Ej., Arm)	Miles de núcleos de procesadores idénticos	Específico de la aplicación: puede incluir IP CORE de terceros
Programación	Sistemas operativos, API ejecutan gran variedad de lenguas de alto nivel	Tradicionalmente HDL (Verilog, VHDL); Los sistemas más nuevos incluyen C / C ++ a través de openCL & SDAccel	OpenCL y la API CUDA de Nvidia permiten la programación de propósito general (por ejemplo, C, C ++, Python, Java, Fortran)	Aplicación específica: TensorFlow opensource para TPU de Google; Fabricantes de CPU (AMD, Intel) incluye herramientas con nuevas versiones de ASIC
Periféricos	Amplia variedad de periféricos analógicos y digitales en MCU; Las MPU incluyen interfaces de bus digital	Los SoC incluyen muchos bloques transceptores, bancos de E / S configurables	Muy limitado; solo memoria caché	Adaptado a la aplicación: puede incluir funciones estándar de la industria (USB, Ethernet, etc.)
Ventajas	Versatilidad, multitarea, facilidad de programación.	Configurable para aplicación específica; la configuración se puede cambiar después de la instalación; alto rendimiento por vatio; acomoda la operación paralela masiva; amplia variedad de características: DSP, CPU	Poder de procesamiento masivo para aplicaciones: procesamiento de video, análisis de imágenes, procesamiento de señales	Diseñado a medida para la aplicación con una combinación óptima de rendimiento y consumo de energía.
Desventajas	La capacidad del sistema operativo agrega alta sobrecarga; optimizado para procesamiento secuencial con paralelismo limitado	Relativamente difícil de programar; segundo más largo en tiempo de desarrollo; bajo rendimiento para operaciones secuenciales; no es bueno para operaciones de punto flotante	Alto consumo de energía, no adecuado para algunos algoritmos; los problemas deben reformularse para aprovechar el paralelismo, pero los APIs proporcionan abstracción	Mayor tiempo de desarrollo; Alto costo; no se puede cambiar sin rediseñar el silicio

1.4. Machine learning

Los orígenes del aprendizaje automático se remontan a principios de la década de 1950, poco después del invención de la primera computadora electrónica de uso general, cuando Alan Turín creó el primer Turing prueba[8], para determinar si una computadora puede pensar, o para decirlo mejor si una máquina puede "aprender". El doctor Turing creía que las máquinas del tiempo dadas podrían "pensar" y "aprender" y finalmente pasar la prueba de Turing. Trabajando en esta suposición, muchos científicos comenzaron a trabajar en esta idea de "Inteligencia Artificial" y en 1952 Arthur Samuel escribió el primer programa de damas que aprendió jugando contra oponentes humanos y ajustó su estrategia y finalmente fue capaz de vencer a los jugadores humanos a principios de la década de 1970. Este término Inteligencia Artificial (IA) más tarde se convirtió en lo que ahora se conoce como Machine Learning. Una vista de diagrama de Venn o cómo todos estos términos están relacionados entre sí es mostrado en la Figura 1.6

Se ha realizado mucho trabajo en los últimos años sobre la idea inicial con avances con diferentes algoritmos descubiertos [9] con sus respectivas mejoras. El mayor avance llegó en redes neuronales cuando los investigadores redescubrieron el algoritmo de retropropagación y sus beneficios para el todo el campo de ML. Usando redes neuronales, que se modelan contra el cerebro humano, el término "Deep learning" que se acuñó en 2006. Las redes neuronales utilizan un enfoque más basado en datos en lugar de enfoques más tradicionales basados en el

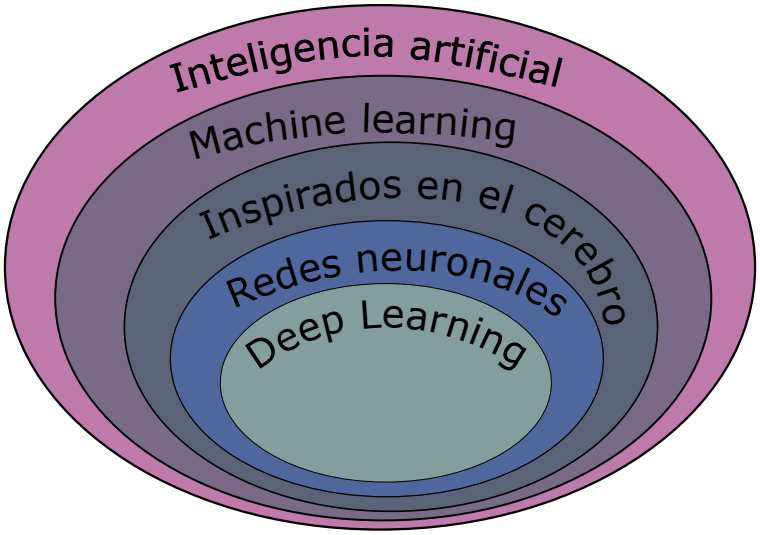

Figura 1.6: Esquema de inteligencia artificial

conocimiento. Los nuevos algoritmos de aprendizaje profundo permiten que las computadoras puedan analizar y distinguir objetos o textos en imágenes y videos.
El aprendizaje automático como se describe anteriormente se reduce a algoritmos que tienen la capacidad de aprender de experiencia mediante el mapeo a través de la hipótesis. En términos generales, estos algoritmos se clasifican en tres tipos:

- Aprendizaje supervisado: Algoritmos de aprendizaje en los que los datos de entrenamiento comprenden.
- Aprendizaje no supervisado: Algoritmos de aprendizaje en los que los datos de entrenamiento consisten en un conjunto de vectores de entrada sin ningún objetivo correspondiente.
- Aprendizaje por refuerzo: Algoritmos de aprendizaje en los que el algoritmo aprende por ejecución repetida de una tarea definida con una recompensa. realimentación. vectores

La clasificación de imágenes es un problema de aprendizaje supervisado en el que se etiquetan datos de entrenamiento proporcionado al algoritmo para aprender. Después de lo cual el algoritmo entrenado puede usar el conjunto de datos para reconocer la futura ocurrencia de la etiqueta en cuestión.

1.4.1. Memorias asociativas

Las actividades humanas han representado un tema de investigación en diversas áreas que tratan de entender los procesos internos involucrados. Sin embargo, la complejidad y la

diversidad de este tema han permitido proponer enfoques en diferentes áreas [10]. Estos enfoques pretenden imitar a través de simulación/emulación algunos comportamientos específicos. En particular, los modelos cognitivos incluyen modelos para explicar el pensamiento, el raciocinio, la inferencia y la abstracción, (por mencionar algunos), representando un desafío por todos los factores y la complejidad de los procesos involucrados. Los primeros intentos fueron inspirados en los enfoques reduccionistas [11], donde los trabajos más significativos[12] pretendían explicar los procesos cognitivos humanos como un conjunto de modelos monolíticos. Estos modelos son caracterizados debido a que se pueden reducir en una abstracción sintética definida, expresada por una aproximación matemática o lógica [13]. Sin embargo, estos enfoques sólo son útiles en escenarios específicos, en los que la información está libre de afectaciones como el ruido o datos incompletos. Después, los enfoques reduccionistas fueron sustituidos por enfoques conectivos. Los enfoques conectivos introducen un nuevo marco y una nueva metáfora para emular y simular procesos cognitivos [12][14][15]. Este enfoque modela diferentes procesos cognitivos como estados interconectados. Los estados representan modelos simples de unidades de proceso de la información y la estructura de las interconexiones define topologías de interacción. Aspectos como la aparición, el cambio de fase, y la auto-organización se explican comúnmente con este enfoque[16]. Además, los modelos computacionales más simples resultan fáciles de implementar y de probar. Por esta razón, la introducción de la redundancia de los datos y la redundancia en el número de unidades de cálculo son capaces de manejar tareas más complicadas, que de otra forma, no se realizarían por una sola unidad de cálculo. Algunos estudios recientes que se ocupan de este tema incluyen [15][17][18], los cuales abordan los problemas teóricos modelados como una red y una interacción de unidades de computación más simples. Uno de estos problemas teóricos se relaciona con el proceso cognitivo realizado por el cerebro, que incluye la tarea de asociar y relacionar conceptos. Este proceso representa una oportunidad de investigación porque hay muchos factores involucrados [19][20] tales como la gestión de la información que incluye la forma de la representación y la forma de manipularlas. Adicionalmente el último factor incluye, cómo se define el proceso de asociación usando las características codificadas. En este orden de ideas, algunos de los trabajos representativos incluyen [15][17][21], donde los autores proponen diferentes modelos basados en la linealidad de los datos expresada por matrices. En este sentido, el reto principal consiste en el establecimiento de un conjunto de características adecuadas para caracterizar un fenómeno a analizar, las cuales son invariantes a ciertas situaciones [15][18][22]. En áreas como el

análisis de imágenes, hay algunos enfoques como [23][24][25], donde los autores muestran algunos criterios, enfocados en la extracción y clasificación de características de la imagen y sus invarianzas para diferentes condiciones del escenario. Por otro lado, los clasificadores, que se utilizan para el agrupamiento de datos, utilizan características altamente confiables sobre su comportamiento, es decir, con características de alto grado de similitud, dentro del espacio de características, son dispersadas en cúmulos; los cuales son factibles para la agrupación a través de cualquier criterio de agrupamiento. En esta rama de la investigación, los enfoques representativos [26][27][28][29] muestran diferentes criterios de agrupación para descubrir patrones en los datos brutos. Los fundamentos y paradigmas utilizados en estos trabajos son diferentes, y por lo tanto el resultado es ligeramente distinto en escenarios similares. Un clasificador muy aceptado es la memoria asociativa [15][18] como un tipo particular de red neuronal artificial (ANN). El enfoque de memoria asociativa (AMA) es de tipo conectivo. Este enfoque utiliza la linealidad expresada en un conjunto de datos, así como una transformación lineal. Este enfoque por lo general es referido como un tipo de red neuronal, es decir, representa un enfoque conectivo donde se utiliza una matriz W es usada para asociar las entradas y salidas. En varios escenarios podría representar una aproximación sólida porque soporta la interferencia de ruido en los datos [21][26]. Los primeros enfoques asociativos fueron desarrollados por Steinbuch [30], quién propone un criterio para definir un clasificador lineal de patrones binarios. Después, una red neuronal particular tuvo influencia para desarrollar otro modelo conectivo: Hopfield Model[15]. Este modelo utiliza la idea de atractores lineales como una manera de codificar la información en clases. Luego, los modelos como [31] y [32] propusieron modelos asociativos basados en la separación de linealidad y la codificación de los datos. Esto es que representan los elementos de memoria como los patrones y establecen un criterio lineal para relacionar cada uno con una clase predefinida. Después, el modelo de ADAM [33] es presentado como un modelo de mejora del modelo de Hopfield [15], añadiendo una matriz extra para indexar los elementos dentro de la memoria.

1.5. Plataforma de implementación

En los últimos años, siglas como RPA, UAS o los archiconocidos drones han aparecido con mayor frecuencia en los medios de comunicación para identificar a las aeronaves no tripuladas. Aunque el desarrollo de este tipo de aparatos no es nuevo, en las últimas décadas su uso se ha ido generalizando en tareas de inteligencia, vigilancia y reconocimiento (ISR) y, con ellos,

su nomenclatura.

RPA, RPAS, UAV y UAS, pequeñas grandes diferencias Si bien es cierto que dron es el nombre más generalizado, en los círculos profesionales hay otros términos para hacer referencia a los distintos tipos de naves y sistemas existentes. Estas diferencias están relacionadas con lo que se conoce como grados de autonomía de un sistema no tripulado o ALFUS, por sus siglas en inglés (Autonomy Levels for Unmanned Systems), definidos sobre tres ejes principales: la independencia de los pilotos, la complejidad de la misión y la dificultad del escenario.

De estos niveles de autonomía se desprenden diferentes nomenclaturas –unas más usadas que otras–, sobre los distintos sistemas inteligentes no tripulados (UMS, Intelligent Unmanned Systems). En línea con la independencia de los pilotos, y según la OACI, usaremos RPA (Remotely Piloted Aircraft) para hacer referencia a las aeronaves tripuladas por control remoto y su variante, RPAS (Remotely Piloted Aircraft System), cuando queramos dirigirnos al sistema en conjunto –la aeronave, el enlace de comunicaciones y la estación de tierra– y no sólo al dispositivo aéreo, como en el caso anterior. En la red también es frecuente encontrar términos como RPAs, pero este haría referencia simplemente a los Remotely Piloted Aircrafts, es decir, el plural del anteriormente mencionado RPA.

Por otra parte tenemos la palabra UAV (Unmanned Aerial Vehicle), que quizás es el término que engloba más aparatos de este estilo. UAV, en español "vehículo aéreo no tripulado", incluye tanto a los dispositivos tripulados (RPAs) como a los que no lo son. La variante UAS (Unmanned Aerial System) hace referencia al sistema en conjunto. Finalmente, llamaremos simplemente aeronaves autónomas a aquellos aparatos capaces de desarrollar una función de forma completamente independiente, sin intervención humana de ningún tipo.

2 Propuesta.

2.1. Introducción

La ideología de utilizar esta fundamentación teórica surge de la necesidad de formar una memoria con la capacidad de funcionar en cualquier microprocesador, al estar conformada principalmente por operadores lógicos contenidos internamente en sus instrucciones. Esto reduce significativamente la complejidad al aprovechar cada ciclo de reloj. A continuación se muestran las reglas.

2.1.1. Espacios binarios con alta dimensionalidad

El espacio $\{0,1\}^k$ es un anillo sobre los operadores descritos anteriormente. Bajo este espacio la distancia d esta definida como una función que se origina de $\{0,1\}^k \times \{0,1\}^k \to \mathbb{N}$ como se muestra en 2.1

$$d(x,y) = \sum_{i=1}^{k} +1 \text{ Si y solo si } x_i \oplus y_i \text{ es verdadero; } +0 \text{ en otro caso.} \tag{2.1}$$

Para $x, y \in \{0,1\}^k$ y x_i representa la i^{esima} componente de x.
De la ultima definición, el operador norma debe ser definido como la distancia de cualquier cadena x y $\mathbf{O}$ donde $\mathbf{O} \in \{0,1\}^k$ denota el origen de referencia identificado por un cadena de longitud k con todos los elementos con el símbolo 0. Este es

$$|x| = d(x, \mathbf{O}); \text{ donde } o = \underbrace{0 \ldots 0}_{k} \tag{2.2}$$

Un espacio binario con alta dimensionalidad tiene varias propiedades[18] Estas hacen posible

el desarrollo de clasificadores. Una de las propiedades mas importantes esta relacionada con una función de densidad de probabilidad de la distancia (2.1). Esta distancia es equivalente a la métrica L_1 de la familia de Minkosky [23]. La distribución de la distancia tiene una distribución binomial

$$d(s^i, s^j) \sim \left(\begin{array}{c} n \\ d \end{array} \right) = \frac{n!}{d!(d-n)!} \tag{2.3}$$

para los elementos arbitrarios localizados en una distancia d. Esta distancia toma los valores cercanos al intervalo $d \in [1, k]$, donde k es la dimensionalidad del espacio y d la distancia entre dos cadenas. Cuando la dimensión k incrementa, la mayoría de las cadenas binarias tienen una distancia cercana a $\frac{1}{2}k$. Las cadenas binarias están localizadas en una distancia, entre ellos, cercana a $\frac{1}{2}n$, son consideradas como ortogonales. El grado de ortogonalidad de un par de cadenas binarias es medido como la diferencia de la distancia entre ellas y $\frac{1}{2}k$, que de denomina por

$$\varphi_k(s^i, s^j) = \left| \frac{1}{2}k - d(s^i, s^j) \right|. \tag{2.4}$$

Que es cero cuando s^i y s^j y mayor de cero se consideran con un menor grado de ortogonalidad. Como consecuencia, cuando n incrementa, la mayor parte de elementos de la cadena del espacio binario tienden a ser ortogonales; Por ejemplo, cualquier par de elementos en $\{0, 1\}^n$ aleatoriamente seleccionado tiene un alto grado de probabilidad de serlo.

2.2. Modelo de memoria asociativa

2.2.1. Conceptos asociativos

Un modelo asociativo consiste en $\mathcal{M}$ tal que para un conjuntos dados $\mathcal{A}$ y $\mathcal{B}$ crea una relación $\mathcal{R} : \mathcal{A} \times \mathcal{B}$. La relación $\mathcal{R}$ tiene propiedades dado un par de elementos $(a, b) \in \mathcal{A} \times \mathcal{B}$, y los elementos están dados por un criterio de distancia definido por d_a en $\mathcal{A}$ y d_b en $\mathcal{B}$ están relacionados respectivamente; por ejemplo, los elementos con pequeñas similitudes con el par $(a, b) \in \mathcal{A} \times \mathcal{B}$ son relacionados de la misma forma.

Nota: El criterio de similitud en d_a debe ser distinto a el criterio de d_b Típicamente las memorias son clasificadas con respecto a la naturaleza de su asociación.

Si: $\mathcal{A} = \mathcal{B}$ la memoria es llamada autoasociativa; Cuando $\mathcal{A} \neq \mathcal{B}$ es nombrada hetero asociativa; Cuando $|\mathcal{A}| > |\mathcal{B}|$ se considera como un clasificador y finalmente, cuando $|\mathcal{A}| \leq |\mathcal{B}|$ es denominada como un transductor [34] o codificador [35].

En general, no hay una expresión particular o metodología que relacione los elementos; es

decir, el proceso esta basado en fundamentaciones teóricas como la linealidad, dependencia o cualquier estructura matemática que ayude a definir un criterio para relacionar ambos conjuntos. $\mathcal{M}$ se construye atendiendo a los fundamentos teóricos principales que definen la clase de la memoria utilizada.
Una vez, tenemos $\mathcal{M}$, las operaciones comunes que debe realizar la memoria asociativa consisten en cualquier combinación de consulta que pueda expresarse en la relación $(\mathcal{A} \times \mathcal{B})$; es decir, por ejemplo, la existencia de un evento en el proceso de la memoria es reducida para verificar si para un evento dado a hay un elemento b que satisface $(a, b) \in \mathcal{R}$; o en el caso de que haya varios b 's tales que estén relacionados con una consulta por similitud en la memoria.

2.2.2. Definición de la memoria asociativa

Para nuestro caso particular, un modelo de memoria asociativa se define como un elemento aleatorio uniforme ortogonal $M \in \ 0, 1\ ^k$. Este elemento se denomina memoria vacía. Las propiedades de los operadores de anillo se utilizan para aprender y almacenar patrones en la memoria vacía. Tres propiedades importantes del operador $\oplus$ son las siguientes:

1. $x \oplus x \leftrightarrow \mathbf{O}$.
2. $x \oplus \mathbf{O} \leftrightarrow x$.
3. $x \oplus y \leftrightarrow y \oplus x$.

La primera y segunda propiedad se utilizan para hacer un criterio de pertenencia para la memoria asociativa. La tercera propiedad se usa para almacenar patrones en la memoria asociativa.
El proceso de aprendizaje asociativo consiste en superponer patrones en una memoria asociativa vacía bajo el operador $\oplus$. Esto es, dado un conjunto de patrones $x_1, x_2, \ldots, x_l$ y memoria asociativa M, un proceso de aprendizaje se define como:

$$M \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_l \tag{2.5}$$

Tenga en cuenta que la memoria asociativa M puede estar vacía o no. La superposición de cada evento bajo $\oplus$ garantiza que cualquier combinación será otro elemento válido en $\{0, 1\}^k$ espacio. Existen varias restricciones que deben seguirse para garantizar una funcionalidad correcta. La memoria asociativa no debe ser $\mathbf{O}$ y $\neg\mathbf{O}$, razón por la cual M

debe ser ortogonal. El proceso de codificación de eventos idealmente también debe mapearse en elementos ortogonales.

Expresamos la adición de un evento particular x_i a una memoria asociativa aprendida dada (2.5), de la siguiente manera:

Tenga en cuenta que hay dos posibilidades: (a) hay un evento aprendido anterior x_i en la memoria; (b) no hay un evento aprendido previo.

En el caso (a), la expresión podría reescribirse como $M \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_l \oplus x_i \oplus x_i$ y, en consecuencia, $M \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_l \oplus \mathbf{O}$. Esto es, el evento x_i se borra de la memoria. (b) para el segundo caso, el nuevo evento se superpone en la memoria como un nuevo evento. Tenga en cuenta que para la segunda restricción cada vez que x_i el evento será un evento afectado por el ruido del evento previamente almacenado $\tilde{x}_i$, la norma de $\tilde{x}_i \oplus x_i$ se acerca a cero.

La consideración anterior se extiende para desarrollar un criterio de pertenencia formal de la siguiente manera.

Para una memoria asociativa dada (2.5) y x_i evento, el criterio de pertenencia se define de la siguiente manera:

$$B(x_i, M) = d(M \oplus (M \oplus x_i), x_i) \tag{2.6}$$

donde $(M \oplus x_i)$ denota la memoria asociativa sin evento x_i, y $M \oplus (M \oplus x_i)$ aísla este elemento previamente almacenado. Tenga en cuenta que siempre que x_i represente un evento de ruido alterado del elemento previamente almacenado $\tilde{x}_i$, operaciones como $M \oplus (M \oplus x_i)$, que desaparecen parcialmente la información del evento de la memoria. En consecuencia, las operaciones de recuperación $M \oplus (M \oplus x_i)$ reconstruyen la mayoría de los eventos similares almacenados. El radio de similitud se utiliza como criterio para establecer un vecindario de similitud. De acuerdo con las propiedades del espacio binario, la vecindad debe seguir el criterio de ortogonalidad; es decir, el radio debe ser inferior a $\frac{1}{4}k$ para considerarse similar.

3

Modelo Experimental y resultados

3.1. Planteamiento de la metodología

El presente proyecto consta de tres etapas fundamentales de desarrollo con la finalidad de ganar tolerancia al error y conseguir la robustez que se necesita en sistemas de vigilancia implementadas en un embebido como se muestra en el siguiente diagrama Figura 3.1.

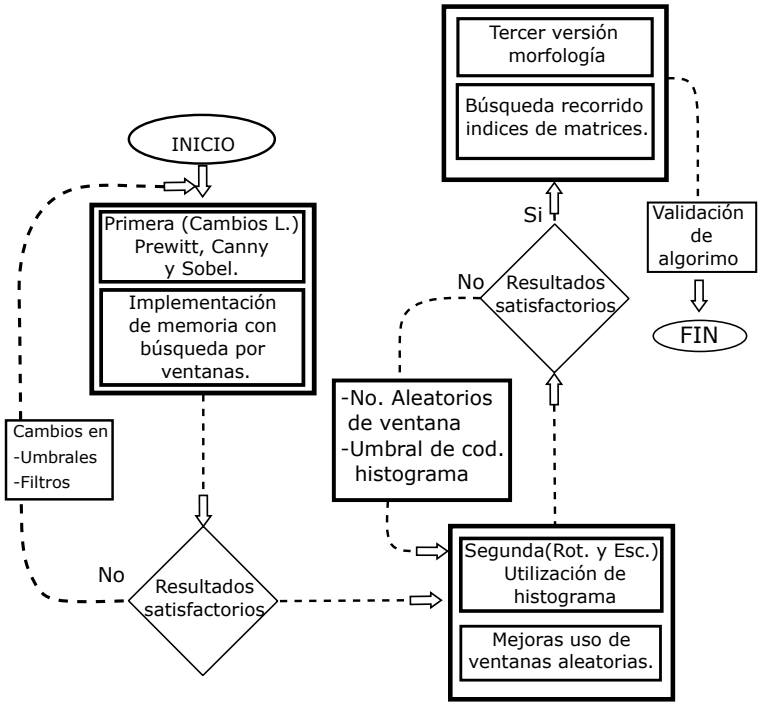

Figura 3.1: Esquema de experimentación

3.2. Etapa de selección de la información

En la primera etapa se propone una la unión de dos técnicas. La primera técnica utilizada es el filtro de *Sobel* que bajo una cierta mascara como se muestra en la Ecuación 3.1 nos da como resultado el gradiente de una imagen.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I} \quad \text{y} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{I} \tag{3.1}$$

Donde I representa la imagen y G el filtro de gradiente.

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2} \tag{3.2}$$

Al utilizar la norma de los elementos de G_x y G_y contrarrestan el problema de cambios lumínicos generales como se muestra en la Figura 3.2

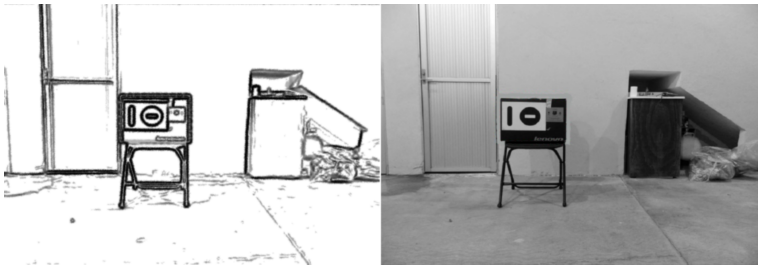

Figura 3.2: Gradiente contra imagen original

En esa etapa se prueba con un recorrido tradicional en la imagen en el cual se desplaza la ventana a lo largo y ancho de cada uno de los lugares de la imagen como se muestra en la Figura 3.3.

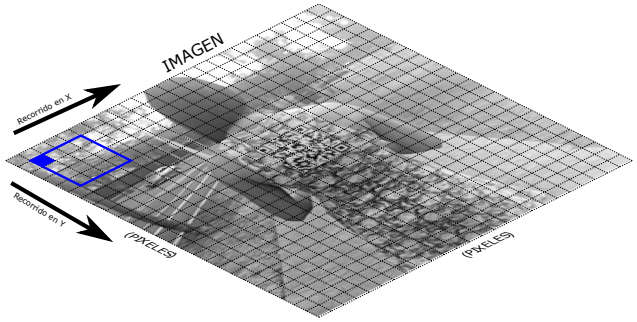

Figura 3.3: Recorrido de la imagen

Aplicando el criterio de la memoria asociativa que se propone la section 2.1 se constituye con el algoritmo 2 que se inicia con la selección del objeto a encontrar y seguir por medio de la creación de una ventana de interés como se muestra en la Figura 3.4.

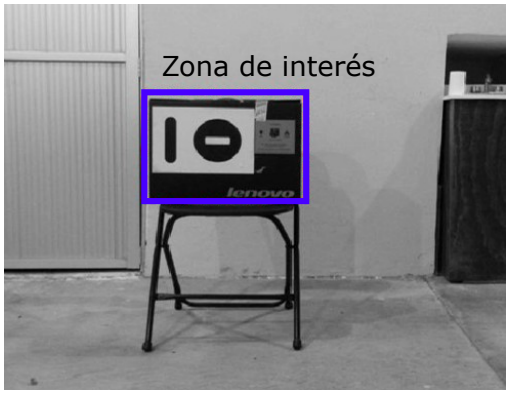

Figura 3.4: Selección de la región de interés

Al final del algoritmo obtenemos una matriz de distancias. En esta podríamos aplicar un umbral para determinar la posición del objeto (usualmente encontramos el centro del objeto aplicando el criterio del mínimo).

Algoritmo 1: Primera versión de la búsqueda

Result: Posición del objeto objetivo

Seleccionar área de interés;

Codifica la ventana de objeto en cadena;

Obtiene gradiente;

Inicia la memoria con una distribución uniforme;

Aprende cadena codificada con memoria;

while *Todas las ventanas de la imagen* **do**

    criterio de pertenencia de la memoria;

    if *n>1/4* **then**

        pertenece a la memoria;

        almacena en matriz de distancias;

    end

end

Verifica mediante umbral;

Gráfica posición del objeto;

Para fines prácticos el algoritmo de búsqueda resulta inviable por el tiempo de computo que se requiere ya que para una imagen de aproximadamente dos megapíxeles se requiere el un

tiempo de procesamiento de aproximadamente un minuto. Un ejemplo del caso exitoso de detección se muestra en la Figura 3.5

Figura 3.5: Resultado de la primera versión de la memoria

3.3. Etapa de Optimización de la población

La segunda técnica que se utiliza en la etapa 2 es el uso de la independencia estadística, haciendo el uso del histograma de una población, en este caso la población seria constituida por el resultado del gradiente de la imagen. Ambas hacen posible el análisis de la estructura de los datos como se muestra en la Figura 3.6.

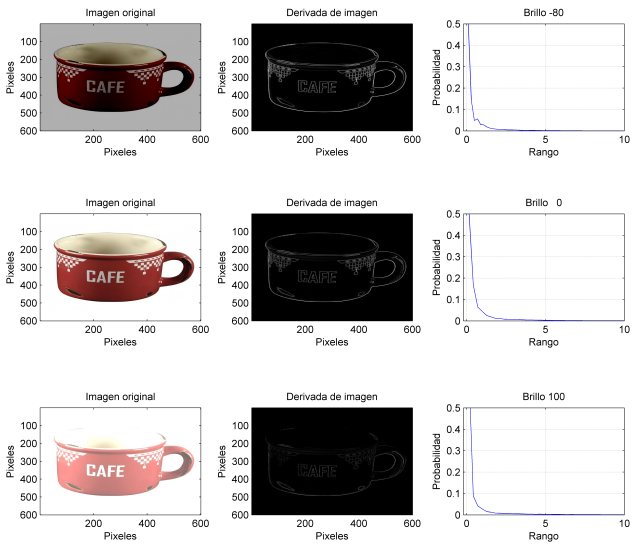

Figura 3.6: En la primera columna se encuentra la imagen original y sus variaciones en brillo, en la segunda el criterio de la deriva utilizado y finalmente se comprueba la similitud en la distribución de los píxeles en la imagen.

Para solventar el problema del tiempo de procesamiento lo que se propone la variante de usar ventanas aleatorias de manera jerárquica. De esta manera se aprovecha el tiempo de computo al no utilizar la totalidad de imagen y su vez utilizar. Se describe un esquema de dispersión de puntos en la Figura 3.7. En el siguiente pseudocódigo se explica las modificaciones que mejoran la primera versión de la búsqueda se agrega la funcionalidad de jerarquía y la tolerancia a rotación y escala.

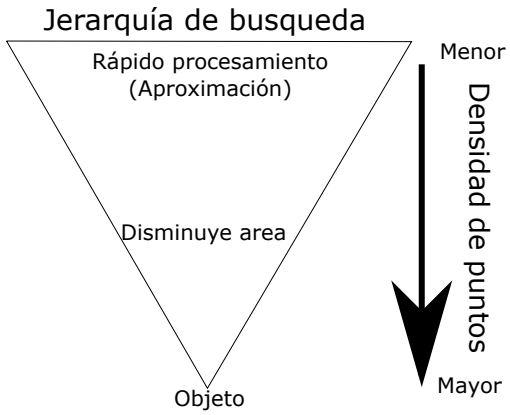

Figura 3.7: Esquema de jerarquía

Algoritmo 2: Segunda versión de la búsqueda

Result: Posición del objeto objetivo

Seleccionar área de interés;

Codifica la ventana de objeto en cadena;

Obtiene gradiente;

Calculo del histograma del gradiente;

Inicia la memoria con una distribución uniforme;

Aprende cadena codificada con memoria;

No. de ventana depende de la jerarquía del esquema;

for *ventana* $\leftarrow 1$ **to** *No. ventana* **do**

    Criterio de pertenencia de la memoria;

    if *n<1/4* **then**

        Pertenece a la memoria;

        Almacena en matriz de distancias;

    end

end

Verifica mediante umbral;

Gráfica posición del objeto;

Se demuestra el resultado de la segunda etapa en la cual se hacen rotaciones y escalado. En la Figura 3.8 que se indica el valor promedio de la distancia al centro del objeto el cual corresponde la el criterio de $\frac{n}{4}$ recordando que n es la cardinalidad de la imagen de *8 bits*; es decir 256.

Figura 3.8: Escenario donde se prueba la rotación y escala.

3.4. Etapa de detección y medición de la similitud

Para la culminación de la tolerancia que se quiere llegar para este proyecto se realizan algunos criterios para garantizar el tiempo de respuesta del algoritmo ante situaciones de ruido del escenario, ya que se requiere que sea en entornos abiertos en donde no siempre se pueden controlar las condiciones físicas como cambios lumínicos, desenfoque de la cámara, que la cámara no logre estar a la misma distancia siempre o inclusive que gire un poco a medida que se mueve. Esto hace que altere la detección de los objetos y sea inviable el seguimiento cuando se esta en el exterior.

Como primer paso definimos para una cadena definida como anteriormente por el histograma del gradiente de la imagen, sin embargo, utilizaremos una codificación basada en el máximo del histograma dividida iterativamente hasta encontrar un balance de elementos binarios como se muestra en la Figura 3.9.

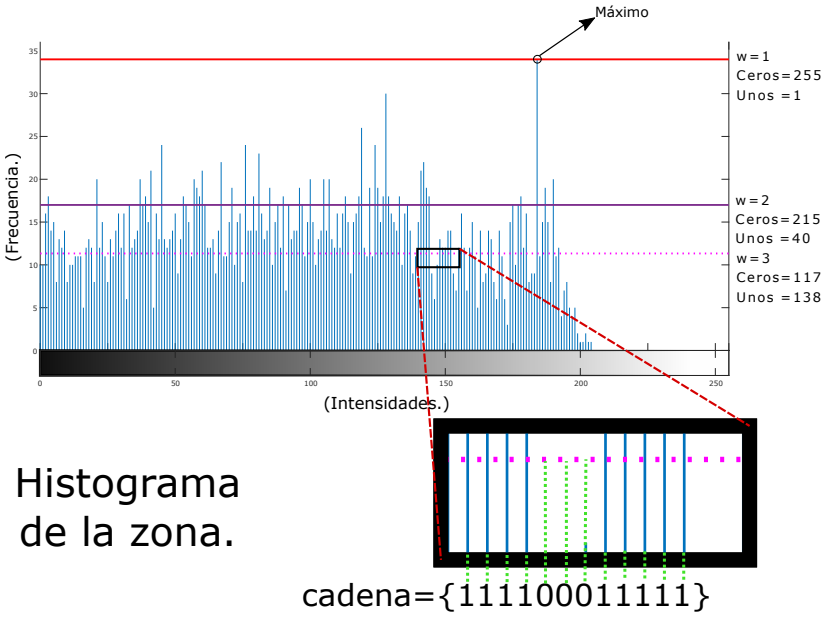

Figura 3.9: Pasos de codificación de la cadena.

Se crea automáticamente un umbral en el cual si se dispone el valor de la columna del histograma por debajo de este se asignará un 0 de lo contrario un 1 y así sucesivamente hasta terminar los 256 valores del histograma.

Una vez creada esta cadena, corresponderá a la representación del objeto y se adaptará a las condiciones de la memoria asociativa. Esta deberá ser usada para el aprendizaje de la memoria asociativa *XOR*. Con la memoria aprendida con el objeto a detectar se procede a una función de dispersión de puntos de una manera uniforme a lo largo de la imagen siguiente a detectar. Las posiciones de cada punto darán la pauta para iniciar el proceso de detección como se muestra en la Figura 3.10

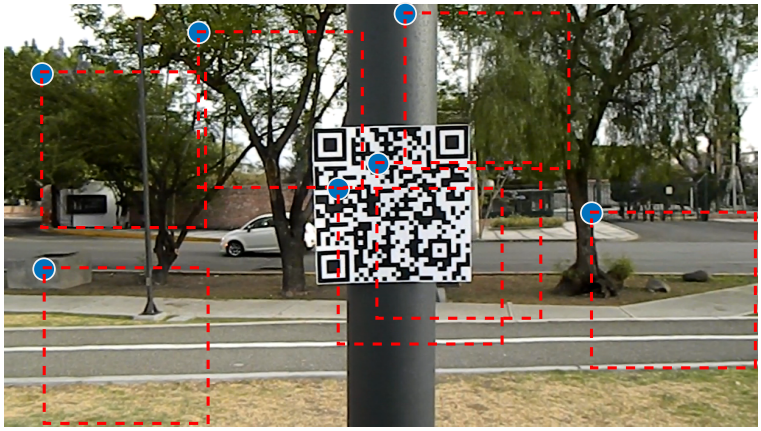

Figura 3.10: Ejemplo de asignación de ventanas aleatorias

Se genera el mapa de distancias en base al criterio de pertenencia de la memoria reiterando la teoría de la capitulo 2 como se muestra en la Figura 3.11

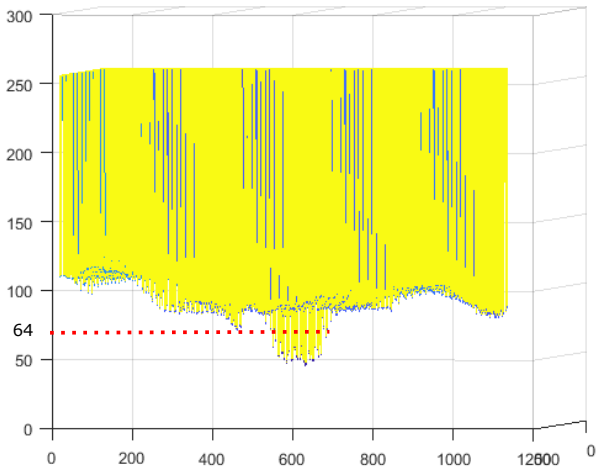

Figura 3.11: Mapa de distancias marcado con el umbral

Sin todos los puntos analizados deja algunos huecos que hacen difícil la ubicación precisa del objeto a detectar por lo que se opta por utilizar una dilatación morfológica definida por la ecuación 3.3 y dada la naturaleza de los puntos se utiliza un elemento estructurante en forma de disco para garantizar la posición del centro del objeto como se muestra en la Figura 3.12

$$A \oplus B = \bigcup_{b \in B} A_b. \tag{3.3}$$

Figura 3.12: Dilatación para unir los puntos del mapa de distancias

Al final se guarda la posición del objeto del cuadro anterior y se utiliza el criterio de jerarquía que se ha propuesto. En dado caso que el objeto no detecte el objeto se reinicia el criterio jerárquico y en el siguiente cuadro se reanuda la búsqueda del punto anterior.
Como parte de la validación se hacen pruebas con variaciones de escalado en escenarios

controlados como se muestra en la Figura 3.13. Se hacen ajustes en el tamaño del elemento estructurante ya que cada caso suele tener un tamaño distinto.

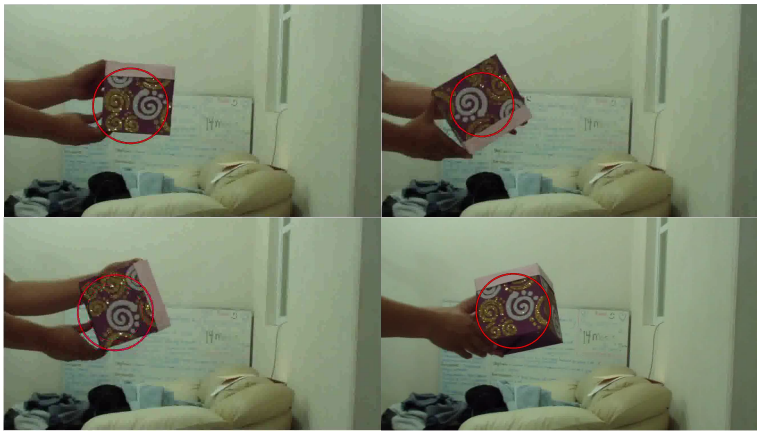

Figura 3.13: Prueba rotación en escenario controlado

De igual manera se busca probar le robustez del método en escenarios controlados pero ahora haciendo la variación de escalado, con esto nos referimos a cambios con la cercanía de la cámara sin presentar problemas de perspectiva como se muestra en la Figura 3.14.

Figura 3.14: Prueba escala en escenario controlado

3.5. Validación con escenarios reales

Como una etapa final se utiliza una base de datos de vídeos grabados por drones en escenarios abiertos para probar la máxima robustez del algoritmo. Aquí se analizarían los principales paradigmas en los sistemas de detección y seguimiento en los vehículos autónomos al encontrarnos con cámara en movimiento y objeto en movimiento.

En el primer vídeo de prueba se observa un automóvil en movimiento captado con un dron y se intenta hacer el seguimiento colocando el área de interés en una parte del carro como se muestra en la Figura 3.15

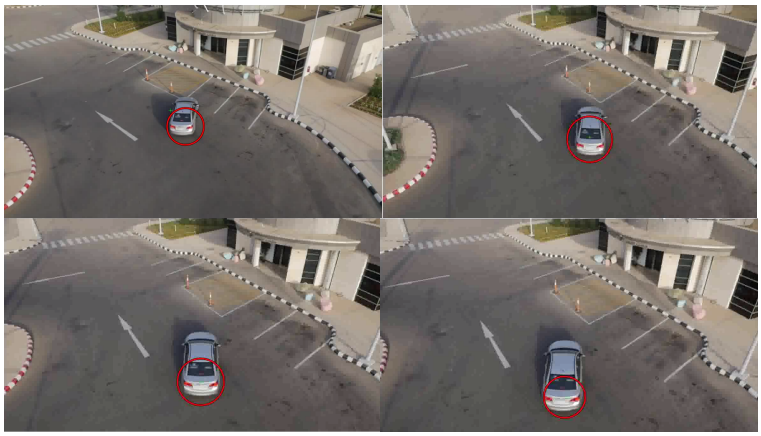

Figura 3.15: Video de base de datos de vehículo

En el segundo video de la base de datos se observa a una persona caminando, la dificultad de este escenario es que la variación de pasto es muy poca como se observa en la Figura 3.16 hace que la memoria llegue a confundirse, este seguidor no necesariamente tiene que seguir todos los cuadros del video. Si llegará a perder el objeto en un cuadro buscara el objeto desde la ultima ubicación detectada.

Figura 3.16: Seguimiento a persona desde dron utilizando la memoria asociativa

3.6. Diseño de arquitectura de vehículo autónomo

Concluidas estas pruebas se hace notar que también se desarrollo una arquitectura para la implementación de vehículo autónomo. Esta esta dividida en tres principales aspectos como se muestra en la Figura 3.17

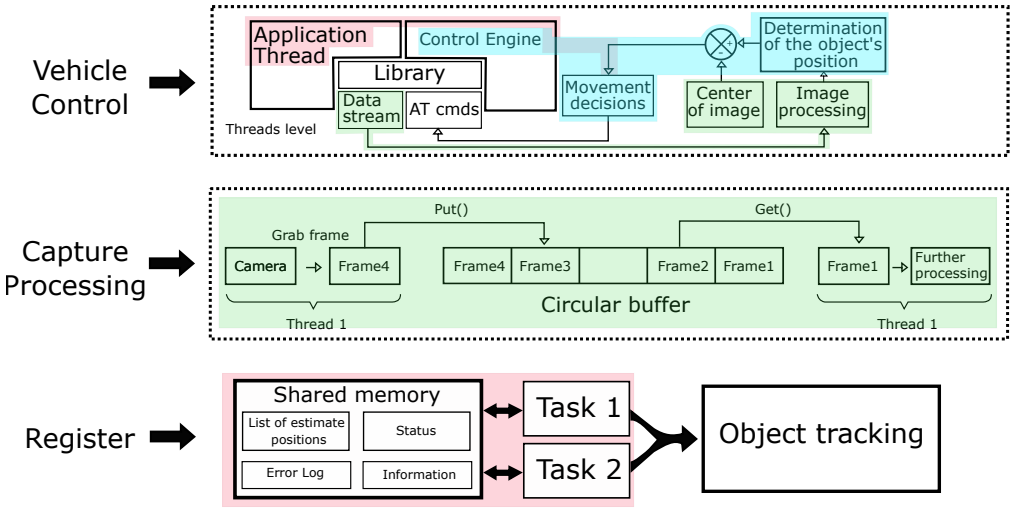

Figura 3.17: Arquitectura del sistema embebido de seguimiento de objetos

También se crean diseños conceptuales de posibles vehículos

Figura 3.18: Vehículo aéreo

4 Conclusiones.

Para las conclusiones finales se pueden decir diversos puntos que se fundamentan en los resultados y en la comprobación de una teoría que hace posible la implementación de una memoria asociativa.

A partir de los resultados obtenidos que se comprueba la posibilidad de detectar un objeto y seguirlo en los cuadros de un secuencia de video empleando una representación dado por cadenas largas binarias (cadenas simbólicas). Utilizando el modelo de muestreo aleatorio jerárquico para hacer mas eficiente el procesamiento de las imágenes. Disminución de uso de batería al tener algoritmos de menor complejidad computacional lo que hace posible el funcionamiento en linea. Se reduce la dimensionalidad de los datos convirtiendo al modelo en una versión computacionalmente más económica en cuestión de utilización de recursos. Se observó que la longitud (dimensionalidad del espacio) de las cadenas tiene relación con la cardinalidad de la calidad del sensor lo cual afecta las capacidades de reconocimiento, es decir, si los sensores tienen más resolución es probable que se pueda hacer mas clara la diferencia en la textura de la imagen, sin embargo, el tiempo de procesamiento crece.

El tiempo de cómputo se reduce significativamente cuando se utiliza los índices de las matrices pero también aumenta el uso de memoria RAM. Demostrando que se pueden realizar en varias operaciones disminuyendo los ciclos de reloj, por consecuencia el tiempo de ejecución.

Se concluye que es viable el uso de procesadores *ARM* con un núcleo Linux ya que cuentan con el set de instrucciones que usa la memoria asociativa XOR. Esto se debe a las pequeñas dimensiones que pudiera tener así como su flexibilidad en los periféricos que se pueden interconectar

4.1. Líneas de investigación

Como trabajo a futuro del trabajo de investigación que se realizó se proponen los siguientes temas:

- Implementación en tiempo real de seguidor de objetos en UAV.
- Reconstrucción 3D de objetos de grandes dimensiones con el uso de drones.
- Optimización de una memoria asociativa para la clasificación de patrones.
- Algoritmos evolutivos optimizados con memorias asociativas.

Bibliografía

[1] R. L. Finn and D. Wright, "Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications," *Computer Law & Security Review*, vol. 28, no. 2, pp. 184–194, 2012.

[2] S. Zhang, "Object tracking in unmanned aerial vehicle (uav) videos using a combined approach," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 2, pp. ii–681, IEEE, 2005.

[3] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 918–923, July 2003.

[4] C. H. Papadimitriou, *Computational complexity*. John Wiley and Sons Ltd., 2003.

[5] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro uav swarms," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 339–353, 2011.

[6] T. Bozkaya and M. Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces," in *ACM SIGMOD Record*, vol. 26, pp. 357–368, ACM, 1997.

[7] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognition*, vol. 44, no. 7, pp. 1357–1371, 2011.

[8] G. Oppy and D. Dowe, "The turing test," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, spring 2019 ed., 2019.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[10] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 74. Prentice hall Englewood Cliffs, 1995.

[11] J. P. Sterba and C. Quinn, "Davis baird on nano tech," *Social Theory and Practice*, vol. 29, no. 2, 2003.

[12] M. Minsky, *Society of mind*. SimonandSchuster. com, 1988.

[13] T. S. Kuhn, *The structure of scientific revolutions* . University of Chicago press, 1996.

[14] B. J. Copeland, *The Essential Turing*. Oxford University Press, 2004.

[15] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[16] Y. Bar-Yam, "Dynamics of complex systems (studies in nonlinearity)," 2003.

[17] B. J. Copeland and R. Sylvan, "Beyond the universal turing machine," *Australasian Journal of Philosophy*, vol. 77, no. 1, pp. 46–66, 1999.

[18] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.

[19] P. Dayan, L. F. Abbott, and L. Abbott, "Theoretical neuroscience: Computational and mathematical modeling of neural systems," 2005.

[20] F. Rieke, D. Warland, R. De Ruyter van Steveninck, and W. Bialek, "Exploring the neural code," 1997.

[21] M. Minsky and S. Papert, "Perceptrons: an introduction to computational geometry (expanded edition)," 1988.

[22] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[23] S. Santini and R. Jain, "Similarity measures," *Pattern analysis and machine intelligence, IEEE transactions on*, vol. 21, no. 9, pp. 871–883, 1999.

[24] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *Acm Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[25] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," tech. rep., Tech. rep., Microsoft Research, 2010.

[26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[27] C. BenAbdelkader, R. Cutler, H. Nanda, and L. Davis, "Eigengait: Motion-based recognition of people using image self-similarity," in *Audio-and Video-Based Biometric Person Authentication*, pp. 284–294, Springer, 2001.

[28] J. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[29] R. T. Collins, R. Gross, and J. Shi, "Silhouette-based human identification from body shape and gait," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pp. 366–371, IEEE, 2002.

[30] K. Steinbuch, "Die lernmatrix," *Biological Cybernetics*, vol. 1, no. 1, pp. 36–45, 1961.

[31] T. Kohonen, "Self-organization and associative memory," *Self-Organization and Associative Memory, 100 figs. XV, 312 pages.. Springer-Verlag Berlin Heidelberg New York. Also Springer Series in Information Sciences, volume 8*, vol. 1, 1988.

[32] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

[33] J. Austin, "Adam: A distributed associative memory for scene analysis," in *Proceedings of First International Conference on Neural Networks*, vol. 4, p. 285, 1987.

[34] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[35] C. E. Shannon and W. Weaver, *The Mathematical Theory of Information*. Urbana, Illinois: University of Illinois Press, 1949.