INSTITUTO TECNOLOGICO SUPERIOR DE MISANTLA

MAESTRÍA EN SISTEMAS COMPUTACIONALES

ALGORITMO BIO-INSPIRADO PARA SEGUIR UNA MANO CON LA CÁMARA DE SILICON RETINA

TESIS QUE PARA OBTENER EL TÍTULO DE

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA

I.S.C LESLIE LORELEI MEZA GIRÓN

ASESOR:

DR. SIMÓN PEDRO ARGUIJO HERNÁNDEZ

DR. LUIS ALBERTO MORALES ROSALES

MISANTLA, VERACRUZ MAYO, 2017

Contenido

Lista de figuras

Lista de Tablas

Lista de abreviaciones

AE	Address Event
AER	Address Event Representation
ANN	Artificial Neural Network
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPG	Central Pattern Generator
DVS	Dynamic Vision Sensor
EP	Programación Evolutiva
FNN	Feedforward neural Network
GA	Algoritmos Genéticos
ISODATA	Iterative Self-Organizing Data Analysis Technique Algorithm
IT	Infero-Temporal
K-NN	k-nearest neighbors
LIF	Leaky Integrate and Fire

MLP	Multilayer Perceptron
MOPSO	Multi-Objective Particle Swarm Optimization
MSO	Multi Swarm Optimization
MST	Área temporal superior media
MT	Área temporal media
NGL	Núcleos Geniculados Laterales
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
ReSuMe	Remote Supervision Method
RNN	Recurrent Neural Network
SNN	Spiking Neural Network
STDP	Spyke-Timing-Dependent Plasticity
SVM	Support Vector Machine
VLSI	Very Large Scale Integration

1 Generalidades

1.1 Introducción

- La percepción provee de información acerca del mundo en que habitamos [8], y la percepción visual es fundamentalmente importante para la mayor parte de la vida inteligente [1].
- La percepción es iniciada por sensores [8] y la visión es por mucho el sensor más útil para lidiar con el mundo físico [8].

Partiendo de estos dos hechos, dentro de la Inteligencia Artificial, el área de Visión Artificial es un campo con un constante avance en cuanto a conocimiento y evolución en búsqueda de mejorar los sistemas artificiales de visión.

Entre los usos posibles de la visión artificial se encuentran los siguientes:

- Manipulación: agarre, inserción, etc., necesita información de forma local y retroalimentación para el control del motor [8].
- Navegación: encontrar caminos libres/claros, evitar obstáculos y el cálculo de la propia velocidad actual y orientación [8].
- Reconocimiento de objetos: habilidad útil para distinguir entre un objeto y otro [8].

Con el incremento de la capacidad, la velocidad y las ventajas económicas de los dispositivos de procesamiento de señales se ha visto un creciente esfuerzo por desarrollar sistemas sofisticados de tiempo real que emulen las habilidades de los seres vivos (especialmente las habilidades humanas), entre ellas la visión, abarcando la identificación, el seguimiento y clasificación de objetos en grupos o categorías de acuerdo a sus

similitudes o semejanzas [6], al tomar en cuenta las tres acciones, se habla del término bio-inspirado o biológicamente inspirado.

La inspiración biológica surge como resultado de observar que el mundo natural puede ofrecer soluciones a los desafíos tecnológicos, biomédicos o industriales al mostrar que el mundo de la biología tiene muchos ejemplos de diseños enormemente funcionales y adaptados gracias a los cuales los animales o las plantas aprovechan las ventajas de la interacción entre ellos y lo que los rodea [15].

Como resultado de los avances tecnológicos y la inspiración biológica surgen los nuevos sistemas para el análisis y procesamiento de imágenes. Ejemplo de estos nuevos sistemas son la visión estereoscópica, la visión ultra-rápida, la visión bio-inspirada en la polarización de imágenes y la visión bio-inspirada por medio de sensores de silicon retina.

Estos sistemas presentan sus propios enfoques:

La visión estereoscópica describe el análisis de una escena capturada desde dos perspectivas diferentes [14]. El objetivo es encontrar puntos de correspondencia en las imágenes izquierda y derecha indicando la proyección de un punto 3D [14].

La visión ultra rápida está basada en el proceso de análisis del cerebro humano, intentando simular la velocidad a la que el cerebro humano procesa las imágenes [].

La visión bio-inspirada en la polarización, esta característica intrínseca de la luz es invisible al ojo humano sin la ayuda apropiada de instrumentos y el principal uso que se le da es para la navegación [16]. Este es el modo en que algunos insectos y animales marinos perciben el ambiente.

El sensor bio-inspirado de silicon retina es un nuevo tipo de sensor el cual entrega información de manera asíncrona y solo si la intensidad de la luz del ambiente cambia [4].

Como generadores de imágenes, los sensores de silicon retina derivan del sistema de visión humana [4] y, en comparación con generadores de imagen CCD o CMOS basados en cuadros estándar, el sensor de retina es un foto-receptor logarítmico, asíncrono y de tiempo

continuo, y cada pixel entrega datos independientemente sólo en cambios de luminancia [4].

Algunas áreas de aplicación para este sensor de visión incluyen alta velocidad bajo condiciones de luz no controladas, redes de sensores inalámbricos, visión industrial para manufacturación o inspección, sistemas de navegación autónoma (ejemplo: búsqueda de tierra, vehículos voladores), dispositivos de interfaz humana (ejemplo: registro visual), y prótesis visuales [11].

En el capítulo 1 de esta tesis se presenta un panorama general de la problemática a resolver así como una breve explicación de la solución propuesta. En el capítulo 2 *Marco teórico* se presentan algunas definiciones que se aplican para el desarrollo del algoritmo propuesto. El capítulo 3 *Estado del arte* muestra algunos trabajos relacionados al estudio y el tratamiento de la información a procesar, estos trabajos se han considerado los más importantes como referencias para el uso de la información en el algoritmo propuesto. En el capítulo 4 *Análisis y diseño del algoritmo propuesto* se describe paso a paso el proceso de desarrollo y la arquitectura obtenida para el algoritmo. El capítulo 5 *Experimentación y resultados* explica cómo es que el algoritmo desarrollado realiza el procesamiento de la información y qué tipo de resultados se obtienen así como una explicación de esa información. Finalmente en el capítulo 6 *Conclusiones y trabajos a futuro* se explica si los resultados obtenidos han sido satisfactorios o su repercusión es positiva para el estudio realizado y qué posibles mejoras se puedan realizar al algoritmo.

1.2 Descripción del problema

Todavía se estudian métodos para aprovechar al máximo el tratamiento de la información generada por el sensor de *silicon retina* ya que, como se dice en [9] y [10], en lugar de utilizar los eventos generados por el sensor, se realiza una conversión de datos para poder trabajar con ellos como se haría con imágenes.

El realizar la conversión de datos es un proceso que consume tiempo y no se ha explotado la ventaja de la entrega de datos asíncronos y la alta resolución temporal del sensor de *silicon retina*.

Al trabajar con datos obtenidos del sensor se carece de colores y formas definidas, únicamente se tienen puntos independientes unos de otros en una tira de datos llamada bus de eventos.

1.3 Justificación

El sensor de *silicon retina* muestra grandes ventajas en comparación con los sistemas de visión convencionales [3], algunas de esas ventajas son:

- Bajo poder de consumo en el procesamiento de la información [3].
- Alta sensibilidad a los reflejos de luz [3].
- No utiliza datos redundantes [3].
- Ventajas increíbles en velocidad, costo computacional y consumo de energía [2].
- La cantidad de información que debe ser procesada decrementa significativamente comparada con los generadores de imágenes como CMOS o CCD [4].
- Representa una alternativa al uso de imágenes.
- Está biológicamente inspirado en la retina [2] e imita ciertos rasgos de esta.

También, al ser un sensor relativamente nuevo presenta características únicas:

- Usa la representación asíncrona de eventos-dirección o AER por sus siglas en inglés (Addres-Event Representation) para transferir la información visual [2], [4].
- No es dependiente de imágenes.
- Es una nueva forma de percibir el ambiente.
- Abre un nuevo campo de investigación.

Conociendo estas cualidades y ventajas que el sensor de *silicon retina* tiene sobre los generadores de imágenes convencionales es comprensible que las investigaciones estén

abiertas a realizar experimentaciones con estos nuevos sensores y posteriormente deseen migrar las arquitecturas de los convencionales sistemas artificiales de visión a sistemas con sensores de *silicon retina*.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar un algoritmo biológicamente inspirado para realizar el seguimiento de objetos en movimiento emulando el modo de caza de los peces abisales utilizando sensor de *silicon retina*.

1.4.2 Objetivos específicos

- Identificar los factores ambientales que afectan al sensor para la obtención de datos.
- Seleccionar una técnica de identificación de regiones que funcione de forma heurística para seleccionar una región en el área de búsqueda con mayor información.
- Seleccionar una técnica heurística para realizar la selección de los agrupamientos con mayor cantidad de eventos.
- Seleccionar un algoritmo de seguimiento independiente del tiempo.

1.5 Hipótesis

Es posible diseñar un algoritmo biológicamente inspirado que realice el seguimiento de objetos en movimiento utilizando datos de un sensor de *silicon retina* que representan los aces de luz reflejados en las paredes de los objetos a seguir, emulando de esta manera el modo de caza de los peces abisales.

1.6 Propuesta de solución

Dadas las dificultades que se presentan al utilizar los datos que arroja el sensor de *silicon retina* algunos autores sugieren diseñar un algoritmo que pueda manejar los datos asíncronos y procesar cada evento de entrada sin tener que utilizar estrategias para generar frames y mejorar los avances que ya tienen [9], [10].

Además, como lo menciona [14], la finalidad de algoritmos para ese tipo de sensores es procesar los datos asíncronos directamente y a la velocidad a la que van siendo entregados por el generador de imágenes.

Tomando en cuenta estos requisitos, se propone el diseño de un algoritmo para realizar la identificación y el seguimiento de objetos procesando los eventos que entrega el sensor.

el seguimiento de objetos utilizando como sistema de visión el sensor de *silicon retina* y teniendo como base del proceso de seguimiento la emulación del modo de caza de los peces abisales, una especie animal marina que habita en lugares profundos donde la luz del sol es nula, desde los 1000 metros bajo el nivel del mar hasta los 6000 metros bajo el nivel del mar incluyendo el plano abisal [7].

1.6.1 Metodología

Paso 1.

Comprender el funcionamiento del sensor de *silicon retina*, el tipo de ambiente y las condiciones que se requieren para la obtención de información y el tipo de datos que genera, en qué tipo de sistemas se ha utilizado y los alcances que han logrado, obtener grabaciones del sensor.

Paso 2.

Realizar una selección de variables tomadas de las señales obtenidas con el sensor.

Las variables obtenidas del sensor son las siguientes:

- Timestamp, es la marca de tiempo en que ocurre un evento.
- x_addr, es la posición en el eje x del plano que se activa con el evento ocurrido en el tiempo marcado como “timestamp”.
- y_addr, es la posición en el eje y del plano que se activa con el evento ocurrido en el tiempo marcado como “timestamp”.
- Polarity, la polaridad de los eventos representa un incremento (evento activo) o un decremento (evento inactivo) del brillo del punto de escena proyectado [68].

De las variables antes mensionadas, se utilizarán x_addr e y_addr que permitirán ubicar en un plano bidimensional las posiciones de los eventos obtenidos, la polarización también será tomada en cuenta ya que tanto eventos actuales como los eventos pasados en una captura de eventos son importantes para tener más información, finalmente los valores de timestamp no se tomarán en cuenta ya que para las pruebas hay independencia de tiempo.

Paso 3.

Se analizarán algunas especies animales que tengan una visión parecida a la del sensor para seleccionar una que pueda compartir la mayor parte de características de funcionamiento con el sensor, se tomarán las características principales de su comportamiento.

Paso 4.

Se diseñará un algoritmo de identificación y seguimiento de objetos tomando como modelo el estilo de caza de una especie marina abisal.

Este algoritmo constará de tres etapas principales:

- Identificación de regiones: en esta etapa el algoritmo identificará dentro de un área de búsqueda, denominada ambiente, el cuadrante o la región donde hay mayor cantidad de información acumulada.
- Agrupamiento de eventos: en esta etapa el algoritmo selecciona el cúmulo de eventos que estén más relacionados unos con otros dentro de la región previamente elegida del área de búsqueda.

- Seguimiento de objeto: del agrupamiento obtenido se asumirá que es un objeto y se realizará su seguimiento.

Paso 5.

Se seleccionará un tamaño de dimensión deacuerdo a las grabaciones de eventos disponibles y se realizarán pruebas con diferentes grupos de eventos para seleccionar el tipo de tamaño en trozos de tiempo que es preferible para las muestras.

Paso 6.

Se realizarán comparaciones con los tamaños de muestras y la velocidad de procesamiento de otros autores.

1.6.2 Inspiración biológica

Para el diseño del algoritmo bio-inspirado se emulará el estilo de cacería *lurk and lure* o *ambushing* también conocido al español como *al acecho* que utilizan los peces abisales [7]. Estas criaturas, al vivir en la zona abisal del océano, carecen de una visión desarrollada, es decir, no ven objetos a colores o con formas definidas, su visión es de escasa distancia, monocromática y se percibe como destellos de luz causados por el reflejo de la poca luminiscencia que choca con los cuerpos de otros peces.

Debido a la desventaja que su visión poco desarrollada puede representar para visualizar e identificar a una presa, para los peces abisales que cazan al acecho es importante la velocidad y gastar la menor cantidad de energía posible pues los alimentos a esas profundidades en el mar son escasos. Además de eso, para apoyarse, los peces abisales poseen una peculiaridad, tienen membranas lumínicas que sirven como carnada. Los peces que habitan en capas superiores a la zona abisal suelen extraviarse en la oscuridad y confunden la luminiscencia de las membranas de los peces abisales con la luz de la superficie, siendo fácilmente atraídos por la luminiscencia de dichas membranas [7].

El proceso de caza al acecho es el mismo en cualquier animal, esperar a que la presa esté a una distancia dentro del rango aceptable para poder ser atrapado y una vez dentro de ese rango realizar un ataque certero.

Desde la perspectiva del cazador, los peces abisales al no diferenciar colores y ser casi ciegos solo se fían de los reflejos de luz mostrados en el cuerpo del pez extraviado que una vez estando lo suficientemente cerca de la membrana como para reflejar su luz, se vuelven presa fácil para el ataque veloz de unas mandíbulas enormes.

La inspiración biológica proveniente de los peces abisales recae de manera particular en el siguiente factor:

- Tanto la cámara como el pez reaccionan a los cambios mínimos del reflejo de luz [3] percibido en las paredes de los objetos en movimiento que se encuentran dentro de su rango de percepción.

1.6.3 Animales del abismo

Los mares cubren el 71% de la superficie terrestre. El océano tiene un grado vertical máximo de casi 11km y alrededor de un 80% del océano tiene más de 1000 metros de profundidad [7]. En el reino mesopelágico (200 a 1000 metros) la luz de la superficie, aunque muy tenue, todavía es visible si el agua es clara. El reino batial (de 1000 a 6000 metros) incluye el plano Abisal pero excluye las trincheras profundas (el reino hadal) [7].

Los peces del mar profundo son las especies que viven en el reino mesopelágico. Las condiciones son extremas. El agua es fría – tiene una temperatura constante de solo 2 grados Celsius, hay poca diferencia entre el día y la noche y la presión es enorme. Los animales que viven ahí han desarrollado muchas estrategias para hacer frente a eso [7].

Debido a que estos peces viven en una región con poca o ninguna iluminación natural, no pueden confiar únicamente en su vista para localizar presas o compañeros. En lugar de eso, muchos peses de las profundidades son luminosos, otros tienen ojos extremadamente

grandes adaptados a la oscuridad o tentáculos largos para ayudarse a localizar su presa en la oscuridad del océano profundo [7].

Las hembras rape conocidas como “anglerfish” son clásicos predadores de “acechar y atraer” [7]. En lugar de buscar activamente por presas, estos peces agitan un señuelo luminoso para atraerlas [7].

Los señuelos son aletas dorsales modificadas y contienen bacterias lumínicas que producen una luz azulada verdosa. La luz y el movimiento de la carnada atraen a la presa dentro del alcance de las mandíbulas abiertas [7]. Su alimentación es, por tanto, cualquier pez que sea atraído por las luces de su prolongación [61].

Conforme se avanza de la superficie hacia aguas profundas, la cantidad de luz cambia, decreciendo con la profundidad. La calidad de la luz también varía con la profundidad.

Con estas características se deduce que el pez abisal, que usa carnadas lumínicas para cazar, se guía de los reflejos de luz que sus presas muestran cuando se encuentran cerca de sus fauces.

2 Estado del arte

2.1 Introducción

Debido a que la tecnología del sensor de *silicon retina* es relativamente nueva, se ha desarrollado una gama de trabajos de investigación tanto para el desarrollo de dispositivos

con la misma tecnología como para el desarrollo de métodos para procesar los datos obtenidos.

En este capítulo se presentan conceptos básicos de los algoritmos, métodos y filtros utilizados por algunos trabajos relacionados en la literatura para el procesamiento de los datos obtenidos del sensor de *silicon retina*.

2.2 Modelos biológicamente inspirados

Modelos de características biológicamente plausibles propuestos por neuro-científicos para tener en cuenta el procesamiento visual en la corriente ventral de la corteza visual [26].

2.2.1 Address-Event Representation

Los generadores de imágenes AER usan un modelo de computación fundamentalmente diferente que es más rápido y más ligero que las convencionales tecnologías de procesamiento de imágenes. En los generadores de imágenes AER la computación inicia al nivel de pixel, haciendo de la información del generador de imágenes selectiva y orientada a eventos. Un pixel puede ser diseñado para detectar características específicas (ejemplo: saturación de luz, movimiento y contornos). Cada pixel genera un evento cuando se satisfacen sus condiciones, así elimina la necesidad de sondear al generador de imágenes por información [23].

2.2.2 Arquitectura basada en eventos

Los Dynamic Vision Sensors (DVS) imitan ciertos rasgos biológicos de la retina. La continua información de salida de esos sensores es una corriente asíncrona de variaciones

de reflejos codificados como address-events. La AER (ejemplo: eventos de reflejo codificados asíncronamente en coordenadas de pixel x,y) se asemeja a los precisamente cronometrados impulsos eléctricos o impulsos de los nervios ópticos derivados de la retina hasta la corteza visual primaria. Los DVS proveen un amplio rango dinámico con latencia de respuesta baja. Los eventos visuales de entrada son procesados en el tiempo de dominio tiempo-real en contraste a las cámaras convencionales que capturan información visual en intervalos regulares (frames) independientemente de su significancia [2].

La retina biológica genera impulsos eléctricos continuos para representar diferencias en sus longitudes de onda de luz perceptibles. Esta información es propagada con los ganglios retinales (nervios ópticos) vía el núcleo genicular lateral hasta la corteza visual primaria. Similar a esto, “Adress-Event-Representation” es un protocolo de transmisión asíncrona de cambios de reflejo en la forma de eventos de impulsos digitales. Esas señales visuales continuas son codificadas en direcciones espaciales x,y de pixeles de los DVS y el tiempo es representado por su ocurrencia [2].

Address-Event Representation (AER), es un protocolo asíncrono biológicamente inspirado para la codificación y comunicación de los datos sensoriales entre un sensor transmitiendo y una unidad de procesamiento receptora [23].

Un canal de comunicación address-event (AE) es un modelo de transmisión de información neuronal en los sistemas sensoriales biológicos [23].

En la terminología AE, los eventos son paquetes de comunicación que son enviados desde un emisor a uno o varios receptores. Para un sensor de imágenes AE sensible a la intensidad de la luz, los eventos son señalizados cuando los pixeles individuales alcanzan una tensión de umbral y solicitan al bus iniciar comunicación con un receptor externo [23].

Un sistema AE está generalmente compuesto por una multitud de celdas/células básicas o elementos transmitiendo, recibiendo o transmitiendo datos. Un evento tiene la forma simple de la dirección del elemento de transmisión (de aquí el origen del término address-event). Una ventaja importante de los sensores de imágenes AER es que no necesitan ser

consultados por información, por el contrario ellos envían la información al receptor una vez que la han reunido [23].

En un sensor de imágenes AER sensible al movimiento solo los pixeles que perciben un cambio en la intensidad de la luz generarán eventos, en un sensor de imágenes AER sensible a la intensidad de la luz el pixel más brillante generará eventos primero y más frecuentemente que los pixeles más oscuros, así la información de esos pixeles inmediatamente se volverá disponible para el receptor, los sensores AER proveen un número no determinista de eventos por intervalo de tiempo [23].

Los sensores de imagen AER pueden proveer compresión y latencia de respuesta reducida de un sistema de reconocimiento al transmitir solo la información relevante y clasificarla de manera que los datos más interesantes sean priorizados. Esta forma de codificar información es el bloque de construcción básico de una red sensorial capaz de detectar características complejas en una escena, como el comportamiento [23].

2.2.3 Corteza visual

La retina transmite sus señales a los núcleos geniculados laterales (NGL) que a su vez proyectan su información a la corteza visual primaria (V1), en donde se cumple un procesamiento a bajo nivel; desde ahí, las salidas son proyectadas a áreas de procesamiento de alto nivel, responsables del análisis de color, forma y movimiento (áreas V3, V4 y V5 respectivamente). Cerca del 25% de las células en V1, ubicadas esencialmente en las capas 4cα y 4B, son selectivas a la dirección del movimiento [5].

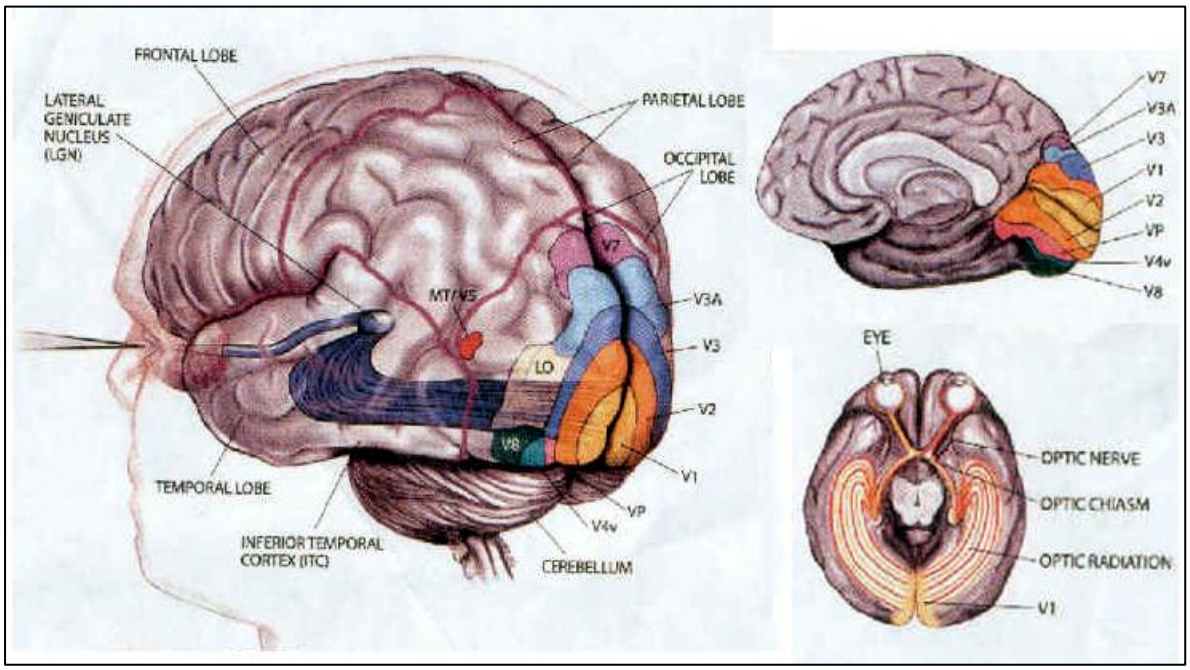

Figura 2.2.3.1 **C**apas de la corteza visual.

En la corteza existen dos corrientes de procesamiento visual: la corriente dorsal y la corriente ventral. La corriente dorsal (la vía del "dónde"), empieza desde V1, sigue en V2, V3 y V5 en humanos o MT en micos, hasta la corteza parietal inferior, se considera está relacionada al movimiento, la representación de localización de objetos y control de ojos y brazos [26]. La corriente ventral inicia desde V1, pasa por las áreas V2 y V4 hacia la corteza ínfero-temporal (IT), se piensa está asociada al reconocimiento de objetos, y por eso también se le llama la vía del "qué" [26].

La mayoría de algoritmos se inspiran solo en la corteza visual primaria –V1, el área temporal media – (V5 en humanos y MT en micos) y/o el área temporal superior media – MST. Las dos primeras son usadas como modelo en la detección de movimiento o incluso solo una de ellas [5].

2.2.4 Células simples y complejas

Los estudios del procesamiento jerárquico en la corteza visual empezaron con Hubel y Wiesel. A través de la investigación de la corteza visual primaria de gatos y macacos [129 y 130 de la tesis 26], ellos propusieron un modelo conceptual jerárquico de células simples a complejas. Las células simples tienen un campo receptivo pequeño y muestran la propiedad de sintonía a estímulos de tipo barra a una orientación y posición particular. Por otra parte, las células complejas tienen un gran campo receptivo y responden mejor también a barras a una orientación particular, sin embargo, muestran fuerte respuesta a la barra en cualquier lugar dentro de su campo receptivo. Hubel y Wiesel sugirieron un modelo jerárquico de la corteza visual, en el cual las células complejas pueden ser construidas integrando entradas convergentes de las células simples [26].

2.3 Trabajos relacionados

Muchos de los trabajos de investigación a pesar de poseer enfoque diferentes, ya sea para seguimiento, clasificación o reconocimiento, han demostrado una cierta similitud empleando algoritmos heurísticos o meta-heurísticos para cumplir sus objetivos y brindar una cierta independencia en la toma de decisiones al momento de que el algoritmo elige los datos a implementar en su proceso de análisis.

Proyectos enfocados en clasificación de objetos como [2], [18] y [21] utilizan redes neuronales y reglas de aprendizaje como ReSuMe, Tempotron y STDP simplificada.

Los proyectos enfocados en el reconocimiento de objetos como [17] y [23] utilizan aprendizaje no supervisado, redes neuronales y reglas de aprendizaje.

Finalmente los proyectos enfocados en el seguimiento de objetos como [3], [19], [22] y [24] no emplean reglas de aprendizaje como los algoritmos de seguimiento y reconocimiento, algunos han empleado redes neuronales y otros han utilizado algoritmos de agrupamiento.

En la tabla 2.1 se pueden apreciar algunos de los trabajos existentes relacionados a la clasificación, el seguimiento y el reconocimiento de objetos, todos empleando la representación de ventos.

PROYECTO	% A FAVOR	FECHA DE PUBLICACIÓN	FINALIDAD	ALGORITMOS	FILTROS	VELOCIDAD DE EVENTOS	TIPO DE APRENDIZAJE	RED NEURONAL	TÉCNICA DE APRENDIZAJE	REGLA DE APRENDIZAJE	CLASIFICACIÓN O SEGUIMIENTO	NÚM. REF.
Gabor feature processing in spiking neural networks from retina-inspired data.		jul-15	Clasificación de figuras		Gabor		Aprendizaje supervisado	(SNN) LIF	Modelo de neuronas de impulsos feedforward leaky integrate and fire (LIF) basado en características Gabor	ReSuMe	clasificación	[2]
AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems.		2006	Clasificación de direcciones de movimiento		Convolución (VLSI winner takes all)		Aprendizaje no supervisado	No	Sub-sistema de aprendizaje de patrones de clasificación espacio-temporal	Aprendizaje de Hebb, identificar las direcciones de movimiento	seguimiento	[3]
Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity.		feb-12	Clasificación de direcciones de movimiento	Algoritmo genético: mutación.			Aprendizaje no supervisado	SNN	Aprendizaje global, aprendizaje por capas.	STDP simplificada	clasificación	[21]
A bio-inspired feed forward system for categorization of AER motion events.	98.5%	2013	Clasificación de posturas humanas	Tempotron	Gabor, operaciones de tipo MAX. Convoluciones (redes de convolución)	intervalos de frame de 20ms	Aprendizaje supervisado	SNN	leaky-integrate type neuron.	Tempotron	clasificación	[18]
Address-event imagers for sensor networks: evaluation and modeling.		abr-06	Reconocimiento de letras								reconocimiento	[23]
Using FPGA for visuo-motor control with a silicon retina and a humanoid robot.		2007						CPG			seguimiento	[19]
Robotic Goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor.	75%	nov-16	Seguimiento de múltiples esferas.	Algoritmo de seguimiento (event driven) mean shift approach	Clustering	2.9 ms		No			seguimiento	[22]

Unsupervised Features Extraction from Asynchronous Silicon Retina through Spike-Timing-Dependent Plasticity.	95%	2011	Detección de carros	Algoritmo genético: evolución genética			Aprendizaje no supervisado	SNN	Aprendizaje global, aprendizaje por capas	STDP	reconocimiento	[17]
Real-time Tracking Based on Neuromorphic Vision.		2015		Compressive tracking algorithm	Banco de filtros multiescala, matriz gaussiana	10 microsegundos					seguimiento	[24]
Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network.	99.48%	2014	Identificación de la postura de la silueta de una persona.	Algoritmo de clustering en el dominio del tiempo.	Gabor, convolución.				LIF neuron	Tempotron, STDP		[25]

Tabla 2.1 **T**abla comparativa de proyectos y características

2.4 Algoritmos

Para realizar la clasificación de objetos, algunos de los métodos más utilizados son k-vecinos más cercanos (K-NN), Máquinas de Vector Soporte (SVM) y Redes Neuronales Artificiales (ANN) [26]. Los proyectos enfocados a reconocimiento y clasificación de objetos [2], [17], [18], [21], [23], [25] tienden a utilizar redes neuronales de impulsos (SNN) y para que estas redes aprendan a reconocer y clasificar los objetos observados por el sensor han requerido implementar técnicas de aprendizaje, algunas de las técnicas aplicadas fueron ReSuMe [2], STDP [17] , [21], [25] y Tempotron [18].

A continuación se realiza una breve descripción de cada algoritmo mencionado anteriormente.

Como parte de la propuesta de esta investigación también se habla de la meta-heurística Optimización por Enjambre de Partículas (PSO), que es una técnica de optimización.

2.4.1 K-NN

El algoritmo K-Nearest Neighbor (K-NN) es un esquema de clasificación simple que clasifica un ejemplo desconocido basado en sus muestras de entrenamiento vecinas. Los vectores características de las muestras de entrenamiento son guardados y etiquetados en la fase de entrenamiento. En la fase de clasificación, un ejemplo de prueba es clasificado asignando la etiqueta que aparece con mayor frecuencia en las muestras de aprendizaje de K-NN [26].

K-NN es un enfoque de clasificación basado en ejemplos en cual requiere una búsqueda entre las muestras de entrenamiento. La constante “k” definida por el usuario usualmente se elige como un número impar en clasificación binaria para evitar empates. Cuando k=1, este se vuelve el algoritmo del vecino más cercano, en el cual el ejemplo desconocido es clasificado asignándole directamente la etiqueta de su vecino más cercano [26].

2.4.2 Support Vector Machine (SVM)

Las máquinas de vector soporte, son un tipo de algoritmo de aprendizaje, originalmente introducido por Vapnik y compañía, y sucesivamente extendido por otros investigadores [33].

Cuando son usados para clasificación, separan un conjunto dado de datos de entrenamiento etiquetados de manera binaria con un hiper-plano que es máximamente distante de ellos (conocido como "el hiper-plano de margen máximo") [33]. Para los casos donde la separación lineal no es posible, pueden trabajar en combinación con la técnica de "kernels", que automáticamente realiza un mapeo no lineal para un espacio de características [33].

2.4.3 Artificial Neural Network (ANN)

Una red neuronal artificial (ANN) o simplemente red neuronal, es un modelo computacional de sistema nervioso [26], consiste de un capa de entrada de neuronas (o nodos, unidades), una o dos (o hasta tres) capas de neuronas ocultas, y una capa fila de neuronas de salida [29], imitando la vasta red de neuronas del cerebro humano [26].

Basado en la topología de la red, una ANN puede dividirse en Feedforward neural Network (FNN) y Recurrent Neural Network (RNN) [26] la cual tiene conexiones de realimentación. La FNN es usada comúnmente en clasificación de datos y reconocimiento de patrones. Las FNN típicas incluyen Multilayer Perceptron (MLP) y Radial Basis Function (RBF).

2.4.3.1 Spiking Neural Network (SNN)

Es sabido que las neuronas biológicas son los bloques de construcción de todo el proceso de información en el cerebro y que están conectadas en patrones complicados para cumplir las actividades diarias, tales como las extremadamente demandantes tareas visuales.

Imitando el comportamiento biológico de las neuronas, los modelos SNN han surgido para mejorar nuestro entendimiento del cerebro e indirectamente aprovechar sus capacidades en aplicaciones potenciales. Las SNN simulan muchas propiedades biológicas con el compromiso principal siendo entre plausibilidad biológica y eficiencia computacional [2].

2.4.3.2 Técnicas de aprendizaje

Los proyectos enfocados a reconocimiento y clasificación que utilizaron redes neuronales de impulsos han requerido implementar técnicas de aprendizaje para el entrenamiento de dichas redes. A continuación se enlistan unas de las técnicas de aprendizaje comúnmente usadas en la literatura haya.

2.4.3.2.1 ReSuMe

Al igual que las células simples de tipo V1 que reciben impulsos de los nervios ópticos, las neuronas LIF se espera que procesen información de entrada de tipo AER. Una regla de aprendizaje se utiliza para que las neuronas LIF reciban instrucciones para trabajar como células simples de tipo V1. Esta regla de aprendizaje es REmote SUpervision MEthod (ReSuMe) y es un método de aprendizaje supervisado biológicamente inspirado para neuronas de impulsos que se concentra en el momento preciso de sus impulsos. Más específicamente, utiliza el enfoque de ventanas de aprendizaje, originalmente introducido por el aprendizaje de Hebb para trenes de impulsos [16 del documento del que lo saqué] en una manera supervisada. Este enfoque de aprendizaje se expande desde el bien conocido método no supervisado STDP [2].

2.4.3.2.2 STDP

La regla de aprendizaje STDP fue demostrada en las neuronas biológicas, y ahora se le considera una base del aprendizaje del cerebro y es extensamente usada, aunque con muchas variaciones, tanto en la ciencia neuro-computacional como en el aprendizaje máquina [17].

Spyke-Timing Dependent Plasticity en su sentido estricto se refiere al cambio en la fuerza sináptica como resultado de provocar eléctricamente pares de potenciales de acción (impulsos) con una diferencia de tiempo fijo entre los potenciales de acción pre-sináptica y post-sináptica [28].

En el contexto computacional, STDP se refiere a reglas de plasticidad que dependen del tiempo de los impulsos pre y post-sinápticos y que están envueltos en varios escenarios de aprendizaje para redes neuronales [28].

Esas reglas de aprendizaje enfatizan la conexión a las biofísicas subyacentes de la modificación sináptica o son minimalistas con respecto a la implementación biológica, o derivan de la maximización de una función de utilidad [28].

Las reglas de aprendizaje están estudiadas en los contextos supervisado, no supervisado o aprendizaje de refuerzo [28].

2.4.3.2.3 Tempotron

Usa el modelo de neurona LIF [25]. El aprendizaje de tempotron es una regla de aprendizaje supervisado basada en gradientes para modelos de neuronas de impulsos que implementan una clasificación binaria de patrones de impulsos multi-neuronal. Un clasificador neuronal cuyas eficacias sinápticas son controladas por la regla de aprendizaje tempotron es referido como tempotron. El tempotron implementa una regla de decisión binaria: Un patrón de impulso es clasificado como patrón objetivo si el tempotron activa o

dispara al menos un impulso de salida en respuesta al patrón. Si la neurona permanece inactiva el patrón es clasificado como un patrón nulo. El tempotron es entrenado en un conjunto de entrenamiento que consiste de objetivos etiquetados y patrones nulos. Cuando el tempotron clasifica mal un patrón de entrada durante el entrenamiento, la regla de aprendizaje ajusta todas las eficacias sinápticas de la neurona de acuerdo a su contribución al máximo potencial de membrana postsináptica: incrementándola cuando la respuesta deseada dispara o activa al menos un impulso y decrementándola si la respuesta correcta permanece en silencio [27].

2.4.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization u Optimización por Enjambre de Partículas es un método estadístico de búsqueda por soluciones óptimas. Este enfoque imita a la inteligencia de partículas y es eficiente para problemas de programación no lineal [36].

PSO es una técnica basada en población, similar en algunos aspectos a los algoritmos evolutivos, excepto que las soluciones potenciales (partículas) se mueven, en lugar de evolucionar, hacia un espacio de búsqueda [32].

Inspirada en la simulación del comportamiento social, fue originalmente diseñada y desarrollada por Eberhart y Kennedy. En PSO, en lugar de usar operadores genéticos más tradicionales, cada partícula (individuo) ajusta su “vuelo” de acuerdo a su propia experiencia de vuelo y la experiencia de vuelo de sus compañeros [34].

La Optimización por Enjambre de Partículas es un método de optimización para funciones no lineales en espacios continuos y discretos, basado en la simulación de un modelo social simple del desplazamiento de cardúmenes y bandadas [42].

Dentro del cómputo evolutivo existen dos ramas los algoritmos evolutivos y la inteligencia de enjambres. Mientras los algoritmos evolutivos poseen una naturaleza centralizada, la inteligencia de enjambres posee un comportamiento distribuido, lo que la vuelve apta para esta investigación [42].

PSO es un método de búsqueda basado en la población, el cual explota el concepto de intercambio social de la información. Esto significa que cada individuo (llamado partícula) de una población dada (llamada enjambre) puede beneficiarse de las experiencias previas de los demás individuos de la misma población. Durante el proceso de búsqueda en el espacio de solución, cada partícula (ejemplo: solución candidata) ajustará su velocidad de vuelo y posición de acuerdo a su propia experiencia de vuelo como la experiencia de otras partículas de compañía del enjambre [43].

2.4.4.1 MSO Multi Swarm Optimization

Los enjambres de Multi-PSO no generalizan inmediatamente, ya que los enjambres no interactúan. Los multi-PSO podrán interactuar si los enjambres tienen acceso a los atractores de otros enjambres; sin embargo, esto reduce a un solo enjambre con diferentes topologías compartiendo información [35].

Un multi-enjambre es, sin embargo, fácilmente construido como una combinación de enjambres de CPSO [35].

2.5 Filtros

Algunos trabajos utilizaron filtros para realizar operaciones de detección de bordes para las capas subsecuentes de la corteza visual [2] y realizar redes de selectividad de objetos [18], otros utilizaron chips de convoluciones [3] para medir pesos de kernesl de convolución integradores de eventos clustering [22] y matriz gaussiana [24] para realizar detección de un objeto en movimiento.

A continuación se realiza una breve descripción de cada filtro mencionado anteriormente.

Como parte de la propuesta de esta investigación también se habla del filtro de Kalman, un filtro muy utilizado para el seguimiento de objetos.

2.5.1 Convolución

La convolución es una forma matemática de combinar dos señales para formar una tercera señal. Es la única técnica más importante en Procesamiento de Señal Digital [40]. Usando la estrategia de descomposición de impulso, los sistemas son descritos por un señal llamada impulso de respuesta [40].

2.5.2 Filtro de Gabor

La función de Gabor es reconocida como una herramienta muy útil en el procesamiento de imágenes computarizado, especialmente en el análisis de texturas, debido a sus óptimas propiedades de localización tanto en el dominio espacial como en el dominio del tiempo [39].

Existen fuertes indicios de que las células simples del córtex visual se pueden modelar mediante filtros de Gabor, sintonizados para detectar diferentes orientaciones y escalas [38].

2.5.3 Matriz gaussiana

En matemáticas, la eliminación de Gauss-Jordan, llamada así debido a Carl Friedrich Gauss y Wilhelm Jordan, es un algoritmo del álgebra lineal para determinar las soluciones de un sistema de ecuaciones lineales, encontrar matrices e inversas. Un sistema de ecuaciones se resuelve por el método de Gauss cuando se obtienen sus soluciones mediante

la reducción del sistema dado a otro equivalente en el que cada ecuación tiene una incógnita menos que la anterior. El método de Gauss transforma la matriz de coeficientes en una matriz triangular superior. El método de Gauss-Jordan continúa el proceso de transformación hasta obtener una matriz diagonal [41].

2.5.4 Filtro de Kalman

El filtro de Kalman recibe su nombre de Rudolph E. Kalman, quien el 1960 publicó su famoso documento describiendo una solución recursiva al problema de filtrado lineal de datos discretos [30].

Escencialmente el filtro de Kalman es un conjunto de ecuaciones matemáticas que implementan un estimador de tipo predictor-corrector que es óptimo en el sentido de que minimiza el error de covarianza estimado – cuando se conocen algunas condiciones. Desde el tiempo de su introducción, el filtro de Kalman ha sido objeto de extensas investigaciones y aplicaciones, particularmente en el área de navegación autónoma o asistida. El filtro de Kalman ha sido usado extensamente para el seguimiento en gráficos interactivos de computadora [30].

La aplicación del filtro de Kalman a sistemas no lineales puede ser difícil [31].

La idea del filtro de Kalman aparece por primera vez en 1960 para lidiar con un sistema lineal bajo la influencia de alteraciones aleatorias. Está diseñado para estimar efectivamente, desde múltiples observaciones acerca de un sistema, su estado en un cierto punto. El método es particularmente efectivo cuando [51].

- La ecuación de movimiento es (quasi)-lineal.
- Hay muchos puntos discretos de mediciones.

- La distancia entre puntos de medición adyacentes es suficientemente corta de tal manera que el estado en el k-ésimo punto está bien aproximado por la extrapolación del estado en el punto (k-1) usando la ecuación de movimiento, incluso en la ‘presencia de la alteración aleatoria entre ellos.

2.5.4.1 Filtro de Kalman extendido

El filtro extendido de Kalman convierte a lineal los modelos no lineales de tal forma que el tradicional filtro lineal de Kalman pueda ser aplicado [31].

Capítulo 3. Análisis y diseño del algoritmo propuesto.

3 Análisis y diseño del algoritmo propuesto

3.1 Introducción

Este capítulo describe al algoritmo bio-inspirado propuesto. Este sistema es la combinación de un sensor de visión de detección en movimiento y una arquitectura bio-inspirada para la identificación y seguimiento de objetos.

3.2 Arquitectura del sistema

Como se habló en el capítulo 1, el sensor de *silicon retina* genera datos asíncronos, por lo tanto los eventos obtenidos no muestran una relación aparente, por esta razón se propone la identificación y el agrupamiento de dichos eventos con el fin de caracterizar un objeto empleando una arquitectura de detección inspirada en un comportamiento biológico.

A continuación se presenta el diseño de la arquitectura propuesta basada en los peces del abismo.

Al realizar una comparacion entre características del sensor de *silicon retina* y los peces del abismo, se encuentras las siguientes similitudes:

- Tanto el sensor como los peces del abismo perciben los reflejos de luz de un objeto siempre y cuando este objeto se encuentre en movimiento y dentro de un rango de percepción.

- Mientras los peces del abismo necesitan un mínimo de luz que se refleje en las paredes de un objeto para poder percibirlo, el sensor de silicon retina puede registrar objetos en movimiento tanto en ambientes con mucha iluminación como en aquellos con iluminación escasa.

- Utilizando sus menbranas lumínicas, los peces abisales atraen a sus prezas a una distancia lo suficientemente cercana para poder atraparlas, esto significa que gracias a la iluminación proporcionada por las membranas, los peces obtienen más información del objeto que están percibiendo, permitiéndoles decidir si lo que se está acercando a ellos es una preza o no. De manera parecida, mientras más información pueda obtener el sensor de *silicon retina* respecto al objeto percibido, más posibilidades se tienen de que durante el procesamiento de los eventos se pueda identificar un objeto.

De esta comparación se crea la propuesta del algoritmo de identificación y seguimiento inspirado en el modo de cacería de los peces del mar profundo (figura 3.2.1).

A continuación se da una breve descripción del proceso a seguir para la identificación y el seguimiento del objeto.

En la parte del sensor de *silicon retina*, el objeto se mueve y el sensor percibe los reflejos de luz que son procesados para determinar si el impulso generado es tomado en cuenta o descartado, para esa toma de decisión se utiliza la umbralización de los datos. Si el impulso obtenido se encuentra en el nivel de aceptación del umbral entonces es enviado como evento en un bus de eventos junto con algún otro impulso que haya ocurrido y también haya sido aceptado en el mismo instante de tiempo.

1.- El sensor percibe movimiento y lo procesa enviando los eventos de los reflejos de luz en el objeto percibido como datos de salida. Estos eventos no contienen información más que una representación de estado activo o inactivo, polaridad y marca de tiempo, es decir, estos datos carecen de otra información como distancias y colores, por tal motivo su interpretación puede ser variada.

2.- Ya que cada evento es independiente del otro, es decir que no existe relación alguna entre un evento y otro, se pueden considerar objetos separados, por lo que es necesario encontrar de entre todos esos objetos una similitud que los relacione y permita crear un conjunto de eventos que pueda ser definido como un candidato a ser un objeto, esta característica en común es la pertenencia a un objeto en la realidad física y ya que no existe información como colores, texturas, sombras u otras características, se lleva a cabo un proceso de identificación del objeto dividido en dos etapas.

2.1.- Para identificar al objeto dentro de todos esos eventos primero se realiza una búsqueda por regiones, la finalidad de esa búsqueda es encontrar la región dentro del espacio de búsqueda que contenga la mayor cantidad de eventos aglomerados. Para la práctica, el espacio de búsqueda se considera en el tamaño de ventana en pixeles que genera cada bus de eventos y el tamaño de esas ventanas está definido por el tipo de sensor que se esté utilizando. En este caso es un sensor DVS-128x128. Para la selección de la región se utiliza la técnica heurística MOPSO.

2.2.- Para la segunda etapa del proceso de identificación, una vez se ha seleccionado la región de eventos, se seleccionan los eventos que tienen mayor relación entre ellos, como criterio para esta búsqueda se utiliza la distancia entre cada impulso o evento. Al conjunto de eventos obtenido de estas dos etapas se le considera el objeto a seguir. Para esta etapa se utiliza una técnica heurística ISOCLUS.

3.- Finalmente se realiza el seguimiento del objeto utilizando como referencia su centroide, el cual se obtiene en la segunda etapa de identificación del objeto. Para el seguimiento del centroide se emplea el algoritmo de seguimiento Kalman.

A continuación el pseudocódigo de la propuesta:

```
Begin
For x=1 to number of iterations
    Obtain events from sensor
    Convert events to readable data
```

Create a matrix bidimentional and store de tada converted

Begin Mopso module
Initialize swarm
Initialize leaders in an external archive
Quality(leaders)
g = 0
While g < gmax
    For each particle
        Select leader
        Update Position (Flight)
        Mutation
        Evaluation
        Update pbest
    EndFor
    Update leaders in the external archive
    Quality(leaders)
    g++
EndWhile
Report results in the external archive
End Mopso module

Begins isoclus module

Initialize seeds which will be the output of the Mopso module

For each coordinate x,y

    Assign each coordinate x,y to their nearest seed

End for

Delete seeds with less coordinates assigned to their clusters

For each seed

Compute the cluster centroid

Compute the mean distance from each coordinate to the cluster centroid

If last iteration or 2k>kinitial

Compute cluster center

End

Else

Compute standar deviation for each cluster and their own vectors

Split clusters too large

Merge clusters too small

Compute average weight for tne new clusters

Back to beginning of isoclus

End if

End isoclus module

Begin E-Kalman module

Obtain the centroid from the previous module

For

Predicted state estimate

Predicted covariance stimate

Compute measurement residual

Compute residual coraviance

Compute kalman gain

Update state stimate

End e-Kalman module

End algorithm.

La figura 3.2.2 ilustra la arquitectura de dicho algoritmo propuesto.

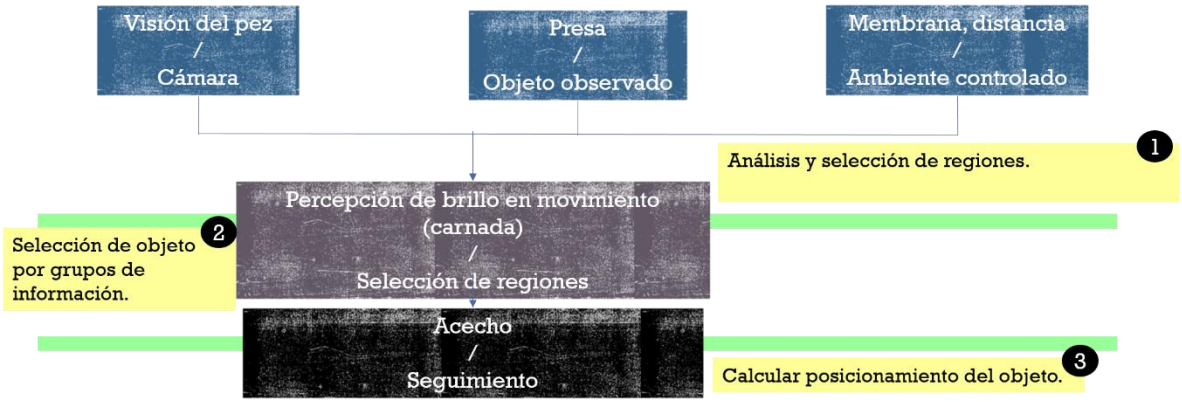

Figura 3.2.1 **C**omparación características pez abisal, sensor de *silicon retina* y arquitectura.

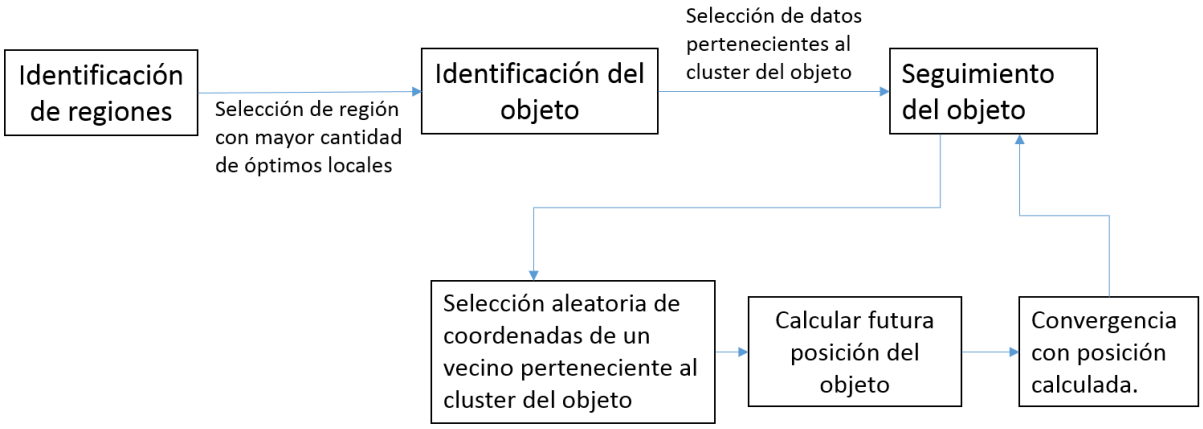

Figura 3.2.2 **D**iagrama de bloques de algoritmo propuesto.

A continuación se explican las etapas del algoritmo.

3.3 Obtención de eventos y extracción de coordenadas

El Sensor de Visión Dinámica (DVS) es un sensor de *silicon retina* con dirección de eventos (address-event) que responde al contraste temporal. Cada dirección de los impulsos de salida representa un cambio cuantificado de la intensidad de registro en un pixel particular desde el último evento en aquel pixel. La dirección incluye un bit de signo para distinguir los cambios positivos de los negativos. Todos los 128x128 pixeles operan asíncronamente y la iluminación de la señal cambia en unos cuantos microsegundos después de ocurrir. No existen frames "completos" de las imágenes, en lugar de eso solo eventos individuales que cambian de señal en una posición espacial particular denotada por una dirección de pixel particular [20].

El sensor puede remover automáticamente el fondo inmóvil y sacar solo el contorno de objetos en movimiento (como un flujo de eventos binarios en movimiento). Esto evita la transmisión masiva de datos en bruto, ahorrando en gran medida el ancho de banda del canal haciéndose así un candidato prometedor de nodo de sensor inalámbrico [26].

Este sensor solo genera imágenes binarias [26].

Los eventos de movimiento generados por el sensor de imágenes de diferencia temporal se guardan y posteriormente son extraídos por el algoritmo [26].

3.4 Análisis y selección de regiones

En esta etapa se pretende identificar las regiones en las que se aglomera la mayor cantidad de información dentro del espacio de búsqueda para posteriormente realizar la búsqueda de objetos en dichos lugares, esto es con la finalidad de facilitar la toma de decisiones respecto a las regiones donde sea preferible buscar la existencia de un objeto.

Para obtener las regiones de interés en este proyecto, se buscó un algoritmo que realizara una búsqueda de posibles lugares donde las probabilidades de encontrar un objeto fueran

buenas, pero se requería que el algoritmo encontrara no solo una sino varias regiones en el espacio de búsqueda. Para lograr dicho objetivo se buscó un algoritmo que cumpliera con esas características de búsqueda.

En matemáticas, estadísticas, ciencias empíricas, ciencia de la computación, o economía, optimización matemática (o bien, optimización o programación matemática) es la selección del mejor elemento de un conjunto de elementos disponibles [59].

De forma general, la optimización incluye el descubrimiento de los "mejores valores" de alguna función objetivo dado un dominio definido, incluyendo una variedad de diferentes tipos de funciones objetivo y diferentes tipos de dominios [59].

La optimización por enjambre de partículas o PSO por su nombre en inglés, es una metaheurística que cumple con las características necesarias para ayudar a encontrar dichos espacios de búsqueda requeridos.

3.4.1 Optimización por Enjambre de Partículas

La optimización por enjambre de partículas (PSO) es uno de los algoritmos heurísticos modernos que pueden ser aplicados en problemas de optimización no lineales y no continuos. Esta es una técnica de optimización estocástica con base en la población para funciones continuas no lineales. PSO fue desarrollado en 1995 por el Dr. James Kennedy, un psicólogo social, y el Dr. Russell Eberhart, un ingeniero eléctrico. El término PSO se refiere a la relativamente nueva familia de algoritmos que pueden ser usados para encontrar soluciones óptimas (o cercanas a lo óptimo) de problemas numéricos y cualitativos. Es fácilmente implementado en la mayoría de lenguajes de programación y ha probado ser tanto efectivo como rápido al ser aplicado en diversos problemas de optimización. PSO fue descubierto a través de simulaciones de un modelo simplificado de un grupo de aves. El Dr. Kennedy y el Dr. Eberhart declararon “La Optimización por Enjambre de Partículas tiene sus raíces en dos componentes principales. Quizás los más obvios son sus lazos con la vida artificial (A-life) en general, y a los grupos de aves, bancos de peces, y la teoría de

enjambres de partículas. También está relacionado al cómputo evolutivo, a los Algoritmos Genéticos (GA) y la Programación Evolutiva (EP).” A diferencia de GAs y EP, PSO es un concepto simple y es muy fácil de implementar [55].

PSO es una técnica de búsqueda heurística (que es considerada por sus autores como un algoritmo evolutivo) que simula los movimientos de las aves en grupos buscando alimentos. La relativa simplicidad de PSO, y el hecho de que es una técnica con base en la población, ha hecho de esta una candidata natural a ser extendida para la optimización multi-objetivo [56].

PSO es una metaheurística, ya que asume pocas o ninguna hipótesis sobre el problema a optimizar y puede aplicarse en grandes espacios de soluciones candidatas. Sin embargo, como toda metaheurística, PSO no garantiza la obtención de una solución óptima en todos los casos [58].

3.4.2 PSO Multi-objetivo

Los problemas de optimización que tienen más de una función objetivo son bastante comunes en cada campo o área de conocimiento. En dichos problemas los objetivos a ser optimizados normalmente están en conflicto con respecto al otro, lo que significa que no hay una solución única para estos problemas. En su lugar, buscamos encontrar buenas soluciones “de compensación” que representen los mejores arreglos posibles entre los objetivos [56].

La Optimización multi-objetivo por enjambre de partículas (MOPSO) es un concepto que surge debido a las limitaciones de optimización para solo un único objetivo que presenta PSO [45]. Con MOPSO se puede realizar optimización para más de un objetivo simultáneamente [45].

Al igual que PSO, MOPSO inicia con una población de soluciones aleatorias, llamadas partículas, que se desplazan a través de un espacio de búsqueda hiper-dimensional [46].

3.4.2.1 Conceptos básicos

Con la finalidad de establecer una terminología en común, a continuación se proveen algunas definiciones de términos técnicos comúnmente usados [47]:

Enjambre: Población del algoritmo [47].

Partícula: Miembro (individuo) del enjambre. Cada partícula representa una solución potencial para el problema a resolver. La posición de una partícula está determinada por la solución que representa actualmente [47].

pbest (personal best): La mejor posición personal de una partícula dada hasta el momento. Es decir, la posición de la partícula que ha proporcionado el mayor éxito (medido en términos de un valor escalar análogo al valor fitness adoptado en los algoritmos evolutivos) [47].

lbest (local best): Posición de la mejor partícula dentro del vecindario de una partícula dada [47].

gbest (global best): Posición de la mejor partícula en todo el enjambre [47].

Líder: Partícula que se usa para guiar a otra partícula hacia mejores regiones en el espacio de búsqueda [47]. Generalmente la partícula *gbest* es elegida como líder [47], pero la selección de la partícula líder depende de la topología a utilizar.

Velocidad (vector): Este vector dirige el proceso de optimización, esto es, determina la dirección en la que una partícula necesita "volar" (moverse), con la finalidad de mejorar su posición actual [47].

Función fitness: Todas las partículas tienen valores fitness los cuales son evaluados por una función fitness para ser optimizados.

Algunos de los parámetros deben ser ajustados, a continuación una lista de parámetros y sus valores típicos [44]:

Número de partículas: el rango típico es de 20 a 40. En realidad para la mayoría de problemas 10 partículas es suficiente para obtener buenos resultados. Para algunos problemas difíciles o especiales, se puede probar con 100 hasta 200 partículas [44].

Dimensión de las partículas: Es determinada por el problema a ser optimizado [44].

Rango de las partículas: También es determinado por el problema a ser optimizado, se pueden especificar diferentes rangos para diferentes dimensiones de partículas [44].

Vmax: Determina el cambio máximo que una partícula puede tener durante una iteración [44]. Generalmente se coloca el rango de las partículas como el Vmax, por ejemplo la partícula (x1, x2, x3) X1 pertenece [-10,10], entonces Vmax=20 [44].

Factores de aprendizaje: Representa la atracción que una partícula tiene hacia otra de sus propios éxitos o de sus vecinos. Se usan dos factores de aprendizaje: C1 y C2. C1 es el factor de aprendizaje cognitivo y representa la atracción que una partícula tiene hacia su propio éxito. C2 es el factor de aprendizaje social y representa la atracción que una partícula tiene hacia el éxito de sus vecinos. Ambos, C1 y C2, generalmente son definidos como constantes [47]. C1 y C2 generalmente equivalen a 2. Sin embargo, otras cantidades también fueron utilizadas en diferentes artículos. Pero generalmente C1 equivale a C2 y rangos de [0,4] [44].

Condición de parada: el número máximo de iteraciones que PSO ejecuta y el requerimiento de error mínimo. La condición de parada depende del problema a ser optimizado [44].

Peso inercial: indicado con la letra w, el peso inercial es utilizado para controlar el impacto del previo historial de velocidades en la velocidad actual de una partícula dada [47]. Se puede promover diversidad por medio del peso inercial [47]. El peso inercial es utilizado para controlar el impacto del historial de velocidades previo al actual [47]. Así, el peso inercial influye en el intercambio/compensación entre las habilidades de exploración global (de amplio alcance) y local (próximo/inmediato) [47]. Un peso inercial grande facilita una exploración global, mientras un peso inercial pequeño tiende a facilitar la exploración local

para afinar el área actual de búsqueda [47]. El valor del peso inercial puede variar durante el proceso de optimización [47].

Versión global vs versión local: la versión global es más rápida pero podría converger en un óptimo local para algunos problemas. La versión local es un poco más lenta pero no es fácil que quede atrapada en óptimos locales. Se puede usar la versión global para obtener resultados rápidamente y se puede usar la versión local para refinar el espacio de búsqueda [44].

Otros términos encontrados en MOPSO:

Función objetivo: Por lo general corresponde al ambiente a ser explorado [48]. Una solución factible que minimiza o maximiza la función objetivo es llamada una solución óptima [48]. La función objetivo contiene múltiples objetivos [45].

El objetivo principal de cualquier algoritmo de optimización multi-objetivo es encontrar el conjunto óptimo de Pareto. Este conjunto óptimo balancea las compensaciones entre los objetivos en conflicto [46].

El concepto de optimalidad de Pareto fue conceptualizado por el economista italiano Vilfredo Pareto, en su trabajo, Manual de Economía Política en 1906. Para definir la optimalidad de Pareto deben ser introducidos unos conceptos básicos [46]:

Dominación: Un vector de posición x_1 domina a un vector de posición x_2 $(x_1 \prec x_2)$, si y solo si [46]:

$$f_k(x1) \leq f_k(x2), \ \ \forall k = 1, \dots n_k \tag{3.4.2}$$

$$f_k(x1) \leq f_k(x2) \ para \ al \ menos \ un \ k. \tag{3.4.3}$$

Óptimo de Pareto: Un vector de posición, $x^* \in F$ es óptimo de Pareto si no existe un vector de posición, $x \neq x^* \in F$ que lo domine [46].

Conjunto óptimo de Pareto: El conjunto de todos los vectores de posición óptima de Pareto conforma al conjunto óptimo de Pareto [46].

Frente de Pareto: Todos los vectores objetivo correspondientes a las posiciones de los vectores del conjunto óptimo de Pareto [46]. Si en lugar de una solución existe un conjunto de soluciones óptimas, se puede elegir las mejores compensaciones en cada caso. Este conjunto de soluciones óptimas es conocido como frente de Pareto en la literatura de la optimización [46]. El objetivo principal de cada algoritmo de optimización multi-objetivo es encontrar el conjunto de Pareto óptimo [46]. Este conjunto óptimo balancea la compensación entre objetivos en conflicto [46].

En PSO, las partículas vuelan a través de un espacio de búsqueda hiperdimensional. Los cambios en la posición de las partículas dentro del espacio de búsqueda están basados en la tendencia psicológica-social de los individuos a emular el éxito de otros individuos [47].

Considerando un espacio de búsqueda *d*-dimensional y *n* partículas, cuya i-ésima partícula en una posición particular $X_i(x_{i1}, x_{i2}, \dots, x_{id})$ se mueve con una velocidad $V_{i(v_{i1},v_{i2},\dots,v_{id})}$. Cada partícula está asociada con su mejor personal, $P_i(p_{i1}, p_{i2}, \dots, p_{id})$ el cual está definido por su propio mejor rendimiento en el enjambre [45]. Cada partícula trata de modificar su posición usando la información de posición y velocidad actuales, distancia entre posición actual y *pbest, y distancia entre posición actual y* gbest.

El movimiento de las partículas está regido por la actualización de sus atributos de posición y velocidad [45]. La posición de cada partícula cambia de acuerdo a su propia experiencia y la de sus vecinos [47]. El vector de velocidad refleja el intercambio social de información [47].

La ecuación (3.4.4) es utilizada para calcular la nueva velocidad de la partícula de acuerdo a su velocidad anterior y las distancias de su posición actual a su mejor posición y la mejor posición dentro del grupo. Luego la partícula se desplaza hacia una nueva posición de acuerdo a la ecuación (3.4.5) [48].

$$V_i^{t+1} = wV_i^t + c_1 r_1\left(x_{pbest} - X_i^t\right) + c_2 r_2\left(x_{gbest} - X_i^t\right) \quad (3.4.4)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (3.4.5)$$

Donde w = Peso inercial, c_1 = Coeficiente d aceleración cognitiva, c_2 = Coeficiente de aceleración social, r_1 y r_2 = Valores aleatorios en el rango [0 1], x_{pbest} es el mejor personal de la partícula y x_{gbest} es el mejor global de la partícula. X_i^t = Posición actual de la i-ésima partícula en la iteración t. V_i^t = Velocidad actual de la i-ésima partícula en la iteración t. V_i^{t+1} = Nueva velocidad de la i-ésima partícula en la iteración $t+1$. X_i^{t+1} = Nueva posición de la i-ésima partícula en la iteración $t+1$.

En MOPSO las ecuaciones de las actualizaciones de velocidad y posición permanecen iguales que en PSO. Todos los parámetros declarados son iguales excepto la función objetivo [45].

3.4.2.2 Algoritmo de MOPSO

La figura 3.4.3.1 presenta el diagrama de flujo del algoritmo MOPSO con base en un criterio de dominación.

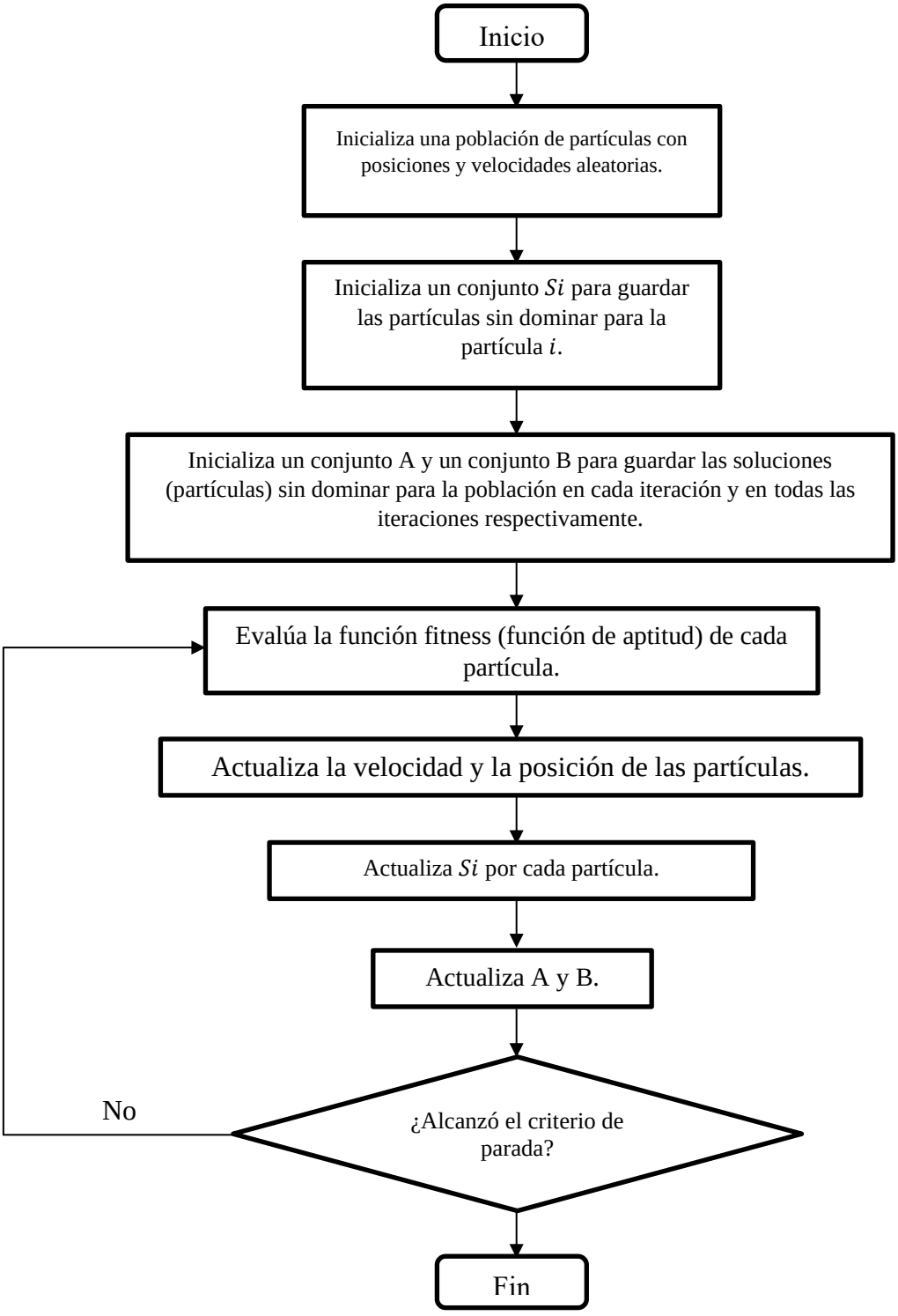

Figura3.4.3.1: **D**iagrama de flujo del algoritmo Multi-Objective Particle Swarm Optimization (MOPSO), con base en algoritmo de [45].

El primer paso en un algoritmo de optimización multi-objetivo es minimizar la distancia entre las soluciones y el frente de Pareto. Para este propósito una función fitness apropiada debe ser definida [46].

Una forma tradicional de asignar la función fitness es el método de agregación, donde la función fitness es una suma de pesos de las funciones objetivo [46].

Sin embargo, este enfoque clásico puede ser muy sensible a la agregación precisa de metas y tender a ser inefectiva e ineficiente donde algunos investigadores propusieron usar métodos muy complejos como las redes neuronales para obtener pesos óptimos de la función objetivo. Algunos nuevos enfoques para la asignación fitness están en la base del dominio de Pareto, donde el fitness es proporcional al rango de dominio de soluciones. MOPSO es un método con base en el dominio que fue propuesto por Coello Coello et al. [11 del artículo]. En este algoritmo las mejores soluciones no dominadas han sido visitadas y están guardadas en un espacio de memoria [46].

Las partes principales del algoritmo funcionan de la siguiente manera [46]:

Después de inicializar una población y un archivo, se deben generar hipercubos del espacio de búsqueda explorado hasta ahora y localizar las partículas usando esos hipercubos como un sistema de coordenadas donde cada coordenada de la partícula está definida de acuerdo a los valores de sus funciones objetivo [46].

La velocidad de cada partícula está calculada por la ecuación (3.4.4).

Para evitar la convergencia a un falso Pareto se propone un operador de mutación. El efecto del operador de mutación decrementa con respecto al número de iteraciones [46]. Esto es controlado con la mutación del parámetro [46].

Después de calcular la posición, cada posición de partícula debe mantenerse dentro del espacio de búsqueda válido. Así cuando una variable de posición vaya más allá de los límites, se deben hacer dos cosas, (1) la variable de posición toma el valor de su límite correspondiente (cualquiera, el superior o el inferior), Y (2) su velocidad se multiplique por (-1) así busca en la dirección opuesta [46].

En el archivo de actualización, este debe permanecer siempre libre de dominio, el tamaño del archivo es finito. Cuando el archivo alcanza su máxima capacidad permitida, esas partículas localizadas en las áreas menos ocupadas del espacio objetivo reciben prioridad sobre esas que recaen en regiones altamente pobladas [46].

En la memoria de actualización de cada partícula, si la posición actual domina al Pbest, Pbest se actualizará con la posición actual [46].

3.5 Identificación del objeto

En esta etapa se pretende reducir la información obtenida, con la finalidad de eliminar aquellos datos que "aparentemente" no son relevantes en nuestro objeto a encontrar y principalmente buscar las aglomeraciones de información que pueden representar a nuestro objeto buscado.

Para lograr dicho objetivo se buscó un algoritmo de agrupamiento que además de realizar los grupos utilizando como semillas las partículas obtenidas del bloque anterior, también pudiera reducir la información, mejorando con esto la forma del objeto.

3.5.1 Clustering no supervisado

El clustering (o análisis de cluster) tiene como objetivo organizar una colección de elementos de datos en grupos, tales elementos dentro de un cluster son más "similares" entre ellos de lo que son con otros elementos en otros clusters. Esta noción de semejanza se puede expresar de maneras muy diferentes, de acuerdo al propósito del estudio, a las suposiciones específicas del dominio y al conocimiento previo del problema [54].

El clustering se realiza generalmente cuando no hay información disponible a cerca de la membresía de los elementos de datos a clases predefinidas. Por esta razón el agrupamiento es visto tradicionalmente como parte del aprendizaje no supervisado [54].

Para apoyar el uso extenso del clustering en la visión por computadora, el reconocimiento de patrones, la recuperación de información, la minería de datos, etc., muchos métodos muy diferentes fueron desarrollados en diferentes comunidades [54].

El clustering no supervisado es una herramienta fundamental en el procesamiento de imágenes para la geociencia y aplicaciones de detección a distancia [49]. Este enfoque es útil cuando datos de entrenamiento confiables son escasos o costosos, y cuando relativamente hay disponible poca información a priori de los datos. Los métodos de clustering no supervisados desempeñan un rol significativo en la búsqueda de clasificación no supervisada [49].

El problema de agrupar puntos en espacios multidimensionales se puede exponer formalmente como uno en un número de problemas de optimización bien conocidos, como el problema Euclideano de k-median cuyo objetivo es minimizar la suma de las distancias al centro más cercano, el problema Euclideano k-center cuyo objetivo es minimizar la distancia máxima, y el problema k-means cuyo objetivo es minimizar la suma de las distancias cuadradas. Se sabe que existen soluciones eficientes solo en casos especiales [49]. Hay soluciones exactas no eficientes conocidas para cualquier problema general de k y algunas fórmulas son conocidas por ser NP-duras [49].

En la práctica, es común usar enfoques heurísticos, los cuales buscan encontrar un clustering razonablemente bueno, pero no proveen garantía en la calidad del resultado. Esto incluye enfoques aleatorios como CLARA y CLARANS y métodos con base en redes neuronales [49]. Una de las heurísticas de clustering más populares y ampliamente utilizadas en la detección a distancia es ISODATA [49]. Se da un conjunto de n puntos de datos en un espacio n-dimensional junto a un entero k indicando el número inicial de clusters y un número de parámetros adicionales. El objetivo general es calcular un conjunto de centros de cluster en d-espacios[49].

Una ventaja significante de ISODATA sobre k-means es que el usuario necesita proveer únicamente un estimado inicial del número de clusters, y con base en varias heurísticas el algoritmo puede alterar el número de clusters ya sea eliminando clusters pequeños, combinando los clusters cercanos o dividiendo grandes clusters difusos [49].

3.5.2 Algoritmo ISOCLUS

La particular variante de ISODATA, llamado ISOCLUS [49], trata de encontrar los mejores centros de cluster mediante un enfoque iterativo. Además usa un número de heurísticas diferentes para determinar si los clusters se fusionan o dividen [49].

En un alto nivel, las siguientes tareas son realizadas en cada iteración del algoritmo: Los puntos son asignados a sus centros de cluster más cercanos, los centros de cluster son actualizados para ser el centroide de sus puntos asociados, los clusters con muy pocos puntos son eliminados, los clusters grandes que satisfacen algunas condiciones son divididos, y los clusters pequeños que satisfacen otras condiciones son fusionados. El algoritmo continúa hasta que el número de iteraciones excede un valor proporcionado por el usuario [49].

Hay un número de parámetros proporcionados por el usuario. Estos incluyen lo siguiente [49]:

k_{init}: número inicial de clusters.

n_{min}: número mínimo de puntos que pueden formar a un cluster.

I_{max}: número máximo de iteraciones.

σ_{max}: desviación estándar máxima de los puntos desde sus centros de cluster a lo largo de cada eje.

L_{min}: distancia mínima requerida entre dos centros de cluster.

P_{max}: número máximo de pares de clusters que pueden ser combinados por iteración.

Sea $S = \{x_1, \dots, x_n\}$ el conjunto de puntos a ser agrupados. Cada punto $x_j = (x_{j1}, \dots, x_{jd})$ es tratado como un vector en un espacio real *d*-dimensional $\mathbb{R}^d$. Sea n el número de puntos. Si el conjunto original es muy largo, todas las iteraciones del algoritmo, excepto la última, pueden ser realizadas en un subconjunto aleatorio de S de un tamaño apropiado. A lo largo del ejercicio, $\|x\|$ es la distancia euclideana del vector x. [49]

(1) Dejando que $k = k_{init}$ muestre aleatoriamente los centros iniciales de cluster $Z = \{z_1, z_2, \dots, z_k\}$ de S.

(2) Asignar cada punto a su centro de cluster más cercano. Para $1 \leq i \leq k$, sea $S_i \subseteq S$ el subconjunto de puntos más cercanos a z_i que a cualquier otro centro de cluster de Z. Esto es, para cualquier $x \in S$,

$$x \in S_j \quad \text{si} \quad \|x \;\; z_j\| < \|x \;\; z_i\|, \forall i \neq j. \tag{3.5.1}$$

(Los lazos con los centros más cercanos son cortados arbitrariamente.) Sea n_j el número de puntos de S_j.

(3) Eliminar los centros de cluster con menos puntos que n_{min}. (Los puntos asociados de S no son eliminados, pero son ignorados durante el resto de las iteraciones.) Ajustar el valor de k y volver a etiquetar acordemente los agrupamientos restantes $S_1, \dots, S_k$.

(4) Mover cada centro de cluster al centroide del conjunto de puntos asociados. Eso es,

$$z_j \leftarrow \frac{1}{n_j}\sum_{x \in S_j} x, \text{ para } 1 \leq j \leq k. \tag{3.5.2}$$

Si algún cluster fue eliminado en el paso 3, entonces el algoritmo va de regreso al paso 2.

(5) Sea $\triangle_j$ la distancia promedio de los puntos de S_j a su centro de cluster asociado z_j, y sea $\triangle$ el promedio general de esas distancias.

$$\triangle_j \leftarrow \frac{1}{n_j}\sum_{x \in S_j}\|x \;\; z_j\|, \text{ para } 1 \leq j \leq k. \tag{3.5.3}$$

$$\triangle \leftarrow \frac{1}{n}\sum_{j=1}^{k} nj \; \triangle_j \tag{3.5.4}$$

(6) Si ésta es la última iteración, entonces colocar $L_{min} = 0$ e ir al paso 9. Además, si $2k > k_{init}$ y es una iteración de número par o $k \geq 2k_{init}$, entonces ir al paso 9.

(7) Por cada cluster S_j, calcular un vector $v_j = (v_1, \ldots, v_d)$ cuya *i-ésima* coordenada sea la desviación estándar de las i-ésimas coordenadas de los vectores dirigidos de z_j hacia cada punto de S_j. Esto es,

$$v_{ji} \leftarrow \left(\frac{1}{n_j} \sum_{x \in S_j} \left(x_i \quad z_{ji}\right)^2 \right)^{1/2} \tag{3.5.5}$$

$$\text{para} \quad 1 \leq j \leq k \text{ y } 1 \leq i \leq d.$$

Sea $v_{j,max}$ la coordenada más grande de v_j.

(8) Por cada cluster S_j, si $v_{j,max} > \sigma_{max}$ y también

$$\left(\left(\triangle_j > \triangle\right) y \left(n_j > 2(n_{min} + 1)\right) \right) \text{ o } k \leq \frac{k_{init}}{2}, \tag{3.5.6}$$

Entonces incrementar k y dividir S_j en dos clusters reemplazando su centro con dos centros de cluster centrados alrededor de z_j separados por una cantidad y dirección que dependan de $v_{j,max}$. Si algún cluster es dividido en este paso, entonces regresar al paso 2.

(9) Calcular las distancias inter-cluster por pares entre todos los diferentes pares de centros de cluster.

$$d_{ij} \leftarrow \left\| z_i \quad z_j \right\|, \qquad \text{para } 1 \leq i < j \leq k. \tag{3.5.7}$$

(10) Ordenar las distancias inter-cluster del paso 9 de manera incremental y seleccionar un subconjunto de máximo P_{max} de los pares de clusters más cercanos, tales que cada par tenga una distancia inter-cluster de máximo L_{min}. Por cada uno

de esos pares (i, j), si ni S_i ni S_j han sido involucrados en una combinación en esta iteración, reemplazar los dos clusters S_i y S_j por un cluster fusionado $S_i \cup S_j$, cuyo centro de cluster asociado sea su peso promedio.

$$z_{ij} \leftarrow \frac{1}{n_i + n_j}\left(n_i z_i + n_j z_j\right). \tag{3.5.8}$$

Volver a etiquetar los clusters restantes y decrementar k acordemente.

(11) Si el número de iteraciones es menor que I_{max}, entonces ir al paso 2.

Si el algoritmo es implementado de la manera más sencilla, y si se asume que el número de clusters k, es mucho más pequeño que el número total de puntos, n, entonces la etapa de mayor tiempo de consumo del algoritmo es el paso 2. Calcular ingenuamente las distancias desde cada uno de los puntos de S hacia cada uno de los k centros por un tiempo total de $O(kn)$ (asumiendo una dimensión fija d) [49].

El algoritmo no necesita calcular explícitamente los centros más cercanos a cada punto. Lo que se necesita es el centroide de los puntos más cercanos a cada centro [49].

Por dentro el algoritmo ISOCLUS está basado en una mejora de una heurística simple y ampliamente utilizada por el agrupamiento k-means, algunas veces llamada algoritmo de Lloyd o el algoritmo k-means [49]. El algoritmo ISOCLUS combina el algoritmo de Lloyd con mecanismos adicionales para eliminar agrupamientos muy pequeños (paso 3), dividir agrupamientos grandes (pasos 7-8), y fusionar agrupamientos cercanos (pasos 9-10) [49].

3.6 seguimiento del objeto

El seguimiento de objetos en tiempo real es la tarea crítica en muchas aplicaciones de visión computacional, tales como vigilancia, interfaces de usuario perceptivas, realidad

aumentada, habitaciones inteligentes, compresión de video con base en objetos y asistencia a los conductores [53].

Dos componentes principales se pueden distinguir en un rastreador visual típico. La representación y localización del objetivo es mayormente un proceso ascendente el cual además tiene que lidiar con los cambios en el aspecto del objetivo. Filtrar y asociar los datos es mayormente un proceso descendente tratando con la dinámica del objeto rastreado, aprendiendo de las escenas previas y evaluando de diferentes hipótesis. El modo en que ambos componentes son combinados y pesados depende de la aplicación y juega rol decisivo en la robustez y eficiencia del rastreador. Por ejemplo, el seguimiento de una cara en una escena llena de gente recae más en la representación del objetivo que en los movimientos del mismo, mientras en vigilancia aérea el movimiento del objetivo y el movimiento de la cámara son los componentes más importantes. En las aplicaciones de tiempo real sólo un pequeño porcentaje de los recursos del sistema puede ser asignado para seguimiento, el resto siendo requerido para las etapas de procesamiento o para tareas de alto nivel como el reconocimiento, interpretación de la trayectoria y el razonamiento. Además, es deseable mantener la complejidad computacional de un rastreador lo más baja posible [53].

La formulación más abstracta del filtrado y el proceso de asociación de datos es a través del enfoque estado-espacio para el modelado de sistemas dinámicos de tiempo discreto. La información caracterizando al objetivo está definida por la secuencia de estados $\{x_k\}_{k=1,\dots}$ cuya evolución en el tiempo es especificada por la ecuación dinámica $x_k = f_k(x_{k-1}, v_k)$. Las medidas disponibles $\{z_k\}_{k=1,\dots}$ están relacionadas a los estados correspondientes a través de la ecuación de medición $z_k = h_k(x_k, n_k)$. En general, f_k y h_k son funciones valoradas como vectores, no lineales y variantes en el tiempo. Cada una de las secuencias de ruido, $\{v_k\}_{k=1,\dots}$ y $\{n_k\}_{k=1,\dots}$ se asume que son independientes e idénticamente distribuidas (i.i.d) [53].

El objetivo de rastrear es estimar el estado x_k dadas todas las mediciones $z_{1:k}$ hasta ese momento, o equivalentemente para construir la función de probabilidad de densidad (pdf) $p(x_k|z_{1:k-1})$. Teóricamente la solución óptima está proporcionada por el filtro recursivo

Bayesiano, el cual resuelve el problema en dos pasos. El paso de predicción usa la ecuación dinámica y la pdf ya calculada del estado en un tiempo $t = k - 1, p(x_{k-1}|z_{1:k-1})$, para derivar la pdf previa al estado actual, $p(x_k|z_{1:k-1})$. Entonces el paso de actualización emplea la función de probabilidad $p(z_k|x_k)$ de la medición actual para calcular la pdf posterior $p(x_k|z_{1:k})$ [53].

Cuando las secuencias de ruido son Gaussianas y f_k y h_k son funciones lineales, la solución óptima está provista por el filtro de Kalman, el cual hace que los siguientes también sean Gaussianos. Cuando las funciones f_k y h_k no son lineales, se obtiene por linearización el filtro de Kalman extendido, siendo la densidad posterior todavía modelada como Gaussiana [53].

3.6.1 Filtro de Kalman

El célebre filtro de Kalman, con raíz en la formulación estado-espacio de los sistemas dinámicos lineales, provee una solución recursiva al problema de filtrado lineal óptimo. Esto aplica tanto a ambientes estacionarios como no estacionarios. La solución es recursiva en que cada estimado actualizado del estado es calculado del estimado anterior y los nuevos datos de entrada, así solo el estimado previo requiere almacenamiento. En adición a la eliminación de la necesidad de almacenar todos los datos pasados observados, el filtro de kalman es computacionalmente más eficiente que calcular el estimado directamente de todos los datos observados en el pasado en cada paso del proceso de filtrado [50].

3.6.1.1 Conceptos

Considerando un sistema dinámico, lineal y de tiempo discreto descrito por el diagrama de bloques mostrado en la figura 3.6.1.1.1. El concepto de estado es fundamental para esta descripción. El *vector estado* o simplemente *estado*, denotado por x_k, está definido como el conjunto mínimo de datos que es suficiente para describir únicamente el comportamiento

dinámico no forzado del sistema; el subíndice k denota el tiempo discreto. En otras palabras, el estado es la menor cantidad de datos en el pasado comportamiento del sistema que es necesaria para predecir su comportamiento futuro. Típicamente, el estado x_k es desconocido. Para estimarlo, se usa un conjunto de datos observados, denotados por el vector y_k [50].

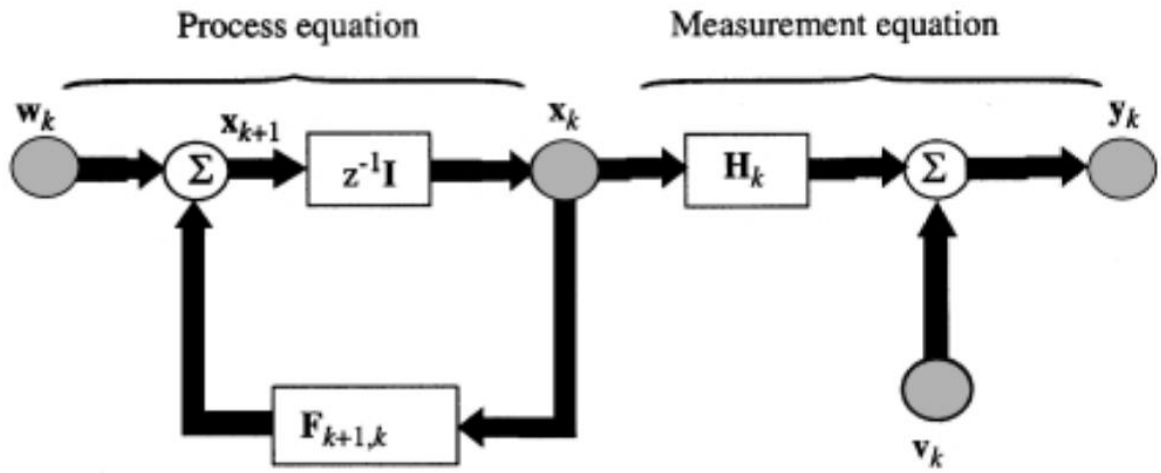

Figura 3.6.1.1.1. **R**epresentación gráfica del flujo de señal de un sistema dinámico lineal de tiempo discreto [50].

En términos matemáticos el diagrama de bloques de la figura 3.6.1.1.1 envuelve el siguiente par de ecuaciones [50]:

1. Ecuación de proceso o predicción

$$x_{k+1} = F_{k+1,k}x_k + w_k, \tag{3.6.1}$$

Donde $F_{k+1,k}$ es la matriz de transición tomando el estado x_k del tiempo k al tiempo $k+1$. El ruido del proceso w_k se asume que es aditivo, blanco y Gaussiano, con una media de cero y matriz de covarianza definida por:

$$E[w_n w_k^T] = \begin{cases} Q_k \; para \; n = k, \\ 0 \quad para \; n \neq k, \end{cases} \tag{3.6.2}$$

Donde el superíndice T denota la transpuesta de la matriz. La dimensión del espacio estado se denota por M.

2. Ecuación de medida o actualización

$$y_k = H_k x_k + v_k, \tag{3.6.3}$$

Donde y_k es el *observable* en el tiempo *k* y H_k es la *matriz de medición. El ruido de medición* v_k se asume que es aditivo, blanco y Gaussiano, con media de cero y matriz de covarianza definida por:

$$E[v_n v_k^T] = \begin{cases} R_k \; para \; n = k, \\ 0 \quad para \; n \neq k, \end{cases} \tag{3.6.4}$$

Además, el ruido de medición $v_k$no está correlacionado con el ruido de proceso w_k. La dimensión del espacio de medición está denotada por *N [50].*

3.6.1.2 Filtro extendido de kalman.

El problema del filtro de kalman considerado hasta este punto en la discusión [50] ha asignado la estimación de un vector estado en un modelo lineal de un sistema dinámico. Sin embargo, si el modelo no es lineal, podemos extender el uso del filtro de Kalman mediante un procedimiento de linearización. El filtro resultante es referido como el filtro de Kalman extendido (EKF). Tal extensión es factible en virtud del hecho de que el filtro de Kalman está descrito en términos de ecuaciones de diferencias en el caso de los sistemas de tiempo discreto [50].

Para sentar las bases para un desarrollo del filtro de Kalman extendido, considere un sistema dinámico no lineal, descrito por el modelo espacio-estado:

$$x_{k+1} = f(k, x_k) + w_k \tag{3.6.5}$$

$$y_k = h(k, x_{k+}) + v_k \tag{3.6.6}$$

Donde, como antes, w_k y v_k son procesos de ruido independientes con una media de cero, blancos y Gaussianos con matrices de covarianza R_k y Q_k, respectivamente. Aquí, sin embargo, el funcional $f(k, x_k)$ denota una función de matriz de transición no lineal que posiblemente también es variante en el tiempo [50].

La idea básica del filtro de Kalman extendido es linearizar el modelo de espacio-estado de las ecuaciones 3.6.5 y 3.6.6 a cada instante de tiempo alrededor del estado estimado más reciente, el cual es tomado para ya sea volverse $\hat{x}_k$ o $\hat{x}_k^-$, dependiendo de cuál funcional particular está siendo considerada. Una vez se obtiene un modelo lineal, se aplican las ecuaciones del filtro de Kalman estándar [50].

Más explícito, la aproximación procede en dos etapas.

Etapa 1. Se construyen las siguientes dos matrices:

$$F_{k+1,k} = \frac{\partial f(k,x)}{\partial x}|_{x=\hat{x}_k}, \tag{3.6.7}$$

$$H_k = \frac{\partial h(k,x)}{\partial x}|_{x=x_k^-.} \tag{3.6.8}$$

Eso es, la ij-ésima entrada de $F_{k+1,k}$ es equivalente al derivativo parcial del i-ésimo componente de $F(k,x)$ con respecto al j-ésimo componente de x. Igualmente, la entrada ij-ésima de H_k es equivalente al derivativo parcial del i-ésimo componente de $H(k,x)$ con respecto al j-ésimo componente de x. En el primer caso, los derivativos son evaluados en $\hat{x}_k$, mientras que en el segundo caso, los derivativos son evaluados en $\hat{x}_k^-$. Las entradas de

las matrices $F_{k+1,k}$ y H_k son todas conocidas (es decir, computables), al tener $\hat{x}_k$ y $\hat{x}_k^-$ disponibles en el tiempo k [50].

Etapa 2. Una vez que las matrices $F_{k+1,k}$ y H_k son evaluadas, se emplean en una aproximación de primer orden de Taylor de las funciones no lineales $F(k, x_k)$ y $H(k, x_k)$ alrededor de $\hat{x}_k$ y $\hat{x}_k^-$, respectivamente [50].

3.6.1.2.1 Ecuaciones aplicadas en el filtro extendido de Kalman

- Modelo del espacio-estado:

$$x_{k+1} = f(k, x_k) + w_k, \tag{3.6.9}$$

$$y_k = h(k, x_k) + v_k, \tag{3.6.10}$$

Donde w_k y v_k son independientes, con media de cero y de ruido Gaussiano, procesos de matrices de covarianza Q_k yR_k, respectivamente.

- Definiciones:

$$F_{k+1,k} = \frac{\partial f(k,x)}{\partial x}|_{x=x_k}, \tag{3.6.11}$$

$$H_k = \frac{\partial h(k,x)}{\partial x}|_{x=x_k^-}. \tag{3.6.12}$$

- Inicialización:

Para $k = 0$, colocar

$$\hat{x}_0 = E[x_0], \tag{3.6.13}$$

$$P_0 = E[(x_0 - E[x_0])(x_0 - E[x_0])^T]. \tag{3.6.14}$$

- Cálculo:

Para $k = 1,2, \ldots$, calcular:

- Propagación del estado estimado:

$$\hat{x}_k^- = f(k, \hat{x}_{k-1}); \tag{3.6.15}$$

- Propagación de covarianza de error:

$$p_K^- = F_{k,k-1} P_{k-1} F_{k,k-1}^T + Q_{k-1}; \tag{3.6.16}$$

- Matriz de ganancia Kalman:

$$G_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}; \tag{3.6.17}$$

- Actualización de estado estimado:

$$\hat{x}_k = \hat{x}_k^- + G_k y_k - h(k, \hat{x}_k); \tag{3.6.18}$$

- Actualización de error de covarianza:

$$P_k = (I - G_k H_k) P_k^-. \tag{3.6.19}$$

Donde I es una matriz de identidad, G_k es la matriz de ganancia Kalman, H_k es la matriz de medición [50].

La traza de la matriz de covarianza de error de estado P decrementa conforme el filtro converge. La traza decrementa con cada lectura de la medición [52].

4 Experimentación y resultados

4.1 Introducción

Este capítulo presenta los resultados obtenidos de la aplicación del algoritmo propuesto. Su desempeño ha sido evaluado utilizando eventos grabados obtenidos de la página de desarrolladores de jAER [57]. Se reconstruyeron algunas imágenes con algunos trozos de eventos, estas imágenes mostraban formas de manos en movimiento en un seguimiento secuencial.

4.2 Algoritmo biológicamente inspirado

El algoritmo obtenido consta de seis etapas, como se muestra en la figura (4.2.1), En la primer etapa, *extracción de datos*, se extraen los datos de paquetes con formato aedat. En la segunda etapa, *conversión de datos*, se utilizan funciones ya definidas para convertir los datos a valores binarios. En la tercer etapa, *creación de matriz 2D*, se genera una matriz con los datos obtenidos de la conversión, esta matriz reduce la cantidad de información y permite una mejor visualización al ojo humano. En la etapa 4, *análisis y selección de regiones*, se utiliza el algoritmo MOPSO para realizar una búsqueda aleatoria de las zonas con mayor cantidad de información. Durante la etapa 5, *selección de objetos por grupos de información*, se generan agrupamientos conservándose los que más cantidad de información contienen, estos agrupamientos posteriormente ayudarán a realizar el seguimiento del objeto en la última etapa *calcular posición del objeto*.

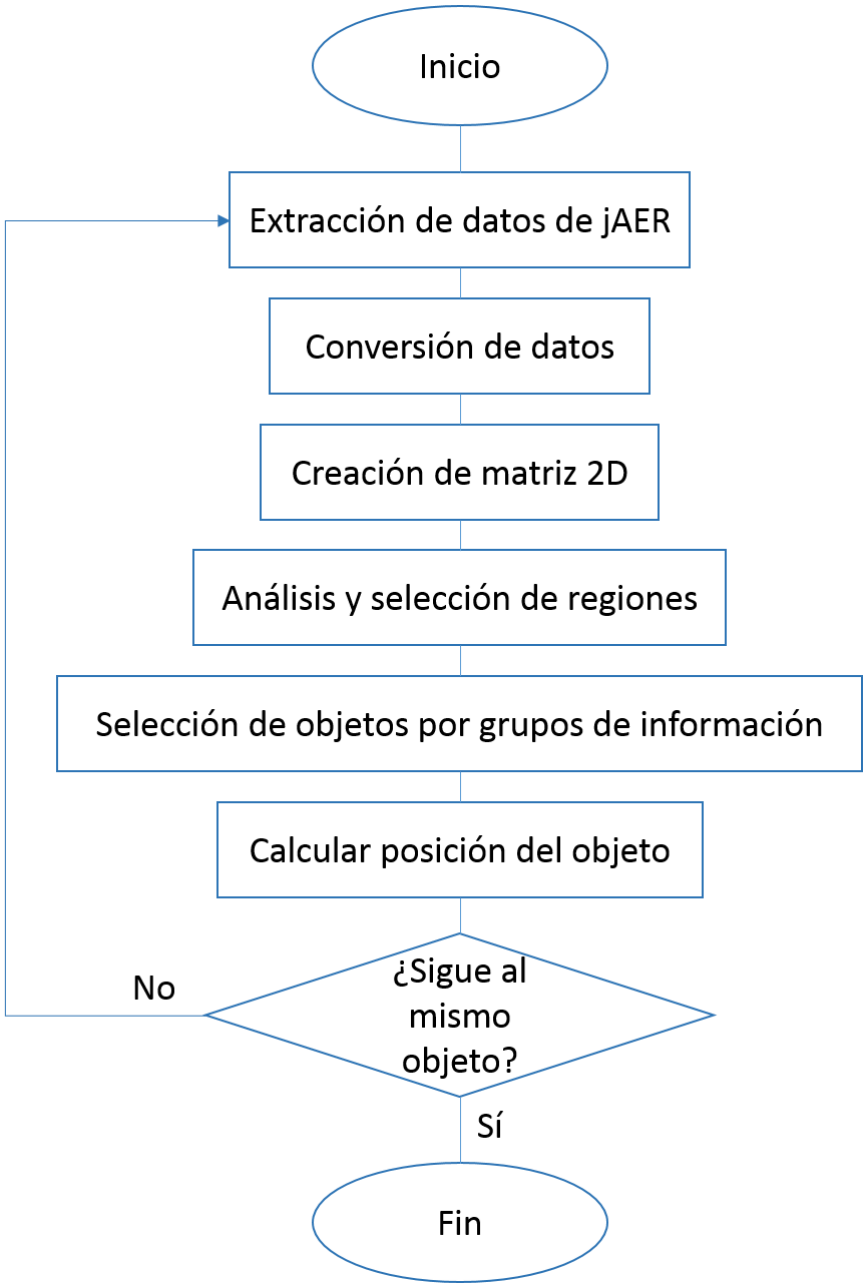

Figura 4.2.1 **D**iagrama de flujo del algoritmo bio-inspirado

Como se mencionó en el capítulo 1. Este algoritmo está inspirado en el método de cacería de los peces abisales, *al acecho* o *acechar y atraer* debido a las características que comparte con el sensor de *silicon retina*.

4.3 Procesamiento de Eventos

Para la extracción de los eventos requeridos por el algoritmo, se utiliza un software diseñado por los creadores del sensor de *silicon retina*, este software es llamado jAER ya que está programado en java y se puede descargar en la página de jAER (https://sourceforge.net/projects/jaer/). La versión utilizada es *jAERViewer1.5_win64* (figura 4.3.1).

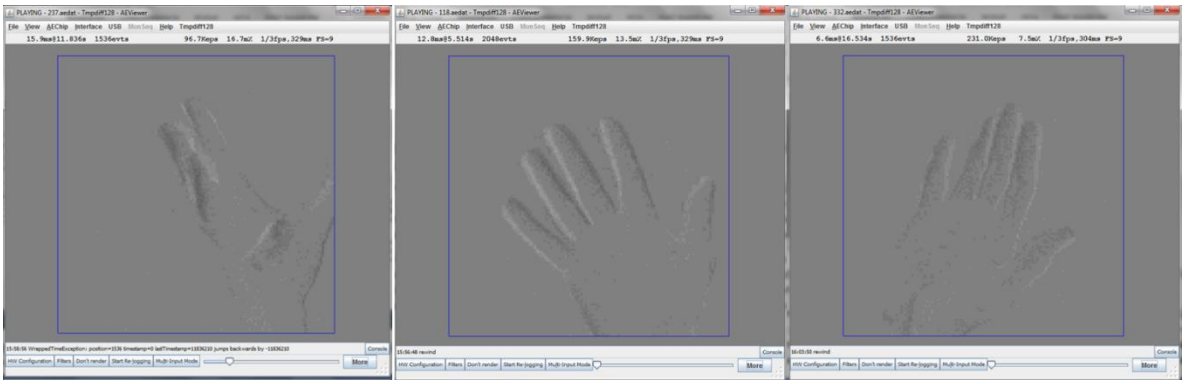

Figura 4.3.1 **P**antalla principal del sistema para visualizar los eventos, jAER, imagen de muestra reconstruida.

Por medio de este software es posible trabajar con información en formato AER (Address Event Representation) ya guardada, es decir, tiras de eventos almacenados que pueden reproducirse utilizando ese software. jAER también puede trabajar con eventos obtenidos en tiempo real, sin embargo para la práctica se utilizan eventos ya guardados. Las acciones capturadas pertenecen a una mano en movimiento (figura 4.3.1).

Es importante mencionar que, debido a que se trabaja con eventos guardados y no eventos obtenidos en tiempo real, se emplea Matlab para realizar la ejecución del algoritmo. Si se elige trabajar con eventos en tiempo real, entonces los códigos utilizados solo deben ser

traducidos a java, que es el único lenguaje en el que se trabaja con eventos en tiempo real; para más información, se encuentra disponible en la página de discusión de jAER [57].

4.4 Obtención de eventos y extracción de coordenadas

La figura 4.4.1 muestra los bloques que conforman las etapas de extracción y conversión de la información.

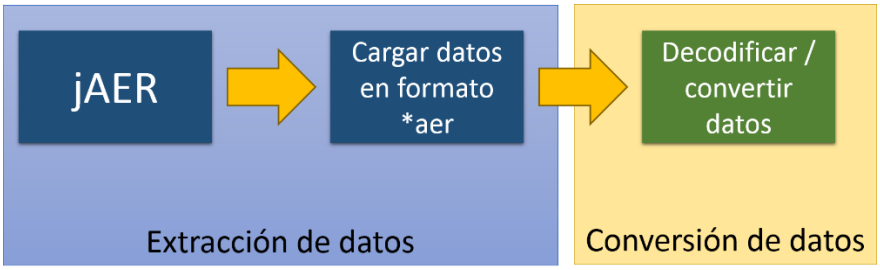

Figura 4.4.1 **P**rimeras etapas del algoritmo.

Para obtener los eventos de una escena ya grabada, se utilizan funciones ya establecidas por los creadores del sensor de *silicon retina*. Estas funciones están disponibles en la página de discusión [57].

Las funciones empleadas para los bloques de extracción de datos y conversión de datos son:

- loadaerdat.m: esta función abre o lee los archivos en formato *.aedat* extrayendo dos conjuntos de información: **allAddr** contiene la información de todas las direcciones como su polarización (figura 4.4.2), **allTs** guarda todas la marcas de tiempo de los eventos guardados.
- extractRetina128EventsFromAddr.m: esta función permite decodificar la información de la matriz allAddr, adquiriendo las coordenadas y la polaridad de los eventos guardados (figura 4.4.3).

allAddr

allAddr <8192x1 uint32>

	1	2
1	12974	
2	5819	
3	19847	
4	13288	
5	20693	
6	12701	
7	16939	

Figura 4.4.2 **a**llAddr son uint32 (o uint16 para grabaciones guardadas) direcciones sin procesar.

Variables - x

x <8192x1 double>

	1	2
1	40	
2	34	
3	60	
4	11	
5	21	
6	49	
7	106	
8	38	
9	71	

Variables - y

y <8192x1 double>

	1	2
1	50	
2	22	
3	77	
4	51	
5	80	
6	49	
7	66	
8	23	
9	99	

Variables - pol

pol <8192x1 double>

	1	2
1	1	
2	-1	
3	-1	
4	1	
5	-1	
6	-1	
7	-1	
8	-1	
9	-1	

Figura 4.4.3 **P**ara tmpdiff128 extrae eventos de retina de un vector addr de 16 bits. addr es un vector de n direcciones de eventos, regresa direcciones *x* e *y*, y polos de polaridad ENCENDIDO/APAGADO con pol=1 para ENCENDIDO y pol=-1 para APAGADO.

Esas coordenadas y polarización obtenidos se utilizan como los datos iniciales (datos de entrada) para el procesamiento y la identificación de los objetos, en total tres vectores de información.

Originalmente los tres vectores de información que contienen las direcciones para X, Y y la polarización pueden llegar a contener información repetida debido a que la representación

del tiempo al que pertenecen puede haber tenido el mismo pixel activo más de una ocasión, generando con esto más información. Una manera de lidiar con las repeticiones es que al diseñar una matriz bidimensional, recreando con esto la imagen del objeto, todos los pixeles aparecen activos solo una vez. Para el caso únicamente se toman en cuenta los pixeles con polarización positiva.

4.5 selección de regiones

Antes de aplicar el algoritmo de Optimización Multi Objetivo por enjambre de partículas, se crea una matriz de 128x128 pixeles o posiciones, esto debido a que los datos procesados del sensor provienen de una escena grabada con una cámara correspondiente a esa cantidad de pixeles. Esta matriz contiene, en resumen, los datos o puntos activos que son necesarios para formar los objetos de manera visual. Al convertir los vectores de coordenadas en matrices, aquellos puntos que fueron activados más de una ocasión son guardados una sola vez, de esta manera se reduce la cantidad de información para su siguiente uso como se muestra en la figura 4.5.1.

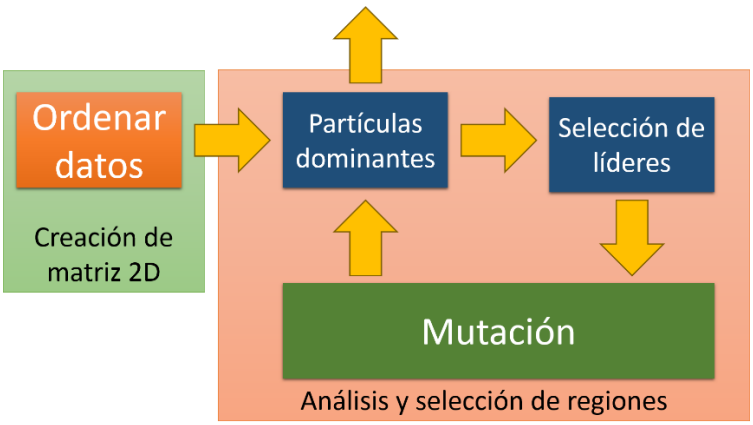

Figura 4.5.1 **E**tapas de creación de matriz 2D y análisis y selección de regiones

Una vez que se han ordenado los datos en una matriz (figura 4.5.2), estos son procesados con el algoritmo Multi Objective PSO.

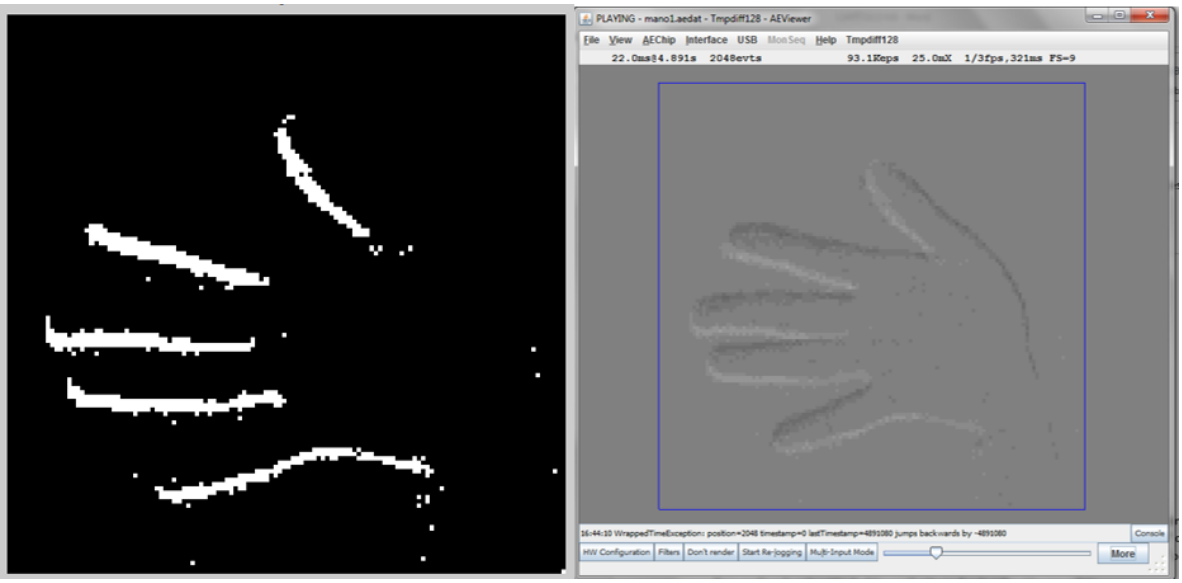

Figura 4.5.2 **I**magen obtenida después de ordenar la matriz.

Para realizar la selección de partículas dominantes se utilizó como criterio de selección la cantidad de puntos activos en vecindad alrededor de cada partícula aleatoria, siendo su *radio* de búsqueda un valor impar elegido por el usuario. Esto sería también el cálculo de valor *fitness* o *costo*.

Para realizar la selección de líderes el criterio fue la selección por ruleta.

Un factor importante respecto al uso de este algoritmo es que al ser una meta-heurística[58], los resultados obtenidos no siempre serán los deseados por el usuario, sin embargo esto también permite que el algoritmo tenga libertad de movimiento.

Otros factores de importancia son las variables de entrada del algoritmo MOPSO:

- La variable que permite definir en cuántas dimensiones se está trabajando, para la práctica esto es bidimensional.
- Recordando que cada dimensión representa a una característica dentro del algoritmo.

- Un kernel de valor variable que se utiliza para contar la cantidad de impulsos activos alrededor de cada partícula del algoritmo.
- El peso inercial.
- La tasa de amortiguación.
- Las variables c1 y c2 que son conocidas como *coeficiente de aprendizaje personal* y *coeficiente de aprendizaje global* respectivamente, estas dos variables son parte primordial para la modificación del comportamiento de las partículas ya que de acuerdo al valor que tengan será la preferencia de comportamiento, mientras una mayor ponderación para el coeficiente de aprendizaje global tiende a reducir el espacio de búsqueda a una mayor velocidad, una mayor ponderación al coeficiente de aprendizaje local permite una mayor expansión en el espacio de búsqueda aunque esto signifique que el tiempo de búsqueda aumente.
- Para la selección de líderes el criterio de selección fue designado al método de selección por ruleta.

El código utilizado es una modificación del código obtenido de la página (http://yarpiz.com/59/ypea121-mopso).

Los resultados obtenidos mostraron mejoras conforme aumentaba el valor del coeficiente de aprendizaje local o personal. La figura 4.5.3 muestra una escena de los resultados obtenidos, representando a las partículas finales en color azul.

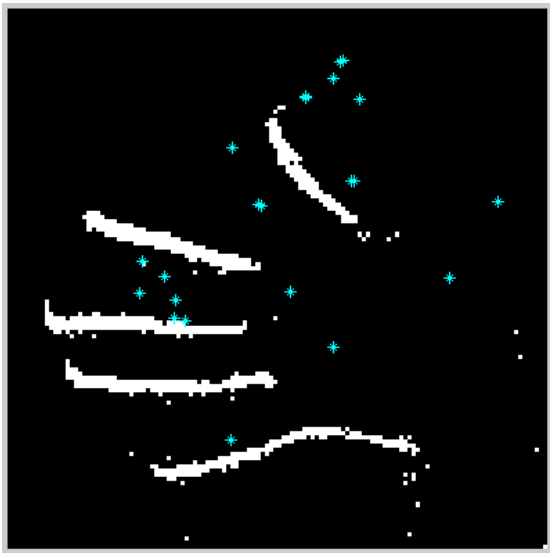

Figura 4.5.3 **R**esultado final de la ejecución del algoritmo MOPSO, los puntos azules indican las posiciones de las partículas dominantes finales.

4.6 Selección de objeto

Para la siguiente etapa, como datos de entrada se utilizaron las partículas obtenidas de la etapa anterior y la misma matriz 2D. Como se muestra en la figura 4.6.1 los pasos a realizar en el algoritmo ISOCLUS se pueden resumir en cuatro etapas: la etapa de agrupamiento de datos, la etapa de eliminación de datos y las etapas de división y combinación de grupos.

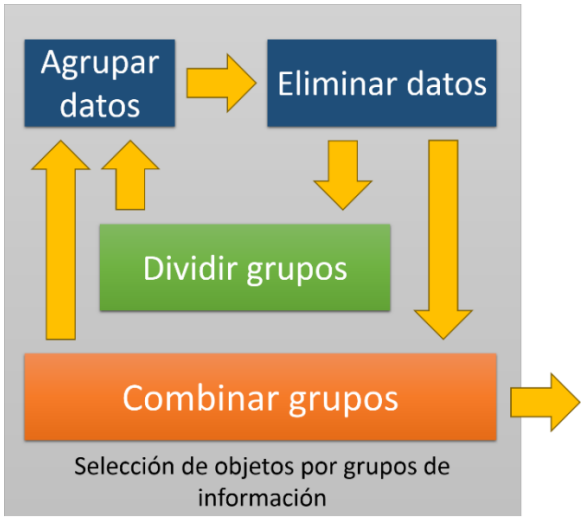

Figura 4.6.1 **D**iagrama de bloques de la etapa de selección.

En la primer etapa (agrupamiento de datos), se usan las partículas como centros de cluster iniciales, para las siguientes iteraciones los centros de cluster estarán cambiando de acuerdo a los criterios que el algoritmo indique. En la segunda etapa (eliminar datos) se eliminan agrupamientos con una cantidad de elementos pertenecientes inferior a una cantidad mínima elegida por el usuario. En la tercer etapa, si algún agrupamiento o cluster tiene una cantidad de elementos pertenecientes mayor a un valor establecido por el usuario, entonces el agrupamiento o cluster es dividido. Si la tercer etapa no generó cambios en los agrupamientos o clusters, entonces la cuarta etapa es realizada y se buscan agrupamientos que sean menores a un valor predefinido y mayores al valor mínimo de elementos en un agrupamiento para ser combinados.

Como criterio de agrupación se utilizó la distancia euclideana entre las partículas y los puntos que conforman al objeto representado en la matriz 2D. Durante la etapa de eliminación de grupos pequeños, debido a que se usa una variable, el resultado puede variar desde eliminar grandes cantidades de información hasta realizar cero cambios. Lo mismo ocurre con el criterio para combinar agrupamientos, puede que ocurran combinaciones y puede que no se realice cambio alguno.

Se menciona en el artículo [49] que el paso donde se tarda más tiempo el algoritmo ISOCLUS es en el paso 2 de diez, generando un tiempo de O(kn).

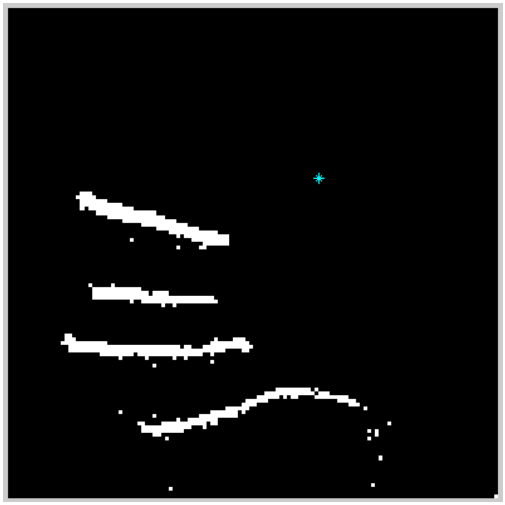

Figura 4.6.2 **R**esultado obtenido de aplicar ISOCLUS.

La figura 4.6.2 muestra el resultado obtenido después de aplicar el algoritmo ISOCLUS a los datos obtenidos previamente. Se puede observar que algunos datos fueron eliminados, por ejemplo el dedo pulgar de la mano y el borde de la muñeca.

El punto marcado en azul es el centroide calculado que se utilizará para hacer el seguimiento con KALMAN.

4.7 seguimiento del objeto

En la etapa para el cálculo de la posición del objeto y su seguimiento (figura 4.7.1), se tomó en cuenta que el fondo es un entorno invariante. Como variable de entrada solo se requiere el centroide del objeto anteriormente obtenido.

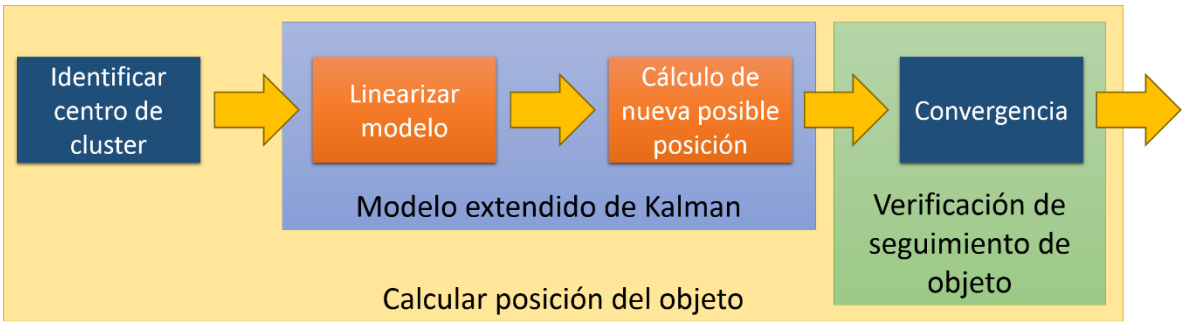

Figura 4.7.1 **D**iagrama de bloques para calcular la posición del objeto.

Para esta etapa (filtro de Kalman) se utilizó una función llamada assignmentoptimal para el problema de asignación rectangular.

El problema de asignación rectangular aparece por ejemplo en aplicaciones de rastreo, donde uno tiene M pistas existentes y N mediciones. Por cada posible asignación, se calcula un costo o una distancia. Todos los valores de costo forman una matriz, donde la columna índice corresponde a las pistas y la columna índice corresponde a las mediciones [66].

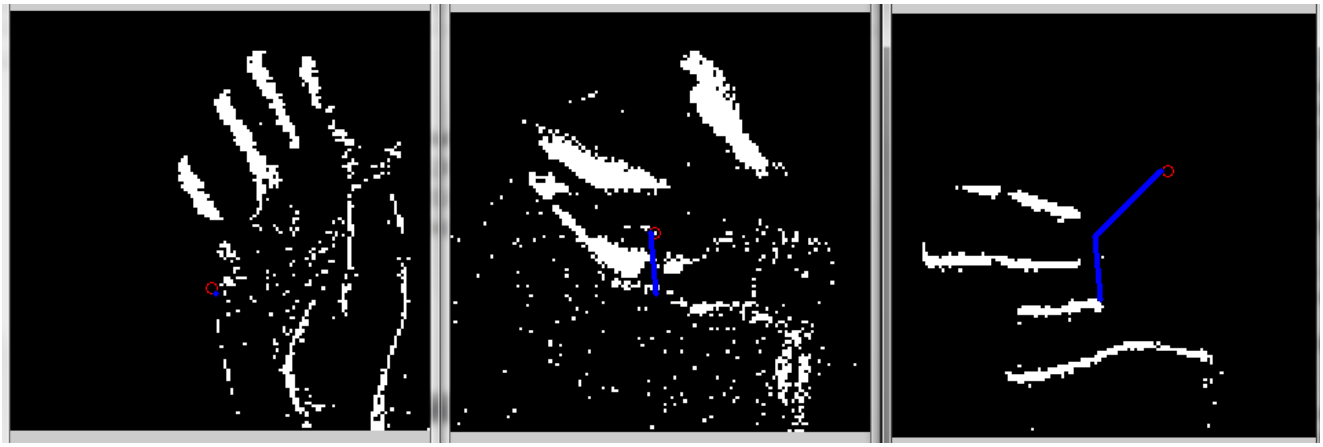

Figura 4.7.2 **F**uncionamiento de algoritmo de Kalman para el seguimiento del objeto.

Para demostrar que el objeto rastreado es el mismo en todo momento, se verifica que las variables aleatorias se muevan en la misma dirección, si se mueven hacia direcciones contrarias entonces el cálculo de la matriz de covarianza es negativo, significando con esto

una divergencia que es interpretada como un objeto diferente al que se está siguiendo. Si por el contrario, la dirección es la misma, es decir convergente, entonces el objeto sigue siendo el mismo.

4.8 Validación.

Las medidas de las diferentes magnitudes físicas que intervienen en una experiencia dada nunca pueden ser exactas y ya que todo instrumento de medición tiene una precisión limitada, no es posible conocer el valor exacto de ninguna magnitud. Cualquier resultado numérico obtenido experimentalmente debe presentarse siempre acompañado de un número que indique cuánto puede alejarse este resultado del valor exacto [69].

En general, se define como error absoluto de una medida a la diferencia existente entre el valor exacto de la magnitud y el valor obtenido experimentalmente. Como no se puede saber el valor exacto, tampoco se puede conocer el error absoluto así definido. El objetivo de la teoría de errores es la estimación de la incertidumbre asociada a un resultado dado. A esta incertidumbre se le denomina también error absoluto [69]. Por tanto, el resultado de la medida m de cualquier magnitud M debe expresarse:

$$m(\pm E) \tag{4.8.1}$$

Siendo E el error absoluto. El doble signo $\pm$ se coloca porque el error puede producirse por exceso o por defecto. No obstante, el error absoluto de una medida no nos informa por sí solo de la bondad de la misma [69].
Por ello, se define como error relativo al cociente:

$$E/m \tag{4.8.2}$$

Que a veces se multiplica por cien, cualificando la incertidumbre en porcentaje de la medida realizada [69].

Para realizar la validación del algoritmo se toman en cuenta las características de los datos a analizar y el resultado obtenido.

Los eventos ocurren de manera aleatoria y son dependientes del movimiento y de los niveles de iluminación, por una parte la dependencia de los niveles de iluminación se refleja en el nivel de umbral que posee el sensor de *silicon retina*, ya que mientras menos luz pueda percibir el sensor, el evento puede que no supere el nivel de umbral establecido necesario para marcarse en la salida de los datos del sensor; por otra parte los periodos de ocurrencia de los eventos no son lienales, esto quiere decir que la cantidad de eventos a procesar es diferente en cada iteración del algoritmo y el resultado en la salida de cada iteración con el algoritmo de seguimiento podría verse severamente afectado en situaciones donde se perciban pocos eventos en la iteración n y en la siguiente se perciban demasiados.

Tomando en cuenta las características anteriores, se opta por realizar el cálculo del error cuadrático de la media o desviación estándar. Empezando por clasificar la búsqueda de un error de precisión de tipo aleatorio o casual debido a las características del experimento, la teoría de los errores casuales proporciona un método matemático para calcular con buena aproximación cuánto puede alejarse del valor verdadero, el valor medido experimentalmente para una magnitud dada M [69].

Definiendo como resultado experimental m de una medida y como error absoluto E de la misma, se realiza la medición en cada iteración del algoritmo obteniendo varias medidas. La caracterización de los errores casuales se hace en este caso mediante la ayuda de la Estadística. La filosofía del método parte del hecho de que el valor exacto de la magnitud es inaccesible, y el proceso de medida es un proceso aleatorio que viene gobernado por una distribución de probabilidad normal o gaussiana [69].
La forma funcional de esta distribución es:

$$P(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) \qquad (4.8.3)$$

$x = \mu$ es el valor más probable al realizar una medida ya que para ese valor la distribución de probabilidad presenta un máximo. El parámetro σ nos da una medida de la anchura de la campana. La probabilidad de que al realizar una medida obtengamos un valor comprendido

en un intervalo cualquiera viene dada por el área que hay bajo la curva gaussiana en ese intervalo [69].

Para determinar con exactitud habría que hacer infinitas medidas. Sin embargo en la práctica se realizarían un número finito de medidas que darían los valores m1, m2, m3, …, mn. Sobre ese conjunto finito de medidas, la Estadística nos permite definir y calcular una serie de estadísticos a saber:

Valor medio o media aritmética de los n valores m_i (i=1,…,n):

$$\bar{m} = \frac{1}{n}\sum_{i=1}^{n} mi \tag{4.8.4}$$

Desviación de la medida m_i respecto de la media:

$$h_i = m_i - \bar{m} \tag{4.8.5}$$

Error cuadrático medio o desviación típica de las n medidas:

$$s = \sqrt{\frac{\sum_{i=1}^{n} {h_i}^2}{n-1}} \tag{4.8.6}$$

El valor de S da una idea de la dispersión de las medidas m_i con respecto de la media $\bar{m}$

Error cuadratico de la media o desviacion estandar de las n medidas

$$s\bar{m} = \frac{s}{\sqrt{n}}\sqrt{\frac{\sum_{i=1}^{n} {h_i}^2}{n(n-1)}} \tag{4.8.7}$$

El valor de $s\bar{m}$ es muy importante porque informa cómo de parecido es el valor Medio $\bar{m}$ delas medidas al valor más probable μ del proceso aleatorio global.

Por conclusión $\bar{m}$ da una estimación de μ, y cuanto menor sea la desviación estándar $s\bar{m}$ más se parece realmente $\bar{m}$ a μ. La desviación estándar decrece a medida que el numero n de medidas es mayor.

Se realizan pruebas midiendo la diferencia entre la posición del centroide obtenida de Kalman después de cada iteración contra las medidas anteriores, siendo cada posición obtenida en ejes x e y las muestras m, n como la cantidad de muestras.

5 Conclusiones y trabajos a futuro

5.1 Conclusiones

Desde la creación del primer modelo del sensor de *silicon retina* VSLI hasta la fecha ya han transcurrido más de 20 años. Y durante este tiempo el interés por experimentar con este nuevo dispositivo ha ido en aumento, probando este sensor en actividades que requieren del procesamiento de imágenes.

Gracias al modo de operar del sensor de *silicon retina*, se tienen muchas ventajas de este al compararlo con los generadores de imágenes convencionales [2, 3, 4]. Y es debido a esas características que este sensor es una excelente alternativa al momento de migrar a otro sistema generador de imágenes.

De igual forma en los recientes años se han realizado diferentes sistemas utilizando como base los datos que genera este sensor, algunos trabajan utilizando redes neuronales, otros trabajan utilizando algoritmos de agrupamiento. La mayoría de investigaciones utiliza métodos o técnicas de inteligencia artificial como técnicas y reglas de aprendizaje (modelo de neuronas LIF [2, 25], aprendizaje global y por capas [17, 21], ReSuMe [2], Tempotron [18], STDP simplificada [21]) con la finalidad de que esos sistemas logren emular mejor el procesamiento de información visual como lo hace el ser humano.

Se propuso una arquitectura de sistema que imita el comportamiento de formas de vida marina que habitan el plano abisal. A diferencia de la mayoría de sistemas desarrollados para el tratamiento de imágenes generadas por el sensor de *silicon retina*, la arquitectura del sistema propuesto no tiene un enfoque dirigido a emular el comportamiento de la corteza visual humana, sino a emular el comportamiento de cacería de un animal marino cuyas características sensoriales son parecidas a las del sensor de *silicon retina*.

Se utilizaron como base los datos de tipo AER obtenidos de la página de desarrolladores [57]. Estos datos originalmente fueron grabados de un sensor de 128x128 pixeles

(ch.unizh.ini.jaer.chip.retina.Tmpdiff128 en la ventana de dispositivos de la aplicación jAER).

Al igual que en otras investigaciones, se optó por trabajar con trozos de eventos que contenían información de periodos aproximados a 20ms, encontrando que la información es basta para poder visualizar la forma de un objeto.

Esta arquitectura está dividida en seis bloques de análisis y procesamiento, y está enfocada al aprendizaje de tipo no supervisado, utilizando técnicas heurísticas (MOPSO e ISOCLUS), un algoritmo genético (selección por ruleta), y un filtro que trabaje con sistemas no lineales (Kalman extendido).

5.2 Trabajos a futuro

Es posible realizar mejoras a la arquitectura presentada. Si bien en la práctica utilizar las técnicas heurísticas ha arrojado resultados positivos, estos todavía se muestran lejanos al nivel de precisión deseado. Algunas mejoras que se pudieran realizar en las pruebas serían el cambio de la función del valor fitness, el cambio de valor en las variables C1 y C2 y la tasa de amortiguación, estos pertenecientes al bloque de Optimización multi-onjetivo por enjambre de partículas. A pesar de que el método de selección de líderes se realiza por ruleta, podría considerarse utilizar otro método de selección, esto en el bloque de ISOCLUS.

Debido a que el sensor de *silicon retina* es independiente del tiempo, un algoritmo que funcione de manera completamente paralela es factible, además, gracias a la antelación de implementar algoritmos no lineales, se puede facilitar una transición en el procesamiento del algoritmo para convertir algunas etapas a procesamiento paralelo. Por ejemplo: se plantea, a mediano plazo, poder convertir a paralelos los bloques de análisis y selección de regiones, y selección de objetos por grupos de información.

También se propone:

- Diseñar y adjuntar el bloque de reconocimiento de objetos para, finalmente, diseñar y adjuntar el bloque de clasificación de objetos, al final del algoritmo.
- La implementación de otros algoritmos en cada bloque, esto con la finalidad de comparar los resultados en busca de mejor precisión.
- Aplicar en diferentes pruebas el algoritmo diseñado, por ejemplo en un ambiente no controlado.

Lograr descartar datos innecesarios con la finalidad de obtener una mejor forma del objeto observado puede mejorar el seguimiento del objeto para dar paso al reconocimiento de sus formas. Esto puede ser de utilidad para los sistemas de clasificación.

Referencias informativas

[1] Vernon D., (1991), *Machine Vision, Automated Visual Inspection and Robot Vision.* Dublin, Ireland, Prentice Hall.

[2] Tsitiridis A., Conde C., Sánchez J., Cabello E. (2015). Gabor feature processing in spiking neural networks from retina-inspired data. 2015 *International Joint Conference on Neural Networks (IJCNN).* **DOI:** 10.1109/IJCNN.2015.7280352

[3] Serrano R., Oster M., Lichsteiner P., Linares A., Paz R., Gómez F., … Linares B. (2005). AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems. *Advances in Neural Information Processing Systems (NIPS), 18(2015).*

[4] Eibensteiner F., Kogler J., Sulzbachner C. y Scharinger J. (2011). Stereo-Vision Algorithm Based on Bio-Inspired Silicon Retinas for Implementation in Hardware. *Computer Aided Systems Theory – EUROCAST 2011 (*624-631), Springer Berlin Heidelberg.

[5] Nope S., Loaiza H., Caicedo E. (2007). Implementación bio-inspirada de sistemas de detección del movimiento por visión artificial. *El Hombre y la Máquina,* 28, 60-67.

[6] Sanz, C. (2002). RAZONAMIENTO EVIDENCIAL DINAMICO, Un Método de Clasificación aplicado al Análisis de Imágenes Hiperespectrales. Universidad Nacional de la Plata Facultad de Ciencias Exactas, La Plata, Argentina.

[7] The Deep sea anglerfish. Recuperado de: http://www.frogfish.ch/deepsea-anglerfish.html

[8] Russell, S. J. y Norving, P., (1995), *Artificial Intelligence, A Modern Approach,* Nueva Jersey, Estados Unidos de América, Prentice Hall.

[9] Kogler J., Suzbachner C. y Kubinger W. (2009). Bio-inspired stereo vision system with silicon retina imagers. Computer Vision Systems (174-183). Lieja, Bélgica: Springer Berlin Heidelberg.

[10] Kogler J., Suzbachner C., Humemberger M. y Eibensteiner F. (2011). Address-Event based Stereo Vision with Bio-inspired Silicon Retina Imagers. *Advances in Theory and Applications of Stereo Vision (165-188)*, AIT Austrian Institute of Technology

[11] Lichtsteiner P., Posch C. y Delbruck T. (2008). A 128x128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE JOURNALS OF SOLID-STATE CIRCUITS, 43*(2), 566-576.

[12] Rahman A., Morel O. y Fofi D. (2012). Visual Behaviour Based Bio-Inspired Polarization Techniques in Computer Vision and Robotics. *Developing and Applying Biologically-Inspired Vision Systems: Interdisciplinary Concepts, Information Science Reference* (247-276), Pensilvania, Estados Unidos: IGI Publishing Hershey.

[13] Mahowald M., (1994), *An Analog VLSI System for Stereoscopic Vision*, Boston, Estados Unidos, Springer US.

[14] Sulzbachner C., Zinner C. y Kogler J. (2011). An Optimized Silicon Retina Stereo Matching Algorithm Using Time-Space Correlation. *Computer Vision and Pattern Recognition 2011 Workshops*.

[15] Vukusic P. y Barr J. (2010). Introducción al diseño bioinspirado. *Gaceta óptica: Órgano Oficial del Colegio Nacional de Ópticos-Optometristas de España*, 448, 74-76.

[16] Lin S., Yemelianov K., Pugh E. y Engheta N. (2004). Polarization Enhanced Visual Surveillance Techniques. *Proceedings of the 2004 IEEE International Conference on Networking, Sensing & Control*, 216-221.

[17] Bichler O., Qerlioz D., Thorpe S., Bourgoin J., Gamrat y C. (2011). Unsupervised Features Extraction from Asynchronous Silicon Retina through Spike-Timing-Dependent Plasticity. *The 2011 International Joint Conference on Neural Networks (IJCNN).* **DOI:** 10.1109/IJCNN.2011.6033311

[18] Bo Zhao, Qiang Yu, Hang Yu, Shoushun Chen, Huajin Tang. (2013). A bio-inspired feed forward system for categorization of AER motion events. *IEEE Biomedical Circuits and Systems Conference (BIOCAS).*

[19] Linares A., Gómez F., Jiménez A. (2007). Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. *2007 IEEE International Symposium on Circuits and Systems, 1192-1195.* **DOI:** 10.1109/ISCAS.2007.378265

[20] Conradt J., Berner R., Cook M., Delbruck T. (2009). An Embedded AER Dynamic Vision Sensor for Low-Latency Pole Balancing. *Computer Vision Workshop (ICCV Workshops).*

[21] Bichler O., Querlioz D., Thorpe S., Bourgoin J. y Gamrat C. (2012). Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks, 32*(2012), 339-348. DOI: 10.1016/j.neunet.2012.02.022

[22] Delbruck T., Lang M. (2013). Robotic Goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience, 7*(223). DOI: 10.3389/fnins.2013.00223

[23] Teixeira T., Culurciello E., Park J., Lymberopoulos D., Barton A. y Savides A. (2006). Address-event imagers for sensor networks: evaluation and modeling. *International Conference on Information Processing in Sensor Networks (IPSN '06),* USA.

[24] Hongmin Li, Jing Pei, Guoqi Li. (2015). Real-time Tracking Based on Neuromorphic Vision. *2015 15th Non-Volatile Memory Technology Symposium (NVMTS),* **DOI:** 10.1109/NVMTS.2015.7457498.

[25] Zhao B., Ding R., Chen S., Linares B., Tang H. (2014). Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network. *IEEE Transactions on Neural Networks and Learning Systems, 26(9),* 1963 - 1978.

[26] Zhao Bo."A biologically inspired human posture recognition system" tesis 20014.

[27] Gütig R. y Sompoinsky H. (2014). Tempotron Learning. *Encyclopedia of Computational Neuroscience (1-3*). Springer New York. DOI 10.1007/978-1-4614-7320-6_685-1.

[28] Senn, W. y Pfister, J.-P. (2014). Spyke-Timing Dependent Plasticity, Learning Rules. *Encyclopedia of Computational Neuroscience (1-10).*

[29] Wang S. (2003). Artificial Neural Network. *Interdisciplinary Computing in Java Programming (81-100)*, Springer US. DOI: 10.1007/978-1-4615-0377-4_5.

[30] Welch G. y Bshop G. (2001). An Introduction to the Kalman Filter. Ciudad/estado y país. University of North Carolina at Chapel Hill, Carolina del Norte, Estados Unidos (2001). Departamento de Ciencias Computacionales.

[31] Julier S. y Uhlmann J. (1997) A new Extension of the Kalman Filter to Nonlinear Systems.

[32] Blackwell, T. y Branke J. (2004). Multy-swarm Optimization in Dynamic Enviroments. *Applications of Evolutionary Computing* (489-500), Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-24653-4_50.

[33] Furey, T. S., Cristianini, N., Duffy, N., Bednarsky, D. W., Schummer, M. y Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *BIOINFORMATICS*, *16*(10), 906-914.

[34] Everhart R. y Shi Y. (2005). Comparison between Genetic Algorithms And Particle Swarm Optimization. *Evolutionary Programming VII (611-616*), Springer Berlin Heidelberg. DOI: 10.1007/BFb0040812.

[35] Tim Blackwell, Jürgen Branke. "Multi-swarm Optimization in Dynamic Enviroments".

[36] Nishida T. y SAKAMOTO T. (2011). Adaptive PSO for Online Identification of Time-Varying Systems. *IEEJ Transactions on Electronics Information and Systems*, 131(7),1642-1649. DOI: 10.1002/ecj.11391.

[37] B. Yegnanarayana. Artificial Neural Networks.

[38] Mariñas, G. (2009). Evaluación de algoritmos supervisados de extracción de características para clasificación de texturas. Universidad Carlos III de Madrid. Departamento de Teoría de la Señal y Comunicaciones.

[39] Abreu, R., Bulloces S. (2012). Implementación de un Filtro de Gabor Modificado para el Mejoramiento de la Imagen de Huellas Dactilares en un Sistema Verificador Biométrico.

[40] Smith, S., (2012). *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing.

[41] Eliminación de Gauss-Jordan, (s. f). En Wikipedia. Recuperado el 26 De Agosto de 2016 de https://es.wikipedia.org/wiki/Eliminaci%C3%B3n_de_Gauss-Jordan

[42] Muñoz, M., López, J. y Caicedo, E. (2008). Inteligencia de enjambres: sociedades para la solución de problemas (una revisión) Swarm intelligence: problema-solving societies (a review). *REVISTA INGENIERÍA E INVESTIGACIÓN*, *28*(2), 119-130.

[43] Paoli, A., Melgani, F. y Pasolli, E. (2009). Clustering of Hyperspectral Images Based on Multiobjective Particle Swarm Optimization, *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, *47*(12), 4175-4188.

[44] Xiaohiu Hu. PSO Tutorial. (2006). Recuperado de: http://www.swarmintelligence.org/tutorials.php

[45] Lalwani S., Singhal S., Kumar R. y Gupta N. (2013). A COMPREHENSIVE SURVEY: APPLICATIONS OF MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION (MOPSO) ALGORITHM. *Transaction on Combinatronics, 2*(1), 39-101.

[46] Chamaani, S., Mirtaheri, S. A., Teshnehlab, M., Shoorehdeli, M. A. y Seydi, V. (2008). MODIFIED MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION FOR ELECTROMAGNETIC ABSORBER DESIGN, *Progress In Electromagnetic Research*, 79, 353-366.

[47] Reyes-Sierra,M. y Coello Coello, C. A. (2006). Multi-Objective Particle Swarm Optimizers: A survey of the State-of-the-Art, *International Journals of Computational Intelligence Research, 2*(3), 287-308.

[48] Muñoz, M., López, J. y Caicedo, E. (2008). Inteligencia de enjambres: sociedades para la solución de problemas (una revisión) Swarm intelligence: problema-solving societies (a review). *REVISTA INGENIERÍA E INVESTIGACIÓN*, *28*(2), 119-130.

[49] Memarsadeghi, N., Mount, D. M., Netanyahu, N. N. y Le Moigne, J. (2007). A FAST IMPLEMENTATION OF THE ISODATA CLUSTERING ALGORITHM*, *International Journal of Computational Geomety & Applications. 17*(1), (71-103).

[50] Haykin, S., (2001), *Kalman Filtering and Neural Networks*, Ontairo, Canada, John Wiley & Sons, Inc.

[51] Extended Kalman Filter. Keisuke Fujii.

[52] Kiriy E. y Buehler M. (2002). Three-state Extended Kalman Filter for Mobile Robot Localization.

[53]Comaniciu D., Ramesh V. y Meer P. (2003). Kernel Based Object Traking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(5).

[54] Grira N., Crucianu M. y Boujemaa N. (2005). Unsupervised ans Semi-supervised Clustering: a Brief Survey*.

[55] Braik M., Sheta A. y Ayesh A. (2007). Image Enhacement Using Particle Swarm Optimization. *Proceedings of the World Congress on Engineering, 1*.

[56] Reyes M. y Coello C. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journals of Computational Intelligence Research, 2*(3), 287-308.

[57] Corradi F., Delbruck T., Longinotti L. y Bamford S. (2006). jAER. Recuperado de: https://sourceforge.net/projects/jaer/

[58] Optimización por enjambre de partículas, (s. f). En Wikipedia. Recuperado el 30 de Agosto de 2016 de https://es.wikipedia.org/wiki/Optimizaci%C3%B3n_por_enjambre_de_part%C3%ADculas

[59] Optimización (matemática), (s. f). En Wikipedia. Recuperado el 30 de Agosto de 2016 de: https://es.wikipedia.org/wiki/Optimizaci%C3%B3n_(matem%C3%A1tica)

[60] National Geographic Society. (1996-2015). Anglerfish Lophius Piscatorius.. Recuperado de: http://animals.nationalgeographic.com/animals/fish/anglerfish/.

[61] EcuRed Conocimiento con todos y para todos. Pez linterna (Rape). Recuperado de: https://www.ecured.cu/Pez_linterna_(Rape).

[62] Black seadevil, (s. f). En Wikipedia. Recuperado el 17 de Noviembre de 2016 de https://en.wikipedia.org/wiki/Black_seadevil

[63] NOAA Ocean Explorer (2016).Red light does not reach ocean depths, so Deep-sea animals that are red actually appear black and thus are less visible to predators and prey. National Oceanic and Atmospheric Administration, U.S. Department of Commerce. Recuperado de: http://oceanexplorer.noaa.gov/facts/red-color.html

[64] Deep sea fish, (s. f). En Wikipedia. Recuperado el 17 de Noviembre de 2016 de https://en.wikipedia.org/wiki/Deep_sea_fish

[65] yarpiz (2017). Multi-Objective Particle Swarm Optimization. Recuperado de: http://yarpiz.com/59/ypea121-mopso

[66] Mathworks (1994-2016). Functions for the rectangular assignment problem. The Mathworks Inc. Recuperado de: https://www.mathworks.com/matlabcentral/fileexchange/6543-functions-for-the-rectangular-assignment-problem?requestedDomain=www.mathworks.com

[67] McGraw-Hill Dictionary of Scientific & Technical Terms, 6E, Copyright © 2003 by The McGraw-Hill Companies, Inc.

[68] Ground Truth Evaluation for Event-Based Silicon Retina Stereo Data, Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, 23-28 June 2013. DOI: 10.1109/CVPRW.2013.98

[69] Manual de Prácticas, Complementos de Física. Universidad de Sevilla, Departamento de Física Aplicada.