Tesis doctoral

Análisis, diseño e implementación de sistemas neuromórficos basados en pulsos para el procesado de información de retinas artificiales

Francisco de Asís Gómez Rodríguez

Sevilla, abril de 2011

Departamento de Arquitectura y Tecnología de Computadores
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla

Análisis, diseño e implementación de sistemas neuromórficos basados en pulsos para el procesado de información de retinas artificiales

por

Francisco de Asís Gómez Rodríguez

PROPUESTA DE TESIS DOCTORAL
PARA LA OBTENCIÓN DEL GRADO DE
DOCTOR INGENIERO EN INFORMÁTICA
SEVILLA, ABRIL 2011

Director: **Dr. Gabriel Jiménez Moreno**

UNIVERSIDAD DE SEVILLA

Memoria presentada para optar al grado de Doctor Ingeniero en Informática por la Universidad de Sevilla

Autor: **Francisco de Asís Gómez Rodríguez**

Título: **Análisis, diseño e implementación de sistemas neuromórficos basados en pulsos para el procesado de información de retinas artificiales**

Departamento: **Departamento de Arquitectura y Tecnología de Computadores**

Vº Bº Director

Gabriel Jiménez Moreno

El autor

Francisco de Asís Gómez Rodríguez

Dedicatoria

Me gustaría dedicar este trabajo a la memoria de mi abuelo Pepe, de quien heredé la curiosidad por saber cómo funcionan las cosas; a la memoria de mi abuelo Juan, de quien aprendí el valor de la constancia para conseguir algo; a la memoria de mi abuela Fina la última en dejarnos y a la que echo mucho de menos y a mi abuelilla Fernanda, la única que aún me queda y que con su cariño hace la vida infinitamente mejor.

Y por supuesto, dedico este trabajo a mis padres Paco y Pilar, a mi hermana Ika, a Lourdes, al chiquitín Sergio y a la más pequeña, Sofía.

"Si he logrado ver más lejos, ha sido porque he subido a hombros de gigantes".

Isaac Newton

Agradecimientos

Me gustaría agradecer a los miembros del grupo de investigación Robótica y Tecnología de Computadores de la Universidad de Sevilla que me ha ayudado en este trabajo, en especial a Fernando por sus magnificas ideas; a Manuel por el DataLogger y la última versión de la aplicación MultiLoadFPGA; a todos los que contribuyeron al diseño y a la construcción de la plataforma USB-AER; a Gabriel, a Alejandro, a Antón, a Rafa y a aquellos que de una manera u otra han facilitado la realización de esta tesis.

También me gustaría dar las gracias a Tobias Delbruck, Shih Chii Liu y a su grupo de investigación.

Índice

Listado de figuras

Listado de tablas

"La posibilidad de realizar un sueño es lo que hace que la vida sea interesante."

Paulo Coelho

Capítulo 1

Introducción

La visión artificial engloba una serie de técnicas cuyo objetivo último es que una máquina "entienda" la información visual, que pueda "percibir". Los términos "entender" y "percibir" están íntimamente ligados a los conceptos de consciencia e inteligencia, es por ello que la visión por computador no deja de ser un subcampo de la inteligencia artificial. Esta relación ha hecho que desde sus comienzos los estudiosos de la visión artificial hayan utilizado la naturaleza como ejemplo, imitándola en lo posible. Sin embargo, desde el principio nos encontramos con grandes diferencias entre los sistemas naturales y artificiales, siendo una de las más significativas cómo se representa la información visual en cada uno de estos sistemas. En los sistemas artificiales tradicionales la información visual viene dada por una secuencia de fotogramas que representan un muestreo, con un periodo fijo, de la realidad; en los sistemas naturales no es fácil determinar cómo es dicha representación de la realidad, pero parece claro que no puede estar basada en fotogramas, más bien debe fluir de forma continua. Desde hace unos años la comunidad científica ha abordado mecanismos alternativos de representación visual (Tobi Delbruck et al. 2010). Aunque esto en sí mismo puede no parecer importante, lo cierto es que abre nuevas

alternativas para abordar los problemas de la visión artificial y resolverlos de forma, cuando menos, diferente; por no decir de forma más eficaz.

Las nuevas retinas artificiales con salida pulsante son un ejemplo de sistemas que pretenden imitar a la naturaleza en lo que se refiere a su funcionamiento y a la representación de la información visual. Este trabajo se apoya en estas retinas para buscar lo mismo que la visión artificial: que una máquina "entienda" qué está viendo. Evidentemente nuestra pretensión con este trabajo no es resolver el paradigma de la inteligencia y la consciencia, aquí estudiaremos y buscaremos nuevos mecanismos para hacer lo mismo que se suele hacer con imágenes tipo fotogramas pero con representación pulsante, comprobando qué posibilidades tienen estas técnicas y qué ventajas presentan.

Es importante indicar desde el primer momento que aunque los resultados que perseguimos con esta tesis son los clásicos de la visión artificial, los métodos que se utilizan en este trabajo no son una mera trasposición de los utilizados en dicha disciplina. Dos de los objetivos clásicos en la visión artificial son el reconocimiento de objetos o patrones y el seguimiento de los mismos. Es evidente que estos objetivos son compartidos por el presente trabajo, pero, a diferencia de muchos otros, aquí se va a tratar desde el primer momento con una representación pulsante de la información, la cual vamos a procesar con un hardware específico, usando nuevos métodos, sin ser transformada en fotogramas en ningún momento. Como es natural para visualizar los resultados y comprobarlos se hará uso de los fotogramas en ciertas etapas, sobre todo en las finales, pero esto es sólo a modo de comprobación.

Por tanto, en esta tesis se presenta un conjunto de celdas neuromórficas[1] para el procesamiento de información visual procedente de retinas artificiales. Se propone además una estructura u organización jerárquica o en cascada del procesamiento. Y se pretende aportar ideas, desarrollos e implementaciones que permitan dar un nuevo paso en la

[1] Los sistemas neuromórficos son aquellos que intentan imitar el funcionamiento de los sistemas neuronales biológicos, copiando de ellos su estructura interna y tratando de resolver los mismos problemas (Indiveri et al. 2009).

construcción de sistemas neuromórficos, con el fin de arrojar un poco de luz a la pregunta de si es posible construir sistemas artificiales que se asemejen en prestaciones, consumo energético y complejidad a los sistemas neuronales biológicos.

1.1. Motivaciones

Las motivaciones para la realización de este trabajo se pueden dividir en tres tipos, las científicas, las personales y las relacionadas con los proyectos en los que está o ha estado implicado el autor del presente trabajo dentro del grupo de investigación de Robótica y Tecnología de Computadores de la Universidad de Sevilla (RTC-US) . Evidentemente las motivaciones científicas son las importantes y las que en última instancia se fusionan con las otras dos, siendo en su conjunto lo mismo.

La principal motivación científica para realizar esta tesis es buscar sistemas de visión artificial cada vez más perfectos, explorando nuevos mecanismos. En este sentido desde hace unos años se dispone, a través de otros grupos de investigación internacionales socios en diversos proyectos, de retinas artificiales pulsantes. Sin embargo, el procesado de dicha información apenas ha sido desarrollado y tratado, es precisamente esta carencia la que ha llevado a la realización del presente trabajo.

En cuanto a las motivaciones personales, analizando la trayectoria profesional del autor se percibe las inquietudes por el diseño y construcción de sistemas de visión artificial. Como proyecto final de carrera presentó un sistema para la detección de obstáculos en sillas de ruedas usando cámaras CCD (en 1999), que le sirvió para trabajar como becario en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla en el grupo de investigación Visión, Robótica y Control (desde 1999 a 2003), donde estuvo involucrado en diversos proyectos (nacionales y europeos) cuyo objetivo principal era la detección y seguimiento de incendios forestales usando cámaras en el espectro visible e infrarrojo. Posteriormente se unió al grupo Robótica y Tecnología de Computadores de la

Universidad de Sevilla (en 2003) donde, hasta la fecha, realiza su labor investigadora en diversos proyectos relacionados con sistemas de visión neuromórficos.

En cuanto a las motivaciones derivadas de las necesidades de los proyectos del grupo de investigación, a continuación se muestran los proyectos en los que ha participado como investigador desde el año 2003:

- Proyecto europeo CAVIAR: Convolution AER[2] Vision Architecture for Real-Time (2002 -2006). El objetivo de este proyecto fue el desarrollo de chips y placas electrónicas para demostrar la viabilidad de un sistema de visión basado en AER, desde el sensado, pasando por el procesado, hasta los actuadores.

- Proyecto nacional VICTOR: Visión por computador en tiempo real (2001-2004). En este proyecto se diseña la conexión de un bus AER a un computador digital mediante interfaz PCI.

- Proyecto nacional SAMANTA: Sistema de visión multi-chip Address-Event-Representation para plataforma Robótica (2004-2006). El objetivo de este proyecto para el grupo RTC fue el diseño e implementación de módulos de interconexión AER con sistemas robóticos digitales.

- Proyecto nacional SAMANTA II: Sistema de visión multi-chip Address-Event-Representation para plataforma Robótica II (2006-2009) y Proyecto andaluz BRAIN SYSTEM (2007-2010). En estos proyectos se continuó la labor realizada en SAMANTA.

- Proyecto nacional VULCANO: Visión Ultra-Rápida por eventos y sin fotogramas. Aplicación a automoción y Robótica cognitiva antropomorfa (2010-2012). El objetivo de este proyecto es dar los pasos necesarios para desarrollar un conjunto de demostradores, validando las técnicas basadas en AER en el sensado y procesado de visión y actuación mecánica para aplicaciones interesantes en el sector industrial.

---

[2] Siglas en ingles de Address Event Representation. Como se verá más adelante AER es un mecanismo neuromórfico para la transmisión de información entre sistemas neuronales artificiales.

Cabe destacar que en todos estos proyectos ha colaborado el Instituto de Microelectrónica de Sevilla del Consejo Superior de Investigaciones Científicas (IMSE-CSIC), y en el proyecto europeo CAVIAR participó, entre otros, el Institute of Neuro Informatics de Zurich (INI-ETZH).

En el presente trabajo, se hace uso de dos desarrollos realizados bajo el amparo de estos proyectos, la plataforma llamada *USB-AER board Inteface*, desarrollada por el RTC-US, que será usada como plataforma hardware en las implementaciones que se aportan. También se hace uso de la retina artificial desarrolla por el INI-ETZH llamada *RetDiff128*. De ambos sistemas pueden encontrarse descripciones más adelante.

Por lo tanto podemos afirmar que la presente tesis es el fruto de años de trabajo en los sistemas de visión artificial, tanto desde el enfoque clásico basado en el uso de sensores CCD y computadores, como desde un enfoque neuromórfico con el uso de hardware específico. Además podemos concluir que si bien las necesidades de los proyectos de investigación en los que el doctorando se ha involucrado han moldeado este trabajo, la idea primigenia de contribuir al desarrollo de sistemas de visión artificial constituye el núcleo del mismo.

1.2. Objetivos

Como se ha indicado anteriormente, el objetivo principal de la tesis es el abordar nuevos sistemas de procesado visual basados en la representación pulsante de las imágenes. Se pretende estudiar diferentes posibilidades sin perder la perspectiva última que nos ofrece la naturaleza en los sistemas neuronales biológicos. También es objetivo de esta tesis implementar y probar en la realidad los nuevos sistemas que se proponen. Para realizar las pruebas pertinentes y que además los nuevos métodos que perseguimos sean aprovechables en el futuro desde un punto de vista industrial, éstos deben poderse implementar de forma sencilla con los elementos que disponemos actualmente en el

mercado. Es por este motivo que se ha optado por utilizar una plataforma hardware basada en FPGA[3]. Por otra parte el procesado visual puede ser muy amplio, es por ello que este trabajo se centra en dos paradigmas clásicos en la visión artificial, el seguimiento y localización de objetos y el reconocimiento de patrones, pero siempre con un procesado pulsante directo. En síntesis, podemos considerar que el objetivo principal de este trabajo es la obtención de nuevos mecanismos de procesado de la visión basado en pulsos para el seguimiento de objetos y reconocimiento de patrones, y la implementación de los mismos con sus correspondientes pruebas.

Conviene definir de manera clara y concisa los objetivos que persigue esta tesis. Se agrupan en generales y específicos.

Objetivos generales

Indudablemente esta tesis se puede enmarcar en el amplio campo científico de los sistemas de Inteligencia Artificial, y dentro de éste en el de los sistemas neuromórficos. Los objetivos generales, por tanto, están encaminados a aportar un poco de luz sobre las posibilidades de los sistemas neuromórficos, y la viabilidad de estos para formar parte de sistemas de más alto nivel en la estructura cognitiva; y son:

1. Explorar la posibilidad de implementación de sistemas de visión artificial neuromórfico
2. Aportar luz a la pregunta de si es posible procesar información que viene codificada en pulsos, ya sea en frecuencia de pulsos o en pulsos temporales precisos.
3. Estudiar la viabilidad de uso de sistemas digitales en la construcción de sistemas neuromórficos, campo patrimonializado por los sistemas basados en electrónica analógica, en los últimos tiempos.
4. Y en un sentido amplio aportar pistas para contestar preguntas sobre las ventajas del uso de eventos en la representación de imágenes; sobre el papel que juegan los

[3] Siglas en ingles de Field-Programmable Gate Array, en español Matriz de puertas programables.

eventos en el transporte de la información en la computación; sobre la conveniencia de la codificación en frecuencia de pulsos o en pulsos temporalmente preciso; y sobre la complejidad, adaptabilidad y aprendizaje en redes de neuronas pulsantes.

Objetivos específicos

Dado lo amplio e incluso ambicioso de los objetivos generales, conviene fijar unos objetivos más concretos y tangibles con los que acercarse a los primeros:

1. Definir una arquitectura para el procesado de información retino-mórfica, usando una retina artificial concreta, en este caso la desarrollada por el INI y llamada RetDiff128.
2. Usar el paradigma neuromórfico, es decir, imitar a la Naturaleza en todo lo posible, incluyendo aspectos como el funcionamiento interno y externo; resolviendo los mismos problemas y de la misma forma.
3. Seguir de manera simultánea varios objetos en movimiento ofreciendo las posiciones relativas de los mismos en el plano de "imagen" del sensor.
4. Obtener parámetros del movimiento de los objetos, tales como velocidad, aceleración etc.
5. Detectar patrones sencillos en la forma de los objetos, tales como orientaciones o formas simple, imitando el funcionamiento y la labor de ciertas neuronas de la corteza visual (detalles sobre el sistema visual pueden ser encontrados más adelante).
6. Diseñar circuitos electrónicos que implementen las funcionalidades antes descritas.
7. Realizar una implementación hardware de los circuitos diseñados con los siguientes requisitos:
    a. La implementación no debe incluir ningún computador convencional en el núcleo del procesado.
    b. La implementación debe ser realizable y realista, modular y que permita demostrar empíricamente la viabilidad de la construcción de sistemas neuromórficos.

c. La implementación debe poder interactuar con la retina artificial, permitiendo la construcción de un sistema de sensado y procesado completo.

1.3. Estructura de la tesis

Esta tesis se articula en 8 capítulos:

Capítulo 1: el presente, dedicado a introducir las motivaciones, los objetivos y la estructura.

Capítulo 2: dedicado a repasar algunos de los fundamentos de los sistemas neuromórficos, a estudiar el sistema visual, y por último a realizar un repaso del estado de los desarrollos neuromórficos actuales basados en AER.

Capítulo 3: detalla las celdas propuestas para el seguimiento de objetos, obteniendo la posición de objetos en movimiento y estimando la velocidad de los mismos.

Capítulo 4: presenta la celda que realiza la detección de patrones visuales en los objetos.

Capítulo 5: contiene una visión general del sistema propuesto, sus diferentes alternativas y las aplicaciones de cada una de ellas.

Capítulo 6: describe la implementación hardware de los desarrollos expuestos en los capítulos anteriores, haciendo uso de la plataforma USB-AER.

Capítulo 7: dedicado a las pruebas y experimentos realizados sobre la implementación hardware del sistema.

Y por último Capítulo 8: relata las aportaciones y las conclusiones a las que se pueden llegar a la vista de este trabajo, así como los trabajos futuros.

"Cada día sabemos más y entendemos menos"

Albert Einstein

Capítulo 2

Los sistemas Neuromórficos

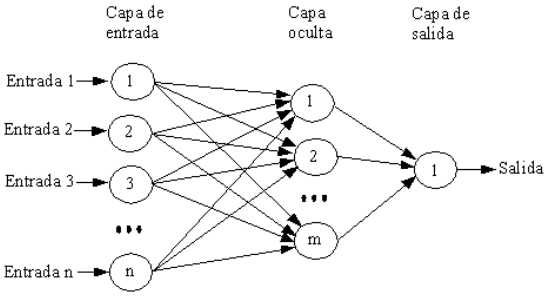
Son muchos en la comunidad científica los que opinan que la Naturaleza ha conseguido a lo largo del tiempo, por medio de la evolución natural, desempeñar tareas complejas con "desarrollos" casi perfectos. Observando como la Naturaleza realiza estas tareas se han conseguido grandes avances en campos muy diversos, algunos ejemplos son: Los trajes de los nadadores olímpicos, inspirados en la piel de tiburón para minimizar la fricción con el agua (Figura 2.1.a). La forma de las alas de los aviones inspirada en las de los pájaros (Figura 2.1.b). La computación basada en redes neuronales artificiales que se inspira en los sistemas neuronales biológicos, tomando de estos la división de la computación en celdas (o neuronas) y la organización de las mismas, desde el punto de vista de la conectividad entre ellas (Figura 2.1.c). Robots humanoides que imitan forma y el comportamiento de seres humanos (Yosehp & Cynthia 2003) (Figura 2.1.b).

Es importante en este punto, reflexionar que se entiende por "desarrollo" natural perfecto. No hay que olvidar que el fin de la Naturaleza es adaptar al individuo al entorno, de manera que le permita sobrevivir en él y perpetuar la especie. Por lo tanto, la Naturaleza debe ser un referente, pero no debe ser imitada en todo para conseguir realizar tareas complejas; este es el caso de la aviación, en que los hombres empezaron a volar cuando

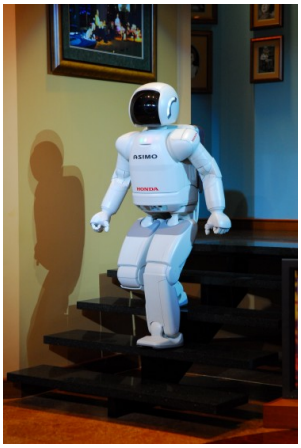
dejaron de imitar la forma de volar de los pájaros, aunque sí tomaron de estos últimos algunos "desarrollos" como son la forma de las alas y la configuración de las mismas dependiendo de la fase del vuelo, por ejemplo.

a)

b)



c)



d)

Figura 2.1: Desarrollos bio-inspirados: a) trajes de los nadadores, b) alas de los aviones, c) sistemas de computación basados en redes de neuronas artificiales, d) robot humanoides.

En las próximas páginas se presentan desarrollos que tienen su inspiración en la biología, de la que toman aquellos aspectos que son útiles para el fin que persiguen, y que modifican otros, debido a limitaciones de la tecnología o simplemente porque no son

adecuados en sistemas artificiales. Por lo tanto, teniendo en cuenta que no es necesario imitar totalmente a la Naturaleza para realizar tareas complejas, este trabajo ofrece algunas aportaciones neuromórficas para el procesado de información visual, que los seres vivos superiores realizan a diario de manera intuitiva.

En este capítulo, en primer lugar, veremos cómo se organizan los sistemas neuronales biológicos, con especial atención a la corteza visual; posteriormente se analizará el modelo de arquitectura de redes neuronales artificiales y los principales modelos de neuronas artificiales. Para terminar nos centraremos en el bus AER y los desarrollos basados en él.

2.1. Ingeniería neuromórfica

El primero en acuñar el término de Ingeniería Neuromórfica[4] fue Caver Mead (Mead 1990) y puede ser definida como el campo de la ingeniería dedicado al diseño y a la fabricación de sistemas neuronales artificiales cuya arquitectura y principio de diseño están basados en el sistema nervioso biológico. A diferencia de los Sistema Neuronales Artificiales, la ingeniería Neuromórfica intenta imitar las funciones neuronales biológicas, tales como el reconocimiento del habla, de objetos o procesos cognitivos (Indiveri et al. 2009). Uno de los desarrollos de la Ingeniería Neuromórfica en el que se basa este trabajo, y que ya se ha mencionado antes, es el bus neuromórfico AER.

En el campo de la Ingeniería Neuromórfica cabe destacar el *Institute of Neuromorphic Engineering* (INE 2011), responsable de la organización de uno de los eventos anuales de esta disciplina más importantes el *Telluride Neuromorphic Cognition Engineering Workshop* (TellurideWorkshop 2011), que se celebra en Telluride en el estado de Colorado (Estados Unidos) desde mediados de los 90; y que tiene un reflejo en Europa en el *CappoCaccia*

[4] No ha de confundirse con la Neuro-ingeniería, orientada al diseño y construcción de sistemas artificiales capaces de interactuar con los sistemas biológicos; puede considerarse una disciplina de la Neurociencia ya que sus desarrollos son muy útiles para ayudar a entender el funcionamiento de los sistemas biológicos.

Neuromorphic Cognition Engineering Workshop (CappoCacciaWorkshop 2011). En ambos eventos se dan cita los investigadores más importantes en la Ingeniería Neuromórfica, así como una gran cantidad de estudiantes que inician sus pasos en dicho campo científico-técnico.

Se entiende, por lo tanto, como sistemas neuromórficos aquellos sistemas desarrollados con las premisas expuestas más arriba: sistemas que imitan el funcionamiento de los sistemas neuronales biológicos, tanto en estructura como en su funcionalidad concreta. Con este trabajo se pretende dar un pequeño paso más en el diseño e implementación de sistemas neuromórficos, con el objetivo final de comprobar si los sistemas guiados por eventos y concretamente la representación de eventos por dirección (AER), de los que hablaremos en profundidad más adelante, son útiles para construir sistemas de procesado de información, en este caso información visual.

2.2. Organización neuronal en la biología

Continuamente, nuestro cerebro procesa la información, que gracias a nuestros sentidos, recibimos del entorno. El procesamiento de esta información nos permite realizar labores muy diversas, desde reconocer comida, detectar un peligro, interactuar con otras personas y un largo etcétera. Todos ellos con una aparente facilidad y simplicidad; aunque la estructura y el funcionamiento del cerebro es tan complejo que hoy en día sólo se tiene conocimiento a muy alto nivel del mismo.

El cerebro humano está compuesto por una cantidad gigantesca de células llamadas neuronas[5]: entre 10 y 100 billones, con una gran variedad de tipos; el tipo de neurona determina el patrón de conectividad con otras neuronas, la respuesta ante estímulos y su tarea dentro del conjunto. En la Figura 2.2, se muestran algunos dibujos realizados por

[5] Santiago Ramón y Cajal fue premiado con el Nobel de medicina en 1906, por sus investigaciones sobre la anatomía y fisiología del cerebro humano.

Santiago Ramón y Cajal (cuya foto aparece en el centro de la figura) hace más de un siglo, donde se pone de manifiesto la gran variedad de tipos de neuronas y la gran riqueza de tipos de patrones de conexiones entre ellas.

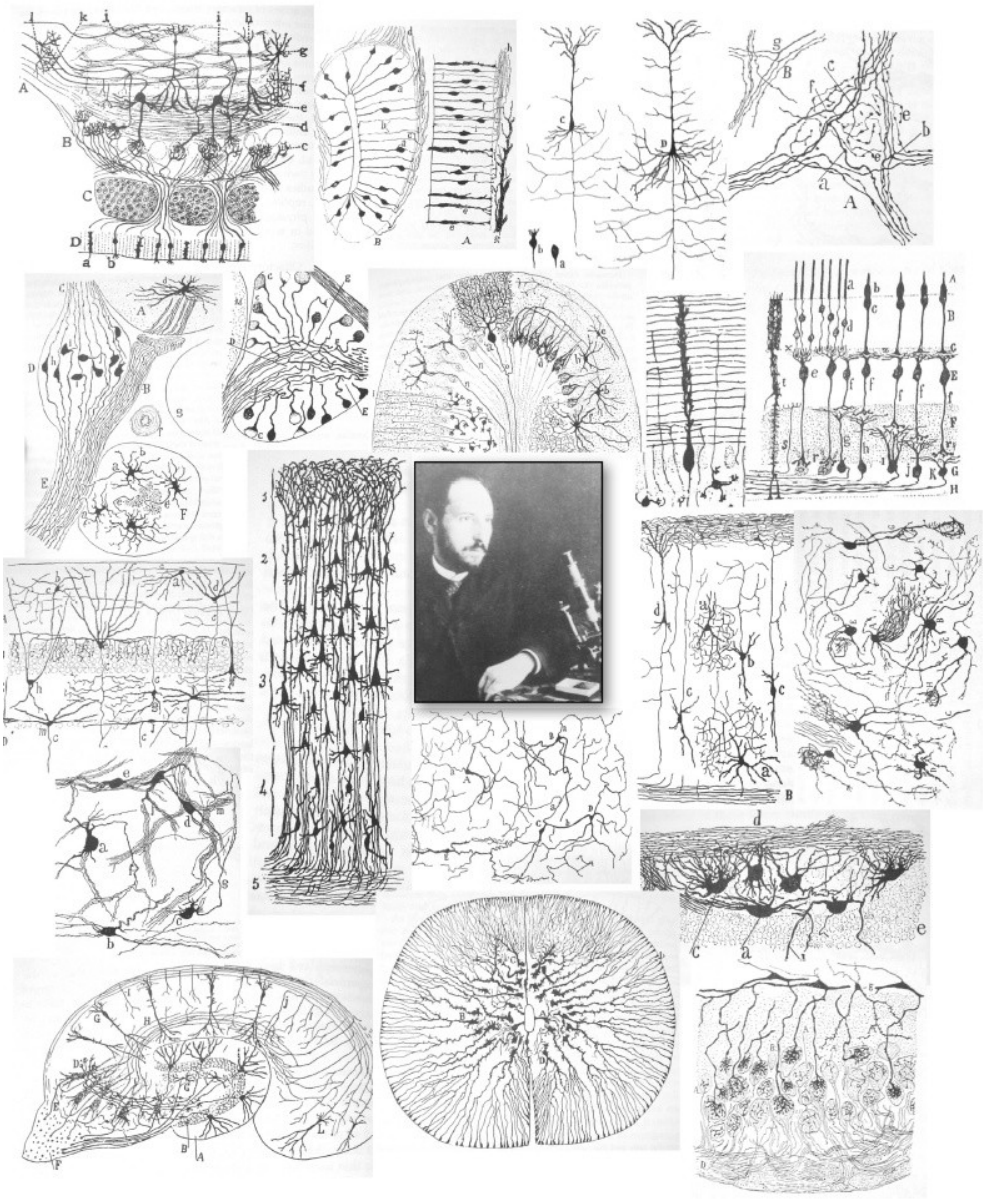


Figura 2.2: Neuronas y circuitos neuronales dibujados por Santiago Ramón y Cajal. (Figura tomada de (Potter 2007))

Lo realmente grandioso del cerebro humano es que esa inmensa cantidad de neuronas no necesita ser programada ni que sus partes dañadas sean sustituidas: es auto-organizado y sólo consume entre 10-40 vatios de energía.

En la Figura 2.3 se muestra las partes de una neurona: el cuerpo, también conocido soma, donde se encuentra el núcleo de la célula; las dendritas por donde recibe los pulsos de otras neuronas y constituye la pre-sinapsis; y el axón, que presenta varias bifurcaciones, y es por donde envía los pulsos generados, constituyendo la post-sinapsis.

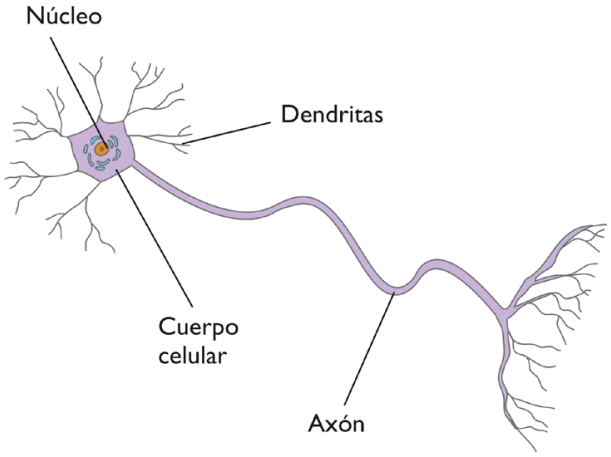


Figura 2.3: Partes de una neurona

A través de estas conexiones las neuronas se comunican unas con otras usando pulsos. El mecanismo de generación de pulsos ha sido estudiado desde hace tiempo y parece estar claro que los pulsos que llegan a una neurona van incrementado su potencial, hasta que éste llega a un punto en el que se supera un determinado umbral. Esta circunstancia provoca que se genere un pulso[6] en el cuerpo (soma) de la neurona que viaja por el axón, donde empieza a bifurcarse y donde puede ser amplificado o incluso duplicado por cada uno de los brazos de las bifurcaciones del axón.

Los pulsos pasan de una neurona a otra por la parte más compleja de la misma: la sinapsis, que está formada por la parte final del axón, el espacio inter-neuronal y la primera

[6] Más detalles pueden encontrarse en (Kandel et al. 2000).

parte de las dendritas de las siguientes neuronas. Al principio se pensaba que la sinapsis sólo servía para transmitir los pulsos entre neuronas, pero hoy se sabe que tiene un papel fundamental como pre-procesador de la información y en el aprendizaje. Cuando el pulso llega al final del axón (pre-sinapsis), la membrana celular reacciona liberando neurotransmisores en el líquido extra-celular que envuelve a las neuronas. Los neurotransmisores deben alcanzar los respectivos receptores en la post-sinapsis (dendritas de las neuronas receptoras). La post-sinapsis puede ser positiva, llamada excitante, o negativa, inhibidora, iniciandose de nuevo el proceso de activación de la neurona siguiente. Cada neurona puede recibir pulsos de otras 10000, cuando la suma de estos pulsos recibidos supera un umbral se produce un nuevo pulso. Después de la generación de un pulso la neurona entra en un estado de relajación, durante el cual no puede emitir otro pulso.

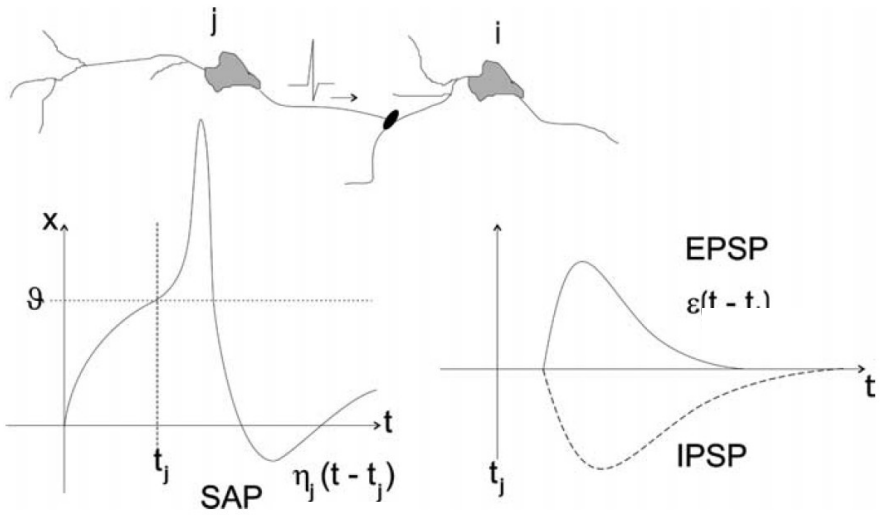


Figura 2.4: Generación y transmisión de pulsos

En la Figura 2.4 se muestra como la neurona *j* genera un pulso cuando el potencial x supera el umbral ϑ en el momento t_j. Después de la generación del pulso la neurona *j* es reiniciada durante un tiempo caracterizado por un potencial llamado SAP[7]. El pulso

[7] Del inglés *spike-after pontencial*, potencial después del pulso

generado tiene una respuesta en la neurona post-sináptica *i*, esta respuesta puede ser excitante (EPSP[8]) o inhibidora (IPSP[9]).

La mayoría de los neurocientíficos están de acuerdo en que a nivel local el funcionamiento de una neurona individual es bien conocido, pero aún se está lejos de conocer como una red formada por neuronas pulsantes procesa la información, en otras palabras aún es un misterio el "código neuronal". Aunque, gracias al desarrollo de nuevos métodos en neurociencia, se ha podido comprobar que las neuronas no se comunican usando solamente frecuencia de pulsos, sino que parte de la información es transmitida usando precisos tiempos entre pulsos consecutivos, es decir, que la información es codificada en el tiempo entre pulsos y no en la cantidad de pulsos recibidos en un determinado tiempo. Este punto es aún, hoy en día, motivo de debate y controversia (W Gerstner et al. 1997)(Laughlin & Sejnowski 2003)(Bohte 2004).

Aunque no se sabe con certeza cuál es el mecanismo exacto por el cual los sistemas neuronales biológicos son capaces de obtener una respuesta a estímulos externos, es bastante aceptado que las interrelaciones entre las neuronas (las sinapsis) se hacen fuertes o se debilitan adaptando la respuesta a los estímulos en función de si el resultado que obtienen es o no útil para el individuo; y es este el punto más oscuro, esta función de utilidad de la cual aún no tenemos más que algunas intuiciones. Se cree que en este proceso de fortalecimiento y debilitamiento las sinapsis provocan a su vez la inhibición de ciertas neuronas en favor de otras, propiciando así que la respuesta se ajuste lo más posible a las necesidades.

La distribución en capas de las neuronas es también comúnmente aceptada. Esta distribución en capas implica que la información proveniente de los sentidos, o de otras capas, es distribuida por todas las neuronas de la siguiente capa, las cuales obtienen una respuesta que será, de nuevo, distribuida entre las neuronas de la siguiente capa. Se piensa

---

[8] Del inglés *Excitatory Post-Synaptic Potencial*, potencial post-sináptico excitador.

[9] Del inglés *Inhibitory Post-Synaptic Potencial*, potencial post-sináptico inhibidor.

que esta distribución de la información también está sujeta a las normas de fortalecimiento y debilitamiento de las sinapsis.

Como evidencia de esta distribución en capas de las neuronas de los sistemas biológicos, la Figura 2.5, tomada de (D. Hubel 1995), muestra una microfotografía de la corteza visual[10] (de la que hablaremos con más detalle más adelante), donde se puede observar la distribución de las neuronas en capas; cada punto corresponde a una neurona; las capas 2 y 3 son prácticamente indistinguibles y la capa 4A es tan delgada que casi es inapreciable. Se piensa que esta distribución en capas es debida en parte a la densidad, al tipo y la colaboración entre las neuronas que forman cada capa.

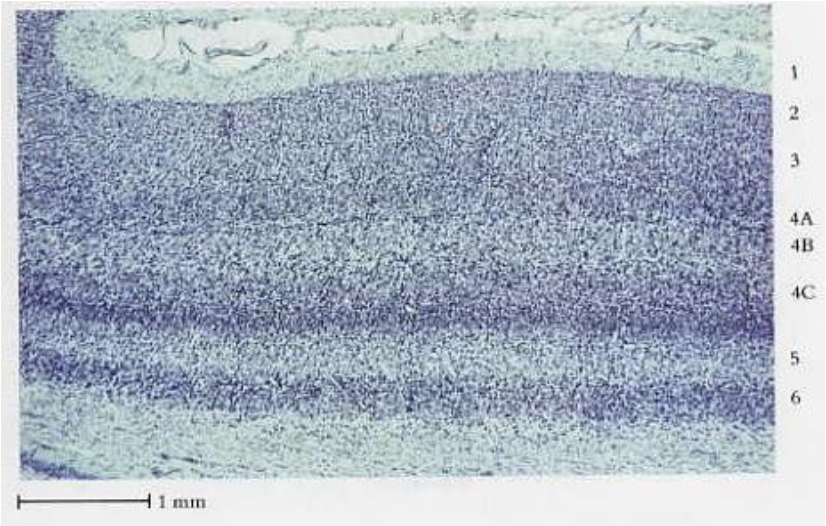


Figura 2.5: Capas de la corteza visual

2.3. El sistema visual

La visión es el sentido más importante en los seres humanos y eso se refleja en la complejidad del sistema de visión y la gran cantidad del área de corteza cerebral que

[10] La corteza visual está situado en el lóbulo occipital, en la parte posterior del cerebro, y se encarga del procesamiento de las señales procedentes de la retina (Cesar Urtubia-Icario 1996).

ocupan las regiones encargadas de esta labor. Se cree que son al menos 32 regiones anatómicamente diferenciables las encargadas de procesar la información visual, 25 de ellas están solamente dedicadas a labores relacionadas directamente con la visión y el resto realizan labores multisensoriales o de control visual del sistema motor.

Si nos movemos desde la retina hasta el cuerpo geniculado lateral (CGL) [11] y desde ahí a las sucesivas capas en la corteza cerebral, nos encontramos neuronas que responden a estímulos cada vez más complejos. Así pues, en la capa V1, también llamada corteza visual primaria, hay neuronas que responden a simples líneas con diferentes orientaciones; mientras que las áreas visuales más altas (como por ejemplo en la corteza temporal inferior) hay neuronas que responden antes rostros.

El sistema visual no está organizado en una ruta secuencial y jerárquica. Sino que aspectos como la forma, el color o el movimiento son analizados de forma separada en rutas paralelas. Estas rutas son normalmente denominadas rutas "qué" y "dónde"(Mishkin et al. 1983) (Ungerleider & Haxby 1994). La ruta "qué" está relacionada con las características del estimulo e identidad de un objeto, y puede ser subdividida en dos rutas: color y forma. La ruta "dónde" está relacionada con el movimiento y el cambio en la forma debido al mismo.

La retina

Desde la retina misma pueden observarse esta división en dos o más rutas. En la retina hay muchos tipos de células ganglionares aunque dos de esos tipos las P y la M[12] constituyen el 90% del total. Las células tipo M (el 10% del total) son sensibles a la longitud de onda y a las altas frecuencias espaciales, y presentan una respuesta lenta; son el inicio del *sistema parvocelular*, ruta relacionada con el detalle y el color. En cambio, las células tipo P (el 80% del total) no son sensibles a la longitud de onda pero sí a las bajas frecuencias espaciales, y presentan una respuesta rápida; son el inicio del *sistema magnocelular*, ruta relacionada con el bosquejo de la imagen y el movimiento.

---

[11] en inglés LGN: *lateral genicule nucleus.*

[12] P deriva de Parvus: pequeño y M de Magnus: grande.

El cuerpo geniculado lateral

Los axones de todas las células ganglionares se unen para formar el nervio óptico, que sale del ojo a través del disco óptico[13] y se proyecta en el cuerpo geniculado lateral dorsal (CGL) (ver Figura 2.6). Los nervios procedentes de ambos ojos se unen antes de alcanzar el CGL para formar el quiasma óptico. En este punto los axones procedentes de la parte interna de las retinas se entrecruzan y continúan hasta el CGL, de manera que cada hemisferio recibe información de ambas retinas. Esto constituye la base de la visión binocular.

El CGL es la estación de relevo principal entre la retina y la corteza estriada; y es el núcleo visual primario más grande y probablemente más importante en el ser humano. Las neuronas del CGL darán lugar a los axones que formarán las radiaciones ópticas. Está formado por un conjunto de neuronas distribuidas en 6 capas, de manera que cada capa recibe información de una sola retina. Las capas 2, 3 y 5 reciben información de la retina situada en el mismo lado que el CGL, mientras que las capas 1,4 y 6 de la retina opuesta. En el CGL se mantiene la misma estructura que en la retina determinan las células ganglionares, de manera que en el CGL hay un mapa completo de la retina. En las capas 1 y 2 se proyectarán las células ganglionares tipo M y en el resto (3, 4, 5 y 6) las de tipo P. Las células del CGL presentan las mismas características de respuesta que de las que reciben la entrada.

[13] El disco óptico mide 1.5x2mm y da lugar al punto ciego del ojo ya que en ese lugar no hay fotoreceptores.

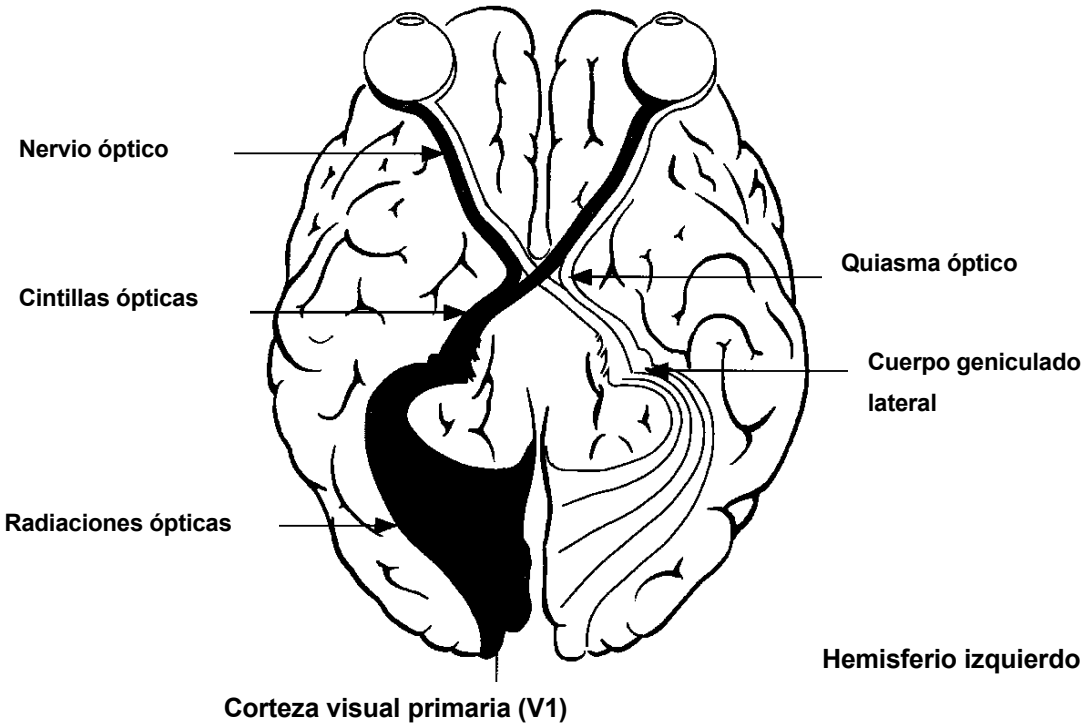


Figura 2.6: Conexión desde las retinas hasta los hemisferios cerebrales (traducido de (Tovée 2008))

La corteza visual

Las neuronas del Cuerpo Geniculado Lateral se proyectan en la corteza visual. La corteza visual es el área cerebral encargada exclusivamente del procesamiento de la información visual, está compuesta por 5 área principales, llamadas V1 (también conocida como Corteza Visual Primaria), V2, V3, V4, V5/MT. Estas áreas están interconectadas entre si con un alto grado de regularidad y precisión.

Las neuronas de la capa V1 (de cuya funcionalidad hablaremos más adelante), mantiene las dos rutas iniciadas en la retina (la del "que" y la del "donde") y esas dos rutas continúan avanzando por la corteza visual, de manera que la ruta "que" involucra las áreas V1→V2→V3→V4 y de ahí al lóbulo parietal; y la ruta "donde", las áreas V1→V2→V5 y

de ahí al lóbulo temporal. Esta separación en dos rutas no es total, si no que existen numerosas interconexiones entre ambas, ya incluso en la capa V1. Una característica común de todas las neuronas de las distintas capas de la corteza visual es que sólo responde a estímulos procedentes de pequeñas áreas del campo visual lo que se ha llamado campo receptivo (en inglés *recptive field-RF*) de la neurona. La separación entre dos campos receptivos contiguos es de menos de 1º, lo que implica un alto grado de solapamiento (DeAngelis et al. 1999). Este campo receptivo aumenta a medida que nos adentramos en la corteza visual, sugiriendo que en las capas más profundas de la corteza visual se dispone de información de un área cada vez mayor del campo de visión.

2.3.1. La corteza visual primaria V1

La corteza visual primaria es donde la, parcialmente procesada, información procedente de la retina y del CGL es separada y empaquetada para un posterior y más refinado análisis en otras capas de la corteza visual. Pero en la capa V1 se realizan algunos análisis fundamentales para la precepción visual, así pues, en la capa V1 hay neuronas sensibles a determinadas orientaciones de los estímulos, al movimiento o incluso al color. Estas sensibilidades a estas características del estimulo visual constituyen un procesamiento del mismo cuyo resultado es enviado a las siguientes áreas cerebrales encargadas de la visión y la percepción. El descubrimiento de estas respuestas de las neuronas de la corteza visual primaria les valió el premio Nobel de Medicina en 1981 a David H. Hubel y a Torsten Wiesel, que llevaron a cabo sus investigaciones entre los años 50 y 70.

Orientación

La respuesta a la orientación del estímulo fue la primera que identificaron Hubel y Weisel en el área V1 (D. H. Hubel & Wiesel 1962). Hay tres tipos de neuronas sensibles a la orientación: las simples, las complejas y las hipercomplejas. Las simples responden a líneas o bordes con una determinada orientación (durante el proceso de maduración de la neurona se hacen sensibles a una u otra orientación). La respuesta de estas neuronas es mayor cuanto más centrado esté el estimulo con respecto a su campo receptivo. La selectividad de estas neuronas varía pero en general decae cuando la orientación cambia

más de 10º respecto a la orientación "preferida". Las celdas complejas responden ante una determinada orientación y en movimiento. Por último, las neuronas hipercomplejas responden más intensamente a una determinada orientación, en movimiento, pero además "prefieren" discontinuidades en el estímulo, como cortes, ángulo o esquinas.

Frecuencia espacial

Las neuronas del CGL son sensibles y se sintonizan con una determinada frecuencia espacial, mientras que las neuronas de la V1 presentan características de filtro de paso de banda (De Valois et al. 1982). De Valois propuso que la corteza visual primaria se comporta como un filtro en frecuencia de 2 dimensiones, donde las neuronas son sensibles a la orientación del estimulo y a la frecuencia espacial simultáneamente.

Textura

En 1992 se descubrió que en la corteza visual primaria hay neuronas que no responden a líneas, a barra o ángulos; pero que sí lo hacen a una determinada señal senoidal o cuadrada, con una determinada frecuencia espacial y orientación. Estas células parece que no realizan un análisis frecuencia, si no que más bien son sensibles a la "textura" del patrón. Los objetos naturales presentan una textura rugosa, estas neuronas son sensibles a la textura y a la orientación de la textura, es decir, no solo son capaces de decir la presencia o no de una determinada superficie, sino también su orientación (Von der Heydt et al. 1992).

Dirección del movimiento y velocidad

Entre un 10% y 20% de las neuronas complejas de las capas superiores de la corteza visual son fuertemente sensibles a la dirección en la que el estímulo se mueve. Un movimiento en una determinada dirección provoca una fuerte respuesta en estas neuronas, pero permanecen inactivas para movimientos del estímulo en otras direcciones. El resto de neuronas complejas no presentan una respuesta marcada ante ninguna dirección de movimiento.

La interconexión entre las neuronas para conseguir esto se muestra en la Figura 2.7. Las neuronas complejas reciben pulsos de dos neuronas simples, siendo éstos excitadores

los provenientes de la primera e inhibidores de una segunda. La segunda neurona simple tiene un campo receptivo tal que es inmediatamente próximo, en una determinada dirección, al de la primera neurona simple. Si el estímulo se mueve en la dirección nula, los efectos excitadores e inhibidores se anulan y la neurona compleja no generará pulsos. En caso contrario, si el estímulo se mueve en la otra dirección, el pulso inhibidor llegará demasiado tarde como para detener la generación de pulsos en la neurona. Para que esto funcione es necesaria una neurona intermedia que añada un cierto retraso.

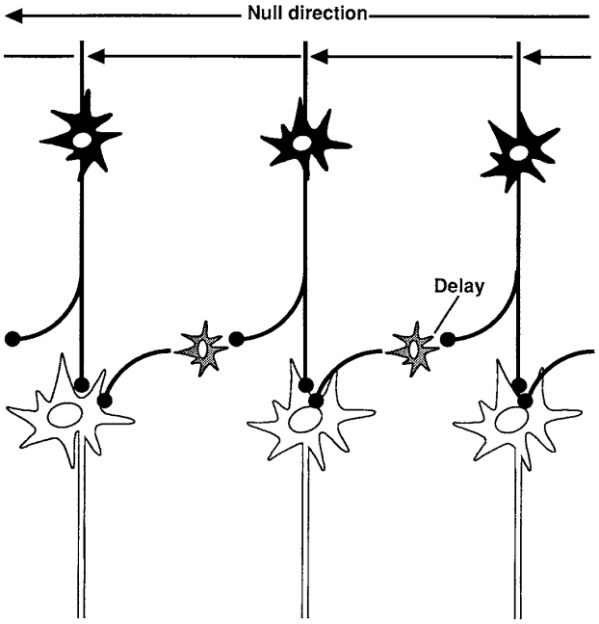


Figura 2.7: Esquema neuronal para la detección del movimiento (figura tomada de(Tovée 2008)).

Si se observa una rejilla moviéndose en una dirección durante varios minutos, el umbral para la detección del movimiento en esa dirección aumenta al doble. Esto es un ejemplo de selectividad adaptativa, producida por la fatiga de las neuronas corticales en esa dirección. Esto constituye una explicación a la ilusión de efecto de cascada que se produce cuando se observa un objeto moviéndose en una dirección durante un largo tiempo, después de ese tiempo si se mira un objeto inmóvil parece que este se mueve en la dirección contraria (Hammond et al. 1985).

Color

El procesamiento del color comienza en la retina donde los conos, células fotoreceptoras sensibles al color, son sensibles a largas, medias y cortas frecuencias de ondas, que corresponden con el color rojo, verde y azul. Las respuestas de los conos son agrupadas en dos combinaciones, roja-verde y azul-amarillo. En la corteza visual primaria (V1) existen tantas neuronas sensibles al color como a la orientación; y también sensibles a ambos.

La cantidad de neuronas sensibles al color exclusivamente varía dependiendo de lo alejado que se encuentra el campo receptivo del centro del campo visual, sin embargo, la cantidad de neuronas sensibles tanto al color como a la orientación es similar en todas las regiones. En general, el 47% de las neuronas que reciben información de la fóvea[14] son sensibles al color, frente al 23% de las que están fuera de la fóvea (Ng et al. 2007).

Aprendizaje (plasticidad)

Trabajos recientes (Bredfeldt & Ringach 2002) han demostrado que se producen cambios dinámicos en los campos receptivos de las neuronas de la V1. Se ha investigado el efecto de estos cambios (Dragoi et al. 2003) y se ha comprobado que las neuronas de la V1 modifican sus respuestas tras cierto tiempo de exposición a un estimulo, de manera que hay neuronas que comienzan a presentar respuestas más intensas ante estímulos que no presentan la orientación "preferida". De igual manera se ha comprobado que si el estímulo presenta la orientación "preferida" no hay cambio en la respuesta.

2.4. Redes de neuronas artificiales

Imitar en un sistema artificial el complejo entramado de conexiones neuronales, donde cada neurona se conecta con otras 10000, que procesan la información de manera continua, con un consumo energético ridículo en comparación con los millones de

[14] La fóvea es el área de la retina donde se concentran los rayos luminosos y permite la visión aguada y detallada.

neuronas existentes, que se auto-organiza y reconecta a lo largo de tiempo; es una labor cuando menos casi-imposible. Aún realizando simplificaciones que apenas reflejan una mínima parte de la riqueza de la biológica, sigue siendo complejo. Aunque a lo largo de la historia han surgido grandes desarrollos que nos permiten acercarnos de manera mínima a los sistemas neuronales biológicos.

2.4.1. Modelos de neuronas artificiales

Así pues, sabemos que las neuronas biológicas envían la información mediante pequeños pulsos eléctricos. Este funcionamiento básico ha sido modelado matemáticamente para su uso en un computador, lo que se ha venido en llamar Redes de Neuronas Artificiales.

Veamos cómo han evolucionado históricamente los modelos de neuronas artificiales. La evolución de las neuronas artificiales es pareja a la de la computación. Las primeras ideas son de hace más de 50 años, cuando McCulloch y Pitts (Wolfgang Maass 1997) desarrollaron la primera red de neuronas artificiales formada con neuronas cuyo funcionamiento conceptualmente era muy simple: una neurona envía un valor binario "alto" si la suma ponderada de las señales de entrada supera un determinado umbral, por tanto, esta neurona solo tiene un valor digital a la salida. Este tipo de neuronas fueron muy usadas, en redes como las redes de perceptrón multicapa o las redes de Hopfield; ya que cualquier función booleana puede ser computada con una red perceptrón multicapa con solamente una capa oculta.

En la segunda generación, las neuronas artificiales no usan una función de umbral para computar sus señales, sino que usan una función de activación continua que permite un rango analógico de señales tanto en la entrada como en la salida. Las funciones de activación más usadas son las sigmoidea o la tangente hiperbólica. Los típicos ejemplos de redes que usan este tipo de neuronas son las "feed-forward" y las redes neuronales recurrentes. Estas neuronas son más potentes que sus predecesoras, ya que son capaces de realizar la misma función pero con bastante menos neuronas(W. Maass et al. 1991). Además son capaces de aproximar mejor cualquier función analógica.

Por otra parte, las neuronas del primer modelo no utilizan pulsos y sus salidas varían entre 0 y 1. Si bien las señales de salida pueden ser expresadas en frecuencia de pulsos durante un periodo de tiempo, donde una alta frecuencia de pulsos corresponde con un nivel alto de salida, este tipo de codificación en frecuencia de pulsos implica un mecanismo de integración en el tiempo, ya que, obviamente, cada pulso es binario: o hay pulso o no lo hay, no hay valores intermedios. Las neuronas biológicas usan frecuencia de pulsos pero con frecuencias de pulsos intermedias; las neuronas con funciones de activación continua pueden modelar esto, y por lo tanto este segundo modelo de neuronas artificiales es más realista y, si se pudiera decir así, más bio-inspirado.

Cambiando los pesos de las entradas de una neurona se altera el flujo de información que fluye por la red. De manera que si las señales de entrada son alteradas, la salida también se verá alterada en el mismo sentido, este es el funcionamiento básico del aprendizaje en este tipo de redes, también conocido como plasticidad. Usando funciones de activación continua es posible usar algoritmos de aprendizaje como el de "error-backpropagation" (en español realimentación del error); muy usado en redes neuronales artificiales para conseguir una respuesta ante un conjunto de entrada.

La tercera generación de redes de neuronas se acerca un poco más a las neuronas biológicas usando pulsos individuales, lo que permite incorporar información temporal en la comunicación y en la computación, como en las neuronas biológicas. En vez de usar frecuencia de pulsos, estas neuronas usan codificación en pulsos, mecanismo en el cual una neurona envía y recibe pulsos individuales, permitiendo así, la multiplexión de información frecuencia y amplitud. Las últimas investigaciones en el campo de la neurociencia apuntan a que las neuronas de la corteza cerebral realizan una computación analógica a muy alta velocidad. Thorpe y sus colegas (Thorpe et al. 2001) han demostrado que los humanos procesamos la información visual (por ejemplo reconocer una cara) en menos de 100ms. Teniendo en cuenta que entre la retina y el lóbulo temporal hay por lo menos 10 saltos sinápticos (es decir, que la información tiene que pasar por al menos 10 neuronas); esto deja 10ms de procesado por neurona, un tiempo demasiado pequeño para usar

codificación en frecuencia de pulsos. Esto no quiere decir que la codificación en frecuencia de pulsos no sea usada, aún cuando la codificación en pulsos sea más rápida.

Estas generaciones de neuronas artificiales han dado lugar a modelos de las mismas que han permitido, por un lado la simulación de redes de neuronas artificiales en computadores; y por otro, la construcción de circuitos digitales, analógicos y mixtos que implementan redes de neuronas artificiales en circuitos integrados (Indiveri et al. 2009).

2.4.2. Arquitectura de las redes neuronales artificiales

Como ya se ha expuesto anteriormente, el sistema neuronal biológico está estructurado en capas, y existe una alta interconexión entre las neuronas. Es por lo que los sistemas neuronales artificiales presentarán un nivel de paralelismo alto y distribuido en capas, en el que las celdas, que imitan el funcionamiento de las neuronas biológicas, "procesan", por medio de sus interrelaciones, la información proveniente de los sentidos o de otras capas.

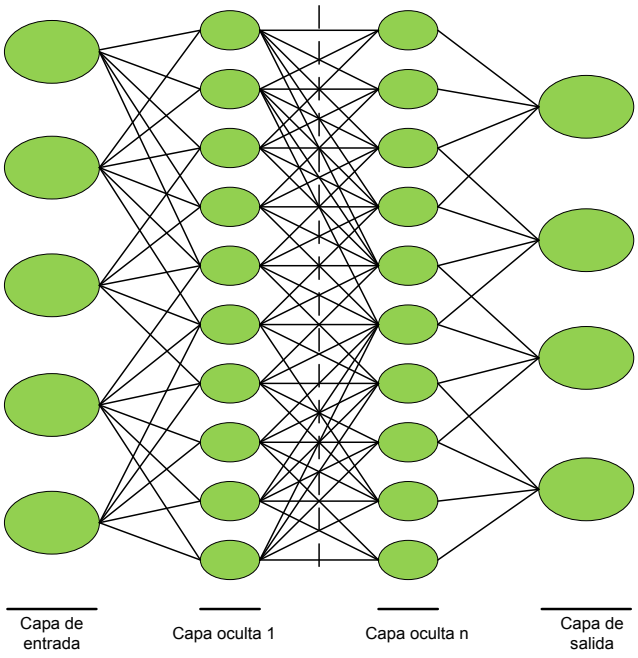


Figura 2.8: Arquitectura en paralelo de los sistemas neuronales biológicos

La Figura 2.8 muestra el diagrama de interconexión de las redes neuronales artificiales, que pretende imitar la estructura de las biológicas, donde cada nodo representa una neurona y las líneas que las unen las sinapsis. La información fluye desde la capa de entrada a la de salida, atravesando varias capas intermedias, que se llaman capas ocultas; la cantidad de capas ocultas determina las capacidades de la red. Obviamente esta arquitectura es una simplificación, en la que no se recoge, por ejemplo, el hecho de que las neuronas físicamente más cercanas comparten más información, o que existen conexiones entre las neuronas de una misma capa.

El alto grado de interconexión de las neuronas biológicas hace bastante difícil la construcción de sistemas artificiales que imiten con exactitud los sistemas biológicos. Si bien, hay que destacar que esta arquitectura ha dado sus frutos y la mayoría de las redes de neuronas artificiales hacen uso de la misma.

2.5. El bus neuromórfico AER

Como ya se ha apuntado más arriba, imitar el grado de interconexión entre las neuronas biológicas (recuérdese que cada neurona puede recibir información de otras 10000) es la principal dificultad que presenta el diseño e implementación de redes de neuronas artificiales. Más aún, si se pretende implementar dicha red en circuitos integrados tipo VLSI[15]. A priori, parece casi imposible construir estas redes de interconexión en circuitos integrados VLSI, máxime teniendo en cuenta que los VLSI son dispositivos en 2 dimensiones. Pero, si simplificamos el grado de conexión de las neuronas biológicas a dos dimensiones, se podría afirmar que no hay nada en la estructura de los sistemas biológicos que no pueda ser imitado. En parte, en un circuito integrado, "sólo" el tamaño plantea un problema; y para poder construir un sistema neuronal suficientemente grande, habría que trocearlo en varios circuitos integrados. De esta forma, el mecanismo de comunicación

[15] VLSI, siglas en inglés de *Very Large Scale Integration:* muy alta escala de integración.

entre los diferentes circuitos que formen el sistema se convierte en un aspecto fundamental.

Una posible solución a este problema de comunicación es el protocolo AER (Deiss et al. 1999) (Mahowald 1992) (Sivilotti 1991) (Mahowald 1994) (Boahen 2000) (Higgins & Koch 1999) (S. Liu et al. 2002). El protocolo AER fue inicialmente propuesto para la comunicación de circuitos neuronales de visión, y más concretamente para enviar la información que generaban las retinas de contraste de las que hablaremos más adelante.

El AER es un protocolo de comunicación digital multiplexado y asíncrono (realmente es un protocolo *handshake* simple). El fundamento del AER está inspirado en el mecanismo de comunicación de las neuronas biológicas. Como ya se ha explicado antes, los pulsos de las neuronas pueden ser entendidos como digitales, en el sentido de que hay o no hay pulso, pero el tiempo que transcurre entre ellos es analógico; y es precisamente en el tiempo entre eventos donde se codifica la información. Teniendo en cuenta que un pulso de una neurona biológica dura aproximadamente 0.5 ms y que la máxima frecuencia de pulsos está limitada, si se usa un sistema de comunicación digital rápido, en el que un evento dure por ejemplo 1 us, puede combinarse en el tiempo la información de varias neuronas, con poca pérdida de información. La pérdida de información se produciría al no respetar la temporalidad del evento, por ejemplo, si se retrasa la emisión del mismo, debido a coincidencias de eventos de dos celdas.

La Figura 2.9 ilustra el funcionamiento del AER. Las neuronas del circuito emisor generan trenes de eventos (en la neuronas biológicas son pulsos) que corresponden con su actividad. Un codificador asigna a cada neurona una dirección única; cada vez que la neurona genera un evento, el codificador emite su dirección por el bus. Debido a que las neuronas operan de forma concurrente, podría darse el caso de que dos o más neuronas emitieran eventos el mismo tiempo; un arbitrador es el encargado de resolver estos conflictos, preservando en la medida de lo posible la correcta información temporal codificada en los trenes de pulso. De esta manera, toda la información de todas las neuronas es combinada en el tiempo y enviada al circuito receptor. En el receptor los

eventos son descodificados y separados, y posteriormente enviado a la neurona correspondiente.

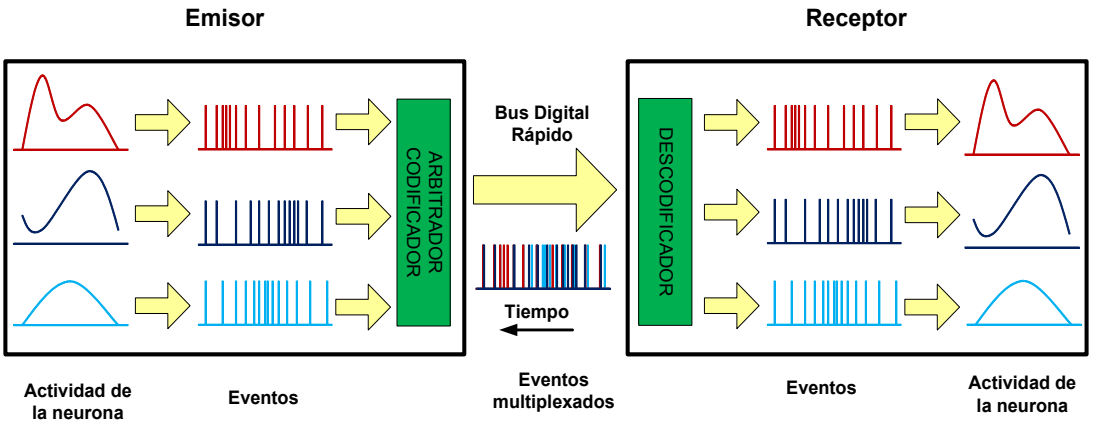


Figura 2.9: Esquema de comunicación Address Event Representation (AER)

En el esquema anterior la actividad de la neurona es codificada en frecuencia de pulsos, pero además de la dirección, es posible enviar de forma explícita otra información asociada a la actividad; o bien codificar la información en el tiempo entre dos únicos eventos, teniendo así un esquema de codificación en pulsos (que no en frecuencia de pulsos), que como ya se apuntó antes parece muy cercano a la forma de trabajo de las neuronas biológicas.

2.5.1. Algunas ventajas del AER

El protocolo AER permite una comunicación con un alto ancho de banda, resolviendo de esta manera el problema de comunicación entre circuitos neuronales que integran cientos o quizá miles de neuronas. La multiplexión en el tiempo, que implica el AER, es la única técnica factible cuando no es posible, por su cantidad, usar cables dedicados a la comunicación entre cada par de neuronas. Es importante destacar que el protocolo AER asume que el canal de comunicación está dedicado a la transmisión de señales significativas. Lo que supone que las celdas del circuito emisor solo deben enviar información cuando ocurra algo importante. Esto presenta una gran diferencia con los

sistemas de comunicación escaneados, en los que las celdas son continuamente interrogadas sobre su estado, y este emitido por el bus. El protocolo AER, por tanto, es un protocolo guiado por los datos o guidado por eventos. Así pues sólo aquellas celdas que tienen algo que informar generan eventos que serán enviados por el bus. Por lo que las celdas que varían poco no contribuyen a la carga del bus.

La representación de información y la comunicación de la misma entre los distintos circuitos que forman un sistema neural artificial son críticas, porque determina el modo en que el sistema computa. Parece que la representación de la información guiada por eventos y el protocolo AER (también guiado por eventos) permiten la construcción de sistemas artificiales cuyas estrategias de procesamiento de la información son similares a los del sistema nervioso. El protocolo AER se ha usado satisfactoriamente en muchos sistemas neuro-inspirados, tanto sensoriales (S.-C. Liu & Tobi Delbruck 2010), como de procesamiento (F. Gomez-Rodriguez et al. 2007) (A. Linares-Barranco et al. 2007) (T. Delbruck & P. Lichtsteiner 2007) y actuación (Angel Jimenez-Fernandez et al. 2009) (los sistemas basados en AER serán analizados más adelante, en este mismo capítulo, con un poco más de profundidad). Estos chips pueden ser fácilmente integrados en otros sistemas más complejos colocándolos en un entorno de diseño AER. Por lo que se puede afirmar que el AER proporciona una organización unificada para la construcción de sistemas multi-chips (R. Serrano-Gotarredona et al. 2005) (Rafael Serrano-Gotarredona et al. 2009).

El uso de direcciones digitales para especificar la neurona emisora hace el mapeado de señales pre-sinápticas sobre destinos post-sinápticos extremadamente flexible porque el evento lleva su posición de origen en sí mismo. La representación AER garantiza el orden temporal de los eventos, el evento puede ser fácilmente decodificado en cualquier ordenación física en el circuito receptor. Alternativamente, el patrón de conectividad se puede cambiar dinámicamente, lo que permite cambiar la estructura de la interconexión de las neuronas artificiales tal y como ocurre en los sistemas biológicos.

2.5.2. Implementación del protocolo AER

En este apartado se presentan las implementaciones más importantes que se han llevado a cabo del protocolo AER, tanto en su versión paralelo como serie. El protocolo AER es un protocolo *handshake*, en el que el emisor indica activando una línea que desea transmitir y el receptor activando otra línea indica que ha recibido la información (véase Figura 2.10). El comportamiento exacto de las líneas depende de la implementación concreta del protocolo, por lo tanto las líneas que intervienen en el protocolo son las siguientes:

REQ: indica una petición de transmisión, es controlada por el sistema emisor.

ACK: confirmación de la recepción, es controlada por el sistema receptor.

DIRECCIÓN: es el bus de datos donde aparecerá la dirección de la neurona emisora.

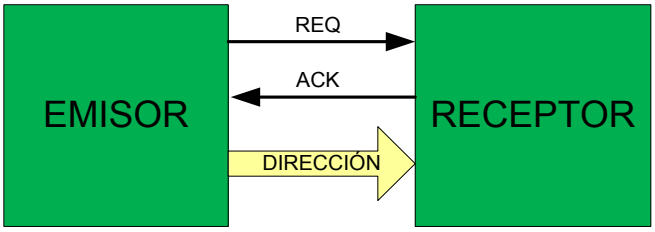


Figura 2.10: Líneas del protocolo AER

En la Figura 2.11 se muestra el protocolo de comunicación *Handshake*, el primer paso consiste en la colocación de una dirección válida (la que se quiere transmitir) en el bus de datos (dirección) al tiempo que se activa la señal REQ por parte de emisor; posteriormente el receptor activa la señal ACK; a lo que el emisor responde desactivando la señal REQ; el último paso consisten en la desactivación de la señal ACK por parte del receptor, lo que concluirá el ciclo de comunicación.

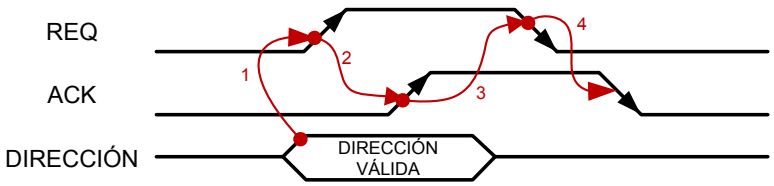


Figura 2.11: Comunicación *Handshake*.

Una trasmisión completa, desde la petición de transmisión hasta la confirmación de la misma es lo que se llama en el entorno del AER un evento. Las primeras implementaciones que se llevaron a cabo fueron la "punto a punto" y la "multipunto" ambas unidireccionales (Boahen 2000) (Higgins & Koch 1999) (Avis et al. 2001).

2.5.3. Estado actual de desarrollos que usan AER

La comunidad de la Ingeniería Neuromórfica está creciendo e incluso existe alguna publicación especializada como es *Frontiers in Neuromorphic Engineering*[16] ; y cada vez hay más investigadores interesados en el AER. A continuación están algunos de los más importantes:

- El grupo *Brain in Silicon* de la Univerdad de Stanford, liderado por Kwabena Boahen.
- El grupo Neuromórfico del IMSE-CNM-CSIC, liderado por Bernabé Linares Barranco.
- El *Institute of Neuroinformatics* de la ETZH de Zúrich.
- El *e-Lab* de la Universidad de Yale, liderado por Eugenio Culurciello.
- El *Computational NeuroEngineering Lab* de la Universidad de Florida, liderado por John G. Harris.

[16] Revista online de acceso libre sobre Ingeniería Neuromórfica: http://www.frontiersin.org/neuromorphic_engineering

- El *Computational Sensory-Motor Systems Lab* de la Universidad Johns Hopkins, liderado por Ralph Etienne-Cummings.
- El grupo de Robótica y Tecnología de Computadores de la Universidad de Sevilla, liderado por Antón Civit Balcells.

Los desarrollos más importantes que usan AER incluyen, módulos de interconexión, retinas, cócleas, circuitos de convoluciones[17] y redes de aprendizaje.

Módulos de testeo e interconexión

En cuanto a los módulos de interconexión son muy importantes los trabajos realizado por el grupo de Robótica y Tecnología de Computadores de la Universidad de Sevilla con el desarrollo de la plataforma USB-AER, que utiliza comunicación AER en paralelo (tal y como fue inicialmente propuesto) (Paz et al. 2005) (F. Gomez-Rodriguez et al. 2006) y (Rivas et al. 2005). En (R. Serrano-Gotarredona et al. 2005), (R. Serrano-Gotarredona et al. 2006) y (Rafael Serrano-Gotarredona et al. 2009), pueden encontrarse sistemas completos de sensado visual y procesamiento formados, en parte, por estos módulos descrito.

Hay algunos desarrollos de sistemas de interconexión en serie, denominado Serial-AER (Miro-Amarante et al. 2006) (Miro-Amarante et al. 2007) (Berge & Hafliger 2007) (Fasnacht et al. 2008) (Zamarreño et al. 2008).

También han de ser destacados los trabajos en el ámbito de los sistemas de interconexión, realizados por el Instituto de Tecnología y Salud de Roma (Dante et al. 2005).

Retinas

La primera retina artificial que usaba AER, fue desarrollada por Mahowald y sus colegas en 1992 al tiempo que se definió el AER (Mahowald 1992). Desde entonces, han aparecido

[17] Los circuitos de convoluciones son circuitos capaces de realizar la operación matemática de convolución de matrices, técnica matemática muy usada en tratamiento digital de imágenes, para realizar todo tipo de filtrados.

diversas retinas artificiales que también usan AER para transmitir la información visual. En la Figura 2.12, tomada de (Tobi Delbruck et al. 2010) se muestran las más recientes, desde 2001 hasta el 2010. Entre ellas, la usada en este trabajo (que como ya se ha comentado es la desarrollada por Lichtsteiner y sus colegas en el *Institute of Neuroinformatics* de Zurich (Patrick Lichtsteiner et al. 2008).

Table 1 Comparison between AER vision sensor devices. Pixel size is given both in lambda (the scaling parameter) and um units. Power consumption is at chip and not board or system level. Best metrics are in bold. (Extended from [5].)

	Prior work						This session		
Year	2001	2003	2005	2006	2008	2009	2010	2010	2010
Source	Zaghloul, Boahen [30]	Rüedi et al. [16]	Mallik et al.[9, 32]	Lichtsteiner et al. [5, 33]	Massari et al. [27]	Ruedi et al.[31]	Posch et al. 2010 [34]	Linares-Barranco et al. [2]	Culurciello et al. [3]
Functionality	Asynchronous spatial and temporal contrast,	Frame-based spatial contrast and gradient direction, ordered output	Temporal frame-difference intensity change detection APS imager	Asynchronous temporal contrast dynamic vision sensor (DVS)	Binary spatial and temporal contrast	Digital log pixel + RISC proc.	Async. Time-based Image Sensor (ATIS)	Async. Weber Contrast (SC), with either rate or TTFS coding	Temporal intensity change or spatial difference can trigger readout
Type (Sec.3)	SC TD AE	SC FE	TD FE	TC AE	SD TD FE	SC, embedded	TC AE	SC AE	TD SD FE
Gray picture output			•			•	•	•	•
Pixel size um (lambda)	34x40 (170x200)	69x69 (276x276)	25x25 (**100x100**)	40x40 (200x200)	26x26.5 (130x130)	**14x14** (311x311)	30x30 (333x333)	80x80 (400x400)	16x21 (??)
Fill factor (%)	14%	9%	17%	8.1%	20%	20%	10%(TC)/20%(gray)	2.5%	**42%**
Fabrication process	0.35um 4M 2P	0.5um 3M 2P	0.5um 3M 2P	0.35um 4M 2P	0.35um 4M 2P	180nm 1P6M	180nm 4M 2P MIM	0.35um 4M 2P	180nm SiGe BiCMOS 7M
Pixel complexity T=MOS,C=cap	38T	>50T, 1C	**6T (NMOS) 2C**	26T(14 anal), 3C	45T	~80T, 1C	77T, 4C, 2PD	131T, 2C	11T
Array size	96x60	128x128	90x90	128x128	128x64	**320x240**	304x240	32x32	128x128
Die size mm²	3.5x3.5	~10x10	3x3	6x6.3	11	5.2x8.4	9.9x8.2	2.5x2.6	??
Power consumption	62.7mW @ 3.3V	300mW @ 3.3V	30mW @ 5V (50 fps)	24mW @ 3.3V	**100uW@2V, 50fps**	**80mW** (11mW sensor)	50-175mW	0.66-6.6mW	<**1.4mW**@3V
Dynamic range	~50dB	120dB	51dB	120dB 2lux to >100 klux scene	100dB	**132dB** 6V/lux s 39dB SNR	**143dB** (static) 125dB@30FPS 56dB SNR	100dB 1lx to 100klx	2V/s/(uW/cm²) @550nm. 1.14uV/e⁻
PD dark current@25C	?	300fA	?	4fA (~10nA/cm²)	NA	44mV/s	1.6nA/cm²	NA	
Response latency, frames/sec (fps), events/sec (eps)	~10Meps	< 2ms 60 to 500 fps	< 5ms? 200 fps?	15µs @ 1 klux chip illumination 2Meps	Max 4000fps	30fps	**3.2us**@1klux 30Meps peak, 6Meps sustained	100us @50klx 66Meps	200-800fps dep. on mode 13Meps
FPN matching	1-2 decades	2% contrast	**0.5%** of full scale, 2.1% TD change	2.1% contrast	10% contrast	**0.8%**		0.87% contrast	

Figura 2.12: Retinas desarrolladas recientemente, tabla tomada de (Tobi Delbruck et al. 2010)

Cócleas

Son interesantes los trabajos realizados por el *Institute of Neuroinformatics* de la ETZH de Zurich en colaboración con otros grupos de investigación con diversos sensores auditivos (Chan et al. 2007) (Chan et al. 2006).

Circuitos convolucionadores

En cuanto a los chip para realizar convoluciones en tiempo real, sólo existen con un reconocimiento internacional los desarrollados por el grupo Neuromórfico del IMSE-CNM-CSIC (Rafael Serrano-Gotarredona et al. 2006) (R. Serrano-Gotarredona et al. 2008). También existen algunos desarrollos de convolucionadores digitales (A. Linares-Barranco et al. 2010) (Rafael Paz-Vicente et al. 2008) (Nageswaran et al. 2009).

Redes de aprendizaje

Sobre los sistemas de aprendizaje en redes neuronales pulsantes basados en STDP[18], de nuevos son importantes los trabajos realizados en este campo por el *Institute of Neuroinformatic* de la ETHZ de Zúrich (Indiveri et al. 2006), y de la universidad de Edinburgo (Yang et al. 2006).

2.5.4. Uso del bus AER en este trabajo

El protocolo AER que se usa en este trabajo fue especificado en el proyecto Europeo CAVIAR y convertido en un cuasi-estándar para muchos desarrollos AER. En el documento de especificaciones del AER en el proyecto CAVIAR (Häfliger 2007) se especifica el conector y el cable (un ATA/133 de 40 pines), y el significado de cada pin puede verse en la Figura 2.13.

Los parámetros temporales del protocolo AER usado en este trabajo son los especificados en el proyecto AER (Figura 2.14). La principal diferencia con las

[18] Siglas de inglés: *Spike-Time Dependant Pasticity,*cuya traducción podría ser plasticidad dependiente del tiempo entre pulsos. La plasticidad es la capacidad de los sistemas neuronales biológicos para cambiar las interconexiones entre las neuronas, que se supone que es la base del aprendizaje.

especificaciones del protocolo AER visto más arriba es que las líneas son activas en nivel bajo, es decir, las líneas REQ y ACK estarán activas cuando tienen valor 0.

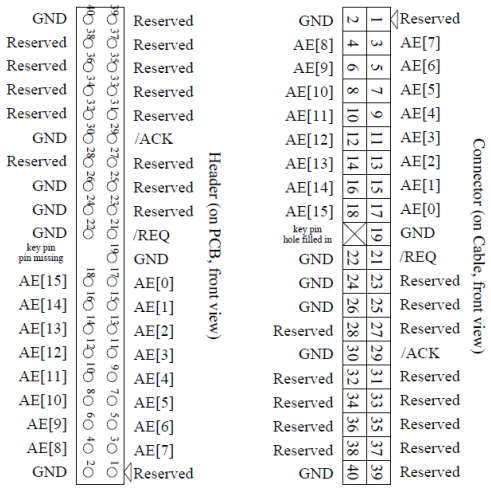


Figura 2.13: Especificación de pines de los conectores y los cables AER del proyecto CAVIAR (tomada de (Häfliger 2007))

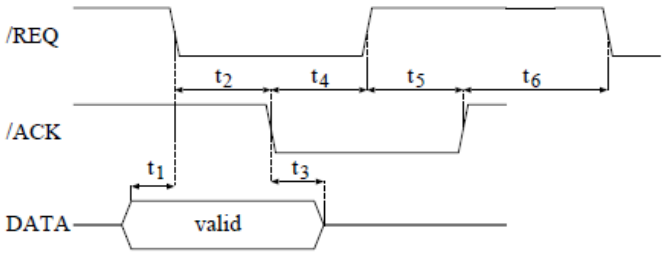


Figure 2: 4 phase handshake. Timing constraints in table 2.

	min	max	avg
t_1	0s	∞	
t_2	0s	∞	$\leq$ 700ns
t_3	0s	∞	
t_4	0s	100ns	
t_5	0s	100ns	
t_6	0s	∞	

Table 2: Timing requirements of 4 phase handshake (figure 2)

Figura 2.14: Especificación temporales del protocolo AER del proyecto CAVIAR (figura tomada de (Häfliger 2007))

2.6. Resumen

La inspiración en la Naturaleza ha guiado, y lo sigue haciendo aún en la actualidad, gran cantidad de las investigaciones científicas, sobre todo en aquellas que tratan de imitar aspectos que algunos seres vivos parecen realizar con aparente sencillez.

Si nos centramos en las investigaciones que se inspiran en los sistemas nerviosos biológicos, existe una disciplina dentro de la ingeniería, la Ingeniería Neuromórfica, que trata de imitar no sólo el funcionamiento de los sistemas biológicos, si no que trata de resolver los mismos problemas, y por supuesto de la misma forma.

Gracias a los grandes avances en el conocimiento del sistema nervioso de los primates superiores, trabajos iniciados por el premio Nobel Santiago Ramón y Cajal, se tiene cierto conocimiento de algunos aspectos fundamentales de la estructura y funcionamiento del cerebro como por ejemplo, la gran variedad de tipos de neuronas que existen, los diferentes patrones de interconexión entre las mismas o el mecanismo de comunicación mediante pulsos.

El sistema visual es uno de los más complejos y estudiados, debido a la gran dependencia que del mismo presentan los seres humanos, es de destacar las primeras investigaciones de, los también premios Nobel, David Hubel y Torsten Wiesel. Estas investigaciones, y las que le siguieron, demostraron que el procesamiento de la información visual, procedente de las retinas, se realiza en un área cerebral denominada Corteza Visual, que está dividida en capas, y que la primera de ellas (la corteza visual primaria V1) tiene un papel importantísimo en el procesamiento de aspectos como el color, la textura, el movimiento, la orientación, etc.

Gracias a estos conocimientos se desarrollaron las Redes Neuronales Artificiales; y dentro de éstas las Redes de Neuronas Pulsantes. Merece una especial mención el protocolo de comunicación para chips neuromórficos *Address Event Representation* (AER), por ser una de las primeras implementaciones hardware que se realizaron de un sistema neuromórfico. El protocolo AER fue desarrollado en 1991 en el CalTech y gracias a él han

visto la luz numerosos sistemas neuromórficos: retinas, cócleas, chip de convoluciones, redes de aprendizaje, módulos de interconexión y testeo entre otros. El protocolo AER es ampliamente usado en este trabajo, como base para la transmisión y el procesado de información.

"Todo hombre puede ser, si se lo propone,
escultor de su propio cerebro."

Santiago Ramón y Cajal

C a p í t u l o 3

S e g u i m i e n t o d e o b j e t o s

En este capítulo se aporta una nueva solución neuromórfica a uno de los problemas clásicos en visión artificial: la determinación de la posición y la estimación de la velocidad de objetos en movimiento en una escena. Se pondrán de manifiesto las diferencias entre el sistema propuesto y los sistemas de visión "clásicos", entendiendo por tales los basados en CCD[19] y procesadores de propósito general para analizar la información que estos proporcionan; y se presentarán las aportaciones que de manera neuromórfica realizan el seguimiento de objetos y la estimación de su velocidad.

3.1. Introducción

Un CCD, de modo simplificado, es un dispositivo que captura la luz y la convierte en corriente eléctrica, de manera que la cantidad de electrones producida es proporcional a la luz recibida. Los primeros dispositivos CCD fueron inventados por Willard Boyle y

[19] Siglas en inglés de *charge-coupled device*: "dispositivo de cargas interconectadas".

George Smith el 17 de octubre de 1969, en los Laboratorios Bell. Ambos fueron premiados con el Premio Nobel de Física en 2009 precisamente por este invento.

Figura 3.1: Willard Boyle (izquierda) y George Smith,
inventores del CCD, en los laboratorios Bell.

La solución al problema del seguimiento de objetos que este trabajo presenta es bastante diferente a las técnicas clásicas, tanto en la técnica para obtener el movimiento como en el sensor usado. Empezando por este último, el sensor es una retina artificial la cual, como se describirá más adelante, proporciona información visual en forma de pulsos; el análisis del movimiento se basa en el procesamiento de los pulsos que emite la retina y no en el procesamiento de fotogramas.

Además el resultado que se persigue tampoco es exactamente el mismo. El objetivo de lo que aquí presentamos va un poco más allá de, por ejemplo, la estimación del flujo óptico, puesto que pretendemos obtener la posición (lo más cercana posible al centro de masas) y la velocidad de los objetos en movimiento; a diferencia del flujo óptico que obtiene un campo de velocidades correspondientes a todos los puntos de la imagen, independientemente de si corresponden o no al mismo objeto. Por tanto la información que se pretende obtener es un poco más refinada, ya que por cada objeto en movimiento sólo se obtendrá un vector de velocidad. En la Figura 3.2, se muestra como es la

estimación del movimiento usando flujo óptico c) aplicado a la secuencia de imágenes mostrada en la parte superior de la figura a) y b); también se muestra como es el resultado que se obtiene usando el sistema que se presenta en este trabajo d), si bien hay que decir que la entrada a este último no son imágenes si no que, como se verá más adelante, es la salida de una retina artificial.

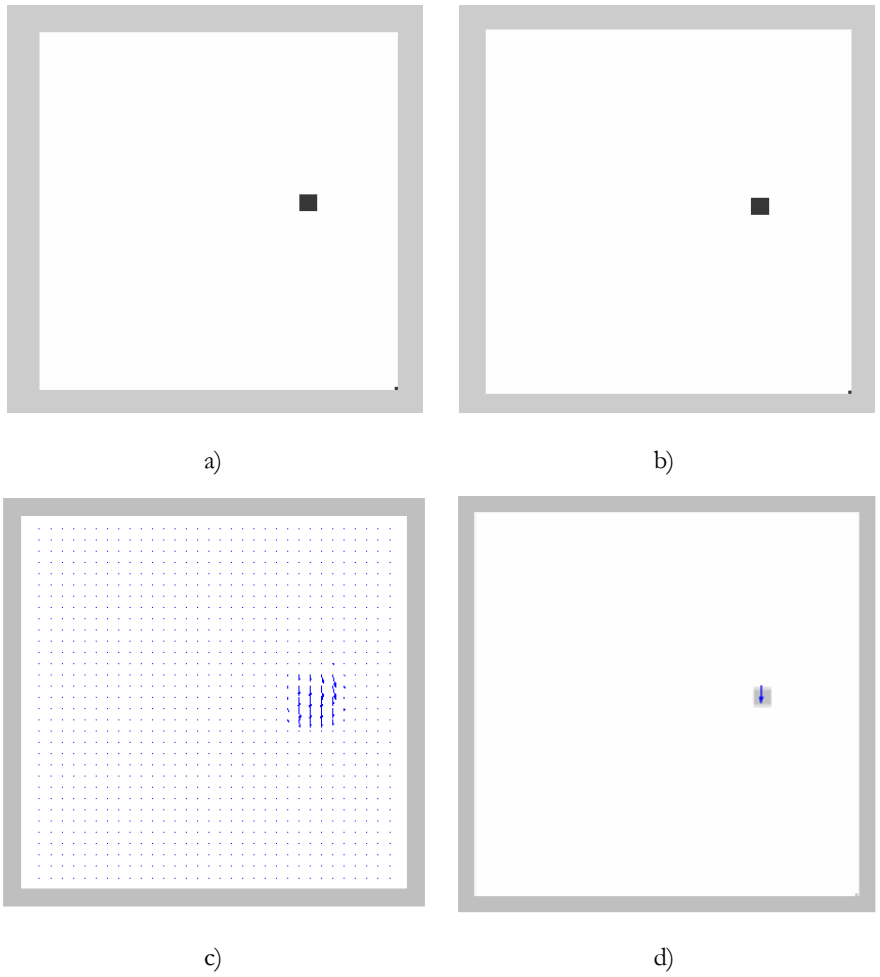


Figura 3.2: Salida del seguimiento de objetos frente al flujo óptico.

A continuación se detallan, algunos métodos "clásicos" para el análisis del movimiento; los aspectos más importantes de la retina usada; semejanzas y diferencias entre el flujo

óptico, diferencias de imágenes y la retina; y para concluir los detalles de la obtención de la posición y de la velocidad de los objetos en movimiento propuesta en este trabajo.

3.2. Algunos mecanismos tradicionales de obtención del movimiento

Veamos algunos de los mecanismos usados en visión artificial "clásica" para el análisis del movimiento: flujo óptico, diferencias de imágenes, correspondencia y segmentación. Como se verá la solución propuesta en este trabajo tiene similitudes, y también algunas diferencias, con todos ellos.

Flujo óptico

Una de las técnicas más potentes usadas en visión artificial para el análisis del movimiento es la "estimación del flujo óptico". Se basa en la comparación, más o menos compleja, de una secuencia de imágenes, obteniéndose el movimiento relativo entre los objetos en la escena y las imágenes. Las primeras teorías sobre "flujo óptico" fueron desarrolladas por James Gibson en los años cuarenta (Gibson 1950), definiendo dicho concepto en base al gradiente de la deformación de la imagen a lo largo del movimiento del observador. El principio consiste en que un objeto móvil delante de una cámara proyecta su movimiento en el espacio en el plano del sensor de visión de la misma, produciendo un campo "aparente" de velocidades bidimensional, en contraposición con el "real", que evidentemente es tridimensional (Verri & Poggio 1989). Por otra parte se supone que la cámara proporciona cambios de luminosidad fundamentalmente debido al movimiento del objeto, por lo que dichos cambios pueden ser relacionados con el campo de velocidad aparente. También se puede utilizar el mismo concepto para descubrir el movimiento propio de la cámara o los cambios de la distancia focal (zoom) que puedan producirse en el objetivo de la misma (Jain et al. 1995).

De lo anterior es fácil deducir que obtener el movimiento real a partir de los cambios de luminosidad de una imagen de vídeo presenta diversos problemas (Verri & Poggio 1989). Por una parte el movimiento detectable es sólo una proyección del real. Además los cambios de luminosidad pueden producirse sin movimiento o haber movimiento sin cambio de luminosidad. El ejemplo clásico con respecto a este segundo problema es el de la esfera con brillo uniforme rotando (Horn & Schunck 1981). Si no cambia la fuente de luz no se produce cambio de luminosidad, aunque haya movimiento (Figura 3.3.a). Y al contrario, la esfera puede permanecer inmóvil pero si cambia el foco de luz puede cambiar la luminancia sin que haya movimiento (Figura 3.3.b).

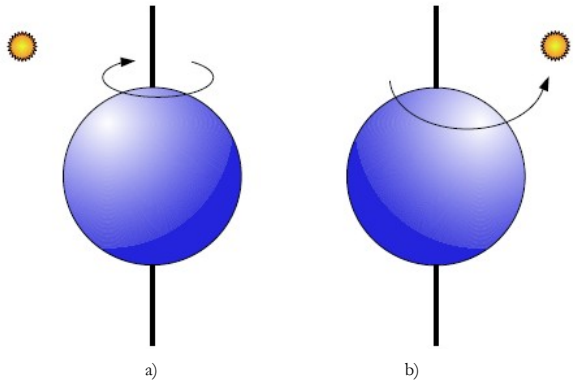


Figura 3.3: Cambio de luminosidad y movimiento

También ocurre el problema de la apertura (Nesi et al. 1995) (Horn & Schunck 1981) (Adelson & Bergen 1985) que no sólo se encuentra en la visión artificial si no que es un fenómeno sensorial humano. El ejemplo clásico del mismo es el del poste de las tradicionales barberías, que giraban sobre su eje (Figura 3.4.a), en dichos postes el cambio de luminancia o flujo óptico (Figura 3.4.b) es perpendicular al campo de movimiento (Figura 3.4.c).

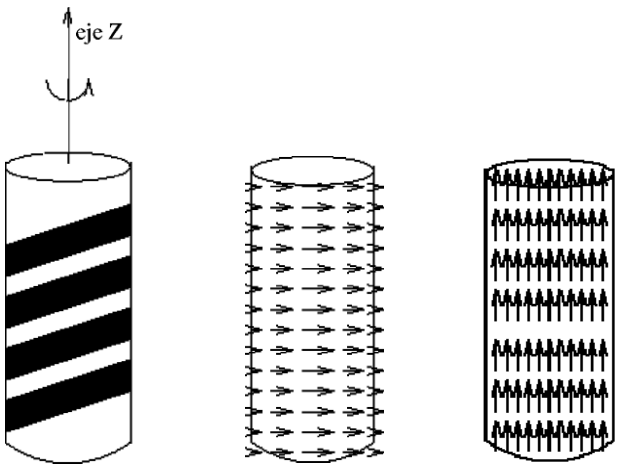


Figura 3.4: Ilusión del poste del barbero: a) poste con franjas de color, b) sentido de giro y c) flujo óptico.

Horn y Schunck (Horn & Schunck 1981) proponen la siguiente formulación del flujo óptico, que posteriormente será conocida como "la ecuación de restricción del flujo óptico.". Suponen que la intensidad de la imagen en un punto dado se conserva en el tiempo, y como ya se ha expuesto las variaciones de intensidad de la imagen se deben únicamente a los desplazamientos de los objetos, sin tener en cuenta los cambios de iluminación. Bajo dichas consideraciones, formalmente, para dos imágenes consecutivas separadas por un tiempo δt, si $I(x, y, t)$ es la intensidad de la imagen en el punto (x, y) y en el instante t, ha de cumplirse que:

$$I(x, y, t) = I(x + u\delta t, y + v\delta t, t + \delta t) \tag{3.1}$$

donde (u, v) es el vector de flujo óptico del píxel desconocido. El vector de flujo óptico depende de cada píxel considerado, siendo $u(x, y)$ la componente sobre el eje *x* y $v(x, y)$ la componente sobre el eje *y*. Considerando que la intensidad luminosa varía de forma suave respecto de *x, y, t*, tomando el límite cuando δt tiende a 0 y desarrollando la expresión anterior se tiene:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{3.2}$$

que puede ser reescrita de forma compacta como:

$$I_x u + I_y v + I_t = 0 \tag{3.3}$$

donde I_x, I_y, I_t son las derivadas parciales de la imagen con respecto a x, y, t, respectivamente, y $V = (u, v)$ representa el vector de flujo óptico en cada punto a determinar.

La bibliografía sobre el cálculo del flujo óptico es muy extensa y los métodos de lo más variados, en este sentido Barron (Barron et al. 1994) clasifica dichos métodos en tres categorías:

- Métodos basados en correspondencias; trabajan comparando los *frames* consecutivos obteniendo la velocidad del desplazamiento a partir del cambio de posición de los objetos.
- Métodos basados en derivadas o gradiente; parten de la ecuación de restricción del flujo óptico y mediante derivadas espaciales y temporales de la iluminación se obtiene la velocidad.
- Métodos que usan la energía, frecuencia o fase; la velocidad se obtiene mediante filtros espacio-temporales organizados en bancos y sintonizados en un rango determinado de las mismas, generalmente se parte del análisis del movimiento mediante la transformada de Fourier.

En nuestro entorno podemos destacar varios trabajos en los que se utilizan las técnicas de flujo óptico en conjunción con sistemas bioinspirados (Botella Juan 2007) y (Matías Casado 2010). El primero realiza un estudio muy completo sobre la materia y propone

nuevas implementaciones partiendo de imágenes tipo *frame*[20], el segundo es una variación sobre algunos algoritmos desarrollados en JAER por el INI para sus retinas (JAER 2011). En esta última referencia sí se utiliza las imágenes continuas de una retina además del bus AER.

Diferencias de imágenes

Una primera aproximación para la obtención del movimiento es la detección del mismo. Los mecanismos más intuitivos de detección de movimiento están basados en la sustracción del fondo (Cucchiara et al. 2003), o sus variaciones: imagen diferencia o imagen de diferencias acumuladas (Jain et al. 1995) (Rosin 1998). Los mecanismos de imagen en diferencia se utilizan en las secuencias de vídeo formadas por *frames*. Consiste en restar píxel a píxel un frame de otro consecutivo, de forma que si el resultado es mayor que un determinado umbral se marca como "uno" dicho píxel, indicando que ha habido una variación significativa, de esta forma se obtiene una imagen en la que los píxeles brillantes marcan la diferencia entre dos *frames*.

Por tanto, la imagen diferencia I_d se puede definir como:

$$I_d(\boldsymbol{p}, t_1, t_2) = I_2(\boldsymbol{p}, t_2) - I_2(\boldsymbol{p}, t_1) \tag{3.4}$$

donde $\boldsymbol{p} = (x, y)$ es el píxel de la imagen y t_1, t_2 son los instantes de tiempo de dos imágenes consecutivas. Si tomamos un umbral T_d la imagen diferencia sería:

$$I_d \begin{cases} 1 & si \;\; I_d(\boldsymbol{p}, t_1, t_2) \geq T_d \\ 0 & en\ otro\ caso \end{cases} \tag{3.5}$$

La imagen de diferencias acumuladas es similar pero comparando una secuencia de *frames*. Un ejemplo lo encontramos en la Figura 3.5, esta imagen resulta casi idéntica a las que posteriormente veremos que se obtienen de las retinas diferenciales, cuando se hace una transformación a *frames* mediante el muestreo de pulsos.

[20] En inglés fotograma, en tratamiento de imágenes es usual usar el término en inglés en lugar de en español.

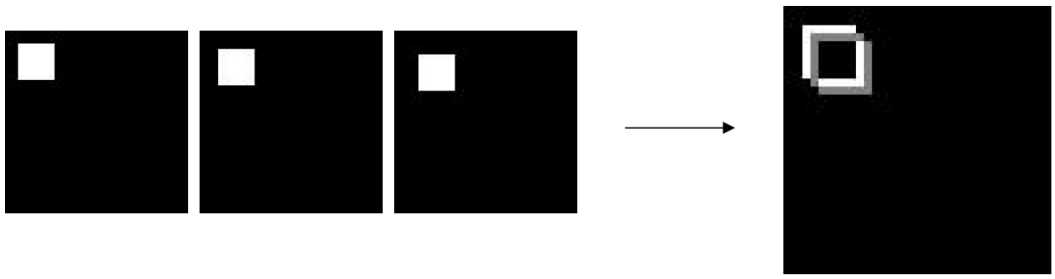


Figura 3.5: Imagen de diferencias acumuladas

Correspondencia de bloques o puntos característicos

Otro método de obtención de movimiento similar a los anteriores es el ajuste de bloques (*block matching*). La idea de dividir la imagen en bloques para después estudiar cada bloque es recurrente en muchos de los métodos de obtención del movimiento. En este trabajo, como veremos más adelante, haremos uso de dicha metodología dividiendo la imagen en bloques que serán tratados por celdas diferentes.

La correspondencia de puntos característicos consiste en identificar ciertos puntos en la imagen, generalmente las esquinas de los objetos, y posteriormente por medio de métodos estadísticos "encontrar" esos mismos puntos en la siguiente imagen de la secuencia (Saeedi et al. 2002).

Por otra parte los métodos de correspondencias buscan los movimientos basándose en hacer corresponder un píxel (Pi) de un frame (k), con otro (Pj), del siguiente frame (k+1) en la secuencia de *frames*. Para hacer dichas correspondencias se suele emplear la segmentación de la imagen en partes, para posteriormente calcular el movimiento de cada una de dichas partes en la secuencia de imágenes. La segmentación pretende extraer los objetos en movimiento de los estáticos.

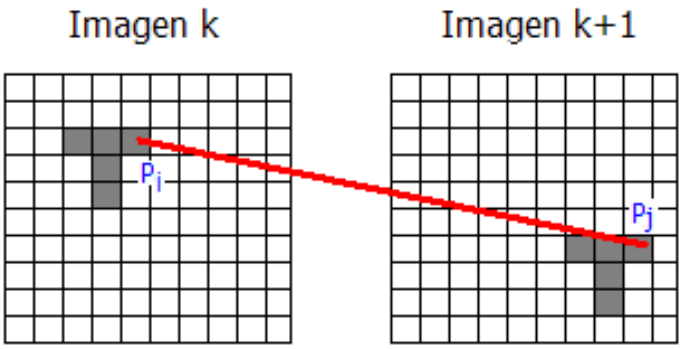


Figura 3.6: Correspondencia de bloques

Segmentación

Los mecanismos de segmentación son muy variados pero se pueden dividir en (Martín et al. 2005):

Algoritmos de segmentación en líneas que extraen el contorno de los objetos. Ejemplos de estos algoritmos se pueden encontrar en (Gu et al. 1996) y (Zhang 1995). En cierta forma la imagen en diferencias acumuladas extrae los bordes de los objetos en movimiento, siendo un mecanismo fácil de implementar. Veremos que la retina diferencial obtiene una información muy parecida.

Algoritmos de segmentación en bloques que modelan las imágenes dividiéndolas en figuras regulares de tamaños variables que se adaptan a áreas de nivel de luminosidad homogénea. El movimiento se determina en base al desplazamiento de los centros de las figuras entre las sucesivas imágenes. Ejemplos de estos algoritmos se pueden encontrar en (Schweitzer 1995), (Panjwani & Healey 1995) y (Szeliski & Shum 1996).

Algoritmos de segmentación de grupos[21] que modelan las imágenes como áreas amorfas adaptadas a áreas de la imagen de características más o menos homogéneas. El movimiento se determina en base al desplazamiento de los centros de masa de cada una de las áreas correspondientes entre dos imágenes consecutivas. Ejemplos de estos algoritmos

[21] En la literatura es común encontrar la palabra cluster, para hacer referencia a agrupaciones de píxeles.

se pueden encontrar en (Kottke & Sun 1994), (Pardàs & Salembier 1994) y (Uchiyama & Arbib 1994).

En este trabajo, como veremos en los siguientes apartados, se propone un mecanismo de seguimiento y obtención de la posición de objetos móviles que en cierta forma guarda parecido con los mecanismos básicos de segmentación mencionados, pero en realidad ni se aplican en secuencias de imágenes basadas en fotogramas (*frames*), ni están basados en algoritmos computacionales clásicos.

3.3. Retina diferencial espacio-temporal

Gran parte de la forma de obtener la posición y la velocidad de los objetos depende de la información que proporciona el sensor, que como ya se ha apuntado más arriba es una retina artificial. Concretamente, en este trabajo se ha usado una retina desarrollada en el Instituto de Neuro-informática de la Universidad de Zúrich por Tobi Delbruck y su grupo (Patrick Lichtsteiner et al. 2008) y (JAER 2011). Esta retina está compuesta por una matriz de 128x128 dispositivos que llamaremos píxeles, por semejanza con las imágenes digitales. Cada uno de estos píxeles, que trabajan de forma asíncrona e independiente unos de otros, es un circuito foto-receptor que capta la luz y emite pulsos según la diferencia de luminosidad en el tiempo que percibe, es decir, sólo es capaz de ver los cambios de luminosidad en el tiempo. Cada píxel emite eventos positivos (ON) cuando cambia de oscuro a iluminado y eventos negativos (OFF) en el caso contrario.

Esa información de cambio de intensidad luminosa es codificada proporcionalmente en frecuencia de pulsos, de manera que un cambio más brusco en la luminosidad producirá más pulsos y un cambio de luminosidad más leve producirá menos pulsos. Para enviar los pulsos, cada píxel es numerado con su posición en la matriz (que llamaremos dirección) y los pulsos de todos estos píxeles son multiplexados en un bus común usando el esquema de trasmisión AER, así pues en el bus aparecerán las direcciones de los píxeles que generan pulsos. De manera que la salida de la retina es un tren de eventos AER (positivos y

negativos) en el que se codifica las diferencias de intensidad luminosa que los píxeles de la retina captan, en un intervalo de tiempo. La Figura 3.7 muestra una foto de la retina artificial a); una microfotografía del chip que implementa la retina b); un esquema básico del circuito de cada uno de los píxeles de la retina c) donde pueden observarse las partes más importantes como son el foto-receptor, el circuito diferenciador y los comparadores para emitir pulsos positivos o negativos dependiendo del sentido del cambio en la luminosidad; por último se muestra el principio de funcionamiento de cada uno de los píxeles d), en el que se observa como dependiendo del sentido de cambio en la luminosidad y de unos umbrales previamente establecidos se producen pulsos positivos o negativos.

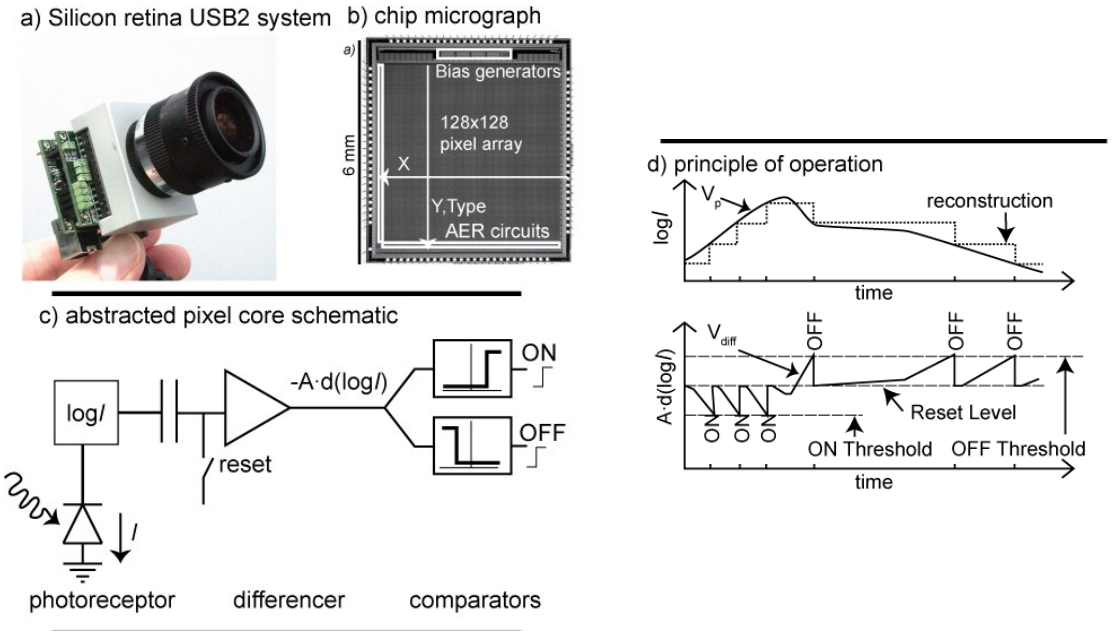


Figura 3.7: Retina artificial diferencial espacio-temporal tomada de (Patrick Lichtsteiner et al. 2008)

Como se ha apuntado antes, la retina capta los cambios de intensidad luminosa que se producen en un intervalo de tiempo, por tanto la retina capta el movimiento de la escena. Así pues, si suponemos que la retina está observando una escena en la que se mueve un objeto, la salida será una serie de trenes de pulsos positivos, correspondientes a los píxeles

que captan el flanco de avance del objeto, entrelazados de manera aleatoria (ya que los píxeles funcionan de manera asíncrona e independientes unos de otros) con trenes de pulsos negativos correspondientes a los píxeles que captan el flanco trasero del objeto.

Para poder observar lo que está captando la retina, se puede realizar una reconstrucción en un fotograma de la salida. Esta reconstrucción se realiza recolectando los eventos producidos y anotando para cada una de las direcciones (que corresponde con la posición del píxel en la matriz) la cantidad de eventos recibidos en un determinado tiempo. Posteriormente se genera una imagen digital en la que la cantidad de eventos recibidos es traducida a niveles de gris, los eventos positivos (ON) están en tonos que van del gris al blanco, siendo el blanco el que corresponde a los píxeles con más actividad; y los eventos negativos (OFF) en tonos que van del gris al negro, siendo el negro el correspondiente a los píxeles con más actividad.

La Figura 3.8 muestra una captura de pantalla de la aplicación "JAERViewer", desarrollada por los mismos investigadores que la retina (el Instituto de Neuroinformática de la Universidad de Zúrich), en la que se puede ver un fotograma en el que se han integrado, durante 77 ms, los eventos AER producidos por la retina cuando "observaba" un cuadrado blanco moviéndose de izquierda a derecha (ver Figura 3.9).

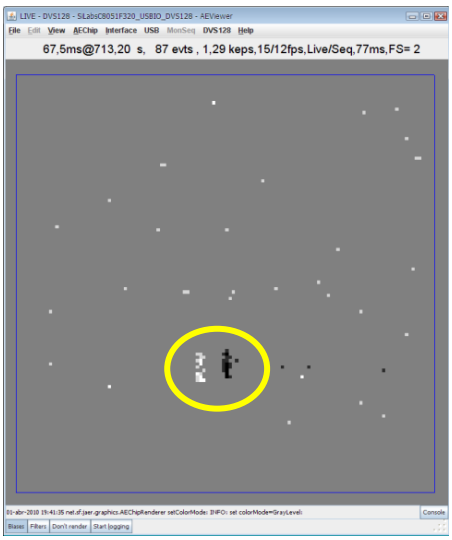


Figura 3.8: Reconstrucción en un fotograma de la salida de una retina artificial.

En la Figura 3.8 marcado en amarillo se observan los eventos correspondientes al movimiento del objeto (el cuadrado). Se puede ver claramente como sólo aparecen eventos en los dos flancos del objeto que producen cambio de estado de los píxeles de la retina. Estos son los correspondientes al flanco de avance del objeto y los correspondientes al flanco trasero, eventos positivos y negativos respectivamente. Además aparecen una serie de eventos en zonas en la que no hay movimiento, esto es debido, según los desarrolladores de la retina, a errores en la fabricación de los píxeles que provocan disparo de pulsos sin causa aparente. Cualquier sistema que use la información de la retina debe tener presente esta circunstancia y no tener en cuenta estos eventos.

Figura 3.9: Cuadrado desplazándose hacia la izquierda

a) b)

Figura 3.10: Escena real a) y salida de la retina b). Figura extraída de (JAER 2011)

En Figura 3.10 se puede observar una escena real a) y la salida que ofrece la retina b), como puede comprobarse solo aquellas partes de la escena que están en movimiento son captadas por la retina (en este caso el tiempo de integración es de 40 ms). La figura ha sido extraída de (JAER 2011).

Toda la información relativa a la retina (incluidas características técnicas, driver, manuales etc.) puede encontrarse en(JAER 2011).

3.4. El flujo óptico, imagen en diferencia y la retina diferencial espacio-temporal

La información que proporciona la retina utilizada en este trabajo es cualitativamente significativa a la hora de la detección del movimiento. Si partimos del caso ideal en el que no hay ruido, y los únicos cambios de luminosidad son debidos al movimiento de un objeto, podemos considerar que en una escena en la que no se produce movimiento no habría emisión de eventos. Por tanto, la sola presencia de eventos en el bus AER evidencia que ha ocurrido algún tipo de movimiento; la ausencia de eventos indica lo contrario: no hay movimiento. Por otra parte, los píxeles cuya dirección aparece en los eventos sitúan de forma clara en la escena el lugar en el que se produce movimiento. Además el tiempo entre eventos nos da idea de la velocidad con la que cambia la luminancia en los píxeles en los que se localiza el movimiento, lo que está relacionado directamente con la velocidad a la que se mueve el objeto. En definitiva, desde el punto de vista cualitativo, la retina diferencial utilizada tiene unas características que la hace bastante adecuada para el estudio del movimiento:

1. El cambio de iluminación total en la escena, si sólo se debe al movimiento englobado en la misma, es cero. Esto normalmente se expresa como la derivada de la iluminación con respecto al tiempo es cero:

$$\frac{d\sum_{i=0}^{n} I_i}{dt} = 0 \tag{3.6}$$

donde I_i es la iluminación del píxel i y n el número de píxeles de la retina.

En nuestro caso si sumamos el número total de eventos producidos en un intervalo de tiempo el resultado final es cero, los eventos positivos se compensan con los negativos, siempre considerando que no hay cambios de luminosidad y que el objeto móvil permanece dentro de la escena con fondo constante.

$$\sum_{i=0}^{n} ep_i - en_i = 0 \qquad \forall\, t \in [0, t_f] \tag{3.7}$$

Donde ep_i es el número de eventos positivos generados por el píxel i, en el tiempo t; análogamente en_i es el número de eventos negativos y n es el número de píxeles de la retina.

2. Los cambios de iluminación locales, o lo que sería la derivada parcial con respecto al tiempo de dicha iluminación para un píxel dado, se obtiene directamente por la frecuencia de los eventos (o tiempo entre eventos) disparados por dicho píxel.

3. Si en una escena en movimiento sumamos todos los eventos con su correspondiente signo para cada píxel de forma individual durante un intervalo de tiempo, la imagen obtenida representa la posición inicial del objeto mediante pulsos negativos y la final mediante pulsos positivos; los cuales si se sumasen entre sí se compensarían unos con otros dándonos cero eventos, como se indicó en la primera característica (de nuevo esto sólo ocurre en el caso ideal y sin ruido).

   Por otra parte, la retina usada tiene un control de ganancia diferente para los eventos negativos y positivos, lo que implica que es muy difícil regular de la misma forma la emisión de eventos positivos y negativos. Es decir, el número de eventos que se emiten cuando cambia la iluminación de menor a mayor es diferente de los que se producen cuando cambia de mayor a menor, aunque los cambios de la iluminación sean de la

misma magnitud pero de signo contrario en ambos casos. Lo que significa que con lo que respecta a esta característica 3 es imposible obtener evidencia experimental de forma precisa.

4. Si consideramos el mecanismo de detección de imagen de diferencias mencionada anteriormente encontramos que la salida de la retina en sí misma es una imagen en diferencia pero continua en el tiempo y no dividida en *frames*, siendo además la información que contiene proporcional al cambio de brillo y no simplemente uno o cero como anteriormente se mencionó para la imagen en diferencia. Además nos proporciona tanto la imagen en diferencia positiva como negativa. Sin embargo, si sumamos eventos durante un intervalo de tiempo como indicamos en el anterior apartado 3, la imagen que se obtiene es algo parecido a la de diferencias acumuladas. Por otra parte, y sólo en el caso ideal y con imágenes simples en las que se conoce la iluminación y el contraste del objeto, se podría obtener el módulo de la velocidad de un objeto móvil usando exclusivamente la frecuencia de eventos.

Llegado a este punto y teniendo en cuenta las cuatro características anteriores nos podemos plantear qué mecanismo sería el más adecuado usar para detectar y localizar el movimiento de objetos mediante la retina diferencial. En principio, podemos pensar que los mecanismos del gradiente o de correspondencia son los que mejor se adaptan a la salida de la retina.

Con lo que respecta al mecanismo del gradiente, si partimos de la ecuación del flujo óptico tenemos que el término de la derivada parcial de la intensidad con respecto al tiempo nos viene dado directamente por la propia retina para cada píxel como se indicó en la característica 2. Parece que podríamos emplear cualquiera de los algoritmos y métodos que en la bibliografía aparecen para obtener de esa forma las componentes de velocidad. Pero surgen algunos problemas que, aunque no son insalvables, complican este mecanismo para la obtención del movimiento. El primer problema se plantea con el cálculo de las derivadas espaciales de la iluminación, las cuales son necesarias para resolver la ecuación de restricción del flujo óptico, teniendo además la dificultad añadida de que la información

que pretendemos procesar está servida en forma de eventos y no como *frames*. Esta dificultad se podría solventar utilizando filtros de convolución y los convolucionadores de eventos que en diferentes trabajos se han propuesto (Paz Vicente 2008) (Alejandro Linares-Barranco et al. 2006). Pero a pesar de ello una vez obtenidos los términos de las derivadas parciales espaciales se deberían acometer la resolución de un sistema de ecuaciones, lo cual también se puede solventar utilizando los mecanismos que se detallan en (Jiménez Fernández 2010) para operar con pulsos o eventos. En definitiva plantearse la obtención de las derivadas espaciales y la resolución de los sistemas de ecuaciones a base de pulsos, sin utilizar algoritmos computacionales tradicionales, parece a priori un trabajo bastante complejo aunque probablemente sea posible, ya que las bases están establecidas en diversos trabajos.

Por otra parte las características 3 y 4 nos hacen pensar que la utilización de la imagen en diferencia y la segmentación de bordes de los objetos móviles resultan mecanismos mucho más inmediatos para la obtención del movimiento y posicionado de los objetos mediante la retina diferencial. La única dificultad a tratar es la eliminación del ruido de la retina, lo cual se resuelve con el filtro *BackGroundActivity Filter* que más adelante se detalla. Podrían plantearse muchos procedimientos para la obtención del movimiento teniendo en cuenta las características 3 y 4 expuestas anteriormente, pero el mecanismo que nos parece más sencillo, una vez se ha obtenido el frente de pulsos positivos y negativos, es localizar la posición del punto intermedio entre ambos frentes y considerar que dicho punto se corresponde con el punto significativo a seguir del objeto. Obsérvese que se pretende procesar la información de forma continua, siguiendo el flujo de eventos sin acumularlos en *frames* , pero inevitablemente el sistema necesitará la llegada de varios pulsos para poder operar y obtener una posición. Una variante de este procedimiento consiste en utilizar sólo el frente positivo o el negativo. En las pruebas finales, en este mismo trabajo, se ensaya un procedimiento de este tipo.

Un mecanismo alternativo basado en una simulación de una retina diferencial, y considerando sólo los frentes positivos, se utiliza en (Matías Casado 2010).Consiste en restar el tiempo entre dos eventos del mismo signo en diferentes píxeles próximos. De

forma simplificada podríamos pensar que los eventos disparados por cada píxel son en realidad indicaciones de que por dicho píxel "pasa" en ese momento el objeto. Restando el tiempo en que se produjeron ambos eventos se obtiene el tiempo que tardó el objeto en desplazarse de un píxel a otro, y por tanto su velocidad. Evidentemente este procedimiento necesita componer el movimiento global a base de la detección de pequeños desplazamientos que pueden no ser todos correctos de forma individual (problema de la apertura). Aunque no se llega a realizar una transformación a *frames*, sí que se realiza un almacenamiento de la imagen en forma de mapa de tiempos; lo cual, en nuestra opinión, modifica en cierta forma la filosofía última del procesamiento pulsante.

En el siguiente apartado se detallará cómo se obtiene el movimiento a partir de la construcción de unas celdas de procesamiento de señales pulsantes distribuidas en cascada, que serán las encargadas de seguir a los objetos móviles.

3.5. Obtención de la posición del objeto.

En este apartado se describe como, a partir de la secuencia de eventos que produce la retina, se obtiene la posición de los objetos en movimiento en una escena.

Antes de entrar en detalles conviene destacar cuales son las diferencias fundamentales con respecto a los sistemas de procesamiento de imágenes clásicos. En primer lugar el sistema que se presenta no debe ser entendido como un sistema de procesamiento de imágenes como tal, puesto que en su funcionamiento no interviene imágenes; el término que mejor se ajusta a su funcionamiento es el de sistema de procesamiento de información visual retino-mórfica. Como se ha apuntado más arriba los sistemas de visión clásicos basan su funcionamiento en el análisis sistemático de una secuencia de imágenes. Este sistema procesa la secuencia de eventos que produce la retina al "observar" una escena, esos eventos son procesados "al vuelo", es decir, en el mismo momento que llegan, sin hacer una integración previa de los mismos.

Publicaciones recientes (T. Delbruck & P. Lichtsteiner 2007), (M Litzenberger et al. 2006) y (Fernández-Domínguez et al. 2010) presentan sistemas que, partiendo de la secuencia de eventos producida por una retina artificial como la usada en este trabajo, son capaces de obtener resultados muy parecidos a los que aquí se presentan, aunque podemos destacar algunas diferencias importantes.

En cuanto al primero de ellos, Delbruck y sus colaboradores realizan una integración de eventos durante un periodo de tiempo, que posteriormente son descargados a un PC y un programa escrito en JAVA procesa, obteniendo la posición y la velocidad de los objetos. La forma de integrar los eventos es diferente a la explicada más arriba, en este caso, los eventos son recolectados y se les asigna una marca de tiempo, de esta manera se tiene toda la información temporal de los mismos, recordemos que la retina codifica la actividad de los píxeles en frecuencia de pulsos. En la página web (JAER 2011) pueden encontrarse algunos procesamientos basados en esta misma técnica: integrar durante un tiempo los eventos marcándolos temporalmente y procesarlos posteriormente en un PC.

En cuanto al segundo de los sistemas, Hosck y sus colaboradores, realizan una integración parecida a la que realiza Delbruck pero en este caso el encargado de procesar los eventos es un DSP[22].

Es precisamente esta integración de eventos la principal y más destacable diferencia del sistema que aquí se presenta. Como ya se ha apuntado, el sistema que aquí se presenta no realiza integración de eventos que luego se procesan. Esta característica hace que este sistema sea "más neuromórfico", si es que puede usarse esa expresión. En realidad, no se sabe a ciencia cierta, cómo es el mecanismo exacto de procesamiento de la realidad en el cerebro, pero se puede afirmar casi con seguridad, que no existe una integración a modo de fotograma, el cual es procesado en un paso posterior, es decir, el cerebro no funciona procesando fotogramas de la realidad. Parece más bien, que el cerebro procesa los eventos al mismo ritmo que los recibe. En una arquitectura altamente paralela e interconectada, de

[22] Siglas en inglés de "digital signal processor": procesador digital de señales

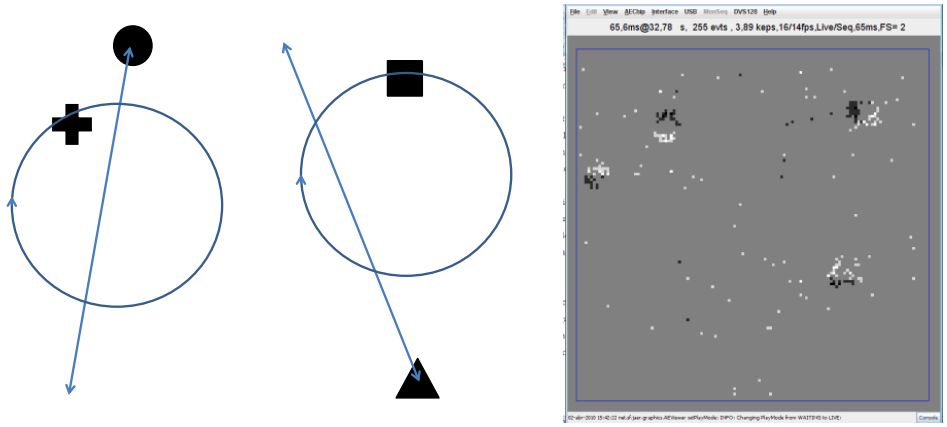
manera que la "realidad" es procesada de manera distribuida en las interconexiones entre las neuronas.

Lo que aquí se presenta, es un sistema basado en celdas o unidades de procesado independientes, estas celdas se encarga de procesar los eventos y generar una respuesta lo más rápidamente posible, tal y como hacen las neuronas biológicas.

En resumen, el sistema que se presenta es un sistema neuromórfico, compuesto por un conjunto de celdas que es capaz de procesar la información que recibe de una retina artificial en forma de eventos. Los eventos son procesados "al vuelo", es decir, en el mismo momento que se reciben. Sin integrarlos a modo de fotograma, ni como lista de eventos marcados temporalmente; y capaz de obtener la posición y seguir objetos en movimiento.

Así pues, la entrada al sistema es una secuencia de eventos producida por una retina artificial, y que corresponden con el movimiento de la escena. Se trata, por tanto, de obtener de esa secuencia de eventos las posiciones de todos los objetos que se mueven. Si analizamos con un poco de detenimiento la salida de la retina (en forma de fotograma) observaremos que por cada objeto en movimiento se genera una nube de eventos agrupados temporal y espacialmente. Además, como ya se ha visto antes, la retina produce dos tipos de eventos: positivos (ON) en el flanco de avance del objeto; y negativos (OFF) en el flanco trasero de objeto.

En la Figura 3.11.a se muestra la configuración de una escena en la que hay cuatro objetos en movimiento (una cruz, un circulo, un cuadrado y un triangulo), cada uno de ellos con trayectorias diferentes; en la Figura 3.11.b se muestra una reconstrucción en forma de fotograma de la salida de la retina, donde se puede observar un grupo de eventos por cada objeto (además de los eventos aislados debidos a errores de fabricación de chip de la retina).



a) b)

Figura 3.11: Escena con 4 objetos diferentes en movimiento a);
salida de la retina b).

Teniendo en cuenta esa agrupación temporal y espacial de los eventos, y los dos tipos de eventos producidos, el sistema propuesto trata de extraer la posición relativa de los mismos, referida a un sistema de coordenadas bidimensional coincidente con el plano de la matriz de píxeles de la retina, obteniendo una posición para cada una de esas nubes de eventos, y por ende la posición de los objetos de la escena.

Como ya se ha comentado, el sistema está compuesto por un conjunto de celdas, cada una de estas celdas estará encargada de obtener la posición de una de estas agrupaciones de eventos y de realizar el seguimiento de la misma, de forma que su salida será la posición de la agrupación, que será actualizada a medida que ésta se mueva, como consecuencia del movimiento del objeto. A estas celdas las llamaremos CMCell.

3.5.1. La celda CMCell

Estas celdas son las encargadas de obtener la posición de los objetos en movimiento en una escena a partir de la secuencia de eventos que produce la retina.

El funcionamiento de estas celdas presupone el uso de una arquitectura en cascada o jerárquica, en la que la primera celda de un nivel recibe toda la secuencia de eventos,

extrayendo de ella los eventos que le son de interés, y reenviado el resto para que puedan ser procesados por la siguiente celda. En el capítulo 5 se verá con más detalle este tipo de arquitectura.

Haciendo una analogía con las neuronas biológicas, cada CMCell tiene cuatro sinapsis, una de entrada y tres de salida. Por la sinapsis de entrada la celda recibe los eventos provenientes de la retina, en el caso de que la celda fuera la primera de la capa; o de la celda anterior en caso contrario. Por una de las sinapsis de salida se enviará el resultado del procesamiento, es decir, la posición del objeto al que la celda esté siguiendo. Por la segunda sinapsis de salida se reenviará aquellos eventos que no son de interés para el seguimiento que se está realizando. Y por la última sinapsis se enviarán los eventos procesados, es decir, aquellos que corresponden con el objeto en movimiento que se está siguiendo.

Todas las entradas y salidas de la CMCell utilizan el esquema de comunicación AER. La Figura 3.12 muestra como es la interfaz externa de la CMCell, donde se observa como la celda extrae de la secuencia de eventos, aquellos que le son útiles y reenvía el resto hacia la siguiente celda (véase el apartado 5.1 dedicado la arquitectura de procesamiento propuesta).

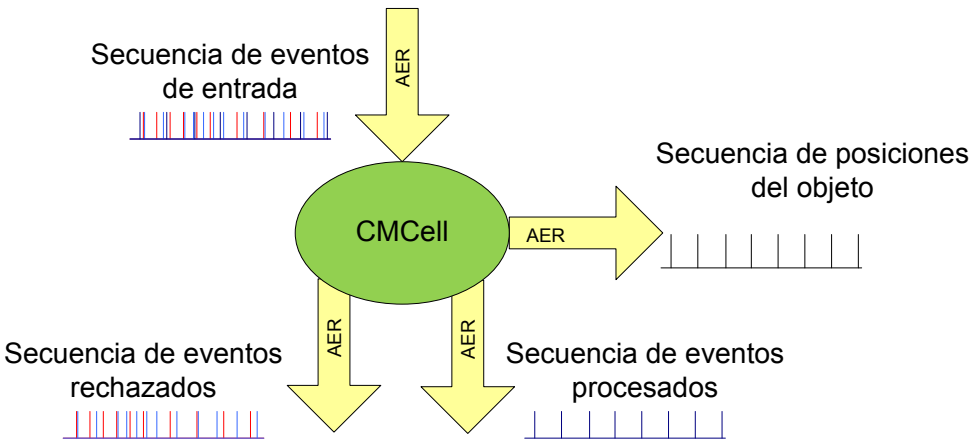


Figura 3.12: Puertos de la CMCell.

Para entender el funcionamiento de la CMCell es necesario definir algunos atributos internos de la misma, algunos de ellos configurables, que permiten adaptar el funcionamiento de la celda a una aplicación concreta:

- **Posición**: es la salida de la CMCell, y por lo tanto corresponde con la posición del objeto que está siguiendo. Se puede configurar la posición inicial que será la posición inicial a partir de la cual la celda comenzará a intentar buscar y seguir un objeto.
- **Ancho**: define el tamaño del campo de visión. Se puede configurar su valor inicial para adaptarlo al tamaño de los objetos esperados. Su valor cambia durante el seguimiento para adaptase al tamaño del objeto.
- **Campo de visión inicial:** es el área en la que la celda espera encontrar un objeto a seguir tras el reinicio. Está definido a partir de la posición inicial y el valor de ancho inicial.
- **Campo de visión**: es el área donde se encuentra el objeto, está definida por la posición (centro) y el ancho. Haciendo una analogía con la biología, el campo de visión hace las veces del campo receptivo de las neuronas de la corteza visual primaria V1.
- **Tiempo de reinicio**: este tiempo permite a la celda volver a la posición inicial si no recibe eventos dentro de su campo de visión durante ese tiempo. Es configurable y permite adaptar el funcionamiento de la celda al tiempo entre los eventos.
- **Umbral de eventos:** es el número mínimo de eventos que se han de recibir para comenzar a seguir a un objeto. Es configurable y permite eliminar en gran medida los eventos aislados que produce la retina y que no corresponden con ningún objeto en movimiento.

Por analogía y para facilitar las explicaciones, se hablará de imágenes cuando en realidad se está haciendo referencia a la matriz de píxeles de la retina. Así, se hablará de

posición en la imagen cuando se refiere a la posición relativa referida a la matriz de píxeles de la retina. Aunque conviene recordar que, como se ha explicado más arriba, el funcionamiento del sistema no se basa en el procesamiento de imágenes o fotogramas, si no en el procesamiento de secuencias de eventos.

Según lo expuesto más arriba, la celda estará posicionada en un punto determinado de la imagen, a partir del cual empezará a seguir objetos que pasen por su campo de visión. Este está definido por la posición inicial y por el ancho inicial. El ancho inicial también marca el tamaño esperado del objeto. El funcionamiento de la CMCell se describe con la máquina de estados finitos de la Figura 3.13.

La celda comienza su funcionamiento en el estado *inicio*, en el que se inicializan las señales internas, entre otras la posición que se inicializa con el valor posición inicial y el ancho que se inicializa con el valor ancho inicial. De este estado se pasa al estado de *espera*: En este estado se está a la espera de recibir un evento por el puerto AER de entrada. Una vez llega el evento se pasa a un estado en el que se discrimina el evento; de manera que si el evento recibido está fuera del campo de visión, éste es reenviado por el puerto AER de reenvío hacia la siguiente celda; en cambio, si el evento recibido esta dentro del campo de visión se incrementa el contador de eventos recibidos. Cuando la cantidad de eventos recibidos es superior al umbral de eventos, se pasa a un estado en el que se calcula la nueva posición. Por el contrario, si la cantidad de eventos recibidos es menor que el umbral de eventos, se volverá al estado de espera, a esperar un nuevo evento.

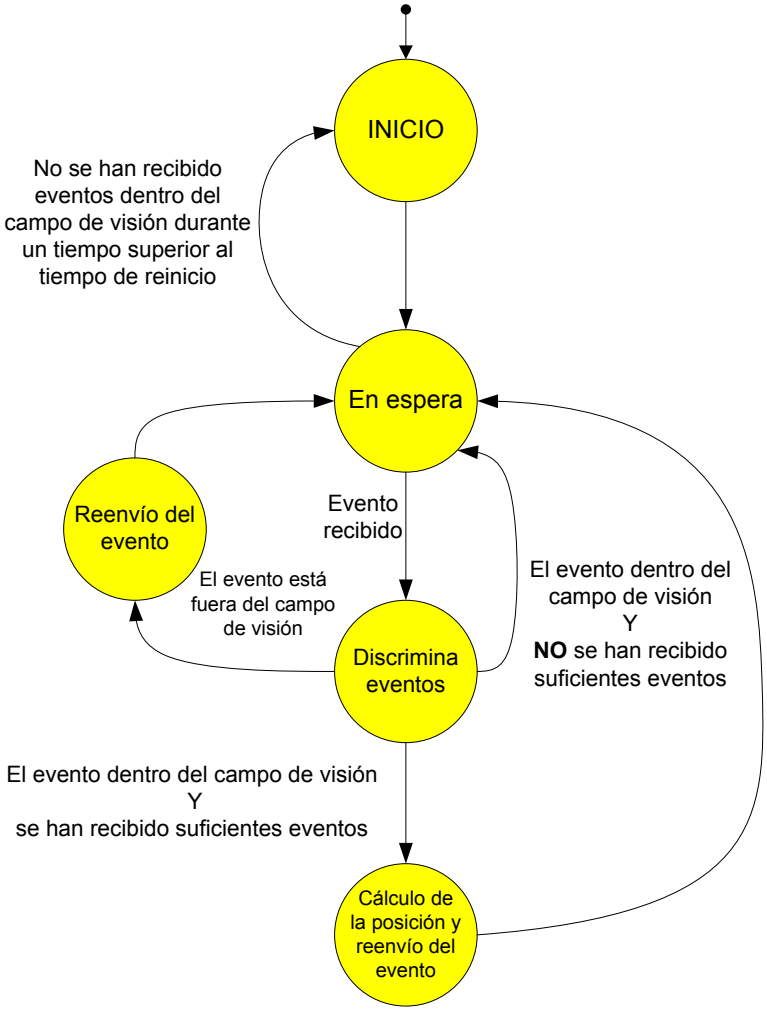


Figura 3.13: Máquina de estados de la CMCell

Cálculo de la posición y seguimiento del objeto

Tras recibir la suficiente cantidad de eventos se tienen suficientes garantías de que los eventos corresponden a un objeto y no a los eventos esporádicos que emite la retina debido a defectos de fabricación. Se consigue así un efecto de filtrado de la señal de la retina.

El propósito del sistema es obtener un punto, lo más cercano al centro de masas, que indique la posición del objeto. Para obtener ese punto se hace uso de los eventos positivos y negativos que emite la retina, esos eventos son producidos por los píxeles de la retina que

captan el movimiento del flanco delantero y trasero (según el sentido del movimiento) del objeto respectivamente. Cada vez que llega un evento positivo se calcula el punto medio entre la dirección del último evento positivo recibido y el que acaba de llegar, de manera análoga se actúa con los eventos negativos. La posición del objeto se obtiene como el punto medio entre los puntos medios positivos y negativos:

$$Pos = \frac{pm^+ + pm^-}{2}$$
$$pm^+ = \frac{\sum_{i=1}^{n} dir_i^+}{n} \quad ; \quad pm^- = \frac{\sum_{i=1}^{n} dir_i^-}{n} \tag{3.8}$$

donde dir_i^+ y dir_i^- son las direcciones de los i-ésimos eventos positivo y negativos respectivamente y n el número de eventos considerados (mayor o igual que 2).

Este cálculo se realiza cada vez que llega un evento dentro del campo de visión de la celda. Con este mecanismo la posición del objeto es actualizada cada vez que llega un evento. Consiguiendo así un procesamiento "al vuelo", uno de los objetivos principales de este trabajo.

Una vez localizado el objeto, el seguimiento se consigue desplazando el centro del campo de visión de las CMCell a la posición del objeto. En la Figura 3.14 se muestra como el campo de visión de la CMCell se desplaza desde la posición en el instante t_0 ($Pos(t_0)$) hasta la posición en el instante actual t_a ($Pos(t_a)$).

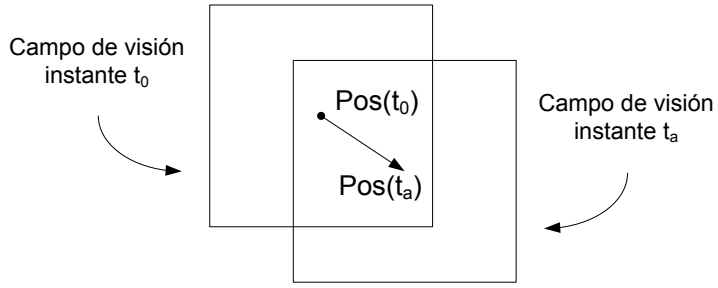


Figura 3.14: Cambio del campo de visión en función de la posición del objeto

Para localizar la posición del objeto en la imagen, es posible usar sólo los eventos de un signo, positivos o negativos, o bien, no tener en cuenta el mismo. Las pruebas realizadas, que no se muestran en este trabajo, demostraron que se obtienen mejores resultados usando el signo de los eventos de la manera descrita más arriba.

Precisión en la localización

La precisión en la localización del objeto depende de la precisión del sensor, concretamente depende de: la óptica de la retina, el tamaño del sensor, número de píxeles del mismo y la distancia al objeto. Conociendo estos parámetros es posible conocer la relación píxel/metro.

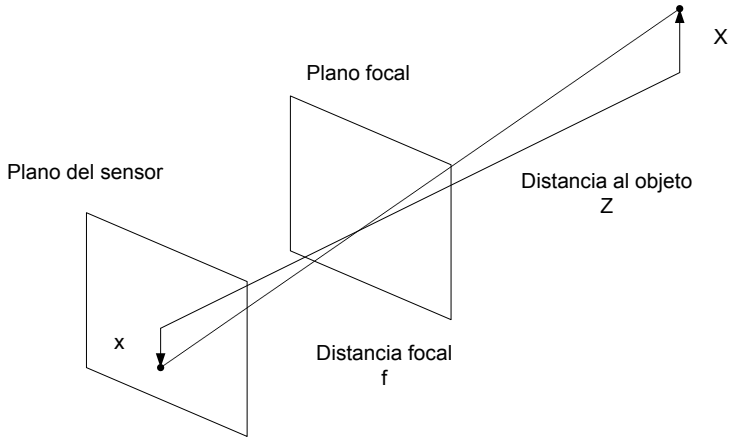


Figura 3.15: Modelo *pinhole* para la calibración de la óptica de un sensor visual

La Figura 3.15 muestra el modelo *pinhole* para la calibración de la óptica de un sensor de modo que:

$$x = X\frac{f}{Z} \tag{3.9}$$

Sabiendo el tamaño del sensor y el número de píxeles (en el caso de la retina artificial 6x6 mm y 128x128 (Patrick Lichtsteiner et al. 2008)) e introduciendo la ecuación (3.9) tenemos la expresión de calibración que nos permite obtener la precisión:

$$x_{pixeles} = \frac{número\ de\ pixeles}{tamaño\ del\ sensor}\frac{f}{Z}X \tag{3.10}$$

3.6. Estimación de la velocidad del objeto

El paso lógico después del cálculo de la posición del objeto es estimar la velocidad del mismo. El cálculo de la velocidad se realiza partiendo de la información que proporciona la CMCell, esta información es una secuencia AER de eventos que son las posiciones del objeto a lo largo del tiempo. Los trabajos publicados recientemente citados anteriormente también hacen un cálculo de la velocidad. La misma comparación hecha para el caso del cálculo de la posición es válida para la velocidad. En este sistema el cálculo de la velocidad lo realizará una celda llamada VCell.

3.6.1. La celda VCell

Como ya se ha comentado las celdas VCell serán las encargadas de estimar la velocidad del objeto. La velocidad será estimada a partir de los sucesivos valores de la posición de los objetos obtenidos por la CMCell.

Haciendo la misma analogía con la neuronas biológicas que hecha con la CMCell. La VCell tiene dos sinapsis; una de entrada conectada la CMCell por la que recibirá la posición del objeto que la CMCell esté siguiendo, y otra de salida por la que se emite la velocidad del objeto que se está siguiendo. La Figura 3.16 muestra esta conexión: la salida de la CMCell es conectada a la entrada de la VCell de manera que ésta procesa los eventos de la CMCell para estimar la posición del objeto. Ambas, entrada y salida, son de tipo AER.

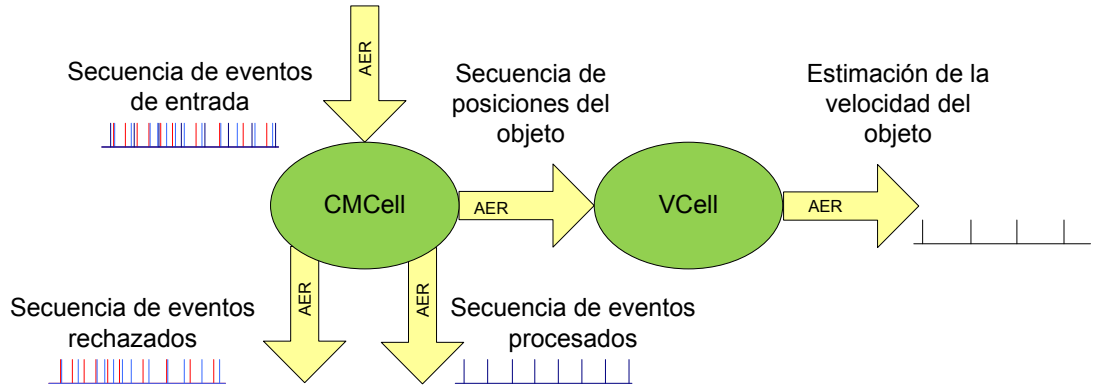


Figura 3.16: Conexión CMCell y VCell

La velocidad es una magnitud física que mide el desplazamiento de un punto durante un determinado tiempo. Por lo tanto para poder estimar la velocidad de un objeto es necesario observar su posición y dejar pasar un tiempo; pasado este tiempo volver a observar la posición; y por último estimar la velocidad como el cociente entre la diferencia de las posiciones observadas y el tiempo transcurrido entre las mismas. El funcionamiento de la VCell se basa en este mecanismo de estimación de la velocidad descrito más arriba. La máquina de estados finitos que gobierna la VCell se muestra en Figura 3.17.

La máquina de estado se inicia en el estado *inicio*$_3$ donde se inicializa el sistema; después se pasa al estado *en espera*, donde se estará hasta que se cumpla el tiempo de espera para realizar la lectura de la posición, con la que estimar la velocidad. Posteriormente se volverá a estado *en espera* y el ciclo comienza de nuevo.

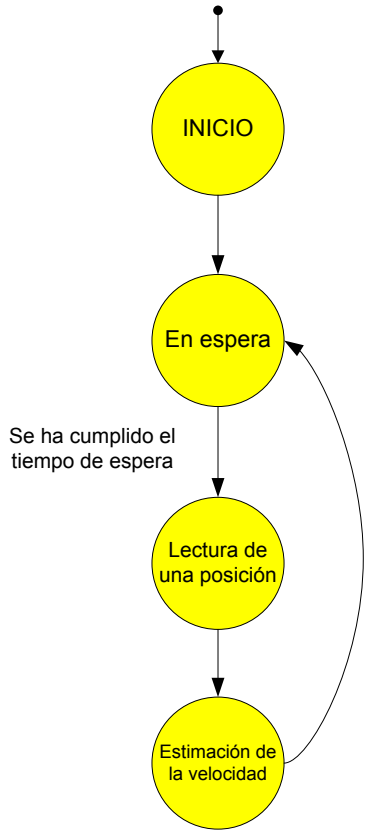


Figura 3.17: Máquina de estado de la VCell

3.7. Resumen

La determinación de la posición y la velocidad de un objeto en movimiento en una escena es uno de los grandes problemas en los sistemas de visión artificial convencional. Entendiendo por tales, los basado en sensores CCD y la computación de las imágenes capturadas por éstos en un computador. Una de las soluciones más potentes para el análisis del movimiento es la estimación del flujo óptico.

La solución que aquí aportamos difiere de estos enfoques en dos aspectos fundamentales. En primer lugar, en este trabajo se usa una retina artificial espacio-temporal que ofrece información de los cambios de contraste en la escena, codificada en AER. Y en

segundo lugar, no se usa ningún computador convencional en el núcleo de funcionamiento.

La localización del objeto en movimiento se realiza usando los eventos que emite la retina, estos eventos son de dos tipos (positivos o negativos), dependiendo del sentido del cambio del contraste. De esta manera, se genera un frente de eventos positivos en el flanco de avance del objeto y un frente de eventos negativos en el flanco trasero del mismo.

La CMCell es la encargada de obtener la posición del objeto. Esta celda concentra su procesado en una región de la imagen AER (por analogía y para facilitar la comprensión del mecanismo se usa la palabra imagen, pero en ningún momento se realiza una integración a modo de fotograma, si no que el sistema funciona procesando eventos tan pronto como llegan). Esta región está caracterizada por una posición y un ancho. Los eventos recibidos fuera de esa región serán reenviados por un puerto AER de salida. En cambio, los eventos que se encuentren dentro de esa región, serán usados para el cálculo de la posición. La posición se obtiene como el valor medio del punto medio de los n últimos eventos positivos y el punto medio de los n negativos.

Para la obtención de la velocidad, la VCell calcula la diferencia entre dos posiciones del objeto separadas un determinado tiempo.

"La verdad es demasiado complicada como para permitir nada más allá de meras aproximaciones".

John von Neumann

Capítulo 4

Reconocimiento de objetos

El reconocimiento de objetos es otro de los grandes problemas en los sistemas de visión artificial. Por reconocimiento de objetos se entiende la capacidad para reconocer formas y en un sentido amplio también texturas.

En este capítulo, se pretende resolver el problema del reconocimiento de objetos de forma neuromórfica. Como se he explicado en el capítulo 2, se tiene la intuición de que las neuronas de la capa V1 de la corteza visual de los seres vivos se hacen sensibles, durante las etapas iniciales del aprendizaje, a determinados patrones visuales. Es decir, que las neuronas de los sistemas biológicos se especializan en "detectar" determinadas características de señales visuales procedentes de las retinas (Ng et al. 2007). Hay suficientes evidencias que permiten pensar que en la capa V1 existen neuronas que se especializan, entre otras labores, en detectar la orientación de los estímulos visuales; es decir se activan al recibir un estímulo visual consistente en líneas verticales, horizontales o diagonales. Precisamente es éste el comportamiento que se pretende imitar. La solución que aquí se presenta se comporta como ciertas neuronas de la capa V1, de manera que es capaz de detectar una determinada orientación en el estímulo visual.

La solución aportada presenta algunas similitudes y algunas diferencias con dos conceptos básicos muy relacionados con el reconocimiento de objetos como son la correspondencia de patrones (o *pattern matching* en inglés) y las redes de neuronas basadas en el perceptrón aplicadas a reconocimiento de patrones.

La inspiración en la biología de esta solución se materializa en el uso, con algunas diferencias, del modelo neuronal "Integrate and Fire".

Veamos una breve introducción a estos tres conceptos.

4.1. Correspondencia de patrones

La correspondencia de patrones (Duda et al. 2001) tiene como objetivo clasificar los objetos presentes en una imagen en una serie de categorías predefinidas. Los dos enfoques más importantes para el reconocimiento de objetos son los basados en datos estadísticos de la imagen, cuyos patrones son descripción cualitativa de la realidad (como por ejemplo el área, la longitud o la textura); o los basadas en la estructura de la imagen, donde los patrones constituyen una descripción cuantitativa (relaciones entre píxeles). A continuación, veremos el primero de ellos ya que muestra algunas semejanzas con la aportación que aquí se presenta.

En primer lugar conviene definir que es un patrón; un patrón, en el ámbito del tratamiento digital de imágenes, es un conjunto de descriptores, que resumen las características de una imagen. Un patrón de clase es una familia de patrones que comparten algunas propiedades, y se denota como $\omega_1, \omega_2, \dots, \omega_W$, donde W es el número de clases. Como ya se ha apuntado, el reconocimiento de patrones consiste en asignar los patrones a su correspondiente clase (Gonzalez & Woods 2002).

Por norma general, los patrones son representados por un vector, donde cada elemento del mismo representa un descriptor.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{4.1}$$

donde $\boldsymbol{x}$ es un patrón, $\mathrm{x_i}$ es el descriptor i-ésimo y n el número total de descriptores en el patrón.

La naturaleza de cada componente del vector $\boldsymbol{x}$ dependerá de la aplicación y la técnica concreta que se use para describir el mundo físico. Así pues, las componentes pueden ser dimensiones, intensidad luminosa, colores, textura, etc. Por lo tanto, la selección de las características que formarán parte del patrón se convierte en una labor fundamental para el éxito del reconocimiento de objetos.

Los métodos basados en la estadística de la imagen se apoyan en el uso de funciones de decisión o discriminación. El problema de estos métodos es: dadas $\mathcal{W}$ clases, $w_1, w_2, \dots, w_{\mathcal{W}}$, encontrar funciones $d_1(x), d_2(x), \dots, d_W(x)$ donde $\boldsymbol{x}$ es un patrón que pertenece a la clase ω_i, entonces:

$$d_i(x) > d_j(x) \qquad j = 1,2, \dots, \mathcal{W}; \quad j \neq i \tag{4.2}$$

La principal dificultad estriba en encontrar las funciones de decisión que validen la ecuación (4.2), permitiendo que la clasificación de los patrones sea unívoca. Veamos la técnica más sencilla: la correspondencia (o *matching)* que forma parte del fundamento del trabajo que aquí se presenta.

En las técnicas basadas en la correspondencia de patrones cada clase es representada por un patrón prototipo. Un patrón desconocido es asignado a la clase que dista menos, usando una métrica preestablecida. Lo más sencillo es calcular la distancia Euclídea entre el patrón desconocido y cada uno de los patrones prototipos. De manera que el patrón será

incluido en la clase cuyo distancia al patrón prototipo sea menor. Este es llamado el clasificador de mínima distancia.

Clasificador de mínima distancia

Supongamos que se define el prototipo de cada patrón de clase como la media vectorial de los patrones de una clase:

$$\boldsymbol{m}_j = \frac{1}{N_j} \sum_{x \in \omega_j} \boldsymbol{x}_j \qquad j = 1,2,\dots,\mathcal{W} \tag{4.3}$$

Donde $N_j$es el número de vectores de patrón de la clase ω_j sobre los que se calcula el sumatorio; $\mathcal{W}$ sigue siendo el número clases y por lo tanto el número de patrones de clase. Una manera sencilla de determinar la clase a la que pertenece un patrón desconocido $\boldsymbol{x}$ es asignarlo a la clase de cuyo prototipo esté más cerca. Usando la distancia Euclídea como función de proximidad el problema se reduce a computar:

$$D_j(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{m}_j\| \qquad j = 1,2,\dots,\mathcal{W} \tag{4.4}$$

Donde $\|\boldsymbol{a}\| = (\boldsymbol{a}^T\boldsymbol{a})^{1/2}$ (^{T} indica trasposición) es la norma Euclídea. Se asignará $\boldsymbol{x}$ a la clase ω_j si $D_j(\boldsymbol{x})$ es la menor distancia. Esto es, la menor distancia implica la mejor correspondencia para esta formulación.

El clasificador de mínima distancia funciona bien cuando la distancia entre las medias es grande comparada con la aleatoriedad de cada clase con respecto a esas medias.

4.2. Redes de neuronas basados en el perceptrón

El perceptrón es un modelo matemático para la simulación por ordenador de redes de neuronas. Tiene la cualidad de que puede ser entrenado para discriminar patrones en dos clases linealmente separables. La Figura 4.1 muestra el modelo del perceptrón para dos clases de patrones.

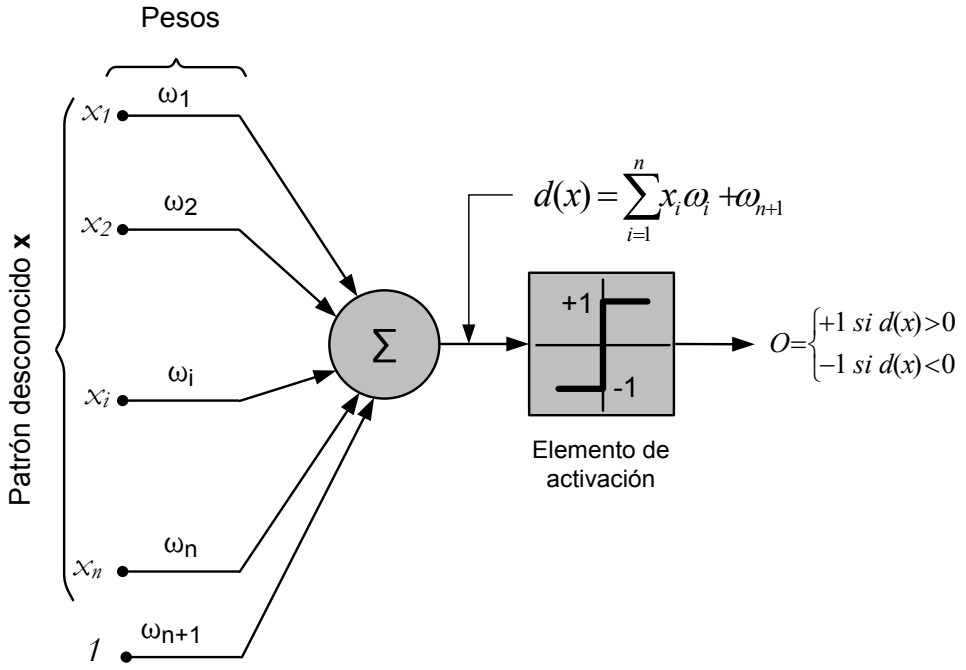


Figura 4.1: Esquema del perceptrón

El funcionamiento del perceptrón se basa en la suma ponderada de sus entradas, es decir:

$$d(\boldsymbol{x}) = \sum_{i=1}^{n} \omega_i x_i + \omega_{n+1} \tag{4.5}$$

La cual es una función de decisión lineal con respecto a las componentes del vector del patrón. Los coeficientes $\omega_i,\ i = 1,2,\dots,n,n+1$, llamados pesos, modifican las entradas

antes de ser sumadas y alimentar el elemento de umbral. Los pesos realizan el mismo papel que las sinapsis en los sistemas neuronales biológicos. La función que convierte la suma ponderada de las entradas en la salida final del perceptrón se llama función de activación.

Cuando $\boldsymbol{d(x) > 0}$, la salida del perceptrón será +1 indicando que el patrón $\boldsymbol{x}$ es reconocido como perteneciente a la clase ω_1. Por el contrario, cuando $\boldsymbol{d(x) < 0}$, indica que el patrón $\boldsymbol{x}$ no pertenece a dicha clase.

Esta formulación presenta algunos problemas cuando $\boldsymbol{d(x) = 0}$. La discusión sobre este problema puede encontrase en el capítulo 12 de (Gonzalez & Woods 2002). Como solución a este problema se propone el uso de un patrón aumentado $\boldsymbol{y}$ que es igual al $\boldsymbol{x}$ haciendo $y_i = x_i, i = 1,2, \dots, n$ al que se le añada un elemento $y_{n+1} = 1$ un 1 al final. Por lo que la ecuación (4.5) quedaría:

$$d(\boldsymbol{y}) = \sum_{i=1}^{n+1} \omega_i y_i = w^T \boldsymbol{y} \tag{4.6}$$

donde $\boldsymbol{y} = (y_1, y_2, \dots, y_n, 1)^T$es el vector de patrón aumentado, y $\boldsymbol{w} = (\omega_1, \omega_2, \dots, \omega_n, \omega_{n+1})^T$es el vector de pesos. El problema ahora consiste en encontrar un $\boldsymbol{w}$, tal que dado un conjunto de patrones de entrenamiento haga que el perceptrón sea capaz de clasificar un patrón desconocido.

Aprendizaje

El aprendizaje es una de las grandes ventajas que presenta la utilización del perceptrón para el reconocimiento de objetos, ya que, gracias a él, los pesos de la suma ponderada son "autoajustados" por el propio perceptrón durante la fase de entrenamiento. Que consiste, de manera sintética, en presentar una serie de patrones conocidos, llamados patrones de entrenamiento, y "forzar" la salida al valor deseado. Hay diversos enfoques que permiten realizar de manera eficiente el entrenamiento. Veamos el caso más sencillo, cuando las clases son linealmente separables.

Sean dos conjuntos de patrones aumentados, $\mathbf{y}(\mathrm{k})$, pertenecientes a las clases ω_1 y ω_2, respectivamente. Denotemos por $\boldsymbol{w}(1)$ el valor inicial del vector de pesos, elegidos aleatoriamente, así pues en la iteración k-ésima, tendremos que si $\mathbf{y}(\mathrm{k}) \in \omega_1$ y $\mathbf{w}^{\mathrm{T}}(\mathrm{k})\mathbf{y}(\mathrm{k}) \leq 0$, $\boldsymbol{w}(\mathrm{k})$ se reemplaza por:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + c\boldsymbol{y}(k) \tag{4.7}$$

donde $\mathbf{c}$ es un incremento de corrección positivo. Por el contrario si $\mathbf{y}(\mathrm{k}) \in \omega_2$ y $\mathbf{w}^{\mathrm{T}}(\mathrm{k})\mathbf{y}(\mathrm{k}) \geq 0,$ se reemplaza por:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - c\boldsymbol{y}(k) \tag{4.8}$$

En otro caso $\mathbf{w}(\mathrm{k})$ permanecerá inalterado:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) \tag{4.9}$$

Este algoritmo solo cambia el valor de $\boldsymbol{w}$ en el caso de que el patrón considerado en la iteración k-ésima sea mal clasificado. La corrección asume un incremento positivo y constante $\mathbf{c}$, por lo que el algoritmo es conocido como regla de corrección de incremento constante.

La convergencia del algoritmo ocurre cuando el conjunto completo de patrones de entrenamiento es presentado en el sistema y no se produce una mala clasificación para ningún patrón de entrenamiento. En el caso concreto de la regla de corrección de incremento constante se puede demostrar que converge en un número finito de pasos, cuando el conjunto de patrones de entrenamientos son linealmente separables. Esto constituye el teorema de entrenamiento del perceptrón (Duda et al. 2001).

Red neuronal multicapa *feedforward*

Cómo es lógico pensar, para la clasificación de objetos no se usa el perceptrón de forma aislada, sino que es usado como elemento de computación dentro de una red conocida como red de *feedforward,* también llamada como red retro-propagación (del inglés *back propagation*) tomando el nombre del algoritmo de entrenamiento más usado en este tipo de redes. La Figura 4.2 muestra la arquitectura básica de la red.

Consta de capas de neuronas estructuralmente idénticas y colocadas de forma que las salidas de las neuronas de una capa son las entradas de la capa siguiente. El número de neuronas de la primera capa, llamada A, es N_A. Siendo $N_A = n$, donde n es la dimensión del vector de patrón de entrada. El número de neuronas de la última capa, llamada Q, es N_Q, siendo $N_Q = W$, donde W es el número de clases que una red ya entrenada puede reconocer. Un patrón $\boldsymbol{x}$ es clasificado en la clase ω_i si la salida i-ésima de la red esta a valor "alto" y el resto a valor "bajo", tal y como se explica más abajo.

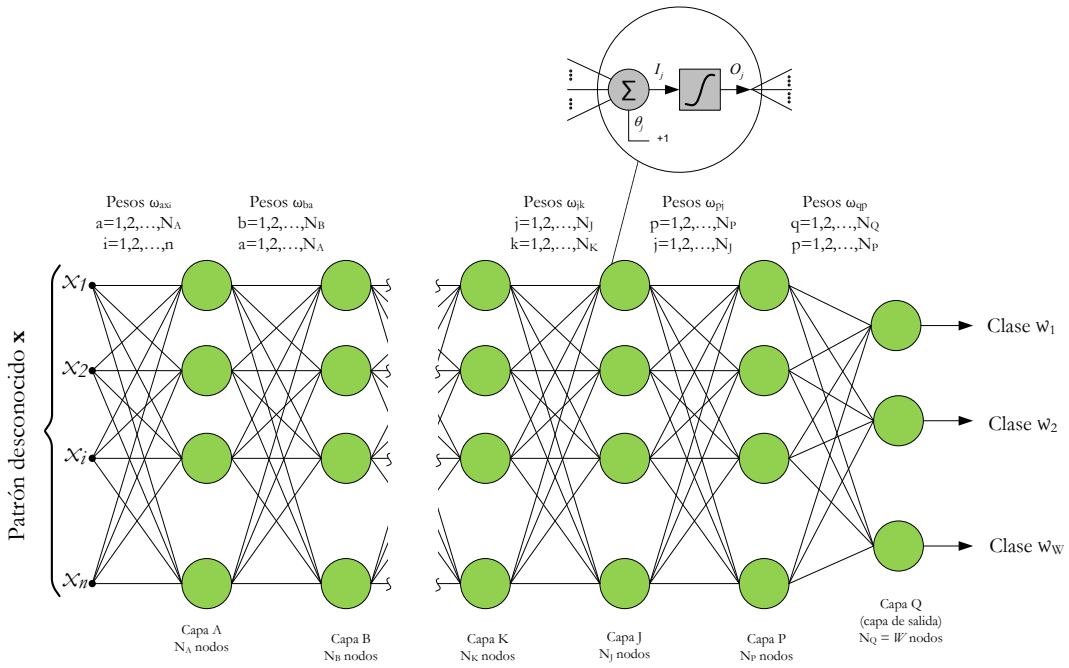


Figura 4.2: Arquitectura de la red retro-propagación

Como se muestra en el globo de la Figura 4.2 las neuronas son perceptrones idénticos a los presentados más arriba, excepto por la función de activación, que aquí es una función sigmoidea y en el modelo original es una función escalón. Esta función (la sigmoidea) es derivable, requisito indispensable para aplicar el algoritmo de aprendizaje *back propagation*. La función sigmoidea tiene la siguiente forma matemática:

$$h_j(I_j) = \frac{1}{1 + e^{-(I_j+\theta_j)/\theta_0}} \tag{4.10}$$

donde $I_j, j = 1,2, \ldots, N_j$, es la entrada de cada elemento de activación de la capa J de la red, θ_j es un parámetro de ajuste y θ_o controla la forma de la función sigmoidal. La Figura 4.3 muestra gráficamente la función sigmoidea, donde se pueden ver los valores "alto" y "bajo" de la respuesta y los parámetros de la función. Así pues la función sigmoidea proporciona un valor "alto" cuando el valor de I_j es mayor que θ_j , y un valor "bajo" en el caso contrario. Obsérvese que la función sigmoidal siempre es positiva.

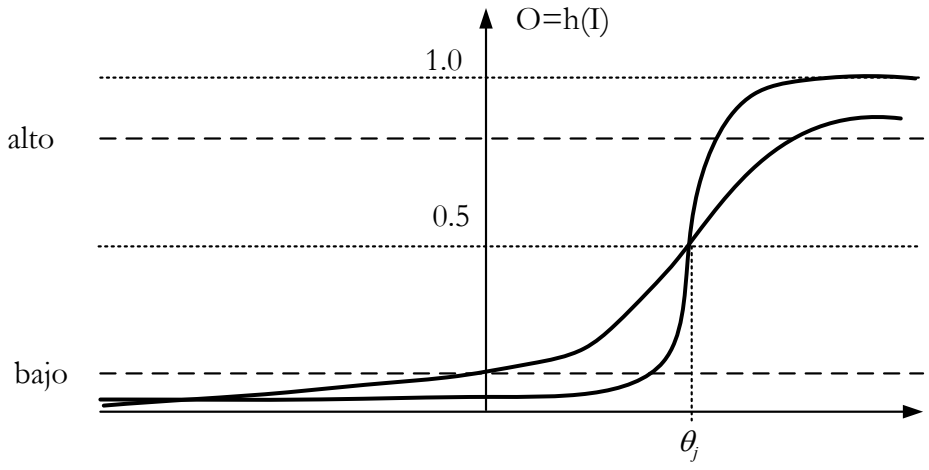


Figura 4.3: Función de activación sigmoidea

Se podrían utilizar diferentes tipos de funciones de activación para las diferentes capas o incluso para diferentes nodos de la misma capa de la red. No obstante, la práctica habitual es utilizar el mismo tipo de función para todas las neuronas de la red.

Por último y sin entrar en detalles, ya que son irrelevantes para el presente trabajo (y que pueden ser encontrados en (Gonzalez & Woods 2002)), el algoritmo de entrenamiento conocido como entrenamiento por retro-propagación (o *training by back propagation*) consiste en aplicar una regla de entrenamiento que permita el ajuste de los pesos en cada una de las capas, buscando un mínimo de la función de error:

$$E_{iQ} = \frac{1}{2}\sum_{i=1}^{N_i}(r_i - O_i)^2 \tag{4.11}$$

donde r_i son las respuestas deseadas, O_i las correspondientes respuestas actuales y N_i es el número de nodos en la capa i.

El ajuste de los pesos, minimizando la función de error, se consigue ajustando los pesos en proporción a la derivada parcial del error respecto a los pesos.

$$\Delta w_{ip} = -\alpha \frac{\partial E_Q}{\partial w_{iq}} \tag{4.12}$$

donde *P* precede a la capa *i* y α es el incremento de corrección constante.

4.3. Modelo *Integrate and Fire*

El modelo neuronal *Integrate and Fire*[23] es un modelo de neurona pulsante y como tal supone que tanto la información que recibe como la que transmite es codificada en pulso o en frecuencia de pulsos. Las propiedades del modelo neuronal *Integrate and Fire* vienen siendo investigadas desde 1907, antes incluso de que se desarrollaran los sistemas de redes de neuronas pulsantes. Este modelo se ha convertido en la aproximación estándar al

[23] Cuya traducción es integra y dispara, donde disparar significa generar un pulso

complejo comportamiento que muestran las neuronas biológicas. Las razones por las que es tan ampliamente usado son (Chris Eliasmith 2003):

1. Fue el primer modelo que ofreció una aproximación válida para una gran variedad de neuronas y en un mayor rango de funcionamiento.
2. Es un caso particular (aunque limitado) del más complejo y bien conocido modelo de Hodgkin-Huxley.
3. Es con diferencia un modelo más realista que las neuronas artificiales "clásicas" (como por ejemplo el perceptrón), ya que contempla la no-linealidad de los pulsos que se observan en las neuronas biológicas.

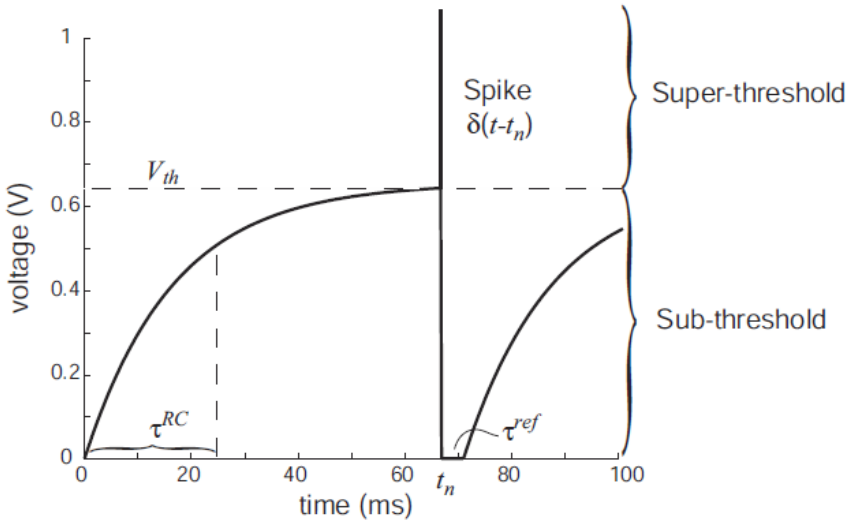


Figura 4.4: Funcionamiento del modelo neuronal "Integrate and Fire" (figura obtenida de (Chris Eliasmith 2003))

El modelo *Integrate and Fire* contempla dos regímenes de funcionamiento: por debajo de umbral y por encima de umbral. En la Figura 4.4 se muestra el comportamiento por debajo de umbral. Ante una llegada continua de pulsos, estos elevan poco a poco el potencial; una vez alcanzado el umbral (V_{th}), conmuta al comportamiento por encima de umbral donde se genera un pulso de delta ($\delta(t\text{-}t_n)$) con área 1(es decir, $\int_{-\infty}^{\infty} \delta(t - t_n)dt = 1$).

Tras el pulso el sistema pasa a la posición cero, donde permanece un tiempo (τ^{ref}) antes de volver al comportamiento por debajo de umbral.

Hay un gran número de características psicobiológicas en este modelo. Primero, el pulso generado es muy parecido al que se observa en la biología, aunque la representación como un pulso delta es sólo un artificio matemático. El pulso generado por el modelo suele ser, típicamente, de 1 a 2 ms de anchura, que es bastante estrecho comparado con el tiempo entre eventos, de 50 a 100ms. Segundo, el tiempo que el modelo permanece en la posición cero después del pulso, llamado periodo reposo, es un buen modelo para un fenómeno que se observa en la neuronas biológicas (Kock 1999). Y tercero, el modelo puede ser extendido a un modelo con pérdidas, modelando la función de "olvido" presente en las neuronas biológicas.

Hay 3 variantes del modelo *Integrate and Fire*, mostradas en la Figura 4.5. Todas tienen dos partes, una de ellas modela el comportamiento por debajo de umbral y la otra es la encargada de la generación del pulso una vez alcanzado el umbral (V_{th}). Una vez que el potencial $V(t)$ sobrepasa el umbral y se genera el pulso, la parte que modela el comportamiento bajo umbral es corto-circuitada durante el periodo reposo.

La primera variante de neurona *Integrate and Fire* es la *perfect or non-leaky Integrate and Fire*, donde la parte correspondiente al comportamiento por debajo de umbral es modelada con un condensador.

La segunda variante conocida como *Leaky Integrate and Fire (LIF)*[24], modela la función de "olvido" de la neuronas biológicas añadiendo una resistencia, la cual provocará una descarga del condensador. Este es el modelo más comúnmente usado.

La tercera variante A*dapting Integrate and Fire*[25], permite modelar la amplitud del pulso añadiendo una resistencia variable al sistema.

---

[24] Cuya traducción es integra y dispara con pérdidas

[25] Cuya traducción es integra y dispara con adaptación.

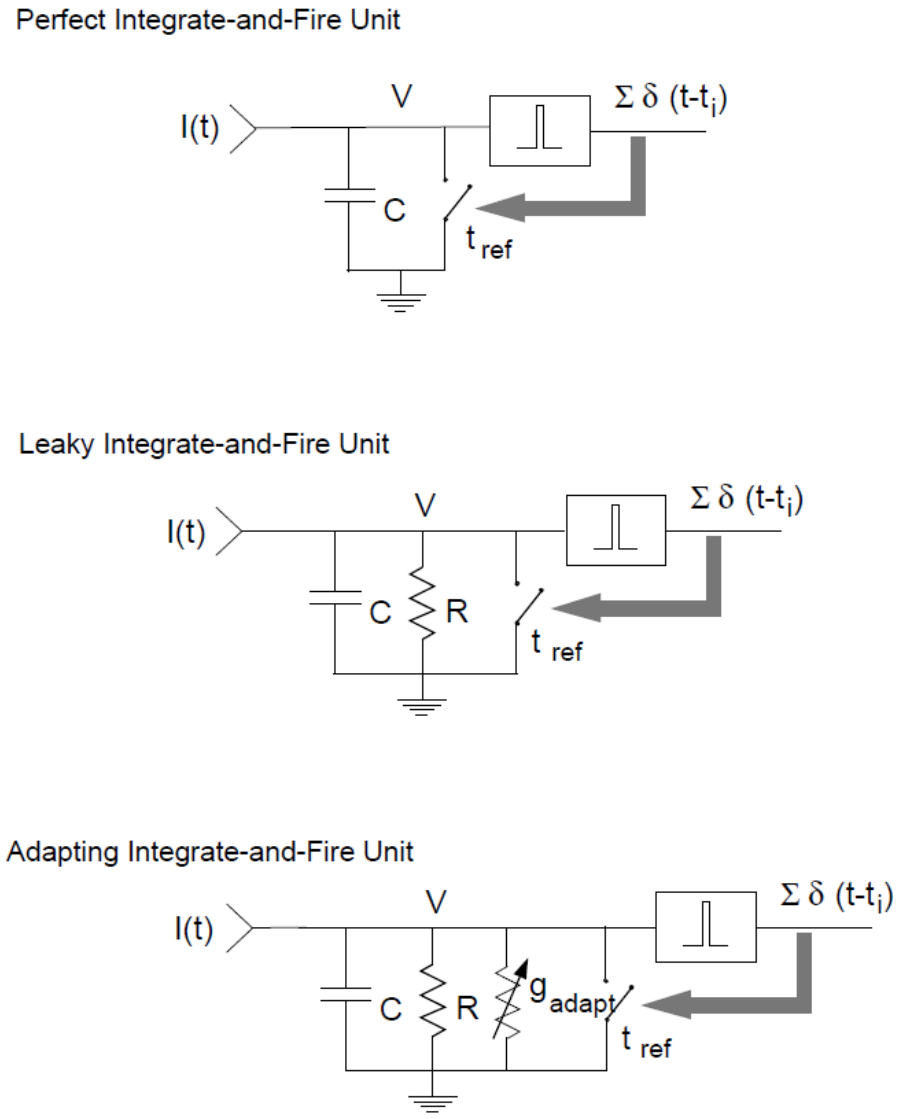


Figura 4.5: Variantes del modelo neuronal "*Integrate and Fire*"
(figura obtenida de (Kock 1999))

.

4.4. Modelo *Pattern Integrate and Fire*

Como ya se ha comentado al principio de este capítulo, se pretende dar una solución neuromórfica al problema de la detección de objetos. La solución se ha denominado *Pattern Integrate and Fire* este nuevo modelo de neurona pulsante está basado en el modelo *Integrate and Fire* y en el perceptrón.

La Figura 4.6 muestra el esquema de este nuevo modelo de neurona, que como se puede apreciar incorpora elementos de los modelos anteriormente mencionados.

La *Pattern Integrate and Fire* producirá un pulso cuando el patrón desconocido de la entrada $\boldsymbol{x}$ coincida, con cierto grado de similitud, con el patrón deseado.

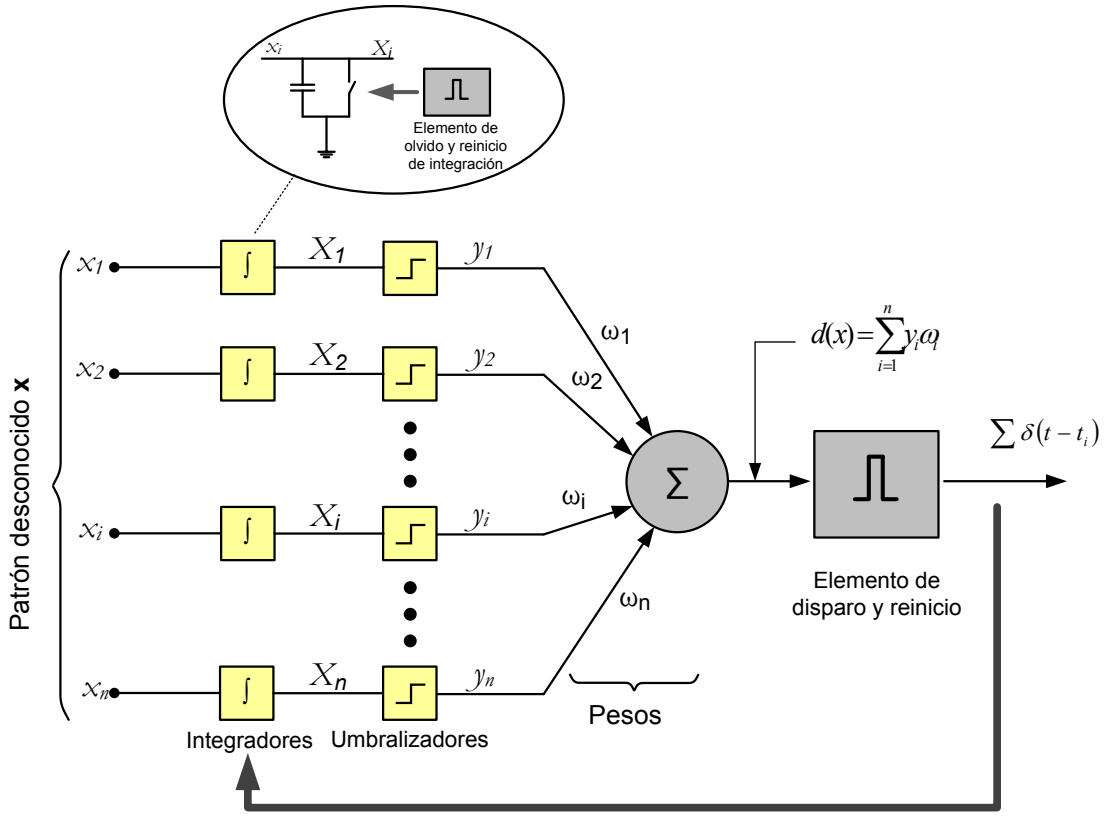


Figura 4.6: Modelo *Pattern Integrate and Fire*

La primera consideración que hay que tener en cuenta es que las componentes del vector de patrón desconocido $\boldsymbol{x} = (x_1, x_2, \dots x_n)^T$, siendo n el número de componentes del mismo, son pulsos, a diferencia que en el perceptrón pero tal y como ocurría en el modelo *Integrate and Fire*. Estos pulsos son integrados en el tiempo de manera que:

$$X_i = \int_{t_0}^{t} x_i dt \tag{4.13}$$

esta integral, de manera ideal, corresponde con la cantidad de pulsos recibidos entre los instantes t_0 y t. El elemento de olvido y reinicio de integración generará un pulso que cada cierto tiempo reiniciará la integración de eventos.

El resultado de la integración de los pulsos de cada componente del vector del patrón desconocido, X_i, es la entrada de un bloque de umbral. De manera que y_i valdrá 0 mientras que X_i esté por debajo de un umbral llamado Umbral de disparo de las entradas:

$$y_i = \begin{cases} 0 \; si \; X_i < U_{Entradas} \\ 1 \; si \; X_i \geq U_{Entradas} \end{cases} \tag{4.14}$$

La umbralización de las entradas y_i constituye la base de la función de decisión $d(x)$:

$$d(\boldsymbol{x}) = \sum_{i=1}^{n} y_i \omega_i \tag{4.15}$$

donde ω_i son los pesos de cada una de la componentes del vector de patrón en el sumatorio final que podrán ser positivos o negativos según se desee que la entrada incremente el potencia o lo disminuya; que alimenta el elemento de disparo y reinicio el cual emitirá un pulso cuando $d(\boldsymbol{x})$ supere un predeterminado umbral, Umbral de disparo. Este elemento de activación producirá un pulso delta, $\delta(t - t_p)$ similar al que presenta el modelo *Integrate and Fire*; además este elemento, tras el pulso, reiniciará los integradores; lo que significará poner en el caso de integradores de pulsos ideales la cuenta a 0. Esto

integradores serán implementados como contadores digitales (véase el capítulo 6 para más detalles sobre la implementación hardware del sistema).

Como puede observarse este nuevo modelo de neurona pulsante presenta similitudes y diferencias con los modelos analizados, que la hacen especialmente útil para la detección de patrones.

En cuanto a las similitudes con el modelo *Integrate and Fire*, encontramos que en ambos casos las entradas y la salida son pulsos; el elemento de disparo y reinicio es idéntico en ambos casos. Esta similitud del nuevo modelo con el *Integrate and Fire* permiten trasladar al modelo *Pattern Integrate and Fire* todas las cualidades del primero. La diferencia fundamental con el modelo *Integrate and Fire* es la forma en la que se acumulan o integran los pulsos: en el modelo original un solo condensador acumula todos los pulsos y cada pulso incrementa en igual cuantía el potencial de la neurona; en el modelo que aquí se presenta, se puede, por una parte, diferenciar varias fuentes de pulsos y por otra se añade la posibilidad de que cada fuente puede variar el potencial de manera diferente, no solo incrementándolo sino disminuyéndolo también, esta funcionalidad se consigue ajustando los pesos, ω_i, con valores negativos.

En cuanto a las similitudes con el perceptrón, encontramos que la función de decisión $d(\boldsymbol{x})$ es idéntica en ambas neuronas y que por lo tanto, si el perceptrón presenta cualidades que permiten su uso para la detección de patrones, este nuevo modelo hereda estas cualidades también.

La principal diferencia la encontramos en las señales sobre las que aplica el sumatorio. En el perceptrón son valores correspondientes a potenciales de las neuronas conectadas en la entrada (en el caso del uso de la función sigmoidea, que el caso más común) y en el nuevo modelo es un valor binario (0 y 1). Aunque esto no presenta ningún problema puesto que el perceptrón fue propuesto, en sus orígenes, como un modelo que sólo podía tomar dos valores, y por lo tanto si se construye una red con preceptrónes binarios, las entradas de las neuronas de las capas internas serán necesariamente binarias.

4.4.1. Utilización del modelo *Pattern Integrate and Fire* con pesos fijos

Como ya se ha explicado más arriba una de las cualidades más importantes del perceptrón es que puede ser entrenado; de manera que no es necesario obtener un patrón tipo para las clases, si no que basta con presentar algunos patrones de entrenamiento e ir corrigiendo los pesos de la suma ponderada para adaptar el funcionamiento de la neurona a la clasificación deseada.

El modelo propuesto deriva en cierta medida del perceptrón, y presenta también la cualidad de ser entrenado. Con la única diferencia que en este modelo hay dos conjuntos de parámetros que pueden ser ajustados: el umbral de disparo de las entradas y los pesos. Pero teniendo en cuenta la naturaleza pulsante del modelo parecen más apropiados los métodos de aprendizaje para este tipo de neuronas.

En 1949 Donald O. Hebb, postuló los fundamentos de la plasticidad sináptica[26]; sobre estos se han desarrollado mecanismos de aprendizaje (Hélène 2006) y entre ellos el que hoy en día se presenta como el más atractivo desde el punto de vista de la implementación artificial, el STDP[27] (Kempter et al. 1999) (van Rossum et al. 2000) (Senn 2002), según la cual una conexión sináptica es potenciada si ocurren pulsos pre-sinápticos un tiempo antes (generalmente decenas de ms) del pulso post-sináptico; si los pulsos ocurren después esa conexión será debilitada.

Basados en este principio se han desarrollado algoritmos de aprendizaje (Strain et al. 2006) (Yang et al. 2006) incluso con implementaciones en hardware (Indiveri et al. 2006) (Ponulak & Kasiński 2010). Aunque por ahora no está demostrada la convergencia de estos mecanismos para un conjunto de clases linealmente separables (Legenstein et al. 2005).

---

[26] La plasticidad sináptica describe, en la neurobiología moderna, la capacidad de adaptación del sistema neuronal biológico.

[27] Siglas en ingles de Spike-Timing Dependant Plasticity cuya traducción será plasticidad dependiente del tiempo de los pulsos.

El aprendizaje en el modelo *Pattern Integrate and Fire* consistirá en alterar los umbrales de las entradas y los pesos dependiendo del patrón temporal de los pulsos, que se reciben y de la respuesta deseada de la neurona.

Debido a las dificultades del aprendizaje expuestas más arriba, y sobre todo en su implementación en hardware, a continuación se presenta un caso particular que permite la detección de patrones, con umbrales y pesos fijos.

Para entender este caso particular hay que tener presente que el campo de aplicación es la información visual, que proporciona una retina artificial como la descrita en el capítulo anterior. La información visual consiste en una secuencia de pulsos que codifican los cambios en la intensidad luminosa debido al movimiento de los objetos de la escena que observa.

Así pues, cada componente del patrón desconocido $\boldsymbol{x}$ es un píxel de la retina, se pretende detectar si la información que la retina ofrece contiene un determinado patrón espacial, como por ejemplo líneas verticales u horizontales; figuras simples como cuadrados, triángulos, etc.; o en un sentido más amplio (y complejo) letras o números.

Al hacer los pesos con valor fijo y predefinido, éstos se convierten en un patrón tipo de la clase que se pretende detectar; este patrón tipo de la clase viene descrito en términos binarios, donde un 1 indicará que en el píxel correspondiente se espera que haya pulsos, que llamaremos **píxeles activos**, y un -1 en caso contrario:

$$\omega_i = \begin{cases} 1 \; si \; en \; x_i se \; esperan \; pulsos \\ -1 \; si \; en \; x_i \; \boldsymbol{no} \; se \; esperan \; pulsos \end{cases} \tag{4.16}$$

Con estos valores el potencial de la neuronal aumentará (caso de $\omega_i = 1$) o disminuirá en caso contrario.

Veamos un ejemplo. Supóngase un patrón de entrada de tamaño 16, que por claridad lo representaremos como una matriz de 4x4. Se desea detectar la existencia de una línea vertical en el centro de ese patrón de tamaño dos píxeles, véase la Figura 4.7.

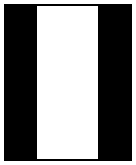


Figura 4.7: Patrón visual: línea vertical

Para este ejemplo el vector de pesos, representado en forma de matriz, es:

$$\begin{Bmatrix} -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \end{Bmatrix} \tag{4.17}$$

El valor de los umbrales de disparo de entrada será también fijo y en principio igual para todas las componentes de vector de entrada, aunque podría ser distinto para los píxeles en los que se espera que hayan eventos y en los que no (nótese que no se hace uso del signo de esos eventos).

A la vista de esta simplificación cada uno de los sumandos del sumatorio de la función de decisión (4.15) tomará solo 3 valores:

$$y_i\omega_i = \begin{cases} -1 \;\; si \;\; y_i = 1 \;\; Y \;\; \omega_i = -1 \\ 0 \;\; si \;\; y_i = 0 \\ 1 \;\; si \;\; y_i = 1 \;\; Y \;\; \omega_i = 1 \end{cases} \tag{4.18}$$

Se puede comprobar fácilmente que, con estos valores de los sumandos, la función de decisión $\boldsymbol{d(x)}$ tomará como valor máximo el número de píxeles activos en el patrón. Este máximo se alcanzará cuando el patrón de entrada coincida exactamente con el patrón tipo de la clase.

4.5. Detección de patrones

La celda PRCell, cuya descripción funcional se presentará más adelante, se basa en el modelo *Pattern Integrate and Fire* con pesos fijos y trata de realizar la detección de patrones. Antes de describir en detalle el funcionamiento de la PRCell es conveniente destacar dos aspectos de la PRCell que la diferencian del modelo. Primero, la PRCell es un circuito digital, y por lo tanto bastante más complicado que los expuestos más arriba. Segundo, los pulsos que recibe no son "simples" impulsos eléctricos sino que son eventos AER.

Veamos en primer lugar como son los patrones que se pretenden detectar y como es la señal de entrada a la PRCell.

Patrones

Los patrones vendrán definidos en una matriz de NxM donde un 0 indicará que no debe haber eventos en esa posición y un 1 que si debería haber eventos. A modo de ejemplo, sea una matriz de 8x8 como la siguiente:

$$\begin{Bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{Bmatrix} \tag{4.19}$$

Correspondería con un estímulo visual que contendría una línea vertical.

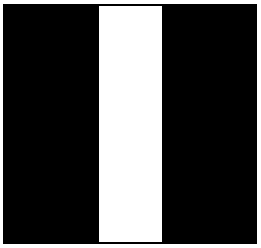


Figura 4.8: Estímulo visual consistente en una línea vertical

Cada una de las 64 posiciones del patrón corresponderá a un punto en el campo proyectivo en la entrada, es decir, en el caso de que la PRCell estuviera directamente conectada con la retina, cada posición correspondería con un píxel de la misma.

Como este esquema se podría detectar cualquier patrón que pudiera ser representado en una matriz de NxM posiciones. Sólo hay que tener en consideración cómo es el estímulo visual que corresponde a ese patrón. Hay que tener en cuenta que la retina, usada en este trabajo, presenta un cierto ruido de fondo (debido a discrepancias en el proceso de fabricación: *missmatch)* y que por lo tanto la respuesta de la misma no es "limpia", entendiendo por limpia la salida que se ajusta perfectamente al estímulo real. Además conviene no olvidar que la retina sólo es capaz de ver los cambios de intensidad luminosa en el tiempo, o dicho de otra manera, sólo es capaz de ver el movimiento.

Para confeccionar el patrón hay que tener en cuenta como es el estímulo que vemos, y como lo "verá" la retina. A continuación se presenta algunos ejemplos en los que se muestran los estímulos reales y como los "ve" la retina, que ayudan a ilustrar las dificultades a la hora de confeccionar buenos patrones. Para cada tipo de estímulo se presentan dos capturas: una en la que se mueven los objetos y otra en la que se hace temblar la retina, mientras que los objetos permanecen inmóviles.

A priori, la trayectoria de los objetos o el hecho de que estén inmóviles y sea la retina la que se mueve, parecen no tener importancia, pero debido a la naturaleza de la retina es más significativo de lo que pueda parecer.

La técnica de mover la retina para obtener movimiento en la escena, está presente en la biología, los ojos de los seres vivos están continuamente en movimiento. Estos movimientos son muy pequeños, involuntarios, aleatorios y, a veces casi, imperceptibles, tanto para el individuo como para un observador. Además hay estudios que demuestran que existe una correlación entre estos movimientos y los pulsos de las neuronas en la capa V1 de la corteza visual (Martinez-conde et al. 2000) y (Engbert 2003).

A continuación se presenta la reconstrucción de la salida de la retina y los patrones que permitirían el reconocimiento de los objetos en la escena. El método para realizar la reconstrucción de la salida de la retina es el mismo, que ya hemos descrito en el capítulo 3.

RETINA INMÓVIL Y OBJETOS EN MOVIMIENTO

En primer lugar, la Figura 4.9 muestra la reconstrucción de la salida de la retina ante estímulos muy sencillos (con un tiempo de integración entre 1 y 140 ms dependiendo del estímulo): líneas verticales (a y b), y diagonales (c, d) que se mueven describiendo trayectorias rectilíneas. El caso de las líneas horizontales y diagonales en el otro sentido, tiene un comportamiento análogo.

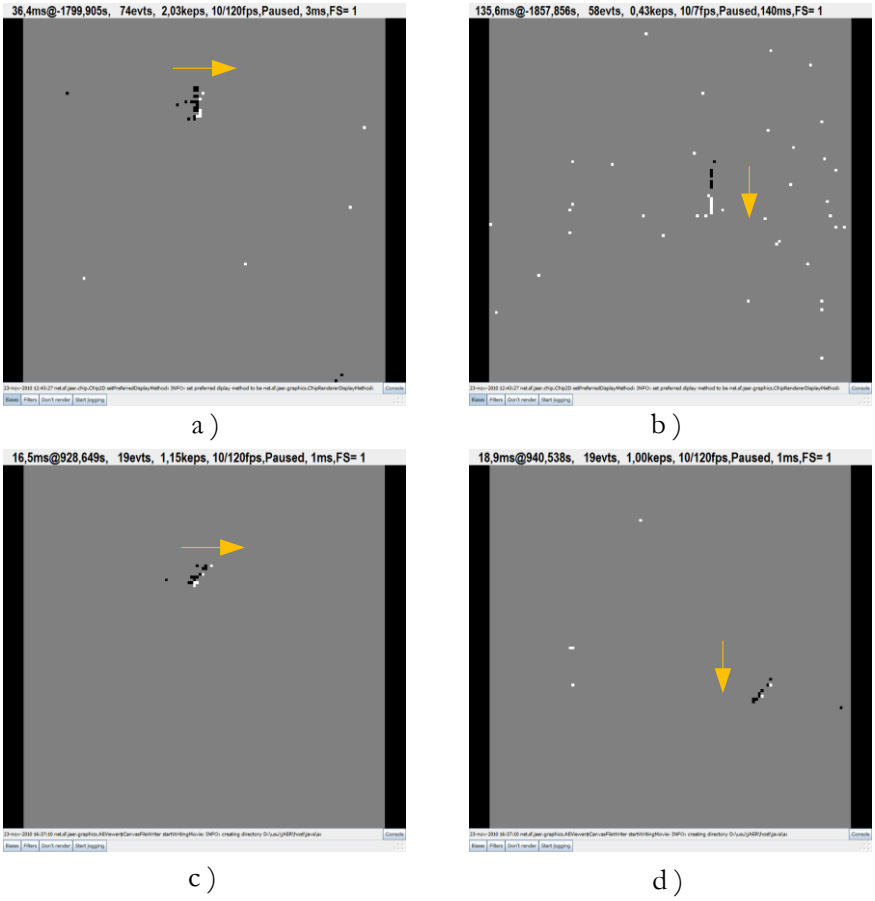


Figura 4.9: Reconstrucción de la salida de la retina para estímulos muy sencillos con trayectorias rectilíneas

A la vista de la Figura 4.9, para detectar estos estímulos sencillos las matrices de los patrones (que por claridad se ha elegido una matriz de 8x8) deberían ser para la línea horizontal y diagonal:

$$\begin{Bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{Bmatrix} \tag{4.20}$$

$$\begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \tag{4.21}$$

Para líneas horizontales y diagonales en el otro sentido basta con trasponer los patrones mostrados en la ecuación (4.20) y (4.21) respectivamente.

Si el movimiento de las líneas describiera un óvalo, en el caso de líneas verticales el patrón seguiría siendo válido, puesto que la salida de la retina es prácticamente igual en todo momento, salvo cuando el movimiento es paralelo a la dirección, que es un poco más delgada. Para las líneas diagonales ocurre lo mismo. Por lo tanto para patrones sencillos (líneas rectas) los patrones son válidos para todo tipo de movimientos.

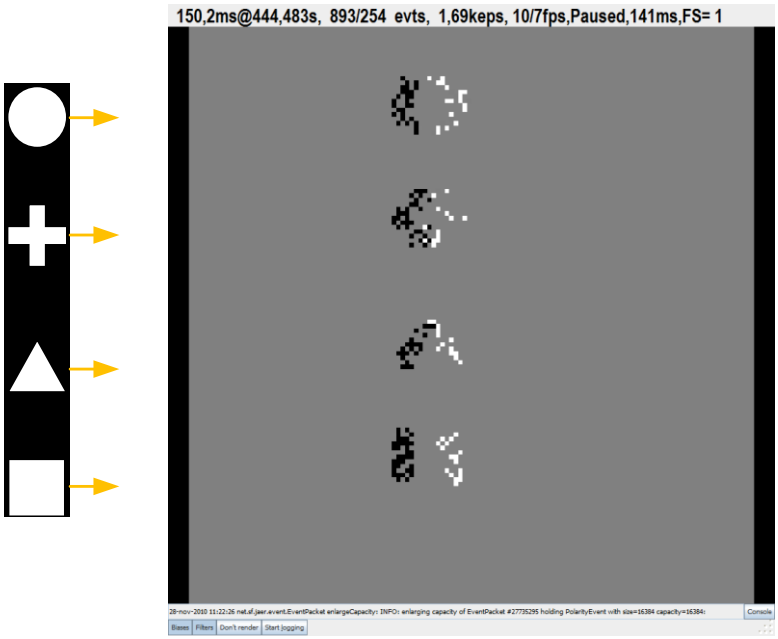


Figura 4.10: Estímulo de figuras sencillas y la salida de retina

En la Figura 4.10 se muestra un estímulo consistente en una serie de figuras sencillas: un círculo, una cruz, un triángulo y un cuadrado; moviéndose hacia la derecha y la reconstrucción de la salida de la retina (con un tiempo de integración de 141ms)[28]. Como se puede observar los patrones para detectar estos objetos en movimiento serán algo más complejos. Podría generarse un patrón para cada una de las figuras, aunque hay que tener en cuenta algunos detalles. Primero, para el caso del cuadrado y el círculo los patrones serán muy parecidos (más si consideramos matrices de 8x8). Segundo, en el caso del triángulo el lado inferior, que es paralelo a la dirección del movimiento es prácticamente "invisible" por la retina, por lo que no debería aparecer en el patrón. Y tercero, el caso de la cruz es bastante complejo y muy parecido al cuadrado.

[28] El tiempo de integración es sensiblemente superior al caso anterior debido a que los objetos se mueven más lentamente y es necesario integrar durante más tiempo para recibir suficientes eventos con los que reconstruir la salida de la retina.

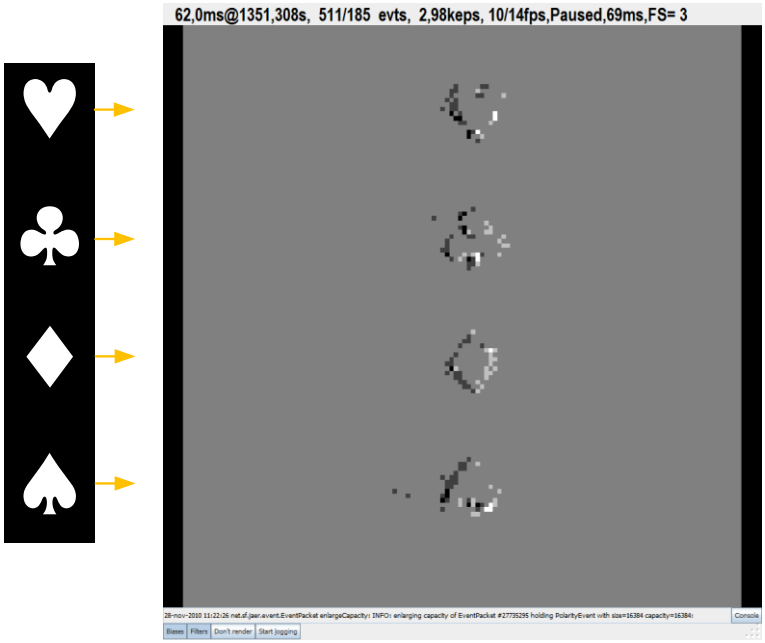


Figura 4.11: Los símbolos de las cartas de la baraja francesa moviéndose con trayectoria rectilínea

En la Figura 4.11 se muestran las figuras de las cartas de la baraja francesa (corazón, trébol, rombo y pica). Como se observa parece muy difícil obtener un patrón que identifique inequívocamente al trébol y lo diferencie de la pica.

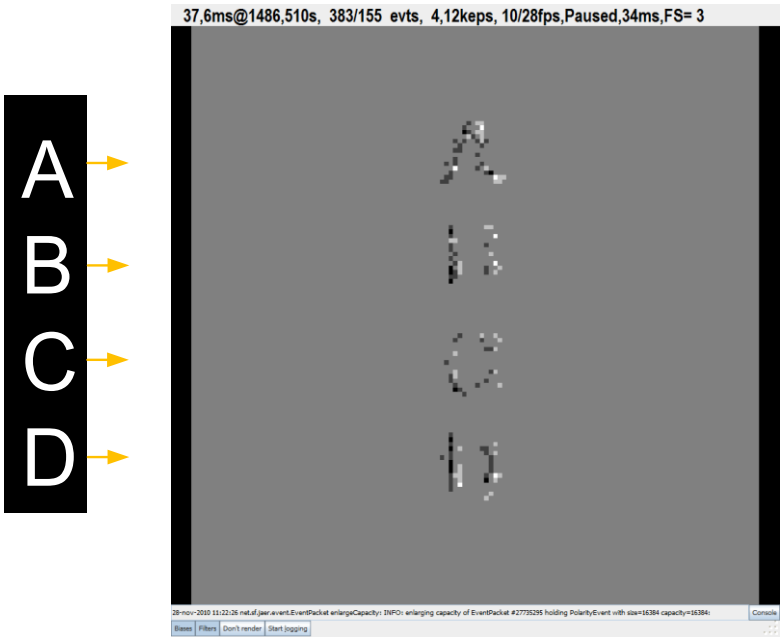


Figura 4.12: Letras en movimiento con una trayectoria rectilínea

En la Figura 4.12 se puede observar la salida de la retina como respuesta a un estímulo consistente en las 4 primeras letras del alfabeto, moviéndose con una trayectoria rectilínea hacia la derecha. Es fácil darse cuenta que los patrones para detectar las letras B y D serán muy similares y que hará muy difícil una hipotética clasificación. Otro aspecto importante es que las líneas horizontales, como por ejemplo, en la letra A son "invisibles", tal y como ocurría con el triángulo.

A la vista de los ejemplos anteriores, parece bastante difícil encontrar patrones útiles para distinguir figuras complejas y letras, cuando éstas se mueven con movimientos rectilíneos.

Sin embargo, para estímulos sencillos, como los mostrados al comienzo (básicamente líneas verticales, horizontales y diagonales), los patrones son fáciles de obtener y como se verá más adelante muy útiles para la detección de estos estímulos.

RETINA EN MOVIMIENTO

Veamos que ocurre cuando los objetos están inmóviles en una posición y es la retina la que se mueve imitando los movimientos de los ojos. Para generar el movimiento de la retina se usa un pequeño *pan & tilt*[29], programado con movimientos aleatorios.

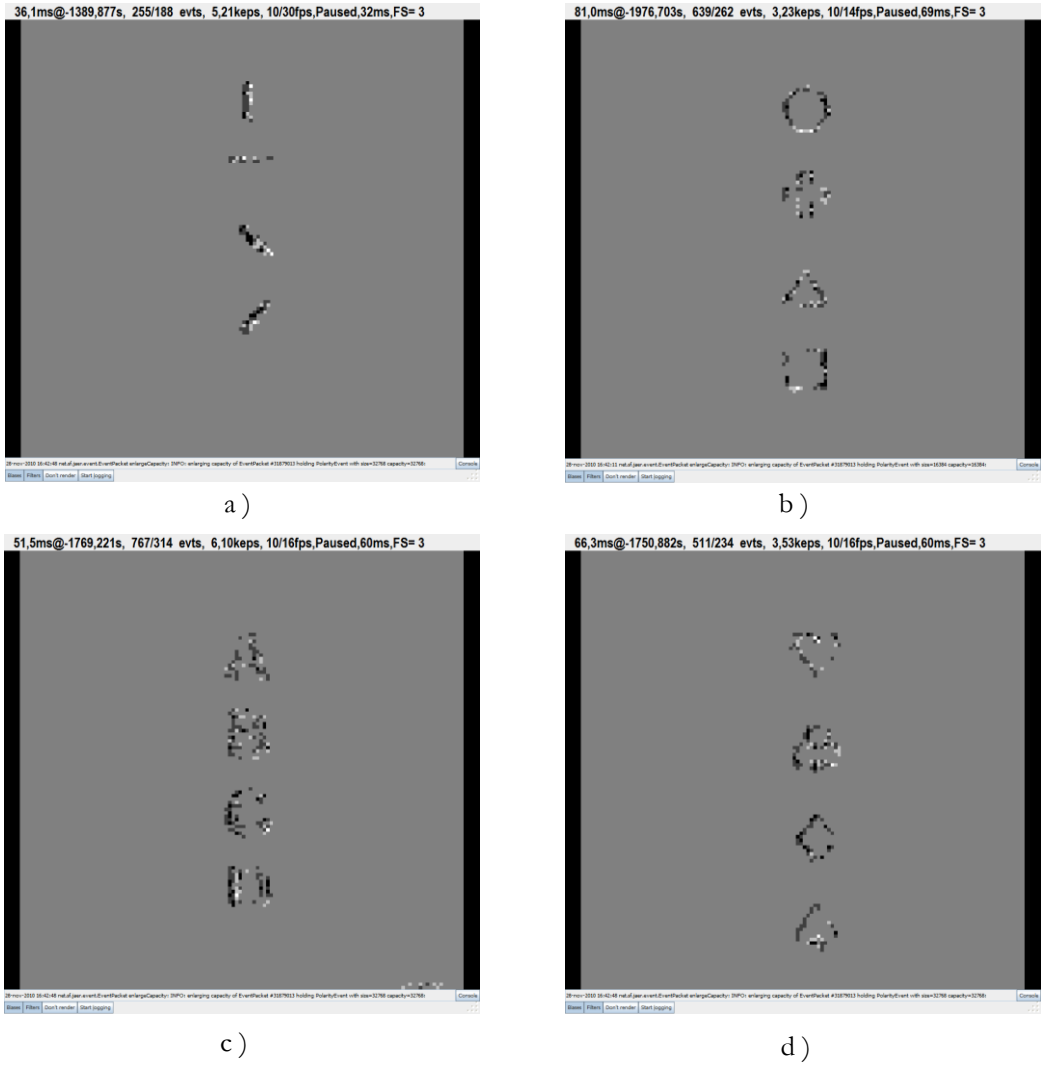


Figura 4.13: Respuesta de la retina cuando los objetos están inmóviles y la retina está vibrando.

[29] Nombre en inglés de un sistema de posicionamiento que permite la rotación sobre el eje Z (panorámico) y sobre el eje Y (inclinación).

La Figura 4.13 muestra la reconstrucción de la salida de la retina para el caso de líneas a), para figuras simples b), para las 4 primeras letras del alfabeto c) y para las figuras de la baraja francesa d). Para las líneas apenas hay diferencias con las imágenes mostradas más arriba, sin embargo para el caso de las figuras sencillas se ve que estas aparecen más definidas y que se pueden observar los detalles "invisibles" en trayectorias rectilíneas (recordemos que eran "invisibles" las líneas paralelas a la dirección del movimiento).

Ahora las letras también aparecen más definidas y las letras B y D parecen más claramente diferenciables; lo mismo ocurre con las figuras de la baraja francesa.

Aunque ante nuestros ojos y después de hacer una reconstrucción de la salida de la retina, que recordemos consiste en integrar todos los eventos durante un determinado tiempo y asignar un tono de gris dependiendo de la cantidad de eventos recibidos en una localización determinada; las figuras y formas serán fácilmente reconocibles; eso no quiere decir que sea trivial el encontrar un patrón útil para realizar el reconocimiento de objetos usando el modelo *Pattern Integrate and Fire* con pesos fijos.

Como ya se ha apuntado el uso del modelo *Pattern Integrate and Fire* con pesos fijos es muy sensible a los patrones usados, y como se acaba de exponer éstos dependen, en gran medida, de los estímulos que se pretenden reconocer y de la forma en la que se obtienen esos estímulos.

4.5.1. La Celda PRCell

La celda PRCell materializa el modelo *Pattern Integrate and Fire*. En primer lugar es necesario definir cómo han de ser las entradas de esta celda.

Entradas de la PRCell

La entrada de la PRCell será una secuencia AER correspondiente a los eventos producidos por los píxeles de la retina. La retina usada en este trabajo tiene una dimensión de 128x 128 píxeles. Es evidente que la PRCell por sí sola no puede detectar un patrón en una posición arbitraria de la imagen. Por lo que necesitará de ayuda para seleccionar la parte de la

imagen que ha de analizar. Esta ayuda será la información que obtiene la CMCell, la cual consigue la posición de un objeto en movimiento en la imagen.

La PRCell es capaz de detectar un determinado patrón predefinido en un área de 8x8 píxeles. Como en el caso de las otras dos celdas presentadas CMCell y VCell; la PRCell tomará como entrada una secuencia de eventos los cuales procesará y obtendrá un resultado, en este caso la existencia o no de un patrón predefinido.

La PRCell tiene 3 puertos de comunicación AER, dos de entrada y uno de salida. Por uno de los puertos de entrada recibirá la secuencia AER con los eventos que debe analizar y determinar si existe o no el patrón deseado; por el segundo puerto de entrada recibirá la posición de la imagen donde debe buscar, esta información la proporciona la CMCell. Por el puerto de salida se enviarán los eventos que se generen cuando se detecte el patrón predefinido, pudiéndose enviar un evento o varios, según se quiera modelar una neurona de ráfagas o no.

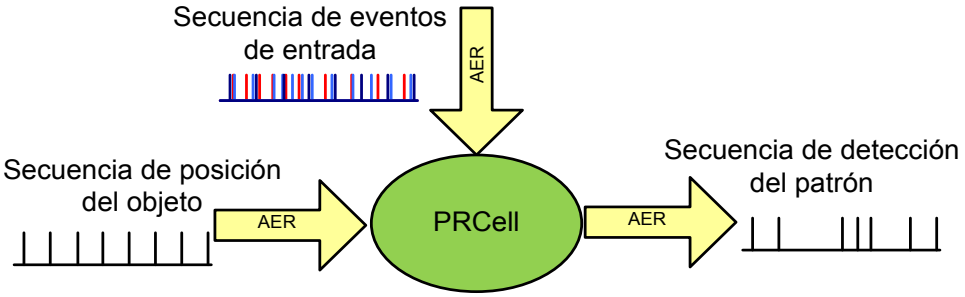


Figura 4.14: Puertos de comunicación de la celda PRCell

Descripción funcional

El funcionamiento de la PRCell requiere de la definición de varios parámetros y atributos que permiten configurar su comportamiento:

- **Patrón o máscara**: es la matriz de 8x8, explicada más arriba, que contiene el patrón a detectar. Cada posición de la misma contiene un 0 en aquellas posiciones en las que no deben haber eventos, ó un 1 en aquellas posiciones donde sí deben haber eventos.

- **Umbral de potencial**: asociado a cada patrón se define un umbral de potencial, que determinará el momento en el que se considera detectado el patrón.
- **Potencial**: corresponde con el potencial interno de la celda, que determinará el régimen de comportamiento de la misma: por debajo de umbral, cuando se están recibiendo eventos; o por encima de umbral cuando se ha superado el umbral de potencial; momento en el que se generan pulsos indicando que se ha detectado el patrón.
- **Tiempo de reinicio**: modela la función de olvido de las neuronas biológicas. La PRCell tiene un tiempo de reinicio, pasado el cual si no se ha detectado ningún patrón se borrará toda la información almacenada en la misma. Parte de la información que se borrará es el potencial y la memoria de eventos recibidos. Esto modela el elementos de olvido y reinicio de integración.
- **Umbral de disparo de las entradas**: para modelar el hecho de que las conexiones entre las neuronas biológicas están ponderadas, la PRCell tiene una matriz de 8x8 en la que se definen cuantos eventos han de recibirse por cada entrada. Se considerará que en esa localización (píxel) hay actividad relevante, y que por lo tanto debe ser tenida en cuenta para detectar el patrón.
- **Memoria de eventos recibidos**: por cada posición se almacenarán los eventos recibidos. Esta memoria es la base para decidir si un píxel ha recibido suficientes eventos para ser tenido en cuenta a la hora de incrementar o disminuir el potencial.

El funcionamiento de la PRCell se describe en la Figura 4.15. Tras el "reset", la PRCell se encuentra en el estado *INICIO*, en el cual se borra la memoria de eventos y el potencial se coloca en la situación de reposo; se pasa entonces al estado *En espera* donde permanerá hasta recibir un evento.

Cuando se recibe eventos se pasa al estado *actualiza cuenta de eventos*, donde se actualiza de memoria de eventos recibidos y se determina si la cantidad de eventos recibidos para esa localización supera o no el umbral de disparo de entradas; en el caso de que no se supere el umbral se vuelve al estado de *en espera*.

Si el umbral es superado, quiere decir que en esa localización se han recibido suficientes eventos como para considerar esa localización, por lo tanto se pasa al estado *Comprueba localización en el patrón*. Si la posición está marcada con un 0 significa que en el patrón en esa localización no debería haber eventos, por lo que se disminuirá el potencial y se pasa a estado de *en espera*. En el caso contrario (que la localización del evento recibido esté marcada con un 1 en el patrón), significa que en esa localización debe haber eventos por lo que se incrementará el potencial. Una vez incrementado el potencial, si es mayor que el Umbral de potencial significará que el patrón ha sido detectado y por lo tanto se emite un evento para indicarlo; en caso contrario (no se supera el umbral) se pasa al estado *en espera*.

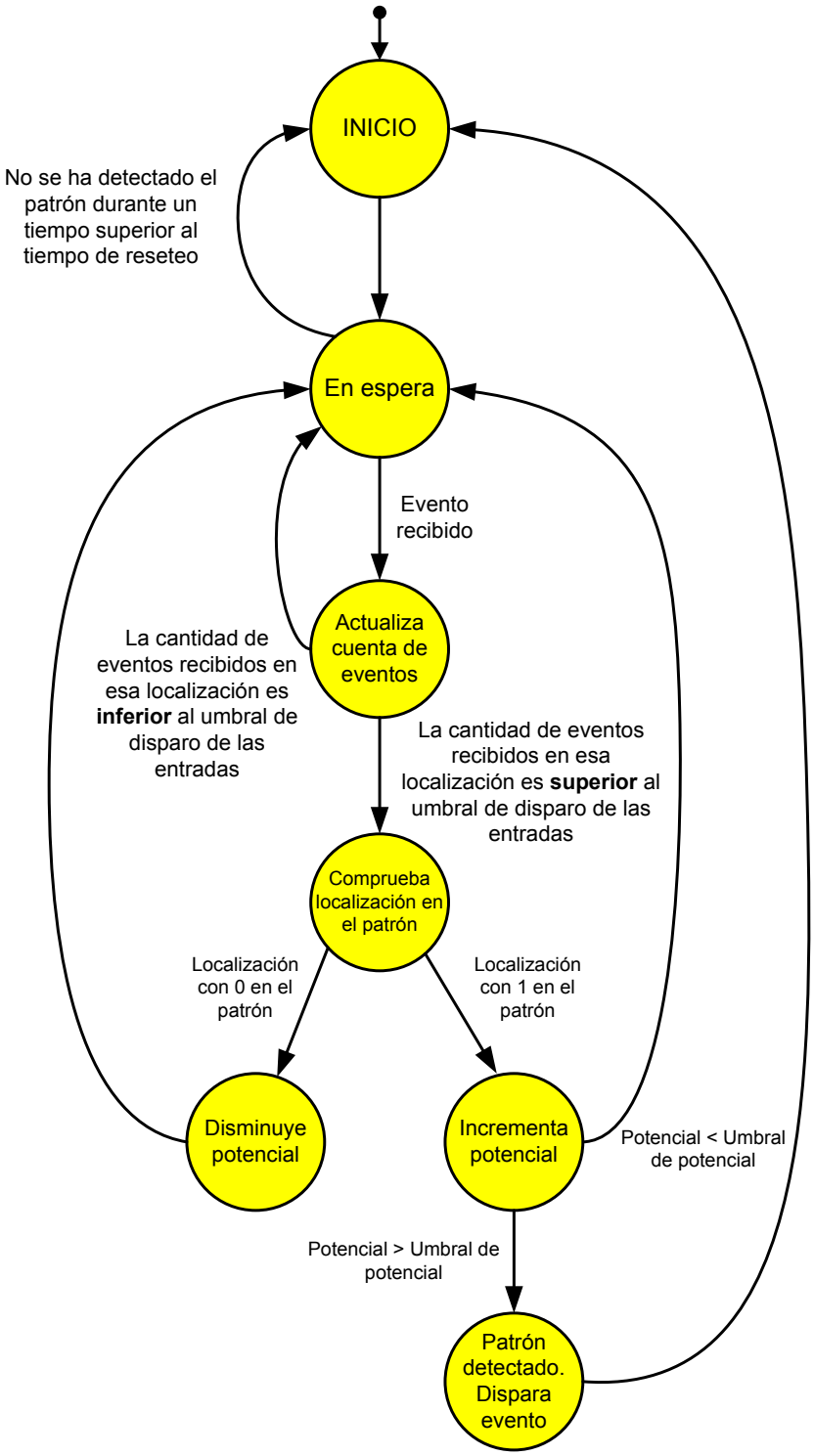


Figura 4.15: Máquina de estados de celda PRCell

4.6. Resumen

El reconocimiento de objetos es el otro de los grandes problemas de la visión artificial, al que se le han dado múltiples soluciones desde diversos puntos de vista. Dentro de los sistemas neuro-inspirados, el desarrollo del modelo de perceptrón, su conexión en red y el algoritmo de entrenamiento *backpropagation* marcaron un hito.

La aparición de las redes de neuronas pulsantes y del modelo neuronal *Integrate and Fire*, más cercano a la biología y más fácilmente realizable en un hardware específico, supone un nuevo reto en la ingeniería neuromórfica.

De la fusión de ambos modelos, el perceptrón y el *Integrate and Fire*, nace el *Pattern Integrate and Fire* que pretende dar respuesta al problema de la detección de patrones en el campo de las redes de neuronas pulsantes. El funcionamiento de este nuevo modelo consiste en la suma ponderada de las entradas activas; una entrada se considera activa cuando la acumulación de los pulsos (en esa entrada), supera un determinado umbral. Cuando el valor de la suma ponderada supera el umbral de disparo se emite un pulso, que indica que la detección del patrón es positiva.

Toda red de neuronas debe tener la capacidad de ser entrenada. Sin embargo, en el caso de las redes de neuronas pulsantes, aunque se ha desarrollado mecanismos de aprendizaje basados en la STDP, no se ha llegado a demostrar la convergencia de los mismos para un conjunto arbitrario de patrones linealmente separables.

El modelo *Pattern Integrate and Fire* puede ser usado sin entrenamiento, con pesos fijos, para detectar un patrón en las entradas. Los pesos son fijados, a 1 ó -1, dependiendo si una determinada entrada debe estar activa o no; de manera que el conjunto de los pesos se convierte en una especie de patrón de clase del patrón que se quiere detectar. La dificultad de este uso estriba en la elección de este patrón, que ha de ser lo suficientemente preciso como para no provocar falsos positivos y a la vez flexible como para soportar ruido o ambigüedades en la secuencia de pulsos de entrada.

Para elegir los pesos, es necesario realizar un estudio sobre la forma en la que la retina responde a un determinado estímulo. Recordemos que la retina artificial sólo es capaz de captar el movimiento, por lo que para poder determinar la existencia de un determinado patrón el objeto debe estar en movimiento. O, como ciertos estudios demuestran, que la retina se mueva (mientras que el objeto permanece inmóvil), este movimiento de la retina provoca un movimiento aparente del objeto, haciéndolo visible para la retina. Hay que destacar que esta forma de obtener "movimiento", no provoca la misma respuesta de la retina, es decir, la retina no lo "percibe" de la misma manera.

La PRCell es una celda AER que responde al modelo propuesto. Consta de 3 puertos AER, dos de entrada: uno por los que recibe la secuencia de eventos y otro por el que recibe la zona de la imagen en la que ha de buscar el patrón; y uno de salida que ofrecerá el resultado de la detección.

La descripción funcional propuesta contempla un tiempo de reinicio, sobrepasado el cual toda la información almacenada en la celda es borrada, implementando así una función de olvido.

"No fracasé, sólo descubrí 999 maneras de cómo no hacer una bombilla."

Thomas Alva Edison

Capítulo 5

El sistema completo: Seguimiento y reconocimiento de objetos

Una vez detalladas las celdas que permiten realizar el seguimiento y reconocimiento de objetos, es conveniente ofrecer una visión de conjunto del sistema completo, donde se propone una arquitectura para el procesamiento neuronal en cascada, en contraposición con la arquitectura paralela "clásica". Se aporta una visión de las diferentes configuraciones posibles, así como, la utilidad de cada una de ellas y las posibles aplicaciones prácticas de las mismas. Por otra parte, dado que el seguimiento y el reconocimiento de objetos es uno de los grandes anhelos en los sistemas de visión artificial, parece claro que el abanico de posibilidades del sistema que aquí se presenta será muy parecido al de los sistemas de visión artificial por computador basados en CCD, obviamente salvando las limitaciones que está nueva tecnología pueda tener y añadiendo las ventajas que pueda presentar.

A continuación, veremos las distintas configuraciones que podemos realizar con estas 3 celdas, CMCell, VCell y PRCell, cuya unión constituye la TrackCell. Al conjunto es lo que hemos denominado el ObjectTracker.

Para concluir, se abrirá la discusión sobre el cambio del modelo de transmisión de la información: debido al aumento y al refinado de la información que se produce capa a

capa, aparece la necesidad de modificar la forma de transmitir la información. Se revisará la filosofía original del protocolo AER para adaptarla a estas nuevas necesidades.

5.1. Arquitectura en cascada

En este trabajo se propone una arquitectura en cascada cuya principal diferencia con la arquitectura en paralelo es que la información, proveniente de los sensores o de otras capas, no se distribuye por todas las celdas, sino que la información que fluye entre una celda y otra es la que no se ha procesado aún; de esta manera se implementa un mecanismo de inhibición implícito entre las celdas de cada capa. En la Figura 5.1, se muestra la idea antes expuesta: la información llega a la primera celda de la capa que "procesa" parte de esa información, el resto de la información es enviada a la siguiente celda de la capa, la respuesta de una capa es enviada a la primera celda de la siguiente capa; y el esquema se repite.

Esta organización presenta algunas ventajas. La interconexión entre las neuronas de una misma capa es más simple. En primer lugar, la comunicación entre las celda es "punto a punto" entre una celda y la siguiente; y segundo no es necesaria la implementación de ningún mecanismo de inhibición explícito, ya que el hecho de que las neuronas no compartan la misma información hace innecesario dicho mecanismo. De manera implícita si existe cierta inhibición, puesto que la información procesada por una celda inhibe a las siguientes de procesar esa misma información. Además, la interconexión entre las capas también se simplifica, porque sólo es necesario enviar la información a una de las celdas (la primera) de la siguiente capa (véase la Figura 5.1 donde se puede observar la gran cantidad de conexiones entre capas que existe).

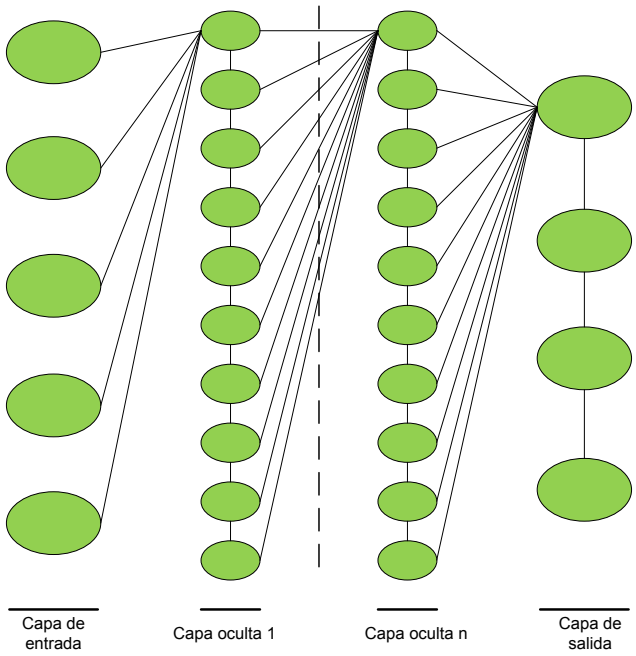


Figura 5.1: Arquitectura en cascada

Estas dos ventajas lo son fundamentalmente en una implementación basada en un circuito digital. La masiva interconexión entre las neuronas de los sistemas neuronales biológicos es uno de los muchos problemas que hay que sortear para construir sistemas neuromórficos digitales. Así, aunque el AER nace con la idea de solucionar este problema, cuando la cantidad de información a enviar es muy elevada, como la procedente de la retina, y además está codificada en frecuencia de pulsos, el bus AER se ve saturado por tanta información y comienzan a haber retrasos significativos en la transmisión de eventos.

5.2. La TrackCell

La TrackCell será la macro celda que incorporará una o varias de las celdas vistas en los capítulos anteriores, permitiendo adaptar el sistema a las necesidades de la aplicación concreta a la que se le quiere dar solución. Desde el punto de vista externo, una TrackCell contará con un puerto AER de entrada por el que recibirá la secuencia AER de eventos a procesar, y dos puerto AER de salida, uno para el envío de los eventos rechazados y otro para el envío de los resultados del procesamiento de los eventos válidos (ver Figura 5.2).

A continuación veremos diferentes combinaciones y sus posibles aplicaciones. La celda CMCell es la única celda imprescindible en todas las combinaciones, no en vano es la que permite localizar la región de interés, sobre el que se harán procesamientos posteriores, en nuestro caso la estimación de la velocidad y la detección de patrones.

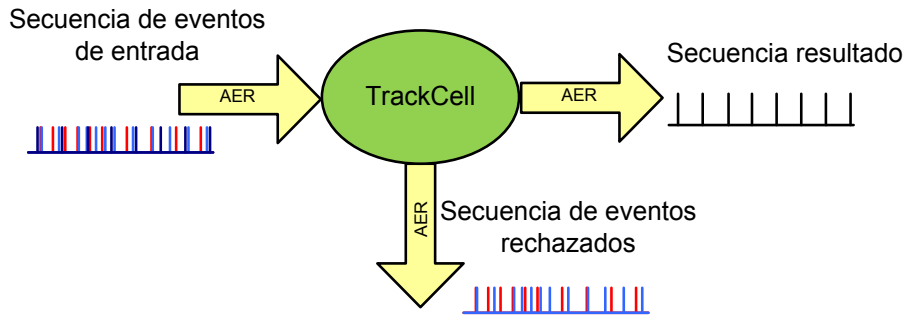


Figura 5.2: TrackCell genérica

5.2.1. TrackCell Tipo S[30]: Obtención de la posición de los objetos en movimiento

Esta primera configuración es la más simple de todas, consta de una única celda CMCell. La finalidad de este tipo de TrackCell, es la localización y seguimiento de un objeto en movimiento en la escena, sin estimación de la velocidad y sin reconocimiento de patrones.

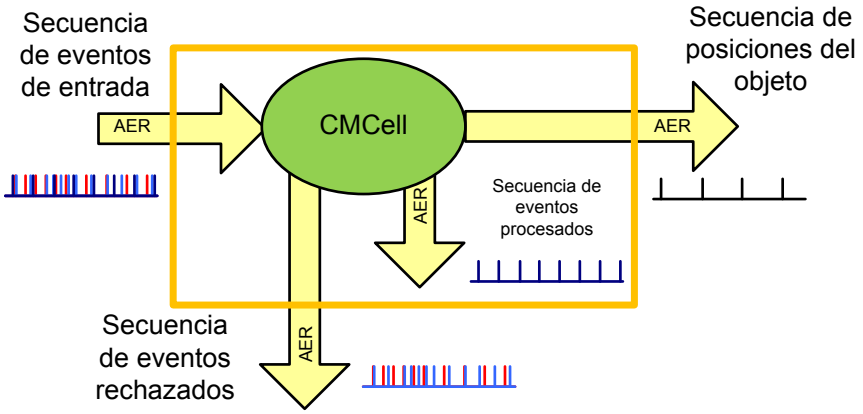


Figura 5.3: TrackCell tipo S

En este caso la TrackCell no es más que un *renombrado* de la CMCell. Su estructura interna se muestra en la Figura 5.3.

5.2.2. TrackCell Tipo B[31]: Obtención de la posición y estimación de la velocidad de objetos en movimiento

Esta configuración podría considerarse como la configuración básica, consta de una celda CMCell y una VCell. Permite la localización de un objeto en movimiento, su seguimiento y la estimación de la velocidad del mismo.

---

[30] S: Simple

[31] B: Básica

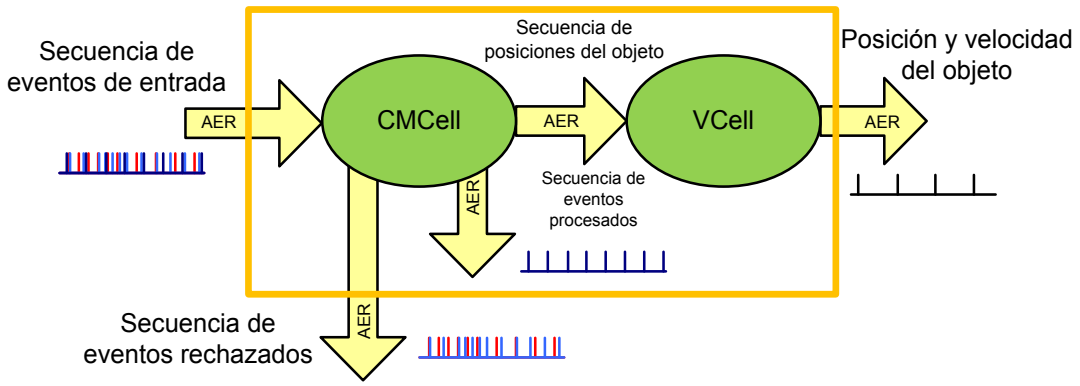


Figura 5.4: TrackCell tipo B

La estructura interna (Figura 5.4) de este tipo de TrackCell ya fue presentada en la Figura 2.1. La entrada de la TrackCell está conectada a la entrada de la CMCell que localizará el objeto, las sucesivas posiciones del mismo serán usadas por la VCell para estimar su velocidad; la salida de la TrackCell es directamente la salida de la VCell.

5.2.3. TrackCell Tipo P[32]: Seguimiento de patrones

Este tipo de TrackCell consiste en una CMCell y una PRCell. Permite la localización y el seguimiento de un objeto, identificando un patrón determinado, es decir, permite comprobar si el objeto que se está siguiendo tiene o no una determinada forma, aunque no permitiría saber su la velocidad.

Es importante remarcar que el seguimiento se realiza tanto si el objeto seguido presenta el patrón como si no, puesto que el seguimiento es independiente y anterior a la detección del patrón. Al igual con los sistemas de detección de patrones en visión por computador, donde la detección del patrón es previa al seguimiento, o dicho de otro modo solo se sigue aquello que presenta el patrón deseado.

La estructura interna de la TrackCell tipo P se muestra en la Figura 5.5, la entrada de la TrackCell está directamente conectada a la CMCell la cual determinará la posición del

[32] P: Patrones

objeto. Los eventos procesados, es decir los usados para obtener la posición del objeto y que por lo tanto corresponden con el objeto en sí, son enviados a la PRCell que determinará si hay correspondencia con el patrón prefijado; la salida de la TrackCell será la salida de la PRCell.

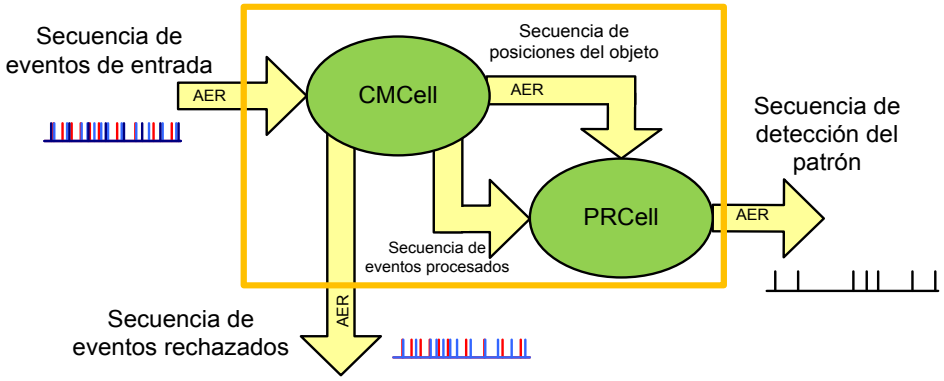


Figura 5.5: TrackCell tipo P

Atendiendo a como es la salida de la TrackCell se obtiene tres variantes:

- Tipo P.1: sólo emite eventos cuando se detecta el patrón.
- Tipo P.2: emite eventos de dos tipos: los que indican que se ha detectado el patrón, y lo que indican que no se ha detectado el patrón (del mismo modo que la retina emite eventos positivos o negativos dependiendo del sentido del cambio de intensidad de la luminosidad).
- Tipo P.3: se emite un pulso indicando la detección del patrón junto el grado de confiabilidad, entendiendo como tal la proporción entre la cantidad de eventos recibidos por entradas deseadas y por entradas no deseadas.

5.2.4. TrackCell Tipo C[33]: Seguimiento de patrones con estimación de velocidad

La TrackCell de tipo C está formada por una celda de cada tipo, es decir, una CMCell, una VCell y una PRCell; permite la localización, estimación de la velocidad de un objeto y detección de la presencia o no de un prefijado patrón en el objeto.

La estructura interna de la TrackCell de tipo C se muestra en la Figura 5.6. La entrada de la TrackCell, de nuevo, es la entrada de la CMCell, la cual determinará la posición del objeto. Esa información es usada tanto por la VCell, para estimar la velocidad, como por la PRCell para determinar la existencia del patrón prefijado; la PRCell necesita también los eventos procesados por la CMCell. La salida de la TrackCell será la fusión de las salidas de la VCell y de la PRCell, de manera que la capa superior pueda saber la posición y la velocidad del objeto y si presenta o no el patrón deseado.

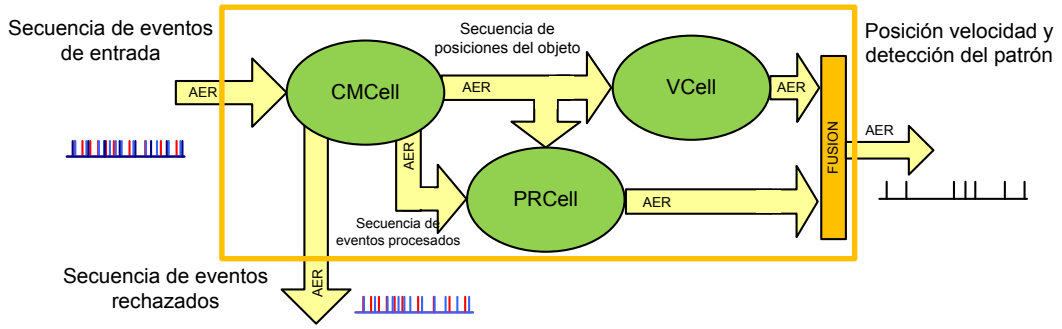


Figura 5.6: TrackCell tipo C

Al igual que en el caso anterior, la salida de la TrackCell da lugar a variantes, según la manera de hacer sea la salida final a partir de las salidas de la VCell y la PRCell. Básicamente, se puede realizar de dos maneras:

- Tipo C.1: inhibe la salida de la TrackCell en el caso de que no se detecte en el patrón.

[33] C: Completa

- Tipo C.2: codifica de alguna manera la detección o no del patrón deseado (evento positivo o negativo).

De forma, que en el primer caso no habría salida a no ser que se detecte el patrón, mientras que en el segundo siempre habría salida pero indicando si se ha detectado o no el patrón. La elección de un tipo de salida u otro dependerá, obviamente, de la aplicación.

5.2.5. TrackCell Tipo CL[34]: Seguimiento y clasificación de patrones con estimación de velocidad

La más compleja de las TrackCells es la tipo CL. Este tipo de TrackCell permite la localización, el seguimiento, la clasificación y la estimación de la velocidad de un objeto en movimiento.

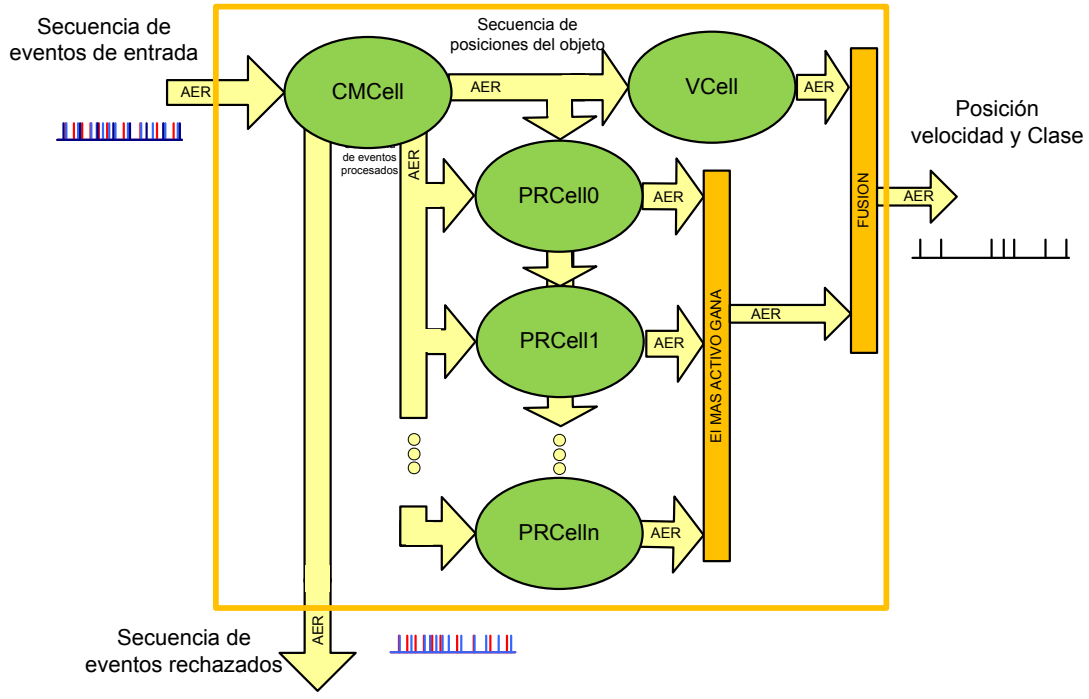


Figura 5.7: TrackCell tipo CL

[34] CL: Clasificadora

La estructura interna de este tipo de TrackCell se muestra en la Figura 5.7. La celda consta de una CMCell, para obtener la posición del objeto, una VCell para estimar la velocidad y hasta n PRCell, cada una de ellas configurada con un patrón distinto que permita determinar que patrón presenta el objeto; en el caso de que varias PRCells detectaran el patrón con el que están configuradas, la más activa de ellas será la ganadora y su salida es fusionada con la de la VCell para generar la salida final de la TrackCell.

A esta celda son aplicables las variantes expuestas para la TrackCell tipo C. Por lo que tendríamos tipo CL.1 y tipo CL.2.

5.2.6. Aplicaciones de las TrackCells

Es evidente que todos los tipos de TrackCell tienen en común el seguimiento de objetos, por lo tanto todas las TrackCells son aplicables a aquellos problemas que requieran conocer la posición de un objeto en movimiento. Las diferencias entre los distintos tipos de TrackCell están en la información adicional que cada tipo ofrece.

Así pues la TrackCell tipo B, (CMCell+VCell), ofrece la información de la velocidad y puede ser muy útil en aquellas aplicaciones donde conocer la velocidad sea un requisito para la toma de ulteriores decisiones. Por ejemplo, en los sistemas de percepción en robots móviles, para la detección de aproximación de obstáculos; en sistemas de vigilancia o supervisión donde se necesite detectar velocidades anómalas de objetos, ya sean altas o bajas.

La TrackCell de tipo P proporciona un análisis de la forma de objeto, a partir de la detección de un patrón visual en el objeto. Esta TrackCell tiene su aplicación en problemas donde solo se necesite seguir ciertos objetos (los que presentan un patrón visual determinado). En robótica su aplicación en los sistemas de percepción está indicada en aquellos problemas en los que se quiera identificar un objeto determinado, como por ejemplo una bola, un cubo; en robots móviles es de utilidad para localizar posiciones o incluso lugares a partir de la existencia de marcas. En sistemas de supervisión su utilidad se fundamenta en la capacidad de discriminación de objetos, permitiendo fijar la atención en solo aquellos que son de interés.

La TrackCell de tipo C, aporta información muy abundante: la velocidad y la existencia de un patrón. Las aplicaciones donde aparte de conocer si el objeto presenta o no un patrón es importante saber además la velocidad, usarán este tipo de TrackCell.

Por último, la TrackCell de tipo CL ofrece la información más completa, de la posición, velocidad de objeto y además que patrón presenta de entre un conjunto prefijado. Las aplicaciones en sistemas de percepción, problemas de clasificación de objetos, donde se requiera conocer su velocidad, para por ejemplo detectar velocidades anómalas en función del tipo de objeto.

En general con los tipos de TrackCell presentados se cubre un gran abanico de aplicaciones propias de los sistemas de visión artificial, que computan el movimiento y la forma de los objetos presentes en una escena. Y prácticamente todos los problemas que se pueden solucionar con un sistema de visión por computador puede encontrar una solución basada en TrackCell. Con esto, no se quiere decir que la solución sea mejor ni peor, si no que será una solución alternativa que en algunos casos puede mejorar las soluciones "clásicas"; lo que sí se puede afirmar con rotundidad es que la solución que aquí se aporta está inspirada en el funcionamiento de la biología, además de imitar su estructura y funcionamiento interno.

5.3. El ObjectTracker

El ObjectTracker es la estructura de más alto nivel organizativo, y englobará todas las TrackCells del sistema, tantas como objetos se desean seguir simultáneamente. El ObjectTracker hace uso de la arquitectura en cascada propuesta al inicio de este capítulo, de manera que las TrackCells son conectadas en forma de cascada. La Figura 5.8, muestra un diagrama de la estructura general de un ObjectTracker compuesto por n TrackCells, Para unir todos los eventos producidos por las TrackCells es necesario un arbitrador, que decida en cada momento que TrackCell ha de poner su información en el bus común, en el caso de colisión temporal de las salidas de varias TrackCell.

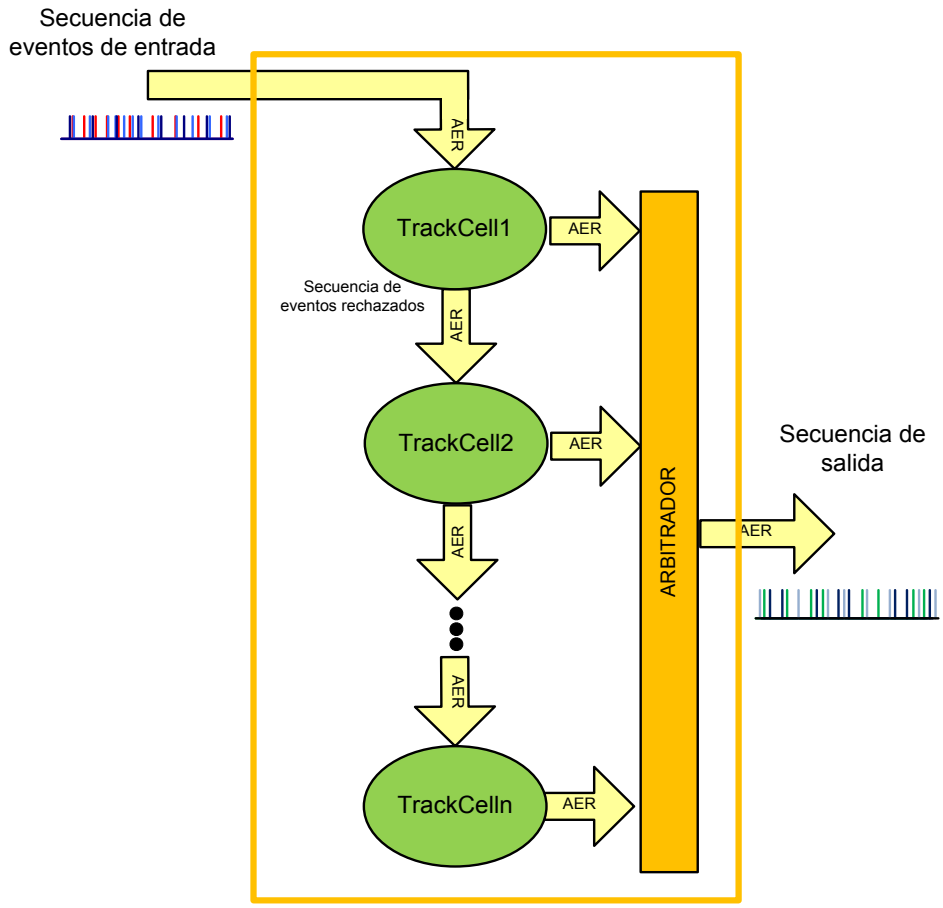


Figura 5.8: Diagrama general del ObjectTracker

La necesidad de un arbitrador no es una novedad en los sistemas AER, que basan su funcionamiento en la multiplexión en el tiempo de todos los eventos producidos por una serie de celdas. Se aprovecha la diferencia de velocidades entre la máxima tasa de emisión de eventos de las celda y del bus, la cual permite enviar todos los eventos de la celda sin originar retrasos significativos, salvo en los casos de coincidencia temporal, en cuyo caso es necesario decidir qué celda emite antes y qué celda debe esperar.

El arbitrador se convierte en una pieza clave para el buen funcionamiento del sistema. El arbitrador más sencillo es el arbitrador de prioridades fijas, el cual tiene pre-asignado el orden en el que deben emitir dos celdas en caso de coincidencia temporal. En el extremo opuesto, en cuanto a prestaciones y complejidad estarían los arbitradores con prioridades

aleatorias, que deciden que celda ha de emitir de manera aleatoria; y los arbitradores con prioridades rotatorias, que en caso de coincidencia de dos celda, cada vez que se produzca la coincidencia se cambia el orden de acceso al bus.

En nuestro caso, el arbitrador que mejor se adapta al funcionamiento del sistema es el que tiene prioridades fijas y además debe asignar la prioridad máxima a la celda que está al principio de cada capa e ir bajando la prioridad conforme se desciende en la capa. Con este esquema de prioridades se garantiza que la primera celda, de la que depende el resto del sistema ya que es la responsable de reenviar los eventos que no necesita a las siguientes celdas, tendrá acceso al bus lo antes posible. En otro caso, podría darse la circunstancia de que el retraso en el acceso al bus de las celdas colocadas al principio de la capa afectara a la temporalidad del sistema. No conviene olvidar que en los sistemas AER no sólo es importante la información en sí, si no que es casi más importante cuándo se produce esa información.

Se pueden construir varios tipos de ObjectTracker dependiendo del número y del tipo de TrackCells que integre, que se denotarán como ObjectTrackerNX, donde N es el número de TrackCells que integra y X el tipo de las misma. Así pues un ObjectTracker4C.2 estará formado 4 TrackCells de tipo C.2. En el caso de que las TrackCell fueran de diferentes tipos, se indicarán todos, anteponiendo la cantidad de cada uno; por ejemplo si el ObjectTracker tuviera una de tipo B y 3 celdas de tipo C.2, se denotaría como ObjectTracker1B3C.2.

5.4. ¿Un nuevo modelo AER?

Tanto en los capítulos anteriores como hasta ahora en el presente capítulo, se ha pasado por alto el incremento, y el refinado de la información, que implica el seguimiento y reconocimiento de objetos.

Al principio, la información que fluía por un puerto AER, indicaba los cambios de intensidad lumínica, codificada en frecuencia de pulsos, de manera que un mayor número

de eventos implica un cambio mayor en la intensidad, mientras que, al contrario, un menor número de eventos un cambio menor en la intensidad. Además siguiendo el protocolo AER cada celda, numerada a tal efecto, sólo tiene que trasmitir su número (dirección) el adecuado número de veces para comunicar el valor de su actividad.

Sin embargo, en este trabajo la información que ha de transmitirse es un poco más compleja que el cambio de intensidad captada por cada píxel de la retina; ahora hay que transmitir la posición del objeto, su velocidad, el tiempo de integración usado para calcular la velocidad, la existencia o no de un patrón (o cual de los patrones predefinidos se ha detectado, en el caso de TrackCell tipo CL), y por último un identificador de la celda que ha elaborado toda esa información.

Si realizamos, como se verá en el capítulo siguiente, una implementación digital de todo el sistema toda esta información puede ser cuantificada en bits, a modo de ejemplo valgan los siguientes valores:

- 16 bits para la posición, lo que supondría un espacio de 256x256 puntos.
- 16 bits para la velocidad, lo que permitiría a un objeto desplazarse de un extremo a otra de la imagen, en un solo paso de integración de la VCell.
- 4 bits para el tiempo de integración de la VCell, lo que permitiría hasta 16 posibles valores.
- 4 bits para los patrones, lo que permitiría en el caso de una TrackCell de tipo CL tener hasta 16 posibles clases.
- 8 bits para el identificador de TrackCell, lo que supone hasta 256 celdas.

En total son 48 bits de información que ha de ser enviada de alguna manera a la siguiente capa de procesamiento.

Siguiendo la filosofía del protocolo AER la información ha de ser codificada en frecuencia de pulsos y enviada trasmitiendo sólo el identificador de la celda que origina el dato. Según eso, con 48 bits, deberíamos contar con 2^{48} (= 281.474.976.710.656) frecuencias distintas para poder soportar todos los posibles valores que pueden tomar los

48 bits. Valor que no parece alcanzable en un sistema digital, y más aún, desde el punto de vista biológico. Aunque se ha observado que las neuronas usan diferentes frecuencias para transmitir diferentes datos no parece que usen un rango tan amplio, además hay que recordar que las neuronas biológicas no son de naturaleza digital sino más bien analógica (Potter 2007) ,y pueden no tener este problema con la trasmisión de la información.

Una solución es enviar la información tal cual, sin codificarla en frecuencia de pulsos, cada vez que ésta se produzca. Por ejemplo, si nos centramos en la TrackCell de tipo B, que recordemos que estaba compuesta por una CMCell y una VCell, y que proporciona la posición y la velocidad de un objeto, la información se enviaría cada vez que se cumpliera el tiempo de integración de la VCell, en este caso la información del periodo de integración estaría también codificada en la frecuencia con la que se envían los eventos.

En el caso de la TrackCell de tipo C, que consta de una celda de cada tipo y proporciona la posición, la velocidad y la presencia o no de un patrón predefinido; la información se enviaría cada vez que se detecte el patrón deseado, en el caso de la variante con inhibición de la salida; o cada vez que se cumpla el periodo de integración, en el caso de la variante con eventos con signo.

Con este esquema de generación de eventos, lo que está codificado en la frecuencia de pulsos es la "actualización" del estado o resultado de la celda que envía el dato y no la información en sí, que es enviada de manera explícita. Un nuevo esquema que rompe de alguna manera con la filosofía original del AER.

En la Tabla 5.1 se muestra de forma resumida lo que cada celda, presentada hasta el momento, emitiría asumiendo los datos de más arriba. Como se puede observar cuanto más compleja es la celda mayor es el volumen de información que genera y por lo tanto sus necesidades de transmisión. En cuanto a la temporización, es decir, cuándo se envía la

información, no existe una periodicidad en los datos salvo, en aquellos casos, que por decirlo de algún modo, es una celda VCell la que tiene el control de la salida[35].

Tabla 5.1: Resumen cuándo y qué transmite cada celda

Tipo de celda	Dato	Tamaño (bits)	Temporización
CMCell	Pos	16	Cada nuevo evento recibido
VCell	Vel	16	Periodo de integración
PRCell	SP	1	Cuando se supera el umbral de carga[36]
TrackCell tipo S	ID y Pos	8+16	Cada nuevo evento recibido
TrackCell tipo B	ID, Pos, Vel y TI	8+16+16+4	Periodo de integración
TrackCell Tipo P.1	ID y Pos	8+16	Cuando se supera el umbral de carga
TrackCell Tipo P.2	ID, Pos y SP	8+16+1	Cada nuevo evento recibido
TrackCell Tipo C.1	ID, Pos, Vel y TI	8+16+16+4	Cuando se supera el umbral de carga
TrackCell Tipo C.2	ID, Pos, Vel, TI y SI	8+16+16+4+1	Periodo de integración
TrackCell PR.1	ID, Pos, Vel, TI y SI	8+16+16+4+1	Cuando se supera el umbral de carga
TrackCell PR.2	ID, Pos, Vel, TI y CL	8+16+16+4+1	Periodo de integración

ID=identificador de la celda
TI= tiempo de integración usado por la VCell para estimar la velocidad
Pos= posición
Vel=velocidad
SP=signo de patrón, indica si se ha detectado o no el patrón predefinido
CL=clase

Por último también se puede observar la variabilidad del tamaño de la información a enviar, punto éste problemático si se quiere estandarizar una transmisión de información entre celdas neuromórficas. Principalmente por dos motivos, primero por la cantidad de información que hay que manipular y segundo por el ancho de los cables necesarios para conectar dos dispositivos AER que quieran compartir información.

Para solucionar el problema de la cantidad de información a enviar y la variabilidad en el tamaño de ésta, proponemos una modificación del protocolo AER, que permite el envío de toda esta información sin cambiar el ancho de los cables.

---

[35] Conviene recordar que la VCell puede ajustar su periodo de integración a la velocidad del objeto y que por lo tanto, también puede cambiar la periodicidad del envío de información, aun en el caso de que una VCell sea la encargada de gobernar la salida.

[36] Que es lo mismo que decir, cada vez que se detecta el patrón; teniendo en cuenta que cuando la PRCell detecta el patrón se reinicia y debe comenzar de nuevo la detección.

5.4.1. AER con multieventos

La idea es sencilla, y se usa en las redes de comunicación telemáticas. Cada evento (del tamaño que sea) será dividido en partes del ancho del cable que una los dispositivos; esas partes serán enviadas como un evento independiente, de forma que ahora la información no fluye como un solo pulso, o evento, si no como un tren o ráfaga o grupo de pulsos o eventos. Obviamente, para garantizar una comunicación efectiva, el emisor y el transmisor han de estar de acuerdo en cuantas partes componen la información.

En la Figura 5.9 se muestra una representación esquemática del AER con multieventos, donde cada uno de los "sub-eventos" es transmitido usando un protocolo *HandShake* de 4 pasos que también usa el AER "clásico".

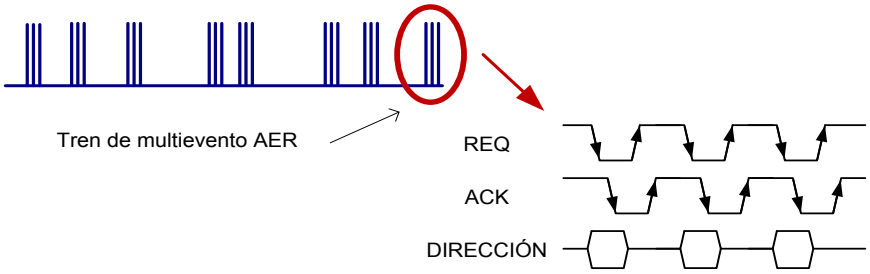


Figura 5.9: AER con multieventos

Este esquema AER permite el envío de más información entre los dispositivos neuromórficos existentes, sin necesidad de modificar los cables que los unen. Aunque sí sería necesario adaptar el funcionamiento interno de los circuitos; que en el caso de estar implementado en FPGA, no presentaría ninguna dificultad, al contrario, que en el caso de circuitos a medida. Este nuevo esquema consiste en realidad en una serialización de los eventos, mandándose éstos en una agrupación indivisible.

La solución aquí propuesta, resuelve el problema de la transmisión de más información usando los mismos cables, en las plataformas existentes.Aunque la solución definitiva, pasa por la adopción de un esquema de comunicación serie, lo suficientemente flexible como para ser adaptado, por medio de parámetros, a cualquier situación. Hasta la fecha no se han sentado las bases de un esquema de comunicación AER serie estándar (Miro-

Amarante et al. 2006), (Miro-Amarante et al. 2007), (Berge & Hafliger 2007), (Fasnacht et al. 2008) y (Zamarreño et al. 2008); tal y como sí ocurrió con la primera versión del AER y la estandarización que supuso, en algunos casos, el proyecto CAVIAR.

5.5. Resumen

Las redes neuronales biológicas, y por imitación, las redes neuronales artificiales presentan un alto grado de conectividad. Implementar en hardware este grado de conectividad es en algunos casos una tarea no exenta de dificultades. Como alternativa a la estructura "clásica" de interconexión neuronal se propone una arquitectura en cascada o jerárquica, donde las celdas de una capa quedan ordenadas y la información fluye desde la primera de ellas a la última. Cada celda extrae del flujo parte de la información y el resto es enviada a la siguiente celda, lo que constituye un mecanismo de inhibición implícita permitiendo que la misma información no sea procesada varias veces.

Los distintos tipos de TrackCell permiten definir diferentes configuraciones adaptadas al problema concreto a resolver. Posibilita la construcción de sistemas que obtengan la posición, la velocidad, la existencia o no de un determinado patrón o la clasificación de un patrón de entrada en una clase, de los objetos en movimiento en un escena; y por supuesto todas las posibles combinaciones de estos elementos. Para ello se combinan las celdas propuestas en los capítulos anteriores: CMCell, VCell y PRCell; siendo la CMCell la celda imprescindible en todas, ya que es la encargada de localizar el objeto sobre el que se realizan los posteriores análisis.

Agrupando varias TrackCells, del mismo o diferentes tipos, se obtiene el ObjectTracker que permite seguir y analizar tantos objetos simultáneamente como TrackCells lo compongan. Las TrackCells serán conectadas usando la arquitectura en cascada propuesta, permitiendo que cada TrackCell siga y analice un objeto y que ese objeto sólo sea seguido por esa TrackCell.

La información obtenida por cada TrackCell dispara las necesidades de comunicación e impone un cambio en la filosofía de codificación de la información inicial del AER, codificación en frecuencia de pulso; obligando a enviar parte de la información de manera explícita, y codificando en el tiempo entre pulsos la frecuencia de cambio del estado de las celdas.

Esta agrupación de distintos tipos de celda se encuentra presente en la biología, las neuronas se agrupan de forma que responden a diferentes estímulos, como son el color, la orientación, etc. y a combinaciones de éstos. Además refleja el hecho de que el sistema neuronal de los seres vivos no está formado por un conjunto homogéneo de neuronas, sino que está formado por una gran variedad de tipos de neuronas que dependiendo de su función específica presentan diversos patrones de conexión entre sí.

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".

Aristóteles

Capítulo 6

Implementación hardware del sistema

Uno de los objetivos específicos de este trabajo es realizar una implementación hardware de un sistema de procesamiento de información retino-mórfica. Dicha implementación no debía incluir, en su núcleo de funcionamiento, ningún computador tradicional, en el que se ejecuten instrucciones. En este capítulo se presentan los detalles del diseño hardware del sistema.

La alternativa utilizada para la descripción de un hardware a medida que implemente este sistema, está basado en el uso del lenguaje VHDL[37] (VHDL Analysis and Standardization Group 2010); este lenguaje permite la descripción de todo tipo de circuitos digitales (e incluso existen extensiones para describir circuitos analógicos).

Por otra parte una vez descrito el circuito que implemente el sistema, es necesario configurar algún tipo de dispositivo que contenga el hardware final del sistema. Este tipo de dispositivos es una FPGA (FPGA 2010).

---

[37] Siglas en inglés de VLSI Hardware Description Languaje y cuya traducción es Lenguaje de Descripción de Hardware para VLSI. VLSI del inglés Very Large Scale Integration, que se podría traducir por muy alta integración; y se refiere al diseño y fabricación de circuitos integrados.

Para realizar las pruebas y experimentos que se presentan en el capítulo 7, se ha utilizado la plataforma USB-AER desarrollada por el grupo de investigación Robótica y Tecnología de Computadores de la Universidad de Sevilla, en el seno del Proyecto Europeo CAVIAR (Caviar Project 2006).

En este capítulo se presentan las implementaciones que se han realizado de algunos de los sistemas propuestos en los capítulos anteriores. Para cada uno de los elementos construidos se ofrecerá el coste hardware medido en slices[38] de la FPGA que tiene la USB-AER (una Spartan II 200).

Otro parámetro que se detallará para cada elemento será la latencia, entendida como el tiempo que transcurre entre que se recibe un evento en la entrada y aparece una salida como respuesta a ese evento. Hay que tener presente que en este trabajo, debido al uso de la retina, sólo se producen eventos cuando la retina capta cambios de luminosidad en la escena, que en general están provocados por el movimiento de los objetos. Por tanto, el valor de latencia que se ofrecerá puede considerarse como la latencia del sistema ante el movimiento de los objetos en la escena. En aquellos casos en los que sea posible se ofrecerá un valor de la latencia real del sistema, es decir, medida sobre la FGPA y el sistema en funcionamiento; en el resto se ofrecerán valores teóricos.

Además, en aquellos desarrollos en los que se pueda medir se presentará en consumo eléctrico del mismo, medido directamente sobre la plataforma USB-AER en funcionamiento.

---

[38] Un Slice es el bloque mínimo de programación de las FPGAs; en la familia Spartan II cada dos Slices forma un CLB (*Configurable Logic Block* = bloque lógico programable); cada Slice está formado por 2 LC (*Logic Cell* = Celda Lógica) que consta de una LUT 4 entradas (*Look-up Table* = tabla de búsqueda) que puede implementar cualquier función lógica de 4 entradas, una RAM de 16x1 bit o bien un registro de desplazamiento de 16 bits; un biestable configurable y un bloque de lógica de control y acarreo (Xilinx Inc. 2008)

6.1. La plataforma USB-AER

Sin entrar en detalles que pueden ser encontrados en (F. Gomez-Rodriguez et al. 2006), (A. Jimenez-Fernandez et al. 2007) y (Paz et al. 2005), a continuación se presentan las características más importantes de la plataforma USB-AER.

La Figura 6.1 muestra una foto de la plataforma USB-AER, donde pueden observarse sus componentes más importantes

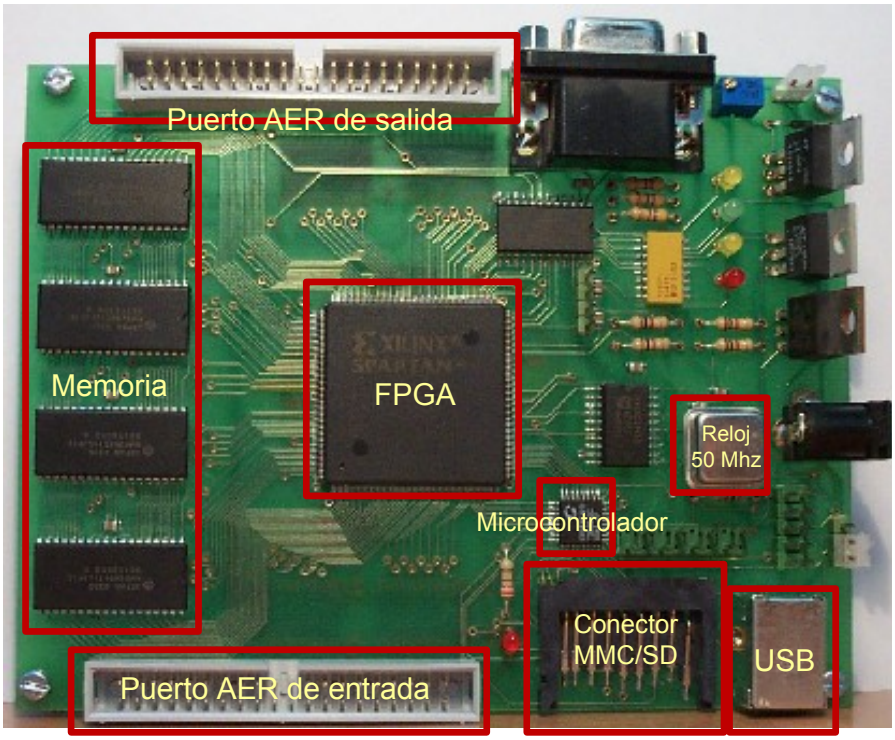


Figura 6.1: Plataforma USB-AER

La plataforma USB-AER está diseñada alrededor de una FPGA de Xilinx Spartan II 200. Esta plataforma fue desarrollada para cubrir las necesidades del proyecto CAVIAR (Caviar Project 2006), concretamente como herramienta para el testeo, depuración e interconexión de sistemas basados en AER. Dispone de 2Mbytes de memoria SRAM de 12 ns, dos puertos AER uno de entrada y otro de salida y conexión USB 1.1 gestionada

por un microcontrolador que también se encarga de la configuración de la FPGA. Ésta puede ser programada desde el USB, o desde un fichero almacenado en una memoria MMC/SD, que es leída por el microcontrolador tras el inicio; de manera que si el microcontrolador encuentra una tarjeta MMC/SD conectada y ésta contiene el fichero de configuración de la FPGA, ésta será programada justo después del inicio, sin necesidad de conexión con el PC. Esta característica permite un funcionamiento totalmente independiente del PC.

La USB-AER es una plataforma muy versátil: existen controladores para Windows y Linux, así como librerías para C++ y Matlab. Desde el punto de vista de la funcionalidad la plataforma puede ser usada de diversas formas, dependiendo del módulo que se cargue en la FPGA:

- **Datalogger**: permite capturar, reproducir y descargar a un PC secuencias de eventos AER.
- **Generator:** permite generar secuencias de eventos AER a partir de una imagen digital almacenada en un PC. Existen diferentes métodos para convertir una imagen digital en secuencias AER (Alejandro Linares-Barranco et al. 2006); algunos de ellos implementados en hardware (F. Gomez-Rodriguez et al. 2005).
- **FrameGabber:** permite reconstruir en forma de fotogramas una secuencia de eventos AER.
- **Mapper:** permite en tiempo real modificar las direcciones de los eventos AER, siendo de gran utilidad para construir sistemas AER a partir de componentes con espacios de direcciones diferentes; para implementar sistemas de transmisión de probabilística, etc.

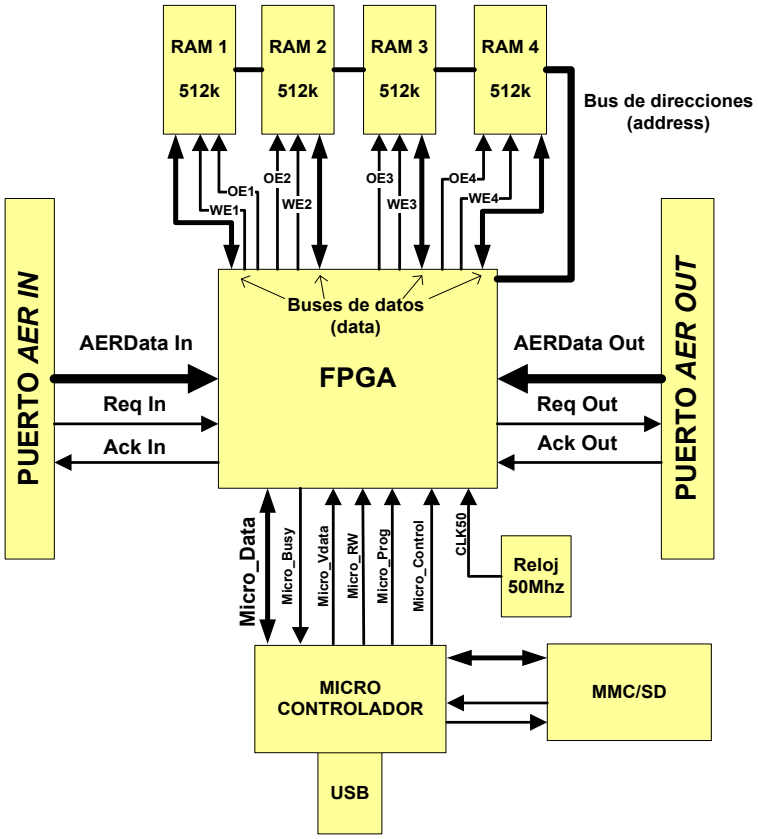


Figura 6.2: Interconexión de los componentes de la plataforma USB-AER

Para realizar nuevos módulos para la USB-AER es necesario conocer como es la interconexión de cada uno de los componentes de la plataforma y la comunicación entre ellos; la Figura 6.2 muestra la interconexión de todos los componentes así como el interfaz que han de tener los nuevos módulos.

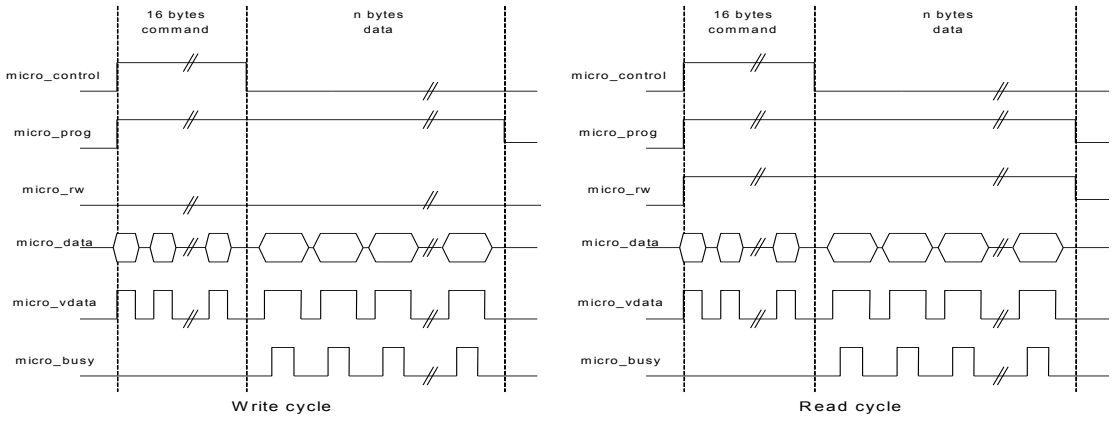
La utilización de los puertos AER de entrada y salida no presenta ninguna peculiaridad que haya que destacar, salvo la necesidad de activar unos buffer de protección que existen en las líneas de datos. El uso de la SRAM sólo requiere cumplir los tiempos detallados en las especificaciones (SRAM_Datasheet 2010). El reloj externo es de 50Mhz que puede ser

multiplicado internamente en la FPGA, ya que la mayoría de los módulos usan un reloj interno de 100Mhz.

La comunicación con el microcontrolador que proporciona la conexión USB tiene algunas peculiaridades que se detallan a continuación. El significado concreto de las señales entre la FGPA y el microcontrolador son:

- micro_data : señal bidireccional por la que se envían los datos.
- micro_vdata: indica si hay un dato válido en el bus de datos (*micro_data).*
- micro_control: indica si en el bus de datos (*micro_data)* hay un dato o un comando.
- micro_prog: señal que indica que la comunicación está activa.
- micro_rw: señal que indica si la comunicación es de lectura o escritura en la FPGA.
- micro_busy: señal que indica que la FPGA está ocupada y que la comunicación debe esperar.

Dependiendo del sentido de los datos, es decir, si los datos van hacia la USB-AER o hacia el PC, hay dos tipos de ciclos de comunicación entre el microcontrolador y la FPGA: ciclo de escritura y ciclo de lectura, respectivamente.



a) b)

Figura 6.3: Comunicación entre la FPGA y el microcontrolador: a) ciclo de escritura, b) ciclo de lectura

La Figura 6.3.a muestra el cronograma del ciclo de escritura, este ciclo se usa cuando los datos van desde el PC a la USB-AER. Tanto en este ciclo como en el de la Figura 6.3.b hay una fase inicial de comandos; consistentes en 16 bytes que siempre se envían desde el microcontrolador a la FPGA y son usados para modificar los parámetros de los módulos de la FPGA. Tras la fase inicial de comandos, comienza la fase de envío de datos; donde cada pulso de la señal *micro_vdata* indica que hay un nuevo dato disponible; el número de bytes a transmitir no está limitado y dependerá de la aplicación en el PC. Como se puede observar el ciclo de lectura es análogo al de escritura con la única diferencia del sentido de los datos y del valor de la señal *micro_rw*.

A continuación se detalla la implementación hardware realizada de cada una de las celdas presentadas, de la circuitería de testeo, depuración y pruebas, y de un módulo para la USB-AER, el *BackGroundActivityFilter*, que permite eliminar el ruido de fondo que tiene la salida de la retina usada en este trabajo.

El consumo eléctrico de la plataforma USB-AER antes de configurar la FPGA es de unos 200mA cuando se alimenta a 6.27 V, la plataforma trabaja internamente a 3.3V para los puertos de AER y los bloques de entrada salida de la FPGA y a 2.5V para el núcleo de la FPGA, por lo que gran parte del consumo se emplea en reducir la tensión de entrada.

6.2. Implementación hardware del BackGround Activity Filter

Para eliminar parte del ruido de la salida de la retina se usa un módulo de la USB-AER, que no estaba contemplado en las especificaciones iniciales de la plataforma, llamado *BackGroundActivityFilter*. Este módulo es la versión hardware de un filtro del mismo nombre que puede encontrarse en (JAER 2011). Esta versión del filtro fue desarrollada por el autor de este trabajo durante una estancia en el *Institute of Neuroinformatics* de Zurich, centro al que pertenece Tobias Delbruck, diseñador de la versión software del filtro y de la retina.

El ruido de fondo que presenta la retina es debido al *mismatch*[39] en el proceso de fabricación del chip de la retina; dicho ruido consiste en la aparición de eventos, procedentes de píxeles aislados, que no corresponden con un cambio de contraste en la escena.

La suposición más importante para tratar este ruido es que los píxeles de la retina no emiten eventos de manera aislada, sino siempre es un grupo de píxeles los que emiten de forma más o menos simultánea. Para eliminar este ruido, cada vez que se recibe un evento ha de compararse la frecuencia de eventos del píxel con sus vecinos[40], de manera, que si la frecuencia de eventos es semejante a alguno de sus vecinos se considera que ese evento es válido, por el contrario si la frecuencia de eventos no es parecida a la de sus vecinos se considera que el evento es ruido y es eliminado de la secuencia AER. La razón de semejanza vendrá determinada por un parámetro *dt* que será el tiempo máximo que puede haber entre el evento actual y último evento recibido procedente de algún vecino.

El método así enunciado, requeriría obtener la frecuencia de eventos de cada píxel, almacenarla de alguna manera y realizar 8 comparaciones con los píxeles vecinos, y eso cada vez que se recibe un evento.

Para evitar tantas comparaciones, el algoritmo consiste en los siguientes pasos:

1.- Cuando llega un evento se marca temporalmente con el valor de tiempo actual: t_1 y ese valor se almacena en todos los vecinos del píxel, sólo en los vecinos y no para sí mismo.

2.- Se compara la marca temporal del evento que acaba de llegar con la información almacenada para ese píxel: t_0, (en virtud del punto 1, esta información habrá sido

---

[39] Entiéndase por *mismatch* las discrepancias entre el diseño y el circuito final que aparecen en el proceso de fabricación de circuitos analógicos integrados, que pueden provocar error, mal funcionamiento o como es este caso un ruido aleatorio.

[40] En tratamiento de imágenes se entiende por vecinos los píxeles que rodean a uno dado. Se puede definir la 4-vecindad, donde los vecinos son los que están situados arriba, abajo, a la izquierda y a la derecha; o también la 8-vecindad donde los vecinos serán los 8 píxeles más cercanos. En este caso se usa la 8-vecindad.

almacenada por algún vecino) de manera que: si t_1-t_0 < *dt* el evento es válido, en caso contrario se considera ruido y no se deja pasar.

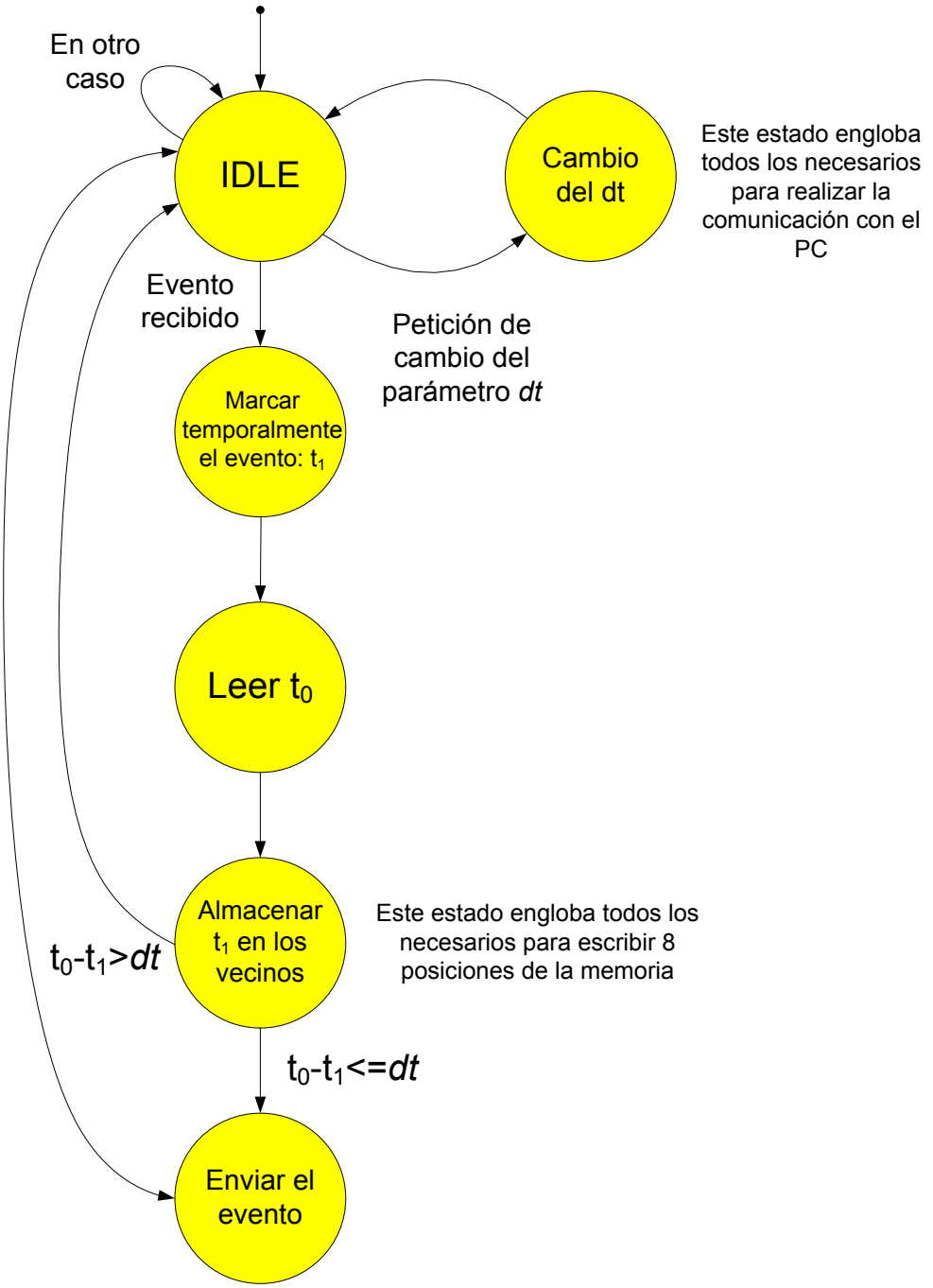


Figura 6.4: Máquina de estados del *BackGroundActivityFilter*

En la Figura 6.4 se muestra una simplificación de la máquina de estados que implementa este algoritmo en hardware (que posteriormente se describió en VHDL), y que es la base del nuevo módulo de la USB-AER. Como se puede observar en la figura tras el inicio, el filtro espera la llegada de una petición de variación del parámetro *dt*, o bien la llegada de un nuevo evento.

Tras la llegada del evento se le asigna una marca temporal, y se lee la información asociada al píxel; posteriormente se almacena el valor de marca temporal en los 8 vecinos; y por último, dependiendo del resultado de la comparación t_0-t_1<*dt,* el filtro irá la estado de *reposo (IDLE)* o procederá a enviar el evento, para terminar también en el estado de *reposo.* Los estados *Cambio de dt* y *Almacenar t_1 en los vecinos* representan en realidad varios estados, que por simplicidad no han sido incluidos en el diagrama.

Esta máquina requiere una circuitería adicional, para la comunicación con el PC a través del puerto USB, con el objetivo de poder configurar el parámetro *dt*. Esta circuitería es muy parecida a la que se explicará más adelante para realizar la comunicación entre el sistema de seguimiento y el PC.

El lugar de almacenamiento será la memoria de la USB-AER que, como se ha comentado más arriba, es una SRAM de 12ns, por lo que la frecuencia del reloj del diseño será de 50 MHz (para garantizar los tiempo de la memoria). Esta frecuencia de reloj y el número de estados que ha de recorrer la máquina de estados determina la latencia mínima que tendrá el sistema. El procesamiento de cada evento necesita, sin tener en cuenta los posibles retrasos en el protocolo handshake, pasar por 23 estados en el caso de que el evento sea considerado como válido y 21 en caso contrario; lo que supone una latencia de 460 ns y de 420 ns, respectivamente. Teniendo en cuenta que la retina puede emitir como máximo 1Mevento/s, lo que supone un evento cada 1 us; el filtro no supone un retraso en el sistema. La latencia máxima no es posible calcularla puesto que depende de los posibles retrasos en el protocolo *handshake*, tanto para el dispositivo conectado a la entrada como a la salida.

Este módulo hace una utilización de la FPGA del 7%, lo que significa el uso de 179 slices.

Para la obtención del consumo se midió el consumo del núcleo de la FPGA de la plataforma USB-AER, (el núcleo está alimentado 2.5V), con la FPGA sin configurar y posteriormente se midió el consumo una vez configurada y en funcionamiento, la resta de ambas medidas es el consumo del diseño. Para este caso 11.1mA.

6.3. Implementación de la CMCell

La implementación de la CMCell contempla todos los elementos descritos en el apartado 3.5 salvo el cambio automático del campo de visión en función del tamaño del objeto; la implementación de la CMCell sólo tiene 2 tamaños posibles, uno inicial con un área de 96x96 píxeles centrada en la imagen, y el de trabajo de 10x10 píxeles; el umbral alrededor del campo de visión también es fijo y su valor es 3 píxeles; el número de eventos para el cálculo de la posición es 2 (por tanto el valor de n en la ecuación (3.8) es 2).

El tiempo máximo que la CMCell espera en una determinada localización la llegada de eventos antes del reinicio es de 100ms. Aunque se puede modificar y ajustar en tiempo de diseño no es configurable una vez sintetizado el código VHDL.

Otro parámetro que se fija en la implementación es la cantidad de eventos que se han de recibir antes de comenzar a emitir valores de posición, que es fijo a 10 eventos, es decir, hasta que no se reciben 10 eventos no se calcula la posición de objeto ni se emiten eventos de posición.

El interfaz de la implementación de la CMCell se muestra en la Figura 6.5 y consta de:

- Señales de sistema: las señales de entrada de reset (*rst*) y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.
- Un puerto AER de entrada (*ACKi*, *REQi*, *ADDi*) por donde recibirá la secuencia AER a procesar.
- Un puerto AER de salida (*ACKo*, *REQo*, *ADDo*) por donde se enviarán los eventos rechazados.

- Un puerto AER de salida (ACKp, REQp, ADDp) por donde se enviarán los eventos procesados.
- Un puerto AER de salida (*ACKnd*, *REQnd*, CM*)* por donde se enviará el resultado del cálculo de la posición de objeto.

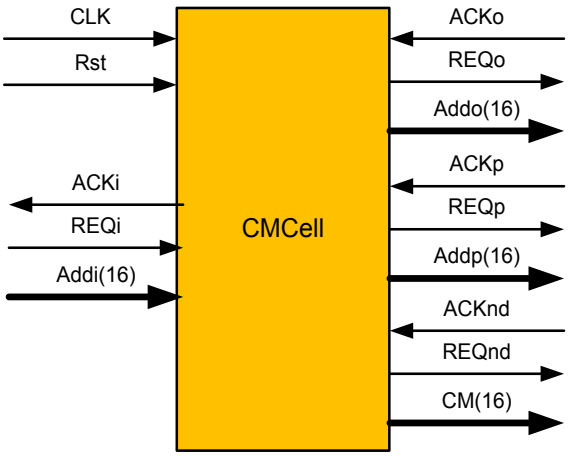


Figura 6.5: Interfaz de la implementación de la CMCell

El cálculo de la latencia teórica[41] es bastante complejo puesto que depende de muchos factores. Se podría calcular la latencia teórica medida desde que se recibe el primer evento correspondiente a un objeto y ésta emite un evento con la posición del mismo, pero dado que antes de calcular la posición del objeto hay que esperar la recepción de 10 eventos y no se sabe a priori a qué velocidad llegan esos eventos, no es posible calcularla.

[41] En el apartado de la implementación del ObjectTracker se ofrecerá, para varias configuraciones, el valor de la latencia real del sistema, medida directamente en la plataforma USB-AER, desde el momento que se envía el primer evento de un objeto hasta que se envía el primer valor de posición y velocidad.

Se podría asumir que se empieza a calcular eventos desde el primer evento, pero para el cálculo de la posición se necesitan un evento positivo y otro negativo, y de nuevo no es posible saber a priori cuando llegarán esos eventos.

La única situación en la que se puede dar un valor de latencia es aquella en la que ya se está siguiendo a un objeto y se actualiza el valor de la posición después de la llegada de un nuevo evento; en ese caso la máquina de estados de la implementación tiene que dar 8 pasos (Figura 3.13) , ignorando los posible retrasos del protocolo AER, lo que representan 80ns.

También se puede calcular la latencia en el envío de los eventos rechazados, en este caso la máquina de estados tiene que dar 3 pasos, lo que supone, si ignoramos los retrasos de protocolo AER, 30 ns.

La latencia para el envío de los eventos procesados es de 20ns, 2 pasos de la máquina de estados.

El coste hardware de la implementación de la CMCell es de 201 slices, que representa un 8% de la FPGA de la USB-AER.

6.4. Implementación de la VCell

La implementación de la VCell se ajusta a la máquina de estados descrita en el apartado 3.6, la única diferencia es que no se usan 8 valores para calcular la velocidad, solo se usan 2: la posición anterior y la actual; este cambio está motivado por el retraso en la estimación de la velocidad que supone usar 8 valores.

En lo que respecta al cambio automático del tiempo de integración para estimar la velocidad, se han definido 15 valores posibles: 2s, 1s, 500ms, 200ms, 100ms, 50ms, 10ms, 5ms, 1ms, 500us, 100us, 50us, 10us, 5us, 1us. El cambio de valor se realiza de la siguiente manera, si el valor de la variación de la posición obtenida, en alguna de sus componentes, es mayor que 15 píxeles se sube un tramo, por lo que el valor de integración será menor;

en caso contrario si el valor de cambio es 1 píxel, se sube un tramo, con lo que el valor de integración será mayor.

El valor de integración determina la frecuencia de pulso que se emite, aunque para facilitar la depuración y las pruebas, se emitirá con el valor de la variación de la posición y posteriormente será representado en la pantalla del PC.

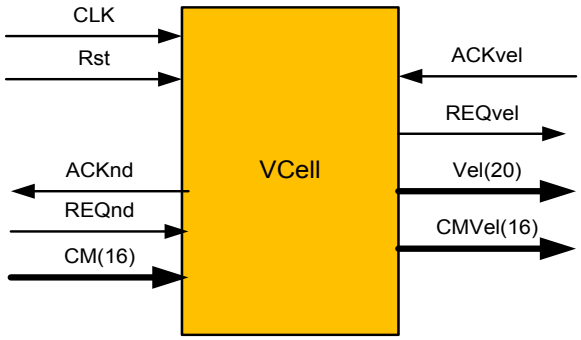


Figura 6.6: Interfaz de la implementación de la VCell

La Figura 6.6 muestra el interfaz de la implementación de la VCell que consta de:

- Señales de sistema: las señales de entrada de reset (*rst*) y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.
- Un puerto AER de entrada (*ACKnd*, *REQnd*, *CM*) por donde recibirá la secuencia AER de las sucesivas posiciones del objeto que está siguiendo la CMCell, a la que la VCell acompaña.
- Y un puerto AER de salida (*ACKvel*, *REQvel*, *Vel*, *CMVel*), por donde se enviarán los valores de la última posición usada para estimar el velocidad (*CMvel*) y el valor de la velocidad, separada por componentes (16 bits), y el valor de integración usado para estimar la velocidad (4bits).

La latencia teórica de la VCell, sin tener en cuenta los posibles retrasos del protocolo AER, dependerá del valor de integración usado para la estimación de la velocidad al que habrá que añadirle 100 ns, correspondiente a los 10 pasos que ha de dar la máquina de

estados. Estos 100 ns son despreciables frente al mínimo tiempo de integración cuyo valor es 1us, y que por lo tanto es la separación mínima de dos eventos consecutivos de estimación de la velocidad.

El coste hardware de la implementación de la VCell es de 109 slices, que representa un 4% de la FPGA.

6.5. Implementación de la PRCell

Al igual que en los casos anteriores, la implementación de la PRCell se ajusta en gran medida a la descripción funcional detallada para la celda (ver apartado 4.5.1). El modelo de tamaño de patrón es de 8x8 píxeles, especificado en un parámetro de configuración llamado *patrón*. El *umbral de disparo de entrada,* en principio, es configurable; pero tras numerosas pruebas se ha podido comprobar que el mejor valor es 1, es decir, dispara al recibir tan solo 1 evento.

En la implementación de la PRCell, el aumento o disminución del potencial se realiza de la siguiente manera: si se dispara una entrada marcada con un uno[42] en el patrón, el potencial de la celda se aumentará en un 1, al contrario, si se dispara una entrada marcada con un cero, se disminuirá en 1. Para evitar el paso por cero y la posibilidad de tener un valor de potencial negativo, se ha fijado el valor de umbral en reposo igual a 64, de manera que si todas las posiciones del patrón deseado estuvieran a 0, y la secuencia de entrada fuera tal que hay eventos en todas la posiciones, el valor de potencial de la celda sería 0; en caso contrario, un patrón con todas las posiciones a 1 (y una secuencia con eventos en todas las posiciones) el potencial de la celda valdría 128.

Se ha comprobado empíricamente que el valor *umbral de potencial* mejor es:

---

[42] Recordemos que en el patrón un valor igual a 0 indica que no deben haber eventos en esa posición, y al contrario un valor igual a 1 indica que esperamos que haya eventos en esa posición.

$$umbral\ de\ potencial = 64 + \frac{n}{2} \tag{4.22}$$

donde n es el número de unos en el patrón de la PRCell.

Con este valor de umbral, no es necesario que disparen todas las entradas presentes en el patrón, dejando un poco de margen a posibles errores en el sensor o en la obtención del patrón, que hagan que la entrada no se ajuste perfectamente al patrón buscado. Este mecanismo además está presente en la biología, según el cual no es necesario ver un objeto en su totalidad para reconocerlo como tal, aunque en este proceso se cree que influye también la experiencia, el recuerdo y el aprendizaje.

El último de los parámetros de la PRCell es el *tiempo de reseteo,* el cual también se ha comprobado empíricamente que un valor adecuado es 20 ms.

Por lo tanto de todos los parámetros de la PRCell, el único que es susceptible de ser alterado de una configuración a otra es patrón.

El interfaz de la PRCell se muestra en la Figura 6.7, que consta de:

- Las señales de sistema *clk* y *rst*.
- Un puerto AER de entrada (*ACKi*, *REQi*, *ADDi*) por donde recibirá la secuencia AER a procesar, que como ya se ha visto más arriba estará conectado al puerto de eventos procesados por el CMCell.
- Un bus para obtener la posición actual de objeto (*CM*), que permite centrar los eventos que se reciben.
- Un puerto AER de salida (*ACKo*, *REQo*, *ADDo*) por donde se enviarán los eventos rechazados. La finalidad de esta puerto es la depuración del sistema y permite comprobar si la celda está procesando todos los eventos, de manera que se puede detectar posibles errores de sincronización entre la CMCell y la PRCell.

- Un puerto AER de salida (*ACKnd*, *REQnd*, *PRnd)* por donde se enviarán el resultado de la detección. Actualmente el valor que se envía cada vez que se detecta un patrón es fijo. La intención futura es enviar el grado de confiabilidad de la detección, aunque en cierto modo esta información ya está codificada en la frecuencia de eventos de la PRCell.

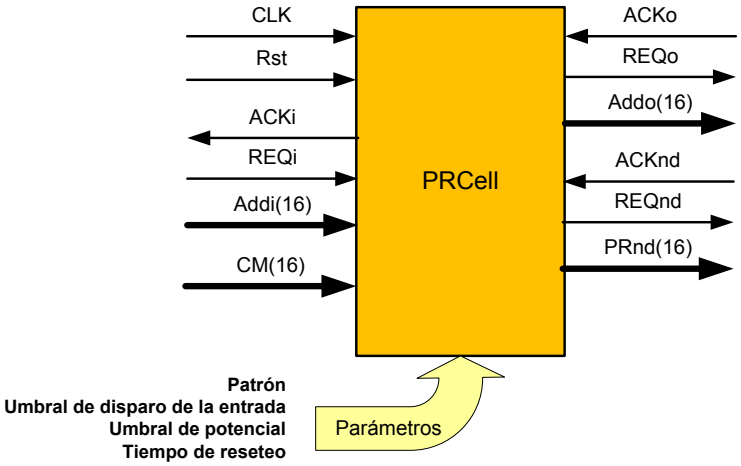


Figura 6.7: Interfaz de la implementación de la PRCell

La implementación de la PRCell incluye una pequeña memoria de simple puerto donde se almacenan la cantidad de eventos que se han recibido por cada entrada, con el objetivo de comprobar si se han recibido suficientes eventos para considerar que dicha entrada ha disparado (supera el umbral de disparo de entrada); aunque como ya se ha comentado más arriba el umbral de disparo de entrada es 1, por lo que dicha memoria no se ha utilizado en los ensayos. Sin embargo, no se ha eliminado de la implementación dado que futuros usos de la PRCell podrían requerir su utilización.

La latencia de la PRCell (sin tener en cuenta los retrasos en el protocolo handshake) para procesar un evento, depende de lo que ocurra con dicho evento; si es rechazado, la latencia es de 4 estados de la máquina de estado lo que significan 40 ns; si es aceptado pero ese evento no implica un disparo de la celda, son 10 estados, 100ns; y por último si el evento provoca un disparo de la celda son 10 estados, 100 ns, a lo que se añade el tiempo

de reinicio, que consiste en borrar la memoria interna de 64 posiciones; en borrar cada posición se tarda 1 ciclo, lo que supone 640ns, en total 740 ns.

El consumo hardware de la PRCell, incluida la memoria para los eventos, es de 73 slices, lo que supone un 3%: la memoria ocupa 1 bloque (de los 14 disponibles en la FPGA) lo que implica un 7% de ocupación.

6.6. Implementación de la TrackCell

La realización 5.2 de la TrackCell responde a lo tratado en el apartado, solo hay que añadir un parámetro que proporcione un nombre a la TrackCell (*Identificador*); su interfaz se muestra en Figura 6.8 que consta de:

- Señales de sistema: las señales de entrada de reset (*rst*) y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.
- Un puerto AER de entrada (*ACKi*, *REQi*, *ADDi*) por donde recibirá la secuencia AER a procesar.
- Un puerto AER de salida (*ACKo*, *REQo*, *ADDo*) por donde se enviarán los eventos rechazados.
- Un puerto AER de salida (*ACKnd*, *REQnd*, *CM, DATA)* por donde se enviará el resultado del cálculo de la posición de objeto *(CM*), el contenido del bus *DATA* dependerá del tipo de TrackCell, se verá más adelante.

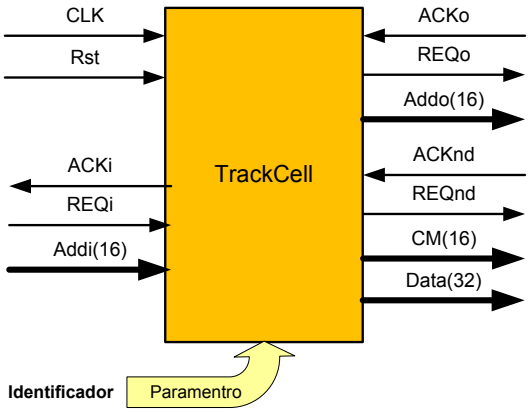


Figura 6.8: Interfaz de la implementación de la TrackCell

En este trabajo se han implementado dos tipos de TrackCell el tipo B y el tipo C.2.

Implementación de la TrackCell tipo B

El tipo B es la interconexión de una CMCell y una VCell (se describió en el apartado 5.2.2). Su implementación no presenta ninguna peculiaridad destacable, salvo el contenido del bus Data que en este caso contiene los 20 bits de la salida de la VCell (20 bits) y el identificador de la TrackCell (8 bits), en total 28 bits de los 32 posibles.

La latencia teórica (sin tener en cuenta los retrasos de comunicación del protocolo handshake) es el resultado de la combinación de las latencias de la CMCell y la VCell, y dependerá de lo que ocurra con el evento:

- Si el evento es rechazado por la CMCell la latencia será de 30 ns.
- Si el evento es procesado por la CMCell pero la nueva posición del objeto obtenida no es tenida en cuenta por la VCell, por que no se ha cumplido su tiempo de integración, la latencia será de 80 ns de la CMCell más 20 ns de comunicación del dato a la VCell aunque ésta lo descarte; total 100 ns.
- Si el evento es procesado por el CMCell y la nueva posición es usada para estimar la posición, la latencias será 80 ns de la CMCell más 100 ns de la VCell (esto ya incluye el tiempo de comunicación entre la CMCell y la VCell); total 180 ns.

El consumo hardware de la TrackCell tipo B es de 306 slices, un 13% de la FPGA.

Implementación de la TrackCell tipo C.2

La TrackCell de tipo C consta de una CMCell, una VCell y una PRCell interconectadas tal y como se detalla en el apartado 5.2.4. En este caso el contenido del bus *Data* es 20 bits de la salida de la VCell, 8 bits de la identificación y 1 bit como salida de la PRCell que indica si se ha detectado o no el patrón; en total 29 bits.

La composición de la salida la lleva a cabo un bloque llamado *fusionC.2*, que funciona de la siguiente manera: emite un evento por cada evento que recibe de la VCell y en el bus *data* pondrá un 1 en el bit reservado para el patrón si entre dos eventos consecutivos procedentes de la VCell recibió por lo menos un evento procedente de la PRCell, en caso contrario pondrá un 0. Este bloque de fusión supone 2 pasos de una máquina de estado, lo que significan 20 ns de latencia adicionales que habría que sumar sólo para la estimación de la velocidad.

La latencia total del sistema será, de nuevo la combinación de las latencias de las 3 celdas involucradas y dependerá de lo que pase con el evento que se acaba de recibir:

- Si el evento es rechazado por la CMCell la latencia será 30 ns.
- Si el evento es aceptado por la CMCell y la nueva posición no es usada para estimar la velocidad y a su vez el evento es rechazado por la PRCell, la latencia será 80ns + 40 ns que tarda la PRCell en rechazar un evento (los 20ns que tarda la VCell en no aceptar el dato se ven enmascarados por los 40ns de la PRCell); en total 120 ns.
- Si el evento es aceptado por la CMCell y la nueva posición es usada por la VCell y el evento es rechazado o aceptado por la PRCell pero sin disparar, la latencia será de 80 ns + 100 ns de la VCell +20 ns de la fusión (el tiempo en rechazar o bien procesar el evento por la PRCell, se ve enmascarado por el tiempo de la VCell); en total 190 ns.

- Si el evento es aceptado por la CMCell y también por la PRCell (e independientemente de la VCell), la latencia será de 80 ns + 740 ns de la PRCell; en total 820 ns.

Por último, el consumo hardware de la TrackCell de tipo C.2 es de 398 slice, un 16% de la FPGA y 1 bloque de RAM para la memoria de la PRCell.

6.7. Implementación del ObjectTracker

Dado que sólo se han implementado las TrackCell de tipo B y de tipo C.2, sólo es posible la implementación de ObjetcTrackers que contengan estos tipos de TrackCell. Debido a la capacidad de la Spartan II 200 se han implementado un ObjectTracker6B y un ObjectTracker4C.2, que contienen 6 TrackCells de tipo B y 4 TrackCells de tipo C2 respectivamente.

El interfaz de un ObjectTracker se muestra en la Figura 6.9 y consta de:

- Señales de sistema: las señales de entrada de reset (*rst*) y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.
- Un puerto AER de entrada (*ACKi*, *REQi*, *ADDi*) por donde recibirá la secuencia AER a procesar.
- Un puerto AER de salida (*ACKo*, *REQo*, *ADDo*) por donde se enviarán los eventos del resultado de la detección y seguimiento de objetos. Esta secuencia AER de salida es la fusión de todos los eventos producidos por todas las TrackCells de las que consta el ObjectTracker. Para realizar dicha fusión es necesario un arbitrador tal y como se detalló en el apartado 5.3.

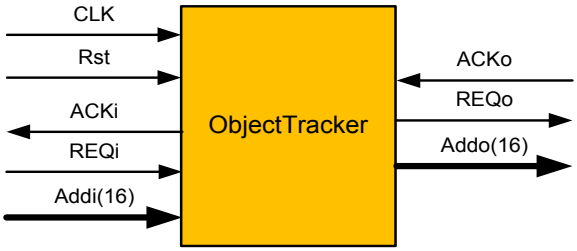


Figura 6.9: Interfaz de la implementación de un ObjectTracker

El interfaz de la Figura 6.9, se corresponde con el ObjectTracker en fase de explotación; no obstante para poder realizar la depuración de errores y las pruebas del sistema, el interfaz se ha modificado para permitir que la salida del sistema sea almacenada en la plataforma USB-AER (en la SRAM) y posteriormente descargada al PC con el objetivo de ser analizada y contrastada con la salida teórica del sistema.

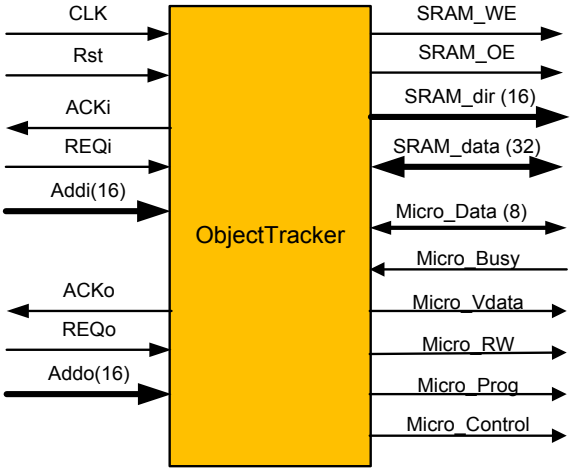


Figura 6.10: Interfaz de depuración y pruebas de la implementación del ObjectTracker

Así pues, se ha implementado un interfaz del ObjectTracker para depuración y pruebas; que se muestra en la Figura 6.10 y consta de la mayoría de las señales de entrada y salida de la FPGA de la plataforma USB-AER, que son:

- Señales de sistema: las señales de entrada de reset (*rst*) y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.

- Un puerto AER de entrada (*ACKi*, *REQi*, *ADDi*) por donde recibirá la secuencia AER a procesar; este puerto estará conectado al puerto AER de entrada de la USB-AER
- Un puerto AER de salida (*ACKo*, *REQo*, *ADDo*) por donde se enviarán los eventos rechazados por todas las TrackCells, este puerto estará conectado al puerto AER de salida de la USB-AER. Esta información puede ser fácilmente recogida por otra USB-AER configurada como *FrameGrabber* de manera que se puede observar que eventos no está procesando el sistema.
- Las señales de control de la SRAM de la USB-AER (*SRAM_WE*, *SRAM_OE*, *SRAM_dir* y *SRAM_data*), permiten al sistema guardar toda la información de procesado en el memoria RAM de la USB-AER.
- Las señales de comunicación con el microcontrolador que hace de puente USB con el PC, (*micro_data , micro_vdata , micro_control, micro_prog, micro_rw, micro_busy*) permite la descarga al PC de toda la información almacenada en la SRAM con el objetivo de analizar dicha información para comprobar el correcto funcionamiento del sistema.

La arquitectura interna del ObjectTracker para depuración y pruebas se muestra en la Figura 6.11, donde se observa como la información del procesado es enviada a la SRAM de la USB-AER haciendo uso de un controlador de RAM, y que posteriormente puede ser enviada al PC a través de un bloque de comunicación con el microcontrolador que hace de puente con el USB.

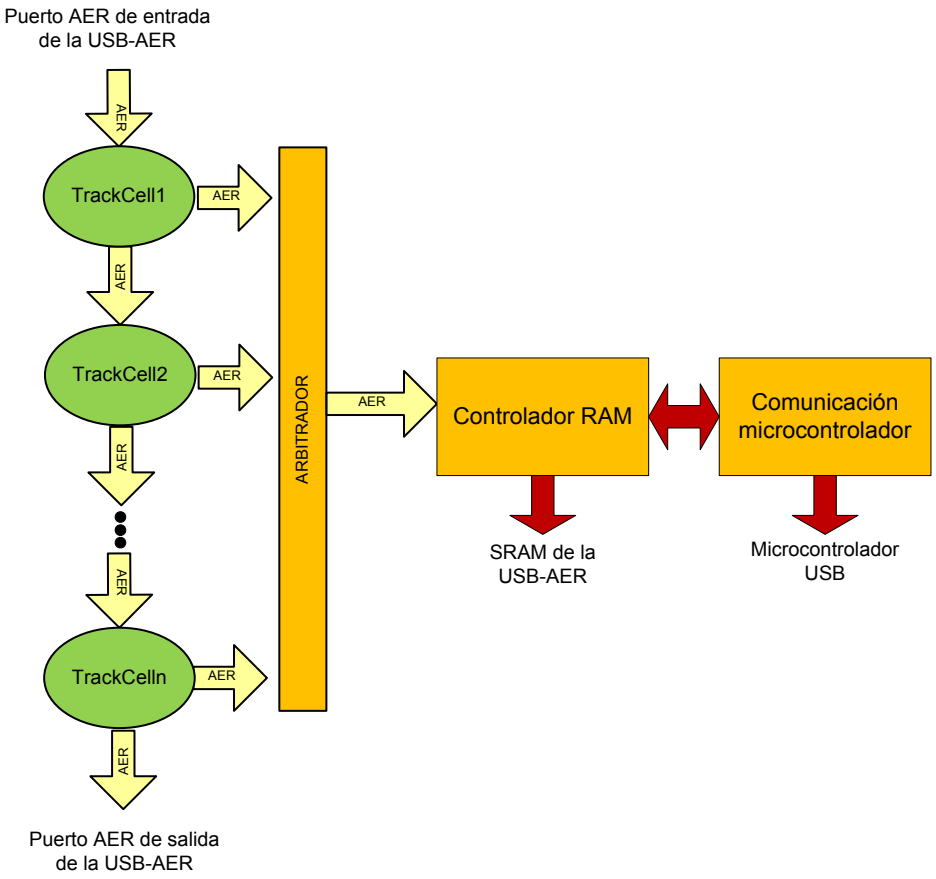


Figura 6.11: Estructura interna del ObjectTracker para depuración y pruebas

Los detalles de la implementación de los circuitos adicionales (arbitrador, controlador de RAM y comunicación microcontrolador) serán expuestos más adelante en este capítulo.

La implementación de los dos ObjectTrackers realizados es exactamente igual, las diferencias vendrán determinadas por el uso de TrackCells de tipos distintos que dan lugar a valores de latencia, de coste hardware y consumo de potencia distintos.

Implementación del ObjectTracker6B

Como ya se ha comentado anteriormente este ObjectTracker consta de de 6 TrackCells de tipo B, por lo que es capaz de seguir y estimar la velocidad de 6 objetos simultáneamente. La latencia teórica[43] del ObjectTracker dependerá de varias circunstancias:

1. Si el evento es procesado, la latencia dependerá del lugar que ocupa la TrackCell que lo procese en la arquitectura en cascada del ObjectTracker, y por supuesto de lo que ocurra con ese evento al ser procesado (véase la discusión sobre la latencia de la TrackCell de tipo B), Concretamente:

   Si el evento es procesado por la TrackCell colocada en la posición N:

   $$(N-1)*30+LTB \tag{4.23}$$

   Donde LTB es latencia de la TrackCell tipo B.

2. Si el evento no es procesado y es enviado por el puerto AER de salida como evento rechazado; concretamente:

   Si el evento es rechazado la latencia será 6x30 ns, es decir, 180 ns.

El ObjectTracker es la celda que se conecta con el exterior de la FPGA, lo que permite realizar una medición de la latencia real. Ésta será el tiempo que transcurre entre un evento de entrada y el correspondiente evento de salida. Para realizar la medición bajo condiciones conocidas y reproducibles se estimulará el sistema con estímulos sintéticos[44].Con ayuda de un osciloscopio se medirá el tiempo entre el primer evento del estímulo y el evento de salida correspondiente a ese evento. La Figura 6.12 muestra la captura de la pantalla del osciloscopio donde se observa en la línea superior la señal REQ del puerto de entrada del ObjectTracker y en la línea inferior la señal REQ del puerto de salida. Se ha modificado

[43] Recuérdese que los cálculos teóricos de la latencia no tiene en cuenta los posibles retrasos que se puedan producir en el protocolo handshake.

[44] Estímulos sintéticos son aquellos generados artificialmente por una USB-AER, en lugar de una retina (véase apartado 7.1 para más detalles)

levemente el diseño para garantizar que la salida siempre corresponde a la TrackCell colocada en el primer lugar en la jerarquía. El tiempo que transcurre entre las activaciones de ambas señales es de 200 ns. Sin embargo, el tiempo teórico calculado más arriba para la primera TrackCell es de 180 ns. Esta discrepancia es debida a la necesidad de incluir dos flip-flop[45] en las entradas de las máquinas de estado para eliminar el riesgo de metaestados inestables, concretamente estos flip-flop afectan a la señal REQ de entrada, y consumen 2 ciclos de reloj (de 100 Mhz), lo que suponen los 20 ns de más.

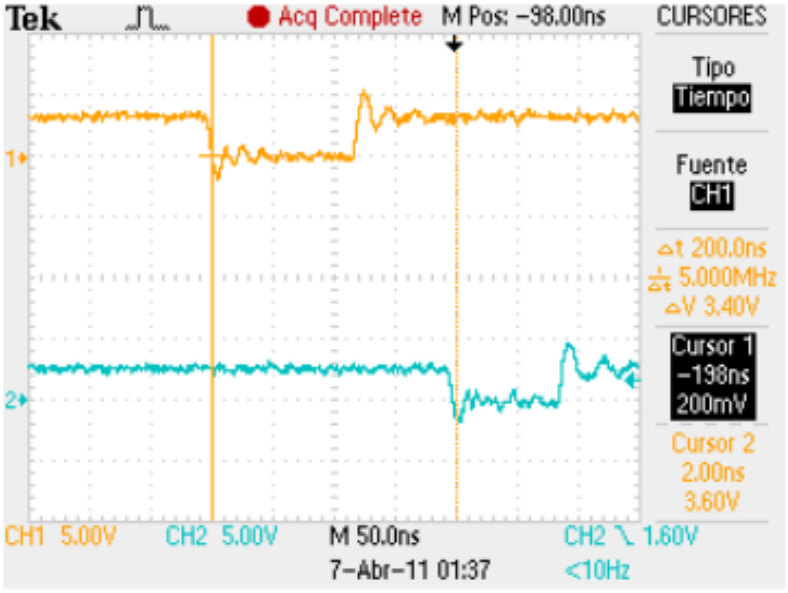


Figura 6.12:Medición de la latencia para la 1ª TrackCell de una ObjectTracker6B

La Figura 6.13 muestra la medición para la segunda TrackCell en la jerarquía, el valor es 230ns. Dato que es correcto y se ajusta a lo calculado teórica si tenemos en cuenta el tiempo extra que añaden los sincronizadores.

[45] Un flip-flop o biestable es un circuito electrónico capaz de permanecer a 0 o a 1 durante un tiempo ilimitado.

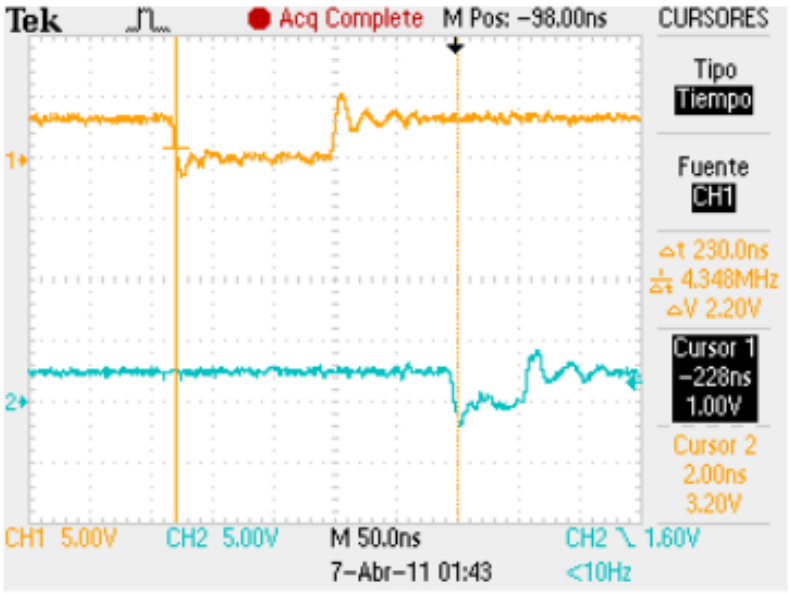


Figura 6.13: Medición de la latencia para la 2ª TrackCell de una ObjectTracker6B

Con estas las dos figuras anteriores se comprueba que los cálculos teóricos de la latencia del sistema son correctos.

El coste hardware de la implementación para depuración y prueba es de 2194 slices de la Spartan II 200, un 93% de ocupación; con el interfaz de explotación, es decir, sin la circuitería adicional (controlador de RAM y comunicación microcontrolador) es de 2056 slices, lo que supone una ocupación del 87%.

Para medir el consumo se ha empleado el mismo procedimiento que en el caso del *BackGroundActivity Filter*, arrojando en este caso un consumo de 53.1 mA (recordemos que el núcleo de la plataforma USB-AER está alimentado a 2.5 V). Esta medición del consumo se ha realizado sin tener el controlador de RAM ni el circuito de comunicación con el microcontrolador.

Implementación del ObjectTracker4C.2

El ObjectTracker4C.2 consta de 4 TrackCell de tipo C.2, con el que se es capaz de seguir y detectar el patrón de 4 objetos simultáneamente. La discusión para el cálculo de la latencia teórica es idéntica al del caso anterior (cambiando el tipo de TrackCell), la única diferencia es que en caso de ser rechazado el evento, dado que son 4 TrackCell, la latencia sería 4x30ns, 120 ns.

La Figura 6.14 muestra la latencia para la primera TrackCell en la jerarquía, 210 ns. 190ns los calculados teóricamente a los que hay que añadir los 20ns de la sincronización.

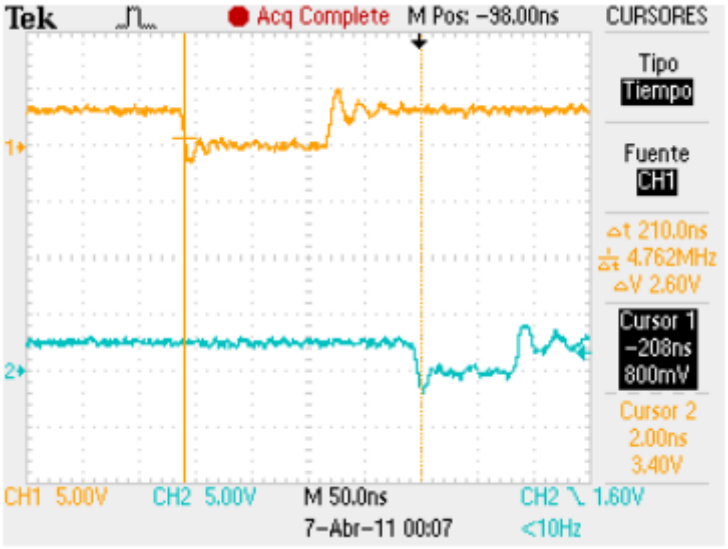


Figura 6.14: Medición de la latencia para la 1ª TrackCell de una ObjectTracker4C.2

De igual manera se observa en la Figura 6.15 el valor de la latencia para la segunda TrackCell.

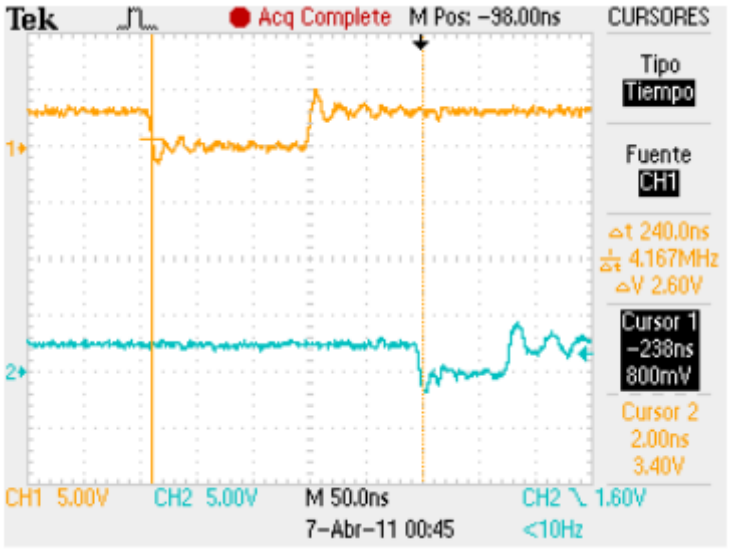


Figura 6.15: Medición de la latencia para la 2ª TrackCell de una ObjectTracker4C.2

De nuevo se comprueba que las mediciones realizadas concuerdan con los cálculos teóricos de la latencia, donde el valor medido es de 240ns, cuando el teórico, que no tiene en cuenta los sincronizadores era de 210ns.

El coste hardware de la implementación para depuración y prueba es de 2084 slices de la Spartan II 200, un 88% de ocupación; con el interfaz de explotación, es decir, sin la circuitería adicional (controlador de RAM y comunicación microcontrolador) es de 1946 slices, lo que supone una ocupación del 82%.

Para medir el consumo se ha empleado el mismo procedimiento que en el caso anterior, arrojando en este caso un consumo de 78.3 mA (recordemos que el núcleo de la FPGA de la USB-AER está alimentada a 2.5 V). Esta medición del consumo se ha realizado sin tener el controlador de RAM ni el circuito de comunicación con el microcontrolador.

6.8. Hardware para de interconexión, depuración y pruebas

Para poder realizar la interconexión, las labores de depuración y las pruebas del sistema se ha tenido que desarrollar cierta circuitería adicional, sin la cual, hubiera sido imposible la implementación hardware del sistema. El hardware desarrollado es: un arbitrador que permite multiplexar todos los eventos de salida de las TrackCells; un circuito para la comunicación con el micro-controlador de la USB-AER que permite la conexión USB con el PC; un controlador de RAM que permita usar la SRAM externa, que es de un puerto de lectura y escritura, como una memoria de doble puerto de lectura y escritura cada uno, donde almacenar la información de depuración y pruebas.

La Figura 6.16 muestra la interconexión entre estos tres elementos. La información de salida de las TrackCells es multiplexada por el arbitrador y es enviada por el controlador de RAM a la memoria de la USB-AER. Cada cierto tiempo esa información es descargada al PC a través del puerto USB y borrada de la memoria. Veamos una breve descripción de cada uno de estos elementos.

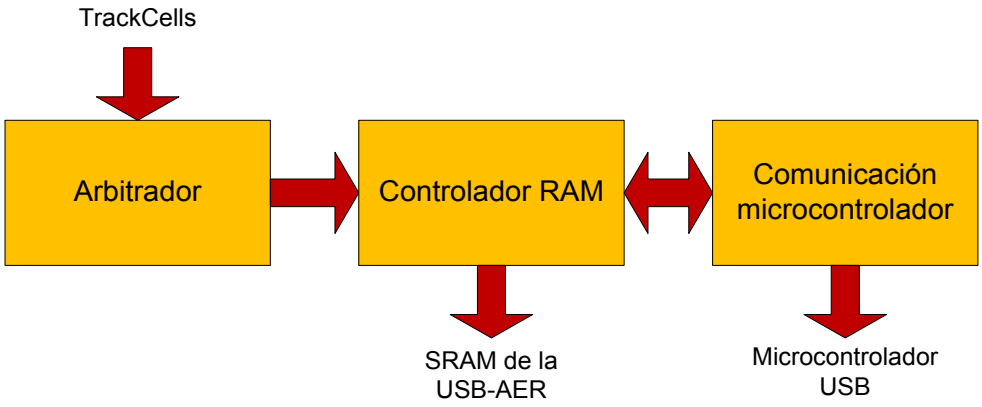


Figura 6.16: Interconexión de la circuitería adicional

6.8.1. Arbitrador

Aunque a la vista de la arquitectura propuesta, pudiera parecer que el funcionamiento del sistema es secuencial y que en cada momento se puede determinar que TrackCell va a emitir un evento, la realidad es que las CMCells computan cada evento en paralelo con el funcionamiento del resto. Esto hace que no se pueda establecer un orden a priori en el que las celdas van a emitir eventos. Por lo tanto, puede darse el caso de que dos o más TrackCells necesiten emitir un evento en el mismo momento. Para organizar la interconexión de todas las TrackCells se ha desarrollado un arbitrador con prioridades fijas, que canaliza toda la información que genera una TrackCell: la posición, la velocidad, el identificador de la celda, el periodo de integración de la VCell y la marca de detección del patrón; en total cada evento que emite una TrackCell son 48 bits. Esa información está empaquetada de la siguiente manera, la señal *diri* corresponde con la posición del objeto y en *datai* está el resto de la información.

Concretamente se ha diseñado e implementado un arbitrador con 8 puertos AER de entrada y 1 puerto AER de salida.

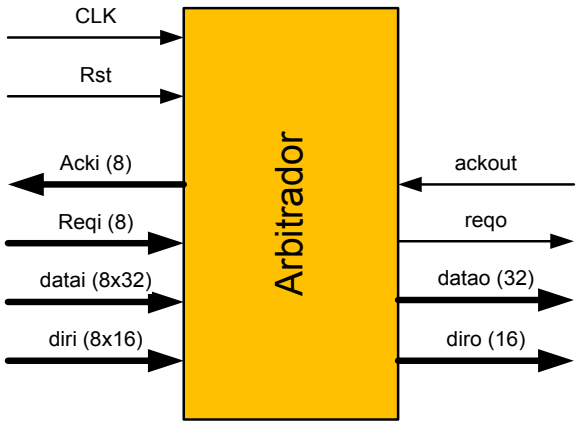


Figura 6.17: interfaz del arbitrador

En la Figura 6.17 se muestra la interfaz del arbitrador que consta de las siguientes señales:

- Señales de sistema: las señales de entrada de reset (*rst)* y de reloj (*clk*), con significado obvio de señal de reinicio y de sincronización respectivamente.

- Ocho puertos AER de entrada: El bus de entrada *diri* es de tamaño 16x8 bits, donde están conectadas las salidas *diri* de 8 TrackCells; el bus de entrada *datai* es de tamaño 32x8 bits, donde se conectan la salidas *datai* de las TrackCells; estos dos buses constituyen la señal de datos de los 8 puertos AER de entrada. El bus de entrada *reqi* es de 8 bits, donde se conectan las señales *req* de las TrackCells que es la señal de petición de transmisión de los 8 puertos AER, y por último el bus de salida *acki* de tamaño 8 bits, donde se conectan las señales *ack* de las TrackCells, que es la señal de asentimiento de los 8 puertos AER de entrada.

- Puerto AER de salida cuya señales son: los buses de salida *datao* y *diro* son de tamaño 32 y 16 bits respectivamente y constituyen la señal de datos del puerto AER, la señal *reqo* es la señal de protocolo de inicio de transmisión del puerto AER; y la señal de entrada *acko* que es la señal de asentimiento del puerto AER.

El arbitrador es una máquina de estado de tres estados que permite enviar a la salida una de las 8 entradas, como hemos visto tanto las entradas como las salidas son AER, es decir usan un protocolo handshake para transmitir el dato.

La Figura 6.18 muestra la máquina de estado del arbitrador. Desde el estado de *idle*, cuando llega algún evento por algún puerto de entrada se pasa al estado *conexión*; en el caso de que haya varias peticiones en varios puertos AER de entrada se selecciona el primero de ellos, quedando el resto para el siguiente ciclo. En el estado de *conexión* se comienza la transmisión del dato por el puerto AER de salida (activando la señal *reqo*) y se envía el asentimiento al puerto AER de entrada seleccionado; cuando se recibe el asentimiento del puerto de salida (activación de la señal *acko*) se pasa al estado *espera_ack*, donde se desactiva la señal de *req* del puerto de salida, y se espera la desactivación de las señales *ack* y *req* del puerto de salida y del de entrada seleccionado respectivamente, para pasar de nuevo al esto de *idle*.

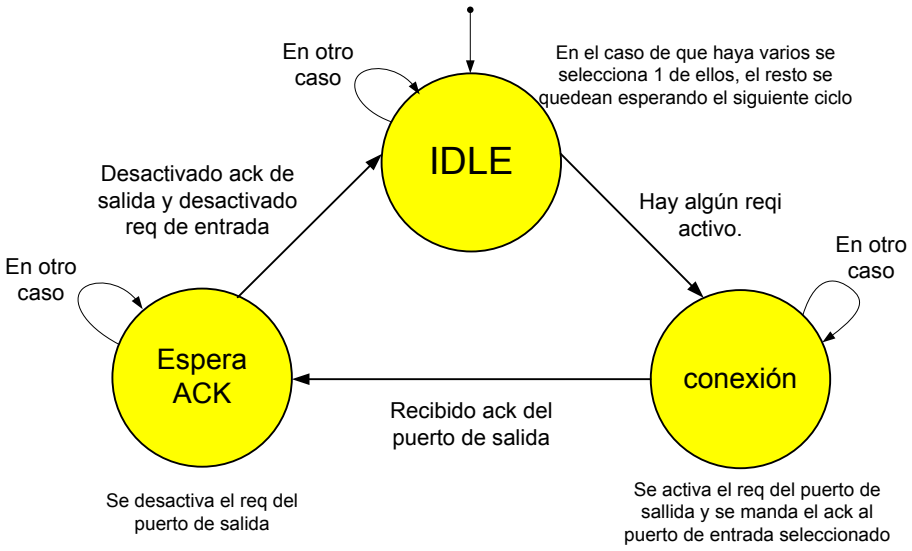


Figura 6.18: Máquina de estado del arbitrador

En el proceso de síntesis el arbitrador ocupa 241 slices de la FPGA de la USB-AER, lo que supone un 10% del total.

La latencia mínima del sistema sin tener en cuenta los posibles retrasos en el protocolo *handshake* es de 30 ns, que correspondería a un paso por cada estado. Pero en realidad, dado que el sistema esta sintetizado en la misma FPGA y con la misma señal de reloj, cada activación y desactivación de las señales de protocolo requieren un mínimo de 2 ciclos de reloj. Como son dos puertos AER con dos líneas de protocolo cada uno, son 4 señales y por lo tanto 8 ciclos de reloj, lo que supone para un reloj de 100Mhz 80 ns como mínimo. La latencia máxima es muy difícil de calcular puesto que depende de la carga del sistema y de la posición de la entrada en el esquema de prioridades.

Si todas las TrackCells emitieran un evento al mismo tiempo, (y sin tener en cuenta retrasos adicionales en el *handshake*), la latencia de la TrackCell conectada en el puerto menos prioritario sería de 560 ns, y además, en ese tiempo, se puede garantizar que ninguna de las anteriores TrackCells podría emitir un segundo evento debido a que el menor tiempo de muestreo que puede soportar la VCell es de 1us (recordemos que el

tiempo de muestreo de la VCell determina la tasa máxima de eventos que emitirá la TrackCell).

6.8.2. Controlador de RAM de doble puerto

Para realizar las tareas de depuración y prueba se usa la memoria de la plataforma USB-AER, que es de un solo puerto y está directamente conectada a la FPGA que es la responsable de su control.

Para nuestros propósitos es necesario poder leer y escribir la SRAM por dos puertos; uno de ellos estará conectado a la circuitería de comunicación con el micro-controlador desde el que es necesario leer la información almacenada, así como borrarla una vez que ha sido leída. El otro puerto estará conectado a la salida del arbitrador, por donde se irá almacenando en la SRAM la salida del sistema; por este puerto sólo es necesario escribir.

Este controlador de RAM manejará la memoria SRAM de la USB-AER haciendo transparente, tanto para el arbitrador como para la circuitería de comunicación con el microcontrolador, el hecho de que la SRAM solo cuenta con un puerto.

El interfaz del controlador se muestra en la Figura 6.19 que consta de:

- Las señales de sistema para el reloj (*clk*) y el reinicio (*reset*).
- La señal de 32 bits (*AcumPeriod*) que indica el tiempo máximo que deben acumular los datos antes de una lectura por parte de PC; es decir, si el tiempo entre dos lecturas consecutivas es mayor que este valor solo se acumula el tiempo que marque esta señal desde la última vez que se leyó.
- El puerto AER de conexión con el arbitrador por el que se leerán los datos que se van a almacenar: *diro* servirá para direccionar la memoria, *datao* serán los datos que que se almacenan y que son los que ofrece la VCell.
- El puerto handshake de comunicación con la circuitería de comunicación con el microcontrolador, que aunque tiene la misma forma que un puerto AER no lo es,

pero por simplicidad y homogeneidad se han nombrado sus señales de forma análoga.

- Todas la señales para el control de la SRAM: bus de direcciones (*SRAM_dir*), bus de datos (*SRAM_data*), habilitación de salida (*SRAM_OE*), señal de escritura (*SRAM_WE*).

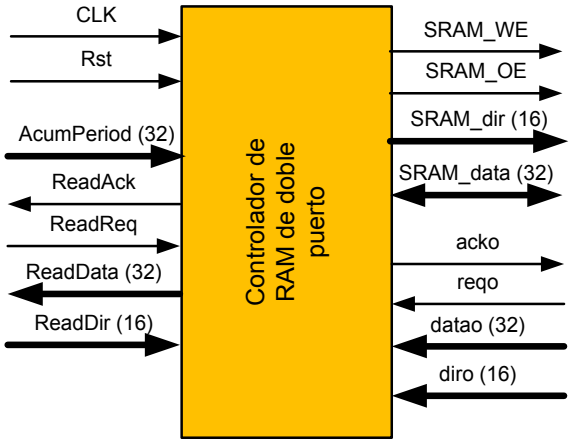


Figura 6.19: Interfaz del controlador de RAM de doble puerto

La máquina de estados del controlador de la de RAM se muestra en la Figura 6.20. Como se puede observar tiene 3 ciclos distintos.

El ciclo de escritura en el que se escriben los datos procedentes de las TrackCells a través del arbitrador.

Y el ciclo de escritura falsa en el que se da el asentimiento a la escritura pero sin escribir nada en la RAM. Este ciclo acurre cuando se ha agotado el tiempo de acumulación especificado en la señal *AcumPeriod*.

En los estados de escritura y lectura de SRAM se colocarán al valor conveniente las señales que controlan la SRAM

El de lectura que consta de 2 estados más el de reposo, recibe las peticiones de lectura por parte del USB a través de la circuitería de comunicación con el microcontrolador, cada

vez que se recibe una petición de lectura se lee la dirección solicitada, y a continuación se borra. La aplicación del PC cada vez que lee solicita todas las posiciones de memoria. Como se verá más adelante el circuito de comunicación con el microcontrolador irá solicitando consecutivamente todas las posiciones de memoria.

Esta lectura consecutiva de toda la memoria no sobrecarga el sistema debido a la diferencia de ancho de bus entre los datos que se leen de la memoria de 32 bits y el de comunicación con el microcontrolador de 8 bits. Esto quiere decir que por cada lectura de la SRAM el circuito de comunicación con el microcontrolador debe realizar 4 envío de datos; por lo tanto entre cada dos lecturas hay tiempo suficiente para atender las escrituras. Además hay que tener en cuenta que la frecuencia del reloj del micro es de 24 Mhz y la de el controlador de RAM de 50Mhz, lo que permite contar con más tiempo para las escrituras. Además en la codificación VHDL del sistema se ha dado prioridad a las escritura de datos provenientes de las TrackCells.

Por todo lo anterior la latencia máxima para una lectura, sin tener en cuenta los retrasos en el *handshake*, será de 80 ns, puesto que leer en realidad implica dos ciclos y borrar los datos también. Para la escritura la latencia máxima, sin tener en cuenta los retrasos del *handshake*, es de 40 ns, puesto que nuevamente la escritura necesita dos ciclos. En el caso de escritura falsa, la latencia se reduce a 20 ns (sin tener en cuenta, de nuevo los posibles retrasos de *handshake*).

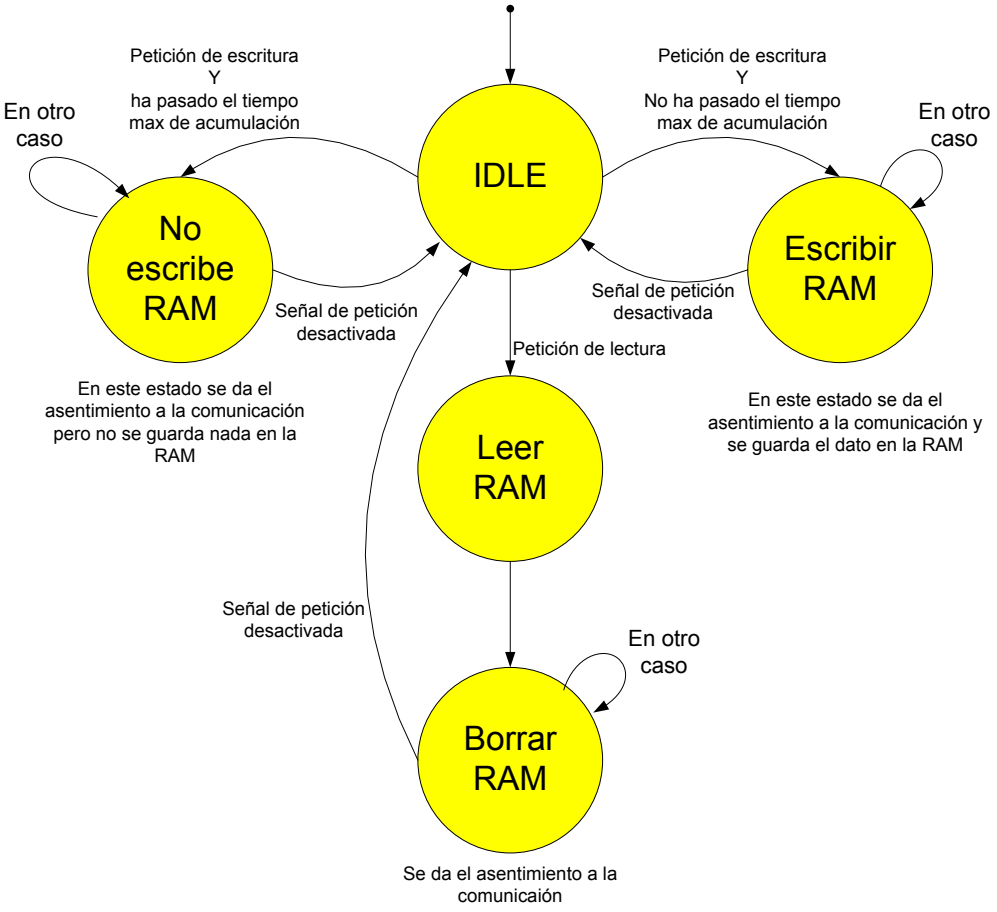


Figura 6.20: Máquina de estado del controlador de RAM de doble puerto

El coste en hardware de este controlador de RAM es de 66 slices, lo que representa un 2% de la FPGA de la plataforma USB-AER.

6.8.3. Comunicación con el microcontrolador

El funcionamiento del sistema de detección y seguimiento de objetos no necesita la intervención de un PC. Pero para poder realizar las labores de depuración y pruebas del sistema es necesario contar con algún mecanismo de conexión con un PC, donde poder observar lo que está ocurriendo en el interior del mismo. Para ello se ha modificado y

adaptado a nuestras necesidades la circuitería que permite la comunicación con el PC a través del puerto USB[46].

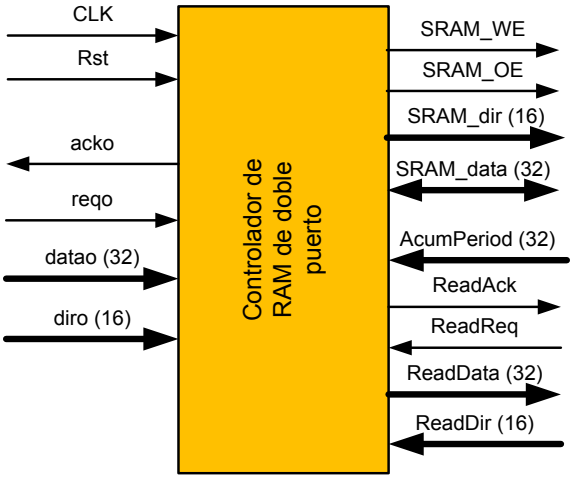


Figura 6.21: Interfaz circuitería de comunicación con el microcontrolador

Esta circuitería está determinada por el protocolo de comunicación entre la FPGA y el micro-controlador de la USB-AER (ver descripción de la plataforma USB-AER al comienzo de este capítulo). El interfaz de esta circuitería se muestra en la Figura 6.21 y consta de:

- Todas las señales de interconexión entre la FPGA y el microcontrolador de la USB-AER.
- Las señales de sistema para el reloj (*clk*) y el reinicio (*reset*).
- Una señal de 32 bits (*AcumPeriod*) que es el periodo de acumulación de datos, parámetro configurable del controlador de RAM de doble puerto que permite cambiar el tiempo de acumulación de datos.

[46] La versión original de esta circuitería fue desarrollada por los diseñadores de la plataforma USB-AER, es prácticamente idéntica en todos los módulos, con ligeras modificaciones como la que se contempla en este trabajo, consistente en la comunicación con el controlador de RAM de doble puerto.

- Un puerto de comunicación handshake con las mismas especificaciones que el AER, aunque no es un puerto AER propiamente dicho, pero por simplicidad y homogeneidad con el resto del sistema se ha optado por este protocolo de comunicación asíncrona. Este puerto estará conectado a la memoria (a través del controlador de RAM de doble puerto), permitiendo descargar al PC la información de depuración y pruebas almacenada en ella.

La circuitería está gobernada por la máquina de estados, cuyo diagrama se muestra en la Figura 6.22. Esta máquina de estados implementa el ciclo de lectura de la comunicación entre el microcontrolador y la FPGA de la USB-AER.

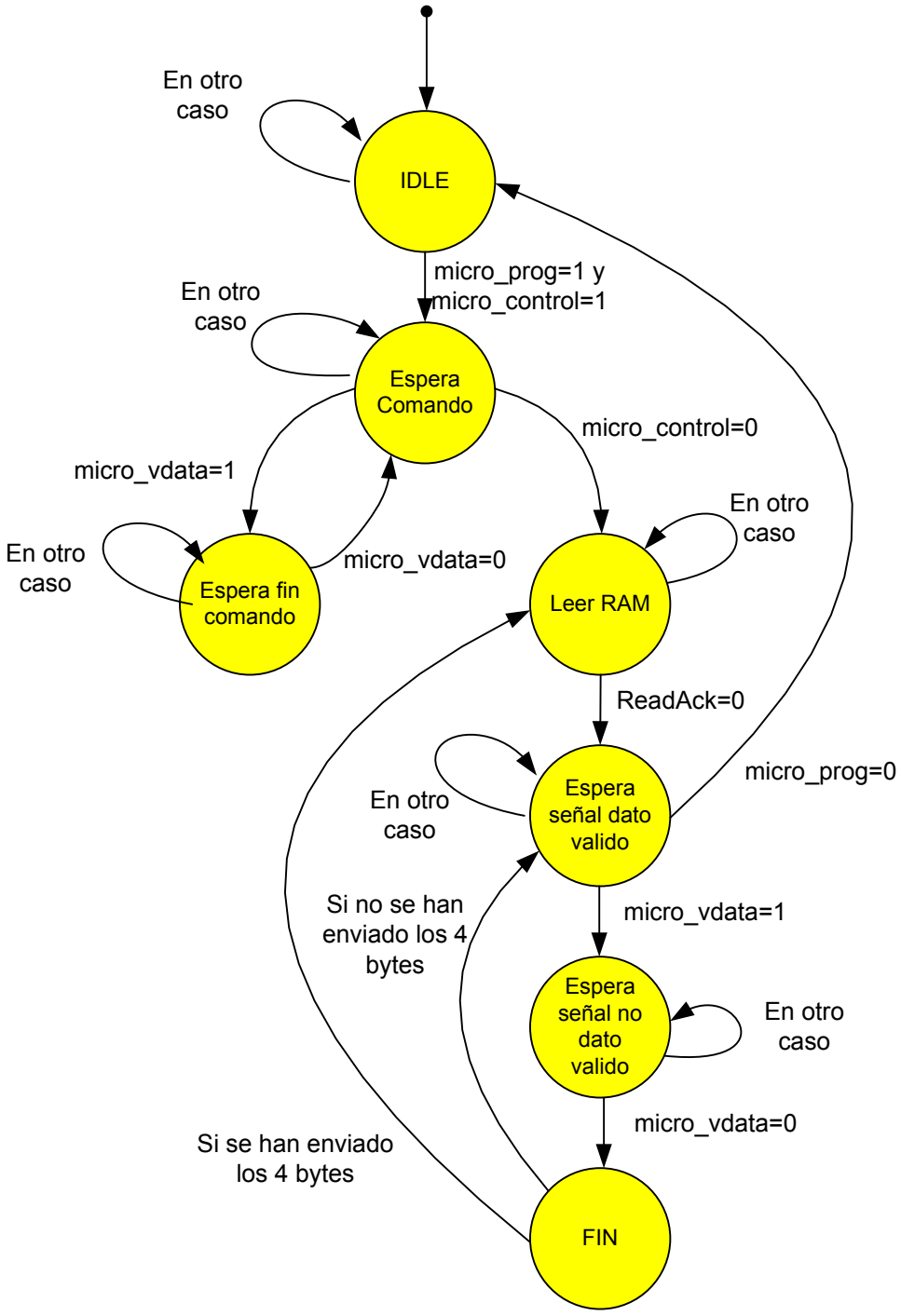


Figura 6.22: Máquina de estado del circuito de comunicación con el microcontrolador.

En el estado de inicio (*idle*), si se activan las señales *micro_prog* y *micro_control* indica que el microcontrolador ha comenzado un ciclo de acceso a la FPGA; este caso es siempre de lectura. En el ciclo de lectura la primera información son los comandos que van desde el PC a la USB-AER, 16 bytes de los que en este caso sólo son útiles 4 (aunque se envían los 16), donde van el valor que hay que transferir por *AcumPeriod*.

Cuando el microcontrolador desactiva la señal *micro_control* quiere decir que empieza la transferencia de datos. En este momento se pone a cero un contador que servirá como valor de índice a la memoria. En el estado de lectura de la RAM se obtiene por parte del controlador de RAM un dato de 32 bits que es la información almacenada en la posición determinada por el contador interno.

Un vez recibida la confirmación de la lectura por parte del controlador de RAM, comienza el envío de los datos de byte en byte al microcontrolador, en cada pulso de la señal *micro_vdata*. Cuando se termina de enviar los 4 bytes se solicita la siguiente posición de la RAM, hasta que el microcontrolador desactiva la señal *micro_prog*.

La aplicación del PC solicitará cada cierto tiempo la descarga completa del contenido de la memoria. Para ello, demandará al driver del USB un bloque de datos del tamaño correspondiente en este caso 128x128x32bits (128x128 es el tamaño de la retina y 32 la cantidad de datos almacenada para cada posición). El microcontrolador solicitará byte a byte la información a la FPGA, el circuito de comunicación con el microcontrolador posee un contador interno, que se incrementa tras cada transmisión, cuyo valor se usa para indicar la dirección de memoria al controlador de RAM.

La latencia de inicio de la transmisión de datos es 10 ns, el periodo de reloj; el tiempo de servicio de un dato es de 60 ns por dato: 20ns debidos al protocolo handshake y 40 ns más debidos a los 4 estados que debe recorrer la máquina (recordemos que el reloj interno es de 100MHz).

El coste hardware de este componente es de 72 slices de la FPGA de la USB-AER que supone un 3% de ocupación de la misma.

6.9. Características principales de la implementación en hardware

Veamos de forma resumida las características de la implementación realizada. Para el sistema completo se han contemplado 2 casos: un ObjectTracker6B y un ObjectTracker4C.2.

Tabla 6.1: Resumen de la latencia de las implementaciones del sistema

Componente	Latencia (ns)
CMCell	20 – 80
VCell	90
PRCell	40 – 740
TrackCell tipo B	30 – 170
TrackCell tipo C.2	30 – 820
ObjectTracker6B 1ª TrackCell	200*
ObjectTracker6B 2ª TrackCell	230*
ObjectTracker4C.2 1ª TrackCell	210*
ObjectTracker4C.2 2ª TrackCell	240*
BackGroundActivity Filter	420 – 460
Arbitrador	80 – 560
Controlador RAM	20 – 80
Comunicación Microcontrolador	10 – 60

* Valor mínimo, medido directamente sobre las plataforma USB-AER.

Sólo es posible obtener el valor de la latencia real para los ObjectTracker, debido a que el resto no tiene una interfaz al exterior de sistema, y es imposible medir la latencia desde fuera de la FPGA. En la Tabla 6.1 se muestran todos los valores máximos y mínimos para cada uno de los componentes, todos los valores han sido calculados teóricamente, dada la imposibilidad de medir estos tiempos dentro de la FPGA. Sólo para el caso de los ObjectTracker es posible medir la latencia desde el exterior, el valor mostrado en la tabla

corresponde al valor mínimo, en un caso ideal. Esto valores verifican los cálculos teóricos realizados.

Tabla 6.2: Resumen del coste hardware de las implementaciones del sistema

Componente	Coste hardware (slices, %)
CMCell	201 , 8%
VCell	109 , 4%
PRCell	73 , 3% (1 Bloque RAM 7%)
TrackCell tipo B	306 , 13%
TrackCell tipo C.2	398 , 16%
ObjectTracker6B	2056 , 87%
ObjectTracker4C.2	1946 , 82%
BackGroundActivity filter	179 , 7%
Arbitrador	241 , 10%
Controlador RAM	66 , 2%
Comunicación Microcontrolador	72 , 3%

El porcentaje hace referencia a la ocupación de la FPGA Spartan II 200 que tiene la USB-AER.

En la Tabla 6.2 se muestra el coste hardware diseño realizado. La PRCell consume sensiblemente menos hardware que las otras dos celdas básicas, aunque si consume memoria RAM. Dado lo reducido de la FPGA de la USB-AER sólo es posible integrar 6 TrackCell de tipo B y 4 de tipo C, lo que supone seguir y detectar sólo 6 y 4 objetos respectivamente, aunque el diseño modular del sistema permite integrar en un sólo sistema cuantas TrackCells se precisen para la aplicación concreta usando una FPGA con el tamaño adecuado (las FPGA Virtex6 tiene 37 veces más capacidad que la Spartan II).

Tabla 6.3: Resumen de consumo de potencia de las implementaciones del sistema

Componente	Consumo de eléctrico (mA) a 2.5V
ObjectTracker6B	53.1
ObjectTracker4C.2	78.3
BackGroundActivity Filter	11.1

En la Tabla 6.3 se muestra el consumo de cada diseño. Para obtener este valor, se ha medido el consumo del núcleo de la FPGA (alimentada a 2.5V). El consumo eléctrico del

BackGroundActivity Filter es sorprendentemente bajo comparado con el resto, pero hay que tener en cuenta que el diseño es bastante simple y su funcionamiento necesita del uso de la memoria RAM de la plataforma USB-AER y el consumo de la memoria no ha sido tenido en cuenta.

6.10. Resumen

Uno de los objetivos de este trabajo es realizar una implementación en hardware del sistema, una excelente forma de implementar en hardware un sistema descrito en términos de máquina de estado, es usar el lenguaje de descripción de hardware VHDL, y posteriormente configurar un dispositivo lógico programable, en este caso una FPGA.

La plataforma USB-AER, especialmente adaptada a sistemas basados en AER, permite la implementación de cualquier circuito digital, ya que está construida en torno a una FPGA, y consta de dos 2 puertos AER, uno de entrada y otro de salida.

La implementación contempla la mayor parte de las características de las celdas propuestas, algunos tipos de TrackCell no han sido implementados, aunque su implementación es sencilla gracias a la modularidad del sistema propuesto.

Las labores de testeo y pruebas requieren de diversa circuitería adicional, ajena al funcionamiento del sistema, pero necesario para realizar la depuración de errores del mismo. El controlador de RAM, el arbitrador, y la circuitería de comunicación con el microcontrolador del USB, son elementos absolutamente necesarios para estas labores.

La latencia del sistema es lo suficientemente baja como para no provocar retrasos significativos en los eventos procedentes de la retina, que como máximo puede emitir 1Mevento por segundo (1 evento cada 1us)

El reducido coste hardware del sistema presentado permite integrar en una USB-AER un ObjectTracker6B o un ObjectTracker4C.2, aunque usando una plataforma con una

FPGA de mayor capacidad es posible realizar un sistema capaz de seguir una cantidad mayor de objetos.

El consumo energético, salvo en el caso del *BackGroundActivity Filter*, es más elvavado del deseable y requerirá una labor de mejora en futuros trabajos.

"El experimentador que no sabe lo que está buscando no comprenderá lo que encuentra."

Claude Bernard

Capítulo 7

Experimentos y pruebas

Este capítulo está dedicado a los experimentos y pruebas realizados con la implementación de los modelos neuronales presentados. Con ello se pretenden mostrar las prestaciones, la viabilidad y utilidad de los sistemas implementados.

En primer lugar se muestran una serie de pruebas destinadas a comprobar el funcionamiento del sistema de localización y estimación de la velocidad. En los primeros experimentos se usan estímulos generados de forma sintética, utilizando la plataforma USB-AER como generador de eventos. Posteriormente se muestran los resultados usando como estímulo la secuencia AER generada por la retina. En este grupo de experimentos se someterá al sistema a situaciones difíciles, en las que los objetos describen trayectorias que se cruzan. En tercer lugar se analizará el comportamiento del sistema con objetos reales: vehículos y personas. Y por último, se muestran experimentos dedicados a la detección de patrones simples consistentes en líneas verticales, horizontales y diagonales.

Como ya se apuntó antes, las medidas que ofrece el sistema de estimación de la velocidad no tienen en cuenta la calibración de la óptica de la retina, por lo que los datos de velocidad están expresados en píxeles por segundo.

El montaje hardware de los experimentos y pruebas está compuesto por 4 placas USB-AER conectadas en serie y una retina artificial. El papel de cada USB-AER dependerá de la naturaleza de éstos, aunque en todos una de ellas implementará el sistema y otra será un *frame-grabber* que permite la monitorización del funcionamiento de la misma. La Figura 7.1 muestra el montaje hardware.

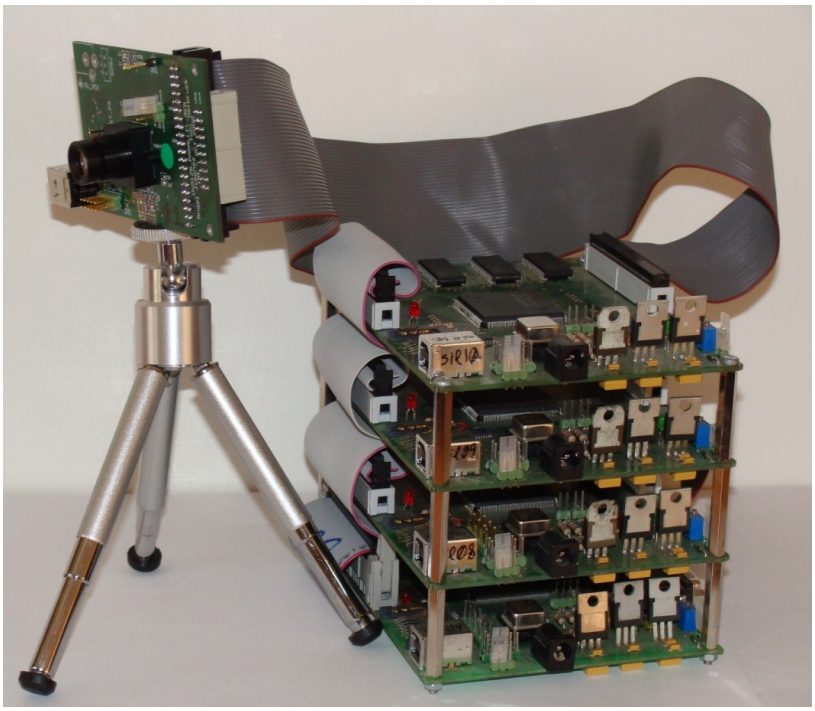


Figura 7.1: Montaje hardware de los experimentos y pruebas

7.1. Seguimiento de un objeto, estimulo sintético

En este grupo de experimentos se usarán estímulos sintéticos. Entenderemos como tales los estímulos que no corresponden a un objeto real y que son generados de manera artificial por un hardware específico. En nuestro caso, los estímulos sintéticos son generados por una USB-AER configurada como generador de secuencias AER

(*DataLogger)*. Para usar la USB-AER como generadora de eventos cuando está configurada como *Datalogger* es necesario crear la lista de eventos, es decir, las direcciones y el tiempo en el que estas direcciones han de ser transmitidas por el bus AER. Para generar estos eventos se ha diseñado una aplicación en Matlab que permite la generación de los eventos correspondientes a objetos de hasta 8x8 píxeles[47]. Esta aplicación permite definir la escala del objeto (con lo que se pueden generar objetos mayores como resultado del escalado de uno de 8x8 píxeles), la velocidad, el tiempo entre eventos, veces que se repite cada píxel antes de "mover" el objeto a la siguiente posición; además la aplicación permite enviar la lista de eventos directamente a la USB-AER.

La Figura 7.2 muestra el interfaz de la aplicación para la generación de estímulos sintéticos. Concretamente en la figura se muestran los parámetros que permiten generar un cuadrado de 4x4 píxel que se mueve con una trayectoria que describe un cuadrado con una velocidad de 50 píxeles/s (o lo que es lo mismo, el objeto se mueve un píxel cada 20 ms).

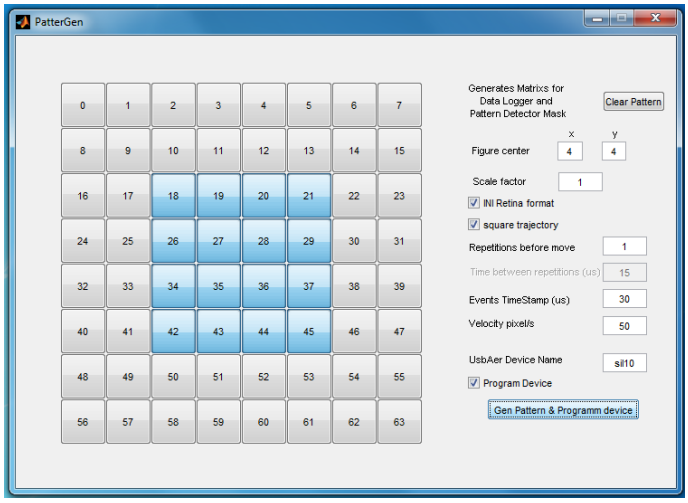


Figura 7.2: Generación de estímulos sintéticos

[47] Este tamaño medido en el plano de un hipotético sensor, es decir, es el tamaño de objeto medido en el plano de proyección de una retina.

En esta serie de experimentos no es necesario el montaje hardware completo, sólo se usarán 3 USB-AER; la retina no es necesaria ya que el estímulo es generado de manera sintética. La Figura 7.3 muestra el esquema del montaje hardware para estímulos sintéticos[48]. La primera USB-AER está configurada como *DataLogger* y es la encargada de generar la secuencia AER sintética; la segunda USB-AER está configurada como *FrameGrabber* con la finalidad de observar una reconstrucción, a modo de imagen, de la secuencia AER; y la tercera está configurada como ObjectTracker que es el sistema que se quiere probar y corresponde con la implementación del sistema neuronal propuesto.

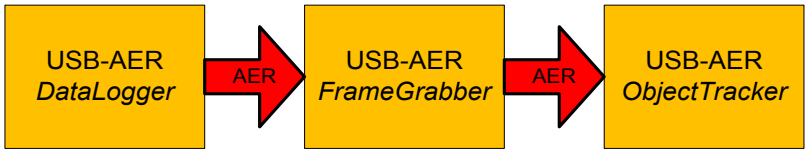


Figura 7.3: Montaje hardware para estímulos sintéticos

Las tres USB-AER están conectadas a un PC usando el puerto USB, con la siguiente utilidad: a la primera de las USB-AER, a través del puerto USB, se le envía la secuencia AER que se reproducirá de manera continua; la segunda USB-AER realizará una reconstrucción en forma de *frame* de la secuencia AER que será descargada al PC periódicamente; y la tercera configurada con el sistema neuronal de seguimiento de objetos almacenará el resultado de seguimiento en la memoria RAM, cuyo contenido también será descargado al PC periódicamente para su visualización.

Para este grupo de experimentos se ha modificado levemente el diseño de la CMCell de manera que no tenga en cuenta los eventos negativos, ya que la aplicación de generación de estímulos sintéticos sólo genera eventos positivos. Esta modificación no altera significativamente el funcionamiento global del sistema cuando el estímulo se corresponde con un objeto pequeño.

---

[48] Este mismo montaje puede ser usado para reproducir secuencias AER previamente grabadas por la USB-AER configurada con *DataLogger*.

7.1.1. Objeto moviéndose a baja velocidad

En este experimento el estímulo sintético consiste en un cuadrado de 1x1 píxel, siguiendo una trayectoria que describe un cuadrado, con una velocidad, en cada lado, de 50 píxeles/s. El cambio de dirección en la esquinas implica que la velocidad en esos puntos no es constante.

La Figura 7.4 muestra varias capturas de la aplicación *MultiLoadFPGA*[49], en la que se observan varias reconstrucciones a modo de *frame* de la secuencia AER con un tiempo de integración de 10 ms. Sobre las capturas se ha dibujado la trayectoria seguida por el movimiento del cuadrado de 1x1 píxel.

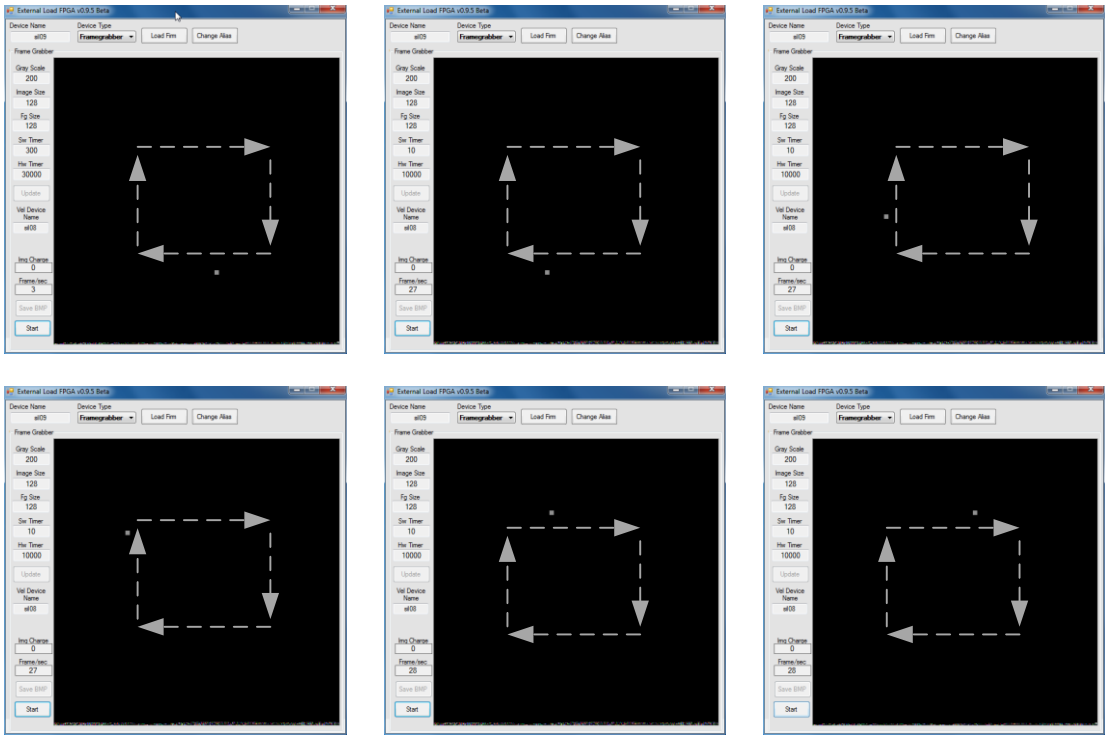
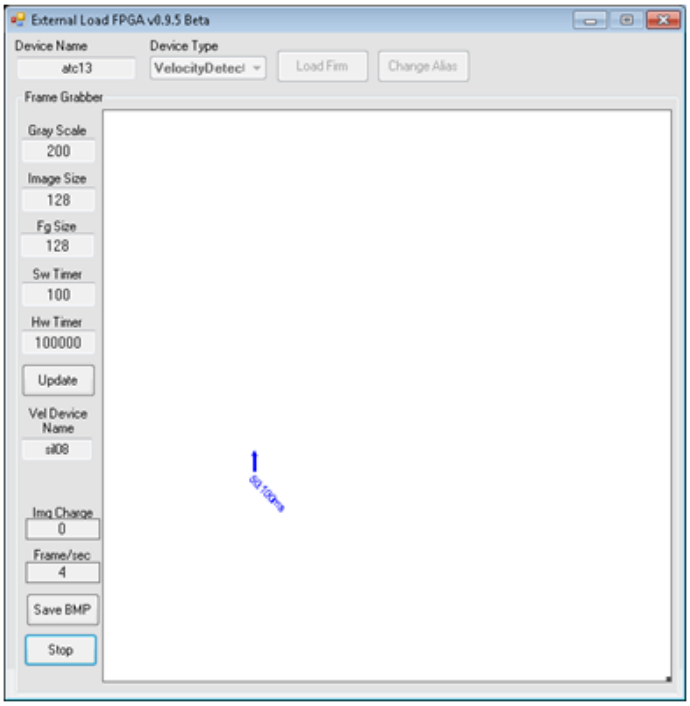
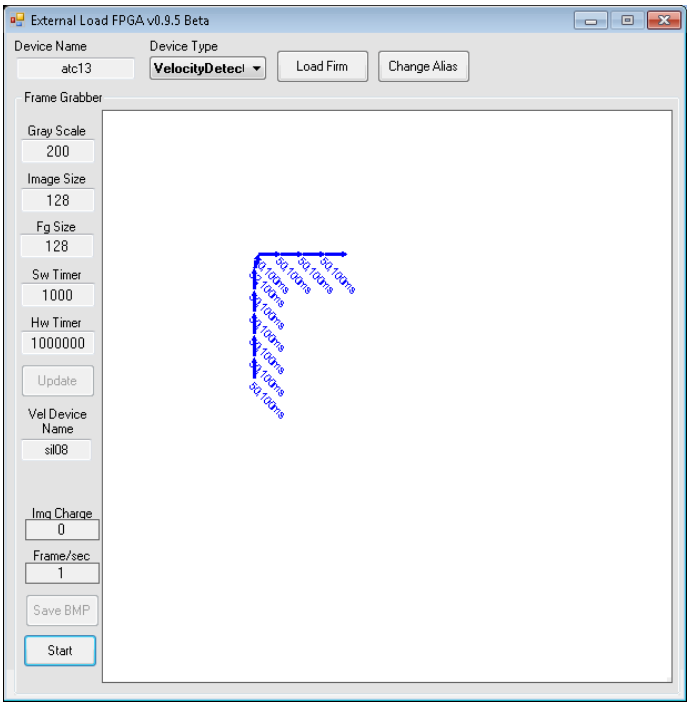


Figura 7.4: Estímulo sintético trayectoria cuadrada

[49] Aplicación para Windows para el manejo de la USB-AER, inicialmente desarrollada en el seno de proyecto CAVIAR. Sobre la última versión, realizada por Manuel Rivas (componente del grupo RTC), se han añadido las nuevas funcionalidades de la USB-AER que se presentan en este trabajo.



a)



b)

Figura 7.5: Respuesta de sistema para estímulos sintéticos a baja velocidad

La Figura 7.5.a muestra la representación de la información grabada en la RAM de la USB-AER durante 100 ms. Se puede observar una flecha que indica la posición del objeto y la magnitud, de manera cualitativa, de la velocidad; junto a la flecha aparecen el valor numérico de la velocidad expresada en píxeles/s, y el tiempo de integración usado por la VCell para realizar el cálculo de la velocidad. En este caso son 50 píxeles/s y 100 ms respectivamente. La Figura 7.5.b muestra la misma información pero acumulada durante 1 segundo. Esta acumulación de información se realiza con el único objetivo de su posterior representación, con el fin de validar y comprobar los resultados obtenidos.

Hay que recordar que todo el análisis del movimiento, la determinación de la posición y de la velocidad del objeto, la realiza el sistema hardware propuesto. En el PC sólo se realiza la representación gráfica de la información que la USB-AER proporciona.

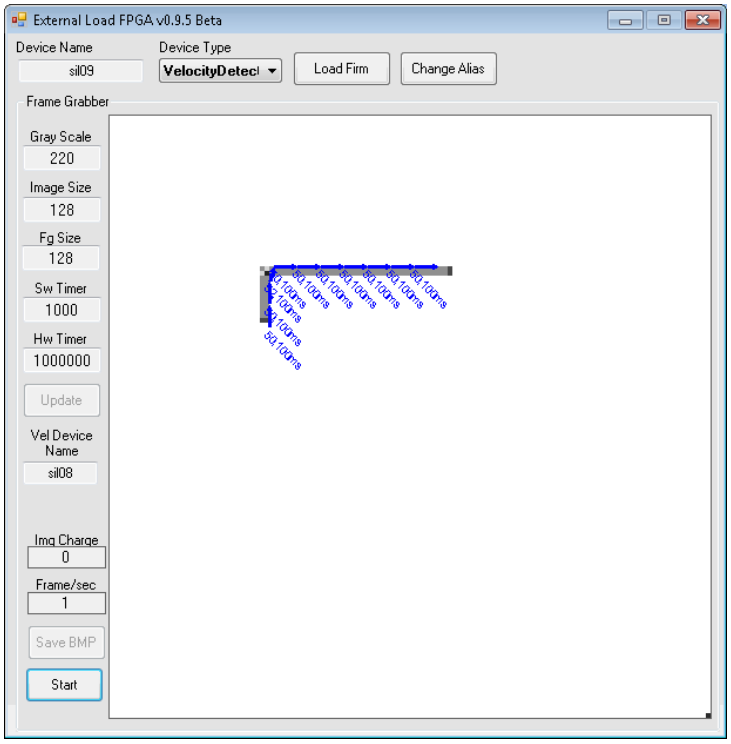


Figura 7.6: Reconstrucción de la secuencia AER fusionada con la respuesta del sistema para estímulos a baja velocidad

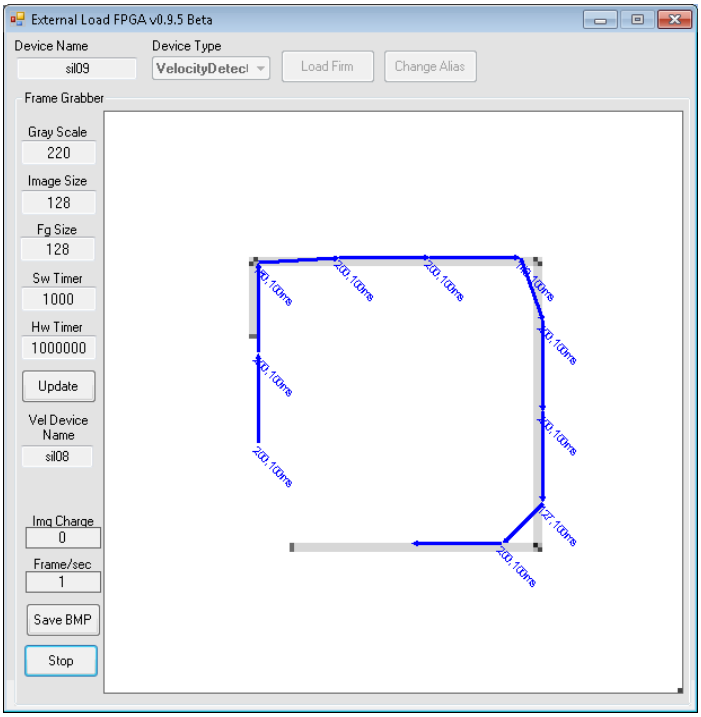
Superponiendo la información del procesado con la reconstrucción de la imagen, es decir, representando juntas la información recogida por la segunda y tercera USB-AER (*framegrabber* y *ObjectTracker* respectivamente), podemos observar como el sistema es capaz de determinar con exactitud la posición del objeto y su velocidad. La Figura 7.6 muestra una reconstrucción de la secuencia AER integrada durante 1 segundo (línea gris gruesa) y la respuesta del sistema (flechas azules).

7.1.2. Objeto moviéndose a alta velocidad

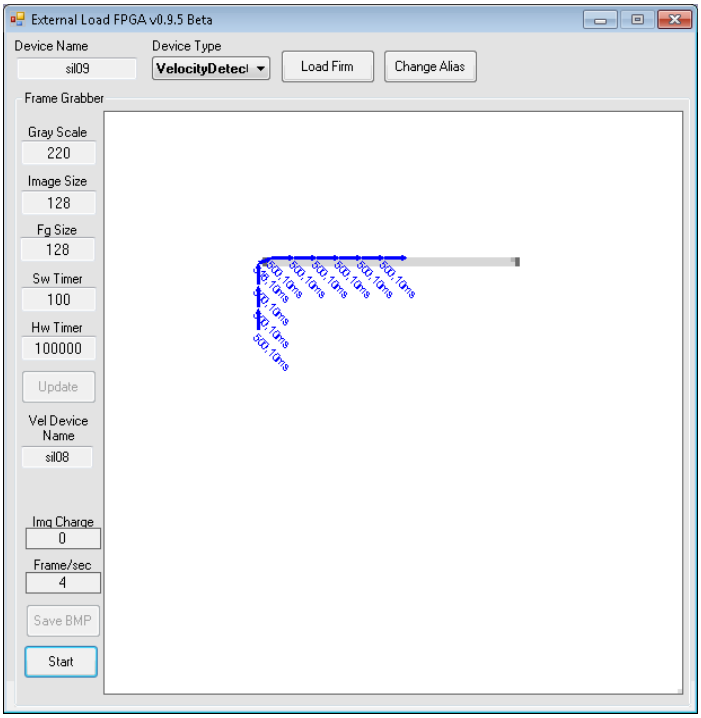
En el experimento anterior se mostraba como el sistema es capaz de seguir un objeto que se mueve a una velocidad relativamente baja (50 píxeles/s), para ello el sistema usa un tiempo de integración de 100ms. En este experimento se pone a prueba el sistema cuando la velocidad del objeto es sensiblemente más alta.

En la Figura 7.7.a se puede observar la respuesta del sistema cuando la velocidad del objeto es de 200 píxeles/s, el tiempo de integración de la VCell es de 100ms (como en el caso anterior), además se ha acumulado el resultado durante 1 segundo también. En la Figura 7.7.b se muestra la respuesta del sistema cuando la velocidad es de 500 píxeles/s, ahora el tiempo de integración de la VCell es de 10 ms. Este cambio en el tiempo de integración de la VCell es automático, es decir, es la propia VCell la que "decide" adaptar este tiempo en función de la velocidad del objeto que está siguiendo. A efectos de llevar a cabo una mejor representación de la información, la USB-AER ha acumulado la información del seguimiento durante tan sólo 100ms.

En cuanto al rastro que deja el objeto moviéndose en el *framegrabber* (línea gris), se observa un ligero desplazamiento con respecto al resultado del seguimiento; esto es debido a la falta de sincronización entre la descarga al PC de la información del *framgrabber* y del *ObjectTracker*. Por lo tanto este desfase ha de entenderse derivado de la imposibilidad del sistema operativo de sincronizar la descarga de ambos dispositivos USB (las USB-AER).



a)



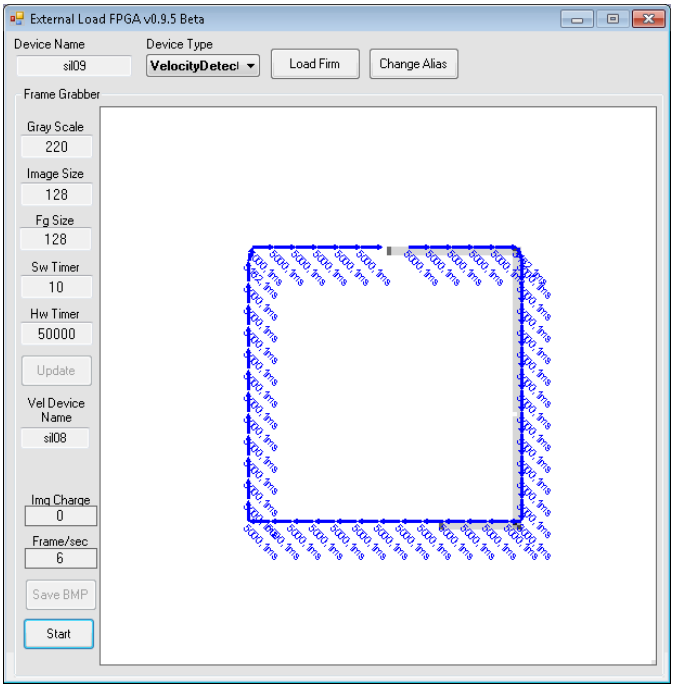
b)

Figura 7.7: Respuesta del sistema para estímulo sintético con velocidad a) 200 y b) 500 píxeles/s

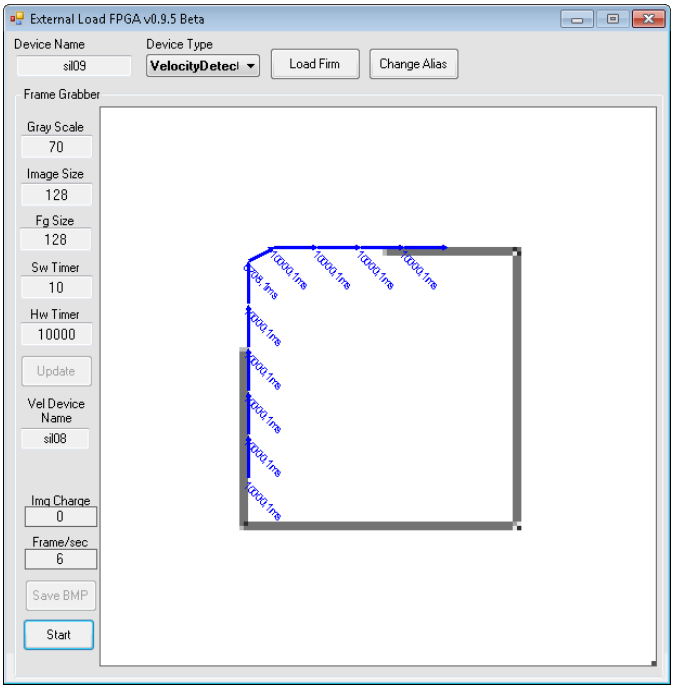
En la Figura 7.8.a se muestra la respuesta del sistema cuando el estímulo se mueve a 5000 píxeles/s (lo que significa que el objeto cambia de posición cada 200us), en este caso la VCell usa un tiempo de integración de 1 ms; se muestra la información acumulada durante 50 ms. En la Figura 7.8.b se muestra la respuesta del sistema para una velocidad de 10000 píxeles/s, el tiempo de integración es de 1ms también, aunque ahora se muestra la información acumulada durante 10 ms.

Por último en la Figura 7.9.a se muestra la respuesta para una velocidad de 30000 píxeles/s, donde el tiempo de integración de la VCell es de 500us y se ha acumulado la información durante 5 ms. En la Figura 7.9.b se muestra la respuesta del sistema para una velocidad de 40000 píxeles/s, el tiempo de integración de la VCell es de 100us y se han acumulado los datos durante 5 ms también.

El hecho de que un objeto se mueva a 40000 píxeles/s significa que el objeto cambia de píxel cada 25 us; dado que la trayectoria tiene un total de 256 puntos, esto significa que se completa un ciclo de la misma cada 6,4 ms; lo que representa una velocidad de giro de 9375 rpm.

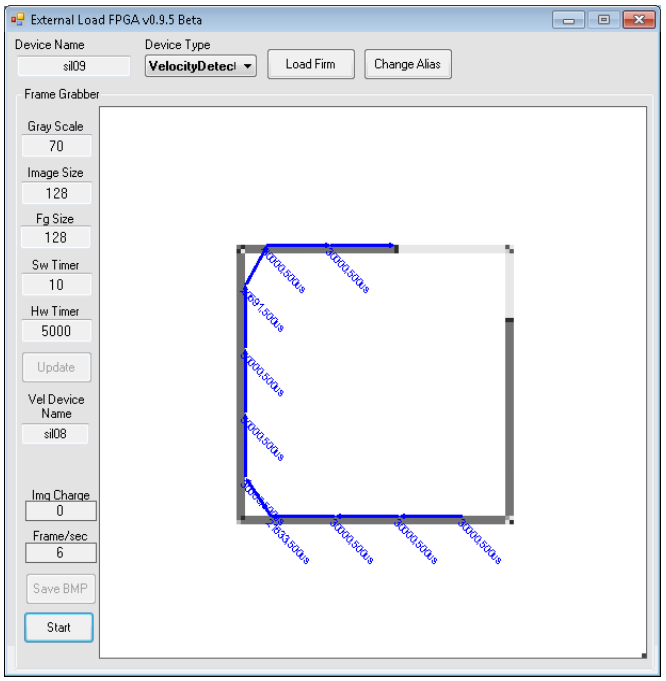


a)

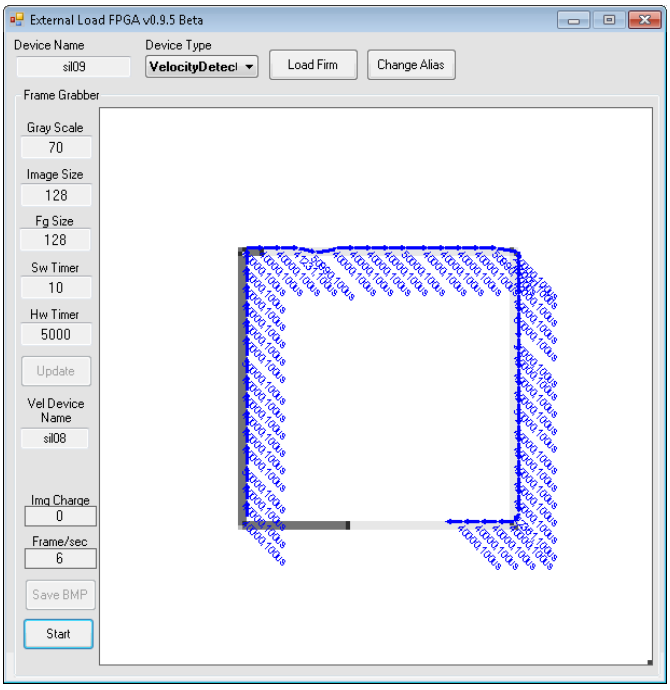


b)

Figura 7.8: Respuesta del sistema para estímulo sintético con velocidad a) 5000 y b) 10000 píxeles/s



a)



b)

Figura 7.9: Respuesta del sistema para estímulo sintético con velocidad a) 30000 y b) 40000 píxeles/s

7.1.3. Objeto moviéndose con velocidad variable

En los experimentos anteriores la velocidad del objeto es constante, excepto en las esquinas. En este experimento la velocidad del objeto cambia, de manera que la velocidad por el lado izquierdo es el doble de la velocidad que por el lado inferior; la del lado superior es 5 veces y la del lado derecho 10 veces la del lado inferior, es decir, las velocidades están multiplicadas por 2, por 5 y por 10.

La Figura 7.10 muestra la respuesta del sistema cuando la velocidad es 1000 píxeles/s en el segmento inferior, 2000 píxeles/s en el izquierdo, 5000 píxeles/s en el superior y 10000 píxeles/s en el derecho. Este esquema de movimiento se repite de manera indefinida. Se puede observar como la VCell adapta el tiempo de integración a la velocidad del objeto, de manera que en el segmento inferior y en el izquierdo es de 5ms, mientras que en el superior y en el derecho es de 1 ms. Además en la Figura 7.10 se observa un redondeado de la trayectoria del objeto en las esquinas del cuadrado, debido al uso de trayectorias continuas en posición y velocidad.

Para una mejor representación de los datos en el PC se han acumulado los datos de la respuesta del sistema durante 100 ms.

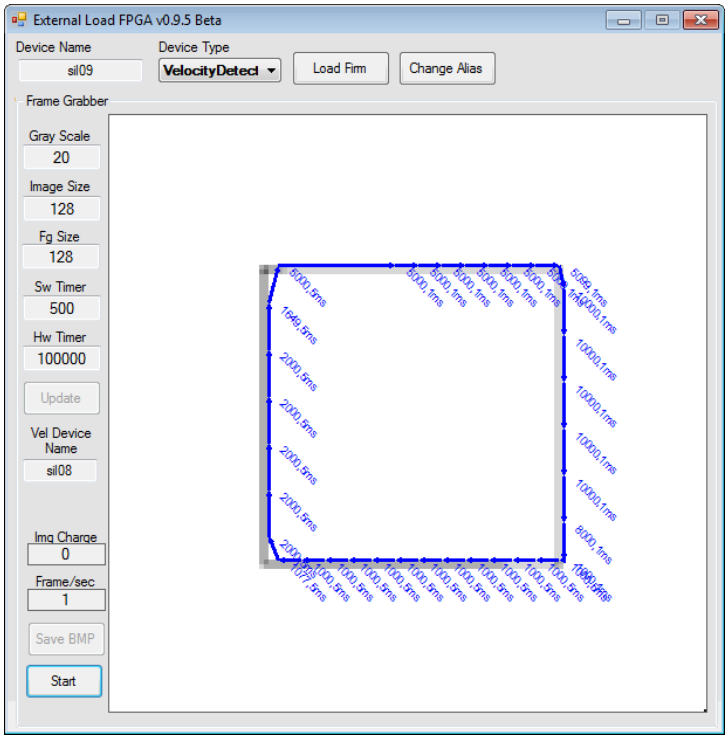


Figura 7.10: Respuesta del sistema para un estímulo con velocidad cambiante, 1000, 2000, 5000 y 10000 píxeles/s

Resumen

Con esta serie de experimentos se pone de manifiesto la capacidad del sistema para obtener la posición de un objeto en movimiento y para estimar su velocidad. Se ha demostrado que el sistema es capaz de seguir y estimar la velocidad con precisión, cuando el objeto se mueve a baja velocidad y también a altas velocidades, incluso a velocidades tan muy altas (40000 píxeles/s), aunque en la actualidad no existen sensores capaces de captar un objeto que se mueva a esa velocidad. Además, el sistema es capaz de adaptarse a cambios de velocidad del objeto y continuar con su labor con la misma exactitud que antes del cambio de velocidad.

Podemos afirmar que si la información del sensor fuera limpia de ruido, el sistema sería capaz de obtener la posición y de estimar la velocidad del objeto con una precisión máxima.

7.2. Seguimiento de objetos sintéticos

En este grupo de experimentos se usarán objetos sintéticos; llamaremos objetos sintéticos a aquellos que son proyectados en un monitor. Se usará una retina que "observará" los objetos que aparecen en el monitor.

El montaje hardware para este experimento está compuesto de una retina y 3 USB-AER. La primera está configurada como *BackgroundActivityFilter* esta funcionalidad permite eliminar algunos errores que presenta la retina, que consisten en la aparición de actividad en píxeles donde no hay cambio de luminosidad y que por lo tanto no corresponden a movimiento en la escena. La segunda USB-AER es un *FrameGrabber*. Y la tercera el *ObjectTracker*.

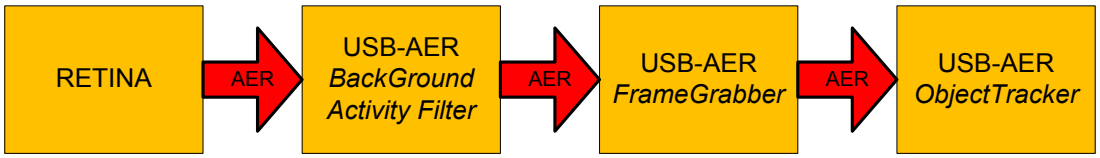


Figura 7.11: Montaje hardware para objetos sintéticos

Al igual que antes todas las USB-AER están conectadas al PC. El *BackGroundActivityFilter* necesita ser configurado para definir el umbral de actividad del píxel que debe ser tenido en cuenta, esta operación sólo es necesaria al inicio del sistema. La conexión del *FrameGabber* y del *ObjectTracker* al PC tiene la misma funcionalidad que en los experimentos anteriores.

7.2.1. Un objeto con velocidad variable

Este primer experimento consiste en una circunferencia de ancho 4 píxeles aproximadamente, que se mueve describiendo un rectángulo, los lados verticales tienen una longitud de 77 píxeles y los horizontales de 80 píxeles. La velocidad de movimiento cambia en cada lado de la trayectoria: la circunferencia tarda 3 segundos en recorrer el lado derecho, 2 segundos el superior, 1 segundo el izquierdo y 0.5 segundos el inferior; lo que

supone aproximadamente unas velocidades (medidas sobre la imagen del sensor) de 26.6 píxeles/s, 40 píxeles/s, 80 píxeles/s y 160 píxeles/s respectivamente.

La Figura 7.12 muestra 4 capturas de la aplicación de monitorización de la retina. Cada captura se corresponde con una reconstrucción de los eventos en forma de *frame* con un tiempo de integración de 30.5 ms, 33.3 ms, 13.7 ms y 51.8 ms (para a, b, c y d respectivamente). Dado que la retina sólo puede captar los cambios de luminosidad, la circunferencia se convierte en dos arcos: el de tonos claros corresponde al flanco de avance y el de tonos oscuros al de cola. Los puntos aislados corresponden al ruido de la retina que se eliminan, en gran medida, gracias el uso del *BackGroundActivityFilter*.

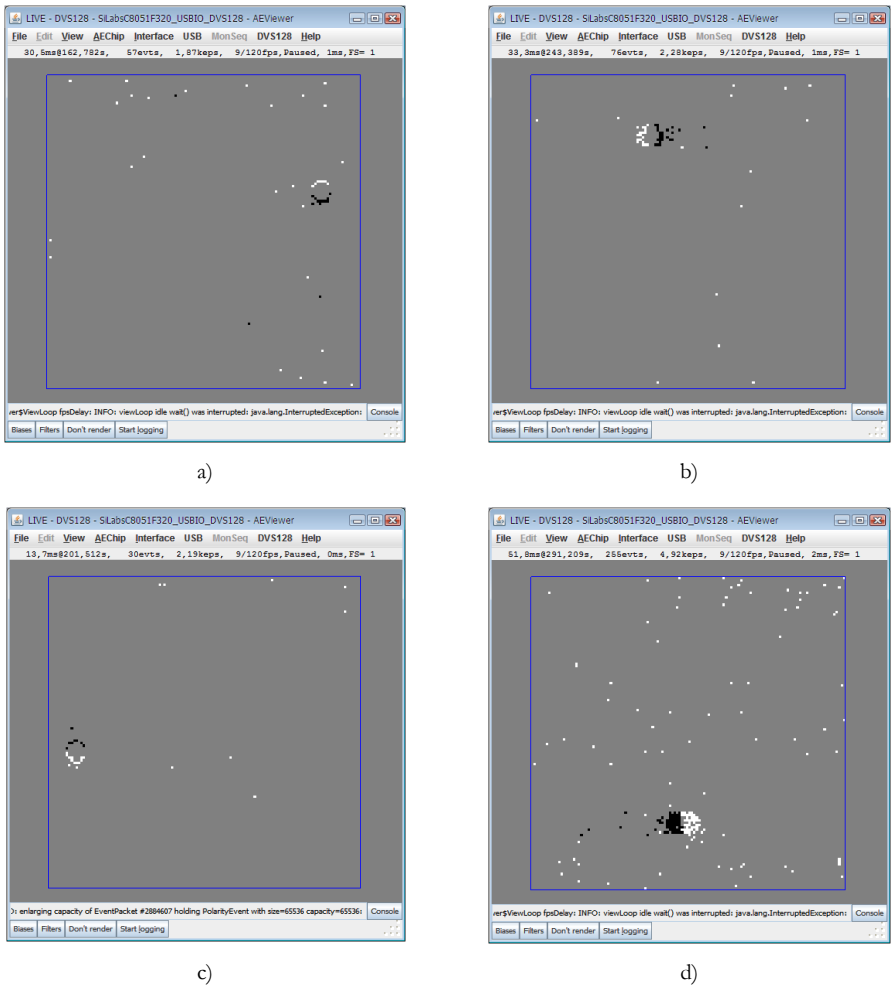


Figura 7.12: Reconstrucción en *frame* de una circunferencia

La Figura 7.13 muestra la respuesta del sistema acumulada durante 6.3 segundos (aunque el objeto tarda 6.5 segundos en completar todo el recorrido). En gris se dibuja la estela de píxeles que deja la circunferencia al moverse; en azul se presenta la velocidad de la circunferencia en cada momento. El cálculo de la velocidad es bastante preciso y además se puede observar el cambio automático de tiempo de integración de la VCell, concretamente en lo valores de la esquina superior izquierda y en la esquina inferior derecha.

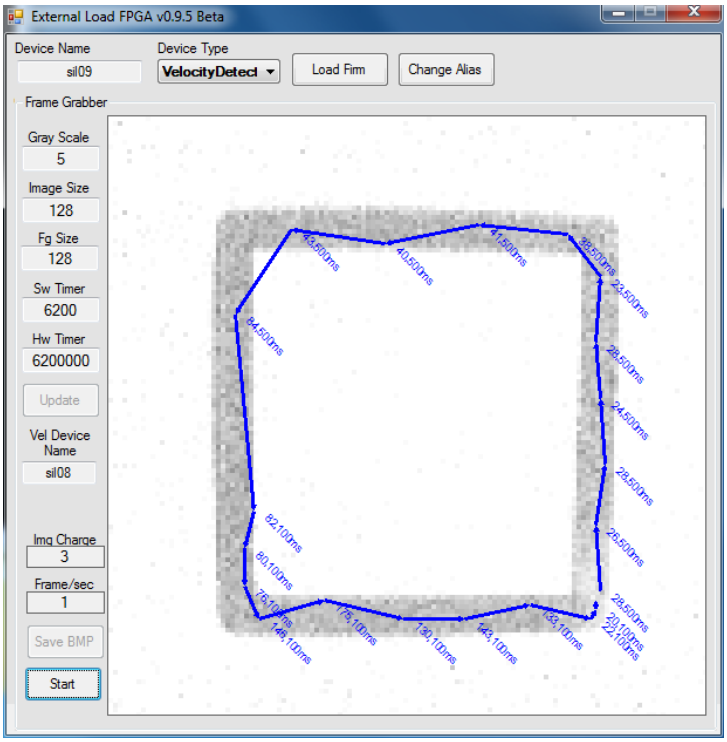


Figura 7.13: Respuesta del sistema para un objeto sintético con velocidad cambiante

En la Tabla 7.1 se describe de forma tabulada la salida del sistema para el segmento izquierdo, donde se puede observar que el error máximo cometido por el sistema en la estimación de la velocidad es del 5%; en la Tabla 7.2 se muestra los valores para el lado derecho, donde el error máximo es de 25 %; en la Tabla 7.3 se presentan los valores para el lado inferior con un error máximo del 19% y en la Tabla 7.4 están los valores para el segmento superior cuyo error máximo es del 8%.

Estos errores en la estimación de la velocidad tienen su origen en el uso de un sensor real, ya que usando estímulos sintéticos la estimación de la velocidad no presentaba errores. El uso de un sensor real, como la retina artificial de trabajo, introduce ciertas incertidumbres a la hora de estimar la posición de los objetos en movimiento. Por un lado, no se puede garantizar que todos los píxeles de la retina emitan la misma cantidad de eventos ante un cambio idéntico en la luminosidad, y por otro los eventos son, en principio, aleatorios. Estos dos motivos hacen que la estimación de la posición del objeto no coincida en todos los casos con el centro de masa del mismo, y por lo tanto la estimación de la velocidad (que se basa en la posición del objeto en cada momento) se vea afectada por este error.

Tabla 7.1: Estimación de la velocidad para el segmento izquierdo

Segmento izquierdo Velocidad 80 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
84	5
82	2
80	0
76	5

Tabla 7.2: Estimación de la velocidad para el segmento derecho

Segmento derecho. Velocidad 26.6 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
28	5
26	2
28	5
24	10
28	5
23	14
22	17
20	25

Tabla 7.3: Estimación de la velocidad para el segmento inferior

Segmento inferior Velocidad 160 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
146	9
175	9
130	19
143	11
133	17

Tabla 7.4: Estimación de la velocidad para el segmento superior

Segmento superior. Velocidad 40 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
38	5
41	2
40	0
43	8

Sobre la estimación de la velocidad hay que destacar que los sistemas neuronales biológicos no realizan una estimación precisa del movimiento de los objetos que se mueven, sino más bien, realizan una estimación cualitativa de dicho movimiento. Es decir, son capaces de detectar cambios en la velocidad, así como determinar que objetos se mueven más deprisa que otros. El sistema que aquí se presenta es capaz de imitar este comportamiento, además de dar una estimación cuantitativa de la velocidad y con una gran rapidez.

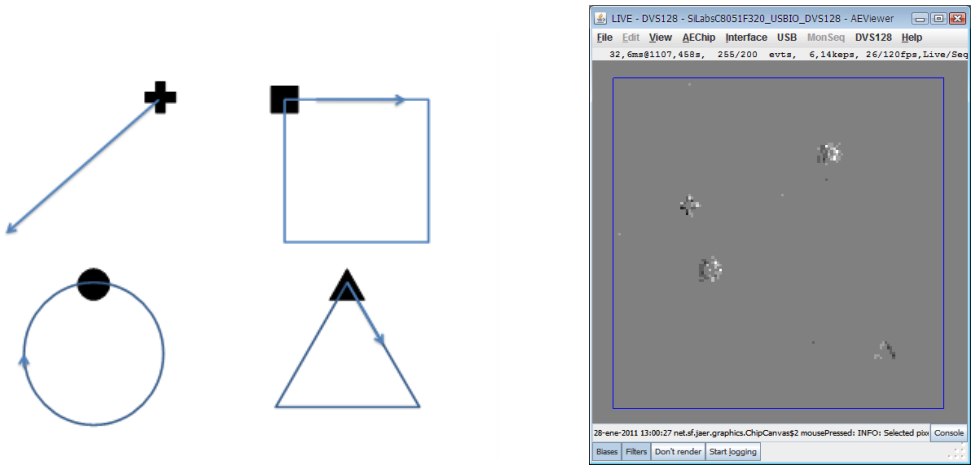
7.2.2. Varios objetos con trayectorias sin cruces y velocidades variables

Este experimento consiste en 4 objetos moviéndose con diferentes trayectorias sin cruces entre ellas, y con velocidades que cambian a lo largo del tiempo. Concretamente la configuración del experimento es la siguiente:

- Un objeto en forma de cruz de unos 8x8 píxeles moviéndose en la parte superior izquierda, cuya trayectoria es una línea recta de longitud 47 píxeles y que tarda en recorrerla 3 segundos. Lo que implica una velocidad de 15.6 píxeles/s aproximadamente.
- Un objeto con forma de cuadrado de unos 8 píxeles de lado moviéndose en la parte superior, cuya trayectoria es un cuadrado de 33 píxeles de lado y que tarda en recorrer toda la trayectoria 2 segundos, lo que significa una velocidad de 66 píxeles/s.
- Un objeto en forma de círculo de ancho 4 píxeles en la parte inferior izquierda, describiendo una circunferencia de ancho 18 píxeles, lo que implica una trayectoria de longitud 113 píxeles y que tarda en recorrer toda la trayectoria 2 segundos, por lo tanto la velocidad es de 55 píxeles/s aproximadamente.
- Y por último un triángulo equilátero de 8 píxeles de lado en la parte inferior derecha, describiendo una trayectoria triangular equilátera de 30 píxeles de lado, que tarda en recorrer toda la trayectoria 5 segundos, lo que implica una velocidad aproximada de 18 píxeles/s.

En todos los caso salvo en el del círculo, se generan aceleraciones en los puntos que los que la trayectoria cambia de dirección bruscamente.

En la Figura 7.14.a se muestra la configuración del experimento; en la Figura 7.14.b se muestra una captura de la reconstrucción de la salida de la retina con un tiempo de integración de 32,6 ms, donde se observan los 4 objetos que forman el experimento.



a) b)

Figura 7.14: Experimento de 4 objetos con trayectorias sin cruces: a) esquema de experimento, b) reconstrucción de la salida de la retina

En la Figura 7.15 se muestra la salida del sistema fusionada con la salida de la retina. Al igual que en el experimento anterior, los datos están acumulados durante 1.9 segundos. Cada color representa la salida de una TrackCell (CMCell + VCell), en este caso el sistema está configurado con 4 TrackCells. El orden de las TrackCell en la arquitectura en paralelo es el siguiente: primero la representada en azul, después la representada en rojo, después la cian y por último la verde.

En la figura se muestran dos detalles interesantes del funcionamiento del sistema. En primer lugar, el funcionamiento independiente de cada una de las TrackCell. Cada TrackCell se encarga de seguir uno de los objetos.

En segundo lugar, se puede observar como cada TrackCell usa un tiempo de integración distinto para estimar la velocidad del objeto al que sigue, adaptando su valor a

la velocidad del objeto. Obsérvese, por ejemplo, el tiempo de integración usado por la TrackCell azul, que sigue al círculo, que es de 200ms; el de la TrackCell rojo, que sigue a la cruz, que es de 500ms; o el de la verde que cambia de 200ms a 100ms.

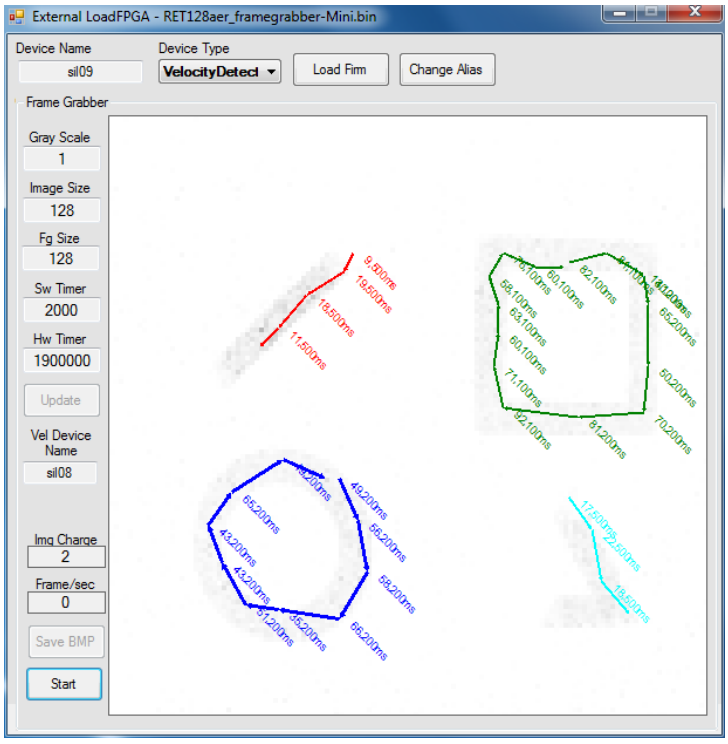


Figura 7.15: Respuesta del sistema para cuatro objetos sintéticos con trayectorias sin cruces

En la Tabla 7.5 se expone de forma ordenada los datos obtenidos por el sistema para el objeto en forma de cruz. Si exceptuamos el primer valor, que corresponde con el punto en el que objeto empieza a moverse, el error máximo es del 29% respecto a la velocidad real del objeto. En la Tabla 7.6 se muestran los datos para el objeto con forma de triángulo donde el error máximo es del 22% con respecto a la velocidad real. En la Tabla 7.7 aparecen los valores para el cuadrado. Exceptuando los valores 13 y 92 que se dan en las esquinas de la trayectoria, el error máximo es del 24%. Los valores de las esquinas están afectados por los cambios bruscos en la trayectoria. En la Tabla 7.8 se presentan los valores del objeto con forma de círculo donde el error máximo es del 22%, si exceptuamos

el valor de 35, anormalmente alto y debido a un fallo en al cómputo de la posición, causado seguramente por las imprecisiones de la retina y por la rapidez de la estimación, que como ya se ha comentado antes, es prioritaria a la precisión en el cálculo.

Tabla 7.5: Estimación de la velocidad para el objeto en forma de cruz

Cruz Velocidad 15.6 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
9	42
19	22
18	15
11	29

Tabla 7.6: Estimación de la velocidad para el objeto en forma de triángulo

Triángulo Velocidad 18 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
17	6
22	22
18	0

Tabla 7.7: Estimación de la velocidad para el objeto en forma de cuadrado

Cuadrado Velocidad 66 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
82	24
81	23
13	80
65	2
50	24
70	6
81	23
92	39
71	8
60	9
63	5
58	12
76	15
60	9

Tabla 7.8: Estimación de la velocidad para el objeto en forma de círculo

Círculo Velocidad 55 píxeles/s	
Velocidad estimada (píxeles/s)	**Error (%)**
49	11
56	2
58	5
66	20
35	36
61	11
43	22
43	22
65	18
49	11

7.2.3. Varios objetos con trayectorias con cruces

Con los dos experimentos que se presentan a continuación se pretende comprobar el comportamiento del sistema cuando las trayectorias de los objetos que se mueven en la escena se cruzan.

La configuración del primer experimento se muestra en la Figura 7.16. Los puntos donde se cruzan las trayectorias están marcados con un círculo rojo. Todos los objetos tardan en completar su trayectoria 3 segundos.

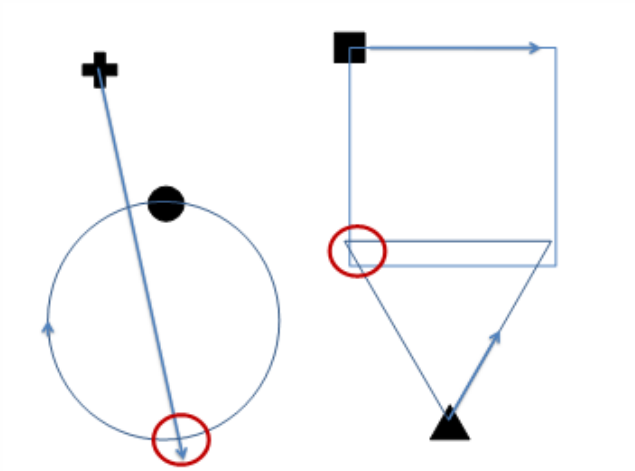


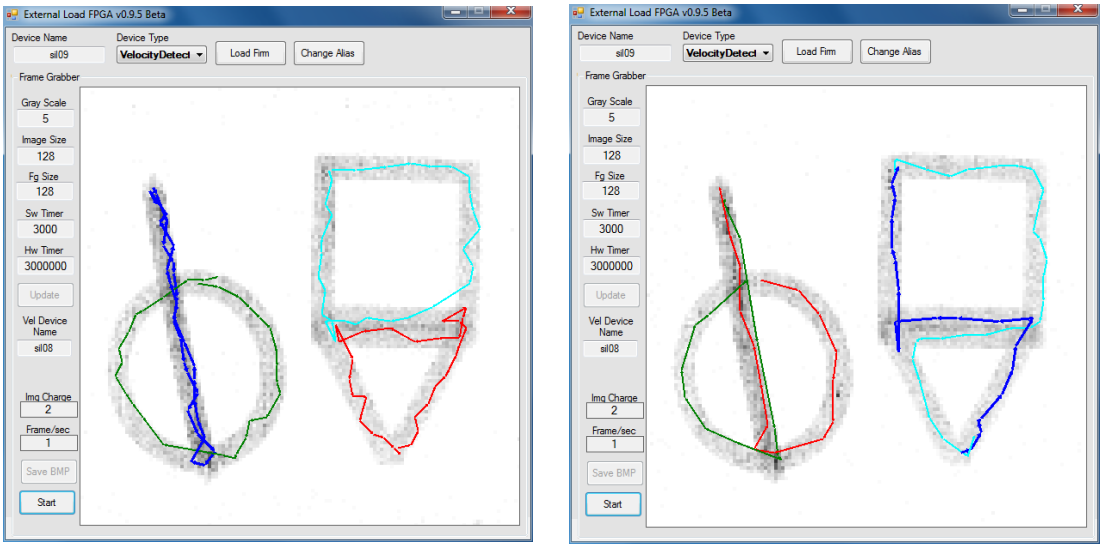
Figura 7.16: Configuración del primer experimento de trayectorias con cruces

En la Figura 7.17 se muestra la salida del sistema en dos momentos diferentes acumulada durante 2 segundos (y, como en los experimentos anteriores, fusionada con la salida de la retina). Se han eliminado los datos de velocidad y tiempo de integración para poder observar mejor como el sistema obtiene la posición de los objetos cuando sus trayectorias se cruzan.

En la captura de la derecha (Figura 7.17.a) se puede observar como el cruce en las trayectorias del círculo y de la cruz no afecta; las TrackCells que estaban siguiendo a cada objeto continúan siguiéndolos, cada uno al suyo, tras el cruce. Lo mismo ocurre con las

TrackCells que siguen al cuadrado y al triángulo. Este comportamiento, que es el deseable, no siempre se consigue.

En la captura de la Figura 7.17.b, se puede apreciar como después del cruce las TrackCells que seguían al círculo y a la cruz intercambian sus papeles, de manera que la TrackCell roja, que antes del cruce, estaba siguiendo al círculo, después de éste empieza a seguir a la cruz. Esto mismo ocurre con las TrackCells azul y cian, que antes del cruce seguían al triángulo y al cuadrado respectivamente, intercambian sus papeles después del mismo.



a) b)

Figura 7.17: Respuesta del sistema al primer experimento con trayectorias con cruces

Veamos un segundo experimento donde el cruce de trayectorias es masivo, lo que provocará, en algunos casos, un intercambio total. El experimento consiste en 4 objetos cada uno de ellos, inicialmente, en una esquina de la escena que se mueven hacia la esquina contraria, cruzándose, los cuatro, en el centro. Todos los objetos tardan en llegar al otro extremo 2 segundos (ver Figura 7.18).

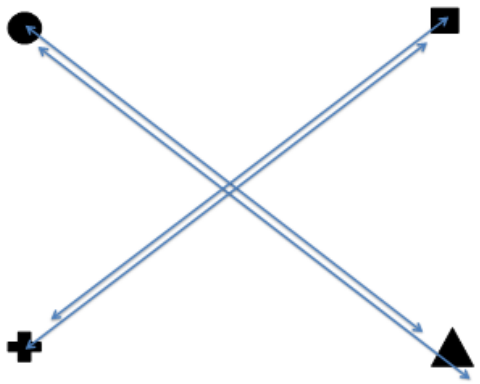
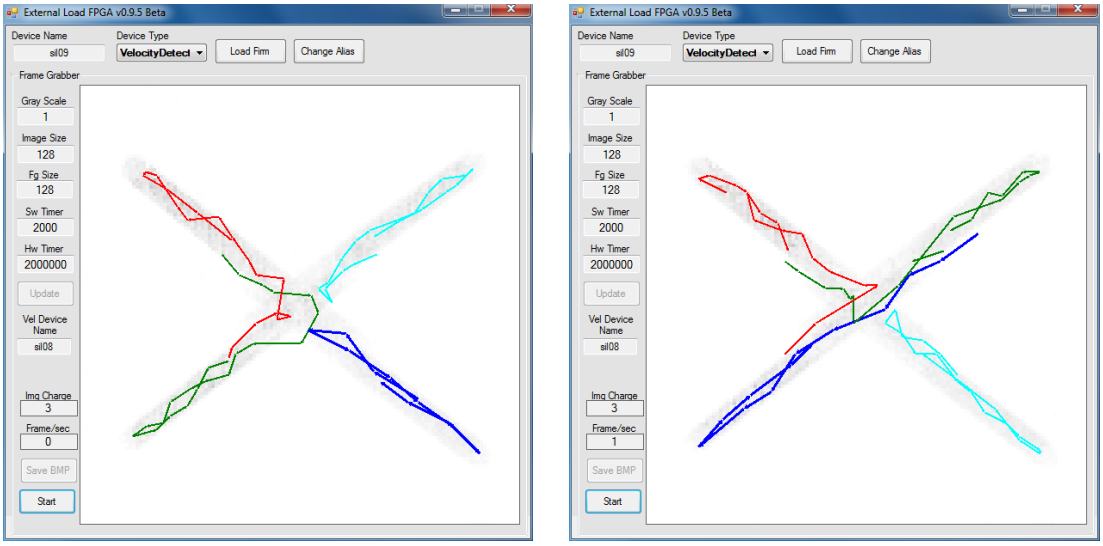


Figura 7.18: Configuración del segundo experimento de trayectorias con cruces

En la Figura 7.19 se muestra la salida del sistema para este segundo experimento, donde el cruce de trayectorias es masivo. En la Figura 7.19.a puede observarse como las TrackCells azul y cian vuelven sobre sus pasos, siguiendo al objeto contrario al que seguían antes del cruce; las roja y la verde intercambian sus objetos tras el cruce. De esta manera, después del cruce, el objeto que seguía la TrackCell azul pasa a ser seguido por la roja; el objeto que seguía la roja pasa a ser seguido por la cian; el que seguía la cian por la verde; y el que seguía la verde por la azul. En la Figura 7.19.b se ve como sólo la TrackCell azul continúa siguiendo al mismo objeto después del cruce, mientras que el resto cambian.

Este intercambio de papeles entre las TrackCells, que sigue los diferentes objetos presentes en la escena, es una consecuencia inherente al funcionamiento del sistema. Es un seguimiento que se podría denominar como local, es decir, la TrackCell no tienen información de cómo es la trayectoria del objeto al que sigue. Además esta información es de más alto nivel y debería ser una capa superior, en el esquema cognitivo, la que se encargue de determinar si el cambio de papeles entre las TrackCells responde o no al cruce de trayectorias.



a) b)

Figura 7.19: Respuesta del sistema al segundo experimento de trayectorias con cruces

Las posibles soluciones a este problema deben venir de dos fuentes: una, la información sobre la forma del objeto; y dos, de la coherencia de la trayectoria detectada, esta coherencia se extraería de un conocimiento sobre las trayectorias de los objetos. Estas posibles soluciones abren una puerta para trabajos futuros, en la línea de completar los siguientes niveles cognitivos.

Resumen

Con esta serie de experimentos se demuestra la capacidad del sistema para tratar la información procedente de un sensor real. Se han probado diferentes circunstancias en las que se demuestra que el sistema es capaz de seguir varios objetos simultáneamente, con velocidades y trayectorias diferentes, con un error máximo en la estimación de la velocidad del 25%, parte de este error es debida al sensor. También se ha comprobado la respuesta del sistema cuando los objetos describen trayectorias que se cruzan, en este caso se demuestra que el sistema es capaz de seguir a todos los objetos después del cruce, si bien, no se puede garantizar que sea la misma TrackCell la que siga al mismo objeto antes y después del cruce. Hay que tener en cuenta que el seguimiento de objeto es local y que las

TrackCells no disponen de información ("trayectorias esperadas", forma de objeto, etc) que le permitan saber si el objeto al que siguen, después del cruce, es el mismo.

7.3. Seguimiento de objetos reales

Con este grupo de experimentos se quiere comprobar el comportamiento del sistema ante estímulos provocados por objetos reales. La configuración del hardware del experimento es la misma que en el caso anterior. La retina "observa" escenas reales. Se usará un ObjectTracker4B como el de los experimentos anteriores, al que se le ha modificado el ancho de las CMCell a 20 píxeles.

7.3.1. Seguimiento de personas

En este primer experimento el estímulo consiste en una escena real donde aparecen tres personas que caminan juntas hasta que a partir de un momento una de ellas se separa. En la Figura 7.20 se muestran dos fotogramas captados con una cámara convencional.

Figura 7.20: Seguimiento de personas, escena real

En la Figura 7.21 se observa la salida del sistema fusionada con la reconstrucción de la salida de la retina. Las capturas de las salidas corresponden al momento en el que la tercera persona se separa del grupo.

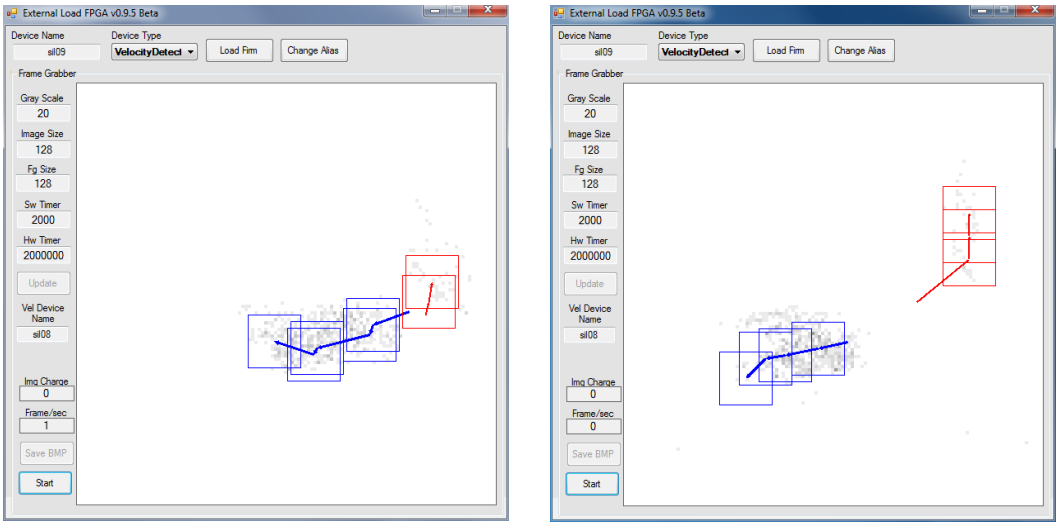


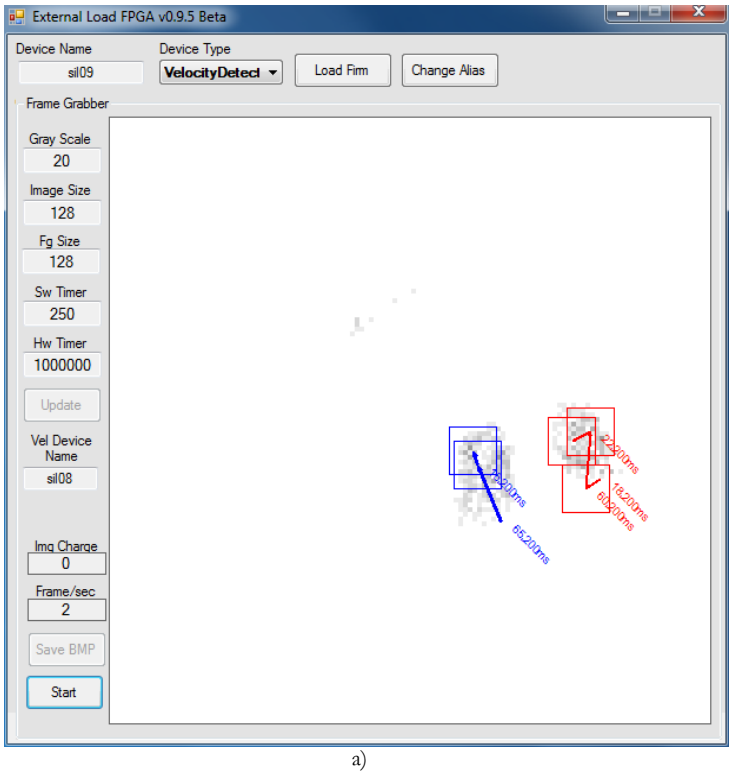
Figura 7.21: Resultado seguimiento personas

Con este experimento se demuestra la capacidad del sistema de seguir objetos reales, en circunstancias reales.

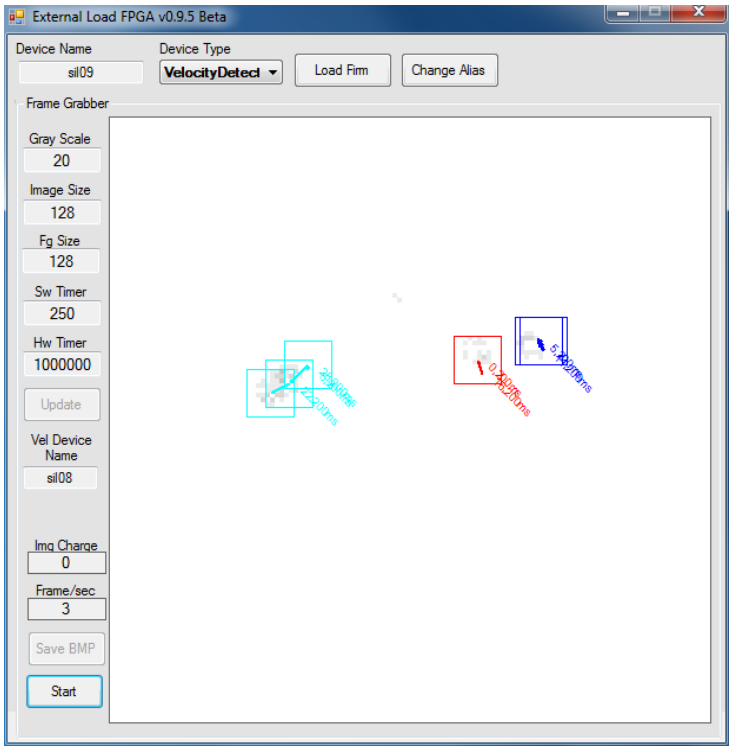
7.3.2. Seguimiento de vehículos

En este experimento la escena se corresponde con el tráfico de vehículos en la vía pública. La Figura 7.22 muestra dos instantes de la escena capturados con una cámara convencional, en la que dos vehículos, que circulan por el carril de la derecha van frenando, y el que se aproxima por la izquierda está acelerando.

Figura 7.22: Seguimiento de vehículos, escena real



a)



b)

Figura 7.23: Resultado seguimiento de vehículos

En la Figura 7.23 se observa la respuesta del sistema donde se puede comprobar cómo el sistema sigue a los vehículos. Además, se muestra la estimación de la velocidad aparente de los vehículos.

Resumen

Estos dos experimentos demuestran que el sistema es capaz de seguir y estimar la velocidad de objetos reales, lo que pone de manifiesto que el sistema puede usarse para resolver problemas reales.

7.4. Detección de patrones con retina en movimiento: detección de orientaciones

En este grupo de experimentos y en los siguientes se pretende demostrar y probar el funcionamiento del procedimiento de detección de patrones propuesto. Para ellos se usarán principalmente sistemas compuestos por TrackCells de tipo C.2.

La configuración del experimento se muestra en la Figura 7.24, que es idéntico al usado para seguimiento de objetos, con la única diferencia que aquí el *ObjetcTracker* está compuesto por TrackCell de tipo C. Por lo tanto, contamos con una retina, conectada a un *BackGroundActivity Filter* que elimina parte del ruido de fondo de la retina; después hay un *FrameGrabber* para observar la entrada al sistema; y por último tenemos un *ObjectTracker*, en concreto un ObjectTracker4C.2.

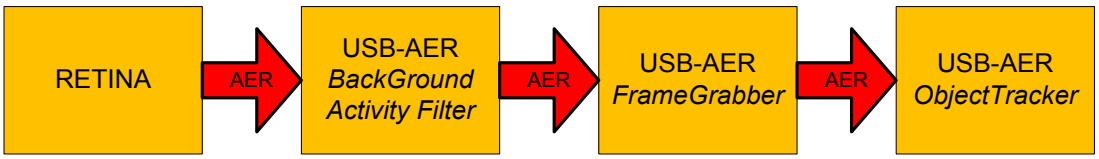


Figura 7.24: Montaje hardware para detección de patrones

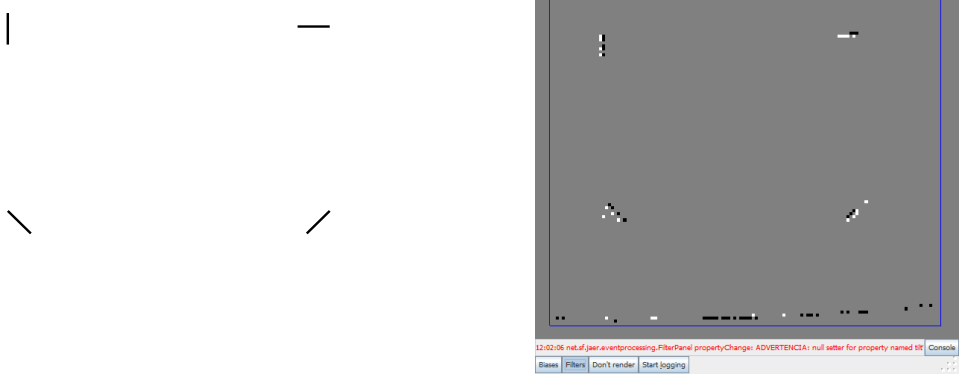
7.4.1. Detección de orientaciones con la retina en movimiento

En este primer grupo de experimentos los objetos están inmóviles, son figuras simples, líneas horizontales, verticales y diagonales. Para crear "movimiento" que pueda ser captado por la retina, ésta está montada sobre un *Pan&Tilt*, que la hace vibrar, creando la ilusión de que los objetos se mueven.

La finalidad de este grupo de experimentos es demostrar la capacidad de detectar orientaciones en los estímulos visuales imitando así el funcionamiento de las neuronas de la capa V1 de la corteza visual (Ng et al. 2007).

Detección de orientación inclinada hacia la derecha

La PRCell está configurada con un patrón para detectar una línea inclinada 45º a la derecha. La Figura 7.25.a muestra el estímulo que se presenta a la retina, que como ya se ha apuntado está inmóvil y es la retina la que se mueve. La Figura 7.25.b muestra la salida de la retina, que a su vez es la entrada del sistema.



a) b)

Figura 7.25: a) Estímulo del experimento de detección de patrones con retina en movimiento, b) salida de la retina

En la Figura 7.26 se muestra la respuesta del sistema. Se puede observar como cada objeto presente en la escena es seguido por una de las TrackCells disponibles en el sistema. Se ha modificado la aplicación del PC para que cuando la TrackCell haya indicado que se ha detectado el patrón, dibuje un cuadrado en la posición donde la detección ha ocurrido. Dicha circunstancia se produce con la TrackCell marcada en color cian. La figura muestra los datos, emitidos por el sistema, acumulados durante 500 ms.

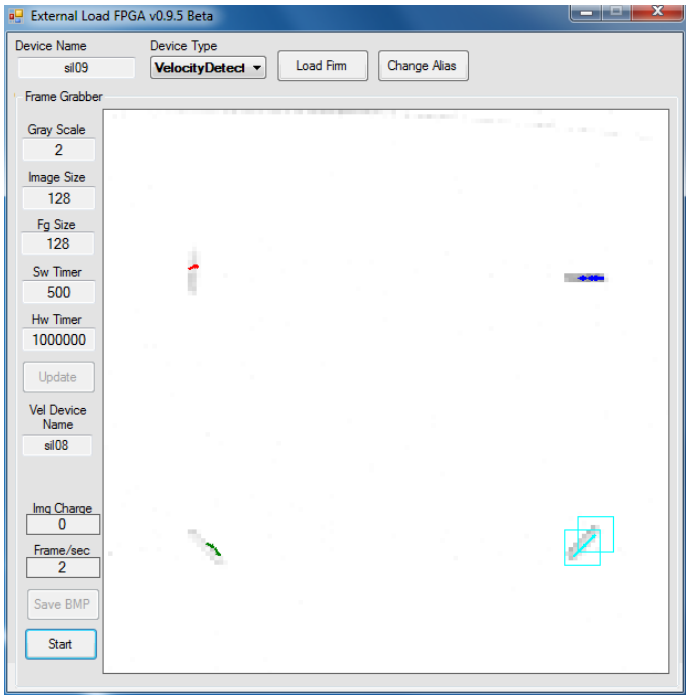


Figura 7.26: Respuesta del sistema detectando una línea inclinada a la derecha, con retina en movimiento

En la Tabla 7.9 se muestra un resumen de los datos que el sistema ha generado durante 1 segundo; durante ese tiempo se han recopilado unos 300 eventos de cada TrackCell, en los que sólo de la TrackCell marcada con el color cian, algunos de ellos, concretamente 176 indicaban que se había detectado el patrón, lo que representa un 58,67%. En el caso de las otras TrackCell, que estaban siguiendo objetos con patrones diferentes al configurado, el número de eventos con detección de patrón positiva fue cero. Se puede afirmar por tanto que cuando el objeto no presenta el patrón deseado no se producen eventos erróneos. Sin

embargo, cuando el patrón si está presente, no hay una detección positiva en todos los casos. Esto es debido, en primer lugar por las imprecisiones a la hora de obtener la posición del objeto, y en segundo lugar por la falta de sincronización entre las salidas de las tres celdas que componen la TrackCell. El funcionamiento de las tres celdas (CMCell, VCell y PRCell) es completamente asíncrono, cada una presenta un tiempo de respuesta diferente:

- La CMCell ofrece un nuevo valor de la localización cada vez que recibe un evento.
- La VCell envía una nueva estimación de velocidad cuando se cumple su periodo de integración, que recordemos se adapta a la velocidad del objeto.
- Y la PRCell envía un evento cada vez que se detecta un patrón en la entrada.

Tabla 7.9: Resultados de la detección de una línea inclinada a la derecha, con retina en movimiento

ID/color	Muestras	Reconocimientos	% reconocimientos
1 azul	300	0	0,00
2 cian	300	176	58,67
3 rojo	316	0	0,00
4 verde	302	0	0,00

A modo de curiosidad se muestra en la Figura 7.27 se muestra la distribución temporal de los eventos producidos por el sistema, en azul y a diferentes alturas los eventos de las celdas 1,2,3 y 4; en rojo y con una altura mayor los eventos de la celda 2 (cian) marcados con detección de patrón positiva.

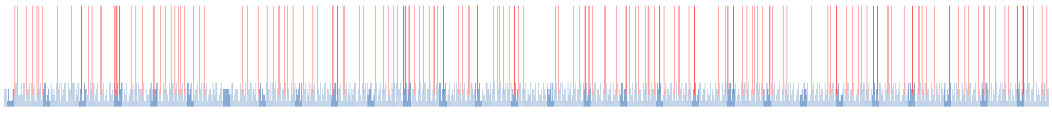


Figura 7.27: Distribución temporal de los eventos de la repuesta a la detección de una línea inclinada a la derecha

Detección de orientación inclinada a la izquierda

En este segundo experimento la configuración hardware y la entrada al sistema son idénticas al caso anterior. Los resultados se muestran en la Tabla 7.10, de nuevo, sólo la TrackCell que sigue al objeto que presenta la orientación deseada emite eventos marcados con detección positiva, concretamente en el 65.69% de los casos.

Tabla 7.10: Resultados de la detección de una línea inclinada a la izquierda, con retina en movimiento

ID/color	Muestras	Reconocimientos	% reconocimientos
1 azul	297	0	0,00
2 rojo	305	0	0,00
3 cian	306	201	65,69
4 verde	307	0	0,00

Detección de orientación vertical

La configuración y la entrada al sistema son idénticas a los casos anteriores para este tercer experimento, pero ahora la TrackCell está configurada para detectar un patrón con una línea vertical, los resultados se muestran en la Tabla 7.11. Los resultados son mejores que en los casos anteriores donde la tasa de reconocimiento del patrón se acerca al 80%, y siguen siendo cero en los casos de las TrackCells que siguen a otros objetos.

Tabla 7.11: Resultados de la detección de una línea vertical, con retina en movimiento

ID/color	Muestras	Reconocimientos	% reconocimientos
1 azul	299	0	0,00
2 rojo	298	0	0,00
3 cian	302	0	0,00
4 verde	299	237	79,26

Detección de orientación horizontal

Igual que en los casos anteriores, pero con las TrackCells configuradas para detectar una línea horizontal. La tasa de detección es del 87,17 en la TrackCell que sigue al objeto que presenta el patrón y cero en aquellas que siguen a objetos que no lo presentan.

Tabla 7.12: Resultados de la detección de una línea horizontal, con retina en movimiento

ID/color	**Muestras**	**Reconocimientos**	**% reconocimientos**
1 azul	304	265	87,17
2 rojo	308	0	0,00
3 cian	306	0	0,00
4 verde	309	0	0,00

Resumen

Como se ha podido comprobar el sistema responde muy bien, en este tipo de experimentos en los que los objetos están inmóviles y la retina vibra para conseguir una ilusión de movimiento. A la vista de los resultados cabe destacar que las TrackCells que siguen objetos que no presentan el patrón deseado, nunca emiten eventos marcados con detección de patrón positiva. La TrackCell que sigue el objeto que sí presenta el patrón emite eventos con detección positiva, aunque no en todos los casos, con una tasa de detección que va desde el 50% al 90%. En los resultados también se puede observar como los patrones horizontal y vertical son detectados un mayor número de veces. Esto podría ser debido a que los patrones son más simples y a que la obtención de la posición es más certera, gracias a que la vibración de la retina es lo suficientemente pequeña como para no cambiar en exceso la posición del objeto.

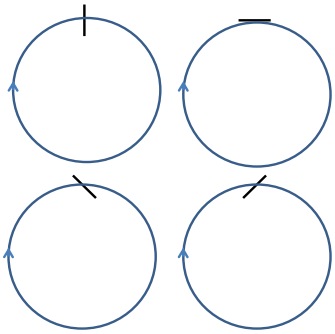
7.4.2. Detección de orientación con objetos en movimiento.

Con estos dos experimentos se trata de demostrar la capacidad del sistema de detectar patrones en los objetos incluso cuando éstos se mueven. La configuración hardware es exactamente la misma que en el apartado anterior.

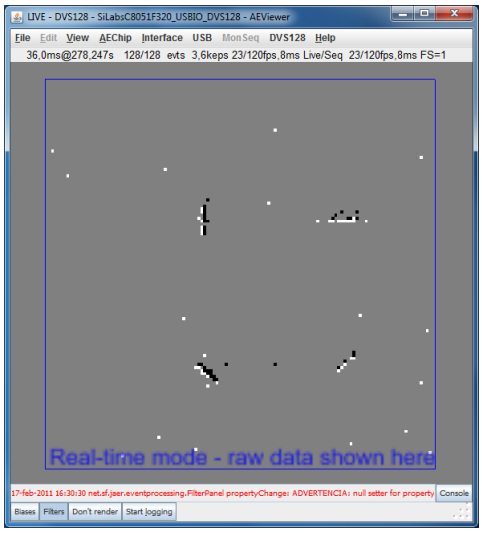
Velocidad baja

El estímulo consiste en los mismos objetos del apartado anterior: una línea vertical, otra horizontal y otras dos inclinadas hacia la derecha y hacia la izquierda; pero en este caso describen una trayectoria circular de 20 píxeles de diámetro aproximadamente. Los objetos

tardan en describir la circunferencia 5 segundos. La Figura 7.28.a muestra el estímulo y la Figura 7.28.b muestra la salida de la retina que a su vez es la entrada del sistema.



a)



b)

Figura 7.28: a) Estimulo experimentos detección de orientaciones con objetos en movimiento, b) salida de la retina

En la Figura 7.29 se muestra la salida del sistema cuando las TrackCells están configuradas para detectar una línea inclinada hacia la derecha, de nuevo la existencia de un cuadrado sobre el objeto indica la detección del patrón, en este caso la línea inclinada hacia la derecha.

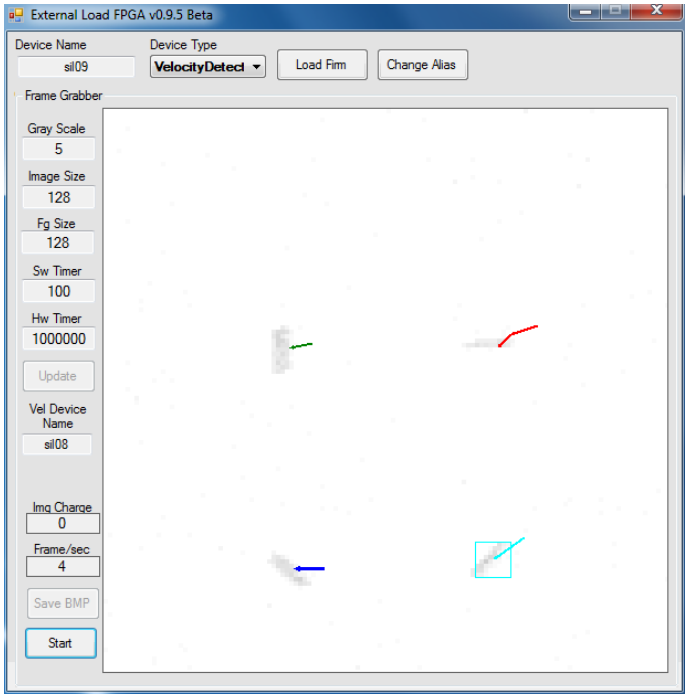


Figura 7.29: Respuesta del sistema detectando una línea inclinada a la derecha, con objetos en movimiento

En la Tabla 7.13 se muestran los resultados para las 4 orientaciones. El porcentaje de detección oscila entre el 41% y el 54%, valores sensiblemente inferiores a los de los experimentos con la retina en movimiento, aunque en los casos en los que las TrackCells siguen objetos que no presentan el patrón los eventos marcados con la detección del patrón siguen siendo 0.

La explicación a la menor tasa de detección hay que buscarla de nuevo en las imprecisiones a la hora de obtener la posición del objeto, debidas, en este caso, al propio movimiento. Además el movimiento en una sola dirección (al contrario que en el caso anterior que la vibración de la retina genera movimiento aparente de los objetos en todas direcciones) dificulta la detección del mismo (véase discusión al respecto en el capítulo 4.5).

Tabla 7.13: Resultados de la detección de orientaciones, con objetos en movimiento a baja valocidad

	ID/color	Muestras	Reconocimientos	% reconocimientos
Línea inclinada a la derecha	1 azul	313	0	0,00
	2 rojo	326	0	0,00
	3 cian	311	130	41,80
	4 verde	307	0	0,00
Línea inclinada a la izquierda	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	306	0	0,00
	2 rojo	306	0	0,00
	3 cian	306	0	0,00
	4 verde	306	167	54,58
Línea vertical	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	324	0	0,00
	2 rojo	324	0	0,00
	3 cian	324	174	53,70
	4 verde	324	0	0,00
Línea horizontal	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	298	142	47,65
	2 rojo	299	0	0,00
	3 cian	298	0	0,00
	4 verde	320	0	0,00

Aún con estas imprecisiones se puede afirmar que el sistema tiene una buena respuesta cuando los objetos se mueven.

Velocidad media

El estímulo es exactamente igual que en el caso anterior salvo por la velocidad de movimiento, los objetos tardan 2 segundos en recorrer las circunferencias. En la Tabla 7.14 se muestran los resultados donde se puede apreciar una disminución de la tasa de reconocimiento, en torno al 16%. La razón de esta disminución es la misma que en el caso anterior, agravada por el aumento de la velocidad.

Tabla 7.14: Resultados de la detección de orientaciones, con objetos en movimiento a velocidad media

	ID/color	Muestras	Reconocimientos	% reconocimientos
Línea inclinada a la derecha	1 azul	557	0	0,00
	2 rojo	583	100	17,15
	3 cian	591	0	0,00
	4 verde	569	0	0,00
Línea inclinada a la izquierda	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	623	3	0,48
	2 rojo	601	101	16,81
	3 cian	594	0	0,00
	4 verde	632	0	0,00
Línea vertical	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	721	0	0,00
	2 rojo	710	0	0,00
	3 cian	693	0	0,00
	4 verde	715	104	14,55
Línea horizontal	**ID/color**	**Muestras**	**Reconocimientos**	**% reconocimientos**
	1 azul	566	0	0,00
	2 rojo	584	0	0,00
	3 cian	580	0	0,00
	4 verde	320	51	15,94

Resumen

Este grupo de experimentos demuestra la capacidad del sistema para detectar patrones simples en una posición determinada de la imagen y previamente calculada por el sistema de localización. El movimiento de los objetos dificulta su localización y por ende la detección del patrón, esta dificultad aumenta con la velocidad. El porcentaje de reconocimientos pasa del 50%, en el caso movimientos a baja velocidad, al 16% cuando las velocidades son moderadas. Si comparamos estos resultados con los experimentos del apartado anterior la reducción de prestaciones es considerable. La corrección de esta reducción ha de ser una línea de trabajo futuro.

"Ciencia es todo aquello sobre lo cual siempre cabe discusión"

José Ortega y Gasset

Capítulo 8

Conclusiones y trabajos futuros

8.1. Resumen de aportaciones

Desde el inicio, este trabajo pretende progresar en el estudio y desarrollo de sistemas neuromórficos, concretamente los basados en AER. Podemos afirmar que este trabajo aporta una demostración de la viabilidad de la construcción de sistemas de visión artificial con una filosofía neuromórfica. Ante la recurrente pregunta sobre la posibilidad del procesamiento de información codificada en pulsos, este trabajo aporta un nuevo elemento encaminado a confirmar esta posibilidad. Además, la construcción de un sistema digital capaz de procesar la información visual supone un acercamiento de los sistemas digitales a los sistemas neuromórficos, como alternativa plausible para la implementación de estos últimos.

Las aportaciones que se muestran en este apartado son todas aquellas que derivan del desarrollo de los objetivos marcados, que recordemos, pretendían la realización de un sistema hardware neuromórfico capaz de detectar patrones, seguir objetos y extraer características del movimiento, como por ejemplo la velocidad; todos ellos usando como sensor una retina artificial y el esquema de comunicación neuromórfico AER.

- Se ha descrito un mecanismo neuromórfico para la localización de objetos en movimiento a partir de la información que proporciona una retina artificial, cuyo funcionamiento se basa en la detección de los cambios de contraste, y que codifica la información en pulsos usando el protocolo AER.

- Se ha descrito un nuevo modelo para la detección de patrones en redes de neuronas pulsantes, *Pattern Integrate and Fire.* El cual nace de la fusión de modelos tradicionales como son el Perceptrón y el modelo *Integrate and Fire.*

- Se ha desarrollado e implementado un dispositivo que se ha denominado CMCell, el cual permite "fijar la atención" en un objeto en movimiento. Obteniendo la información de dicho objeto de una parte de la secuencia AER de la retina, del mismo modo que los campos receptivos de la corteza visual, sólo que en este caso este "campo receptivo" no es fijo, sino que se desplaza al tiempo que el objeto se mueve. La CMCell constituye una pieza fundamental de este trabajo.

- Se ha desarrollado e implementado un dispositivo que se ha denominado VCell, cuyo funcionamiento permite conocer la velocidad del objeto que la CMCell ha localizado y que se adapta a la velocidad de movimiento, permitiendo una estimación de la velocidad más fina y precisa.

- Se ha desarrollado e implementado un dispositivo que se ha denominado PRCell. Esta celda, basada en el modelo *Pattern Integrate and Fire,* permite la detección de patrones visuales simples, tales como la orientación de los mismos, de igual modo que ciertas neuronas de la corteza visual primaria V1. Constituyendo el primer paso para el reconocimiento de formas más complejas.

- Se ha desarrollado e implementado un dispositivo que se ha denominado TrackCell, unión de varias de las celdas anteriores, que permite construir la solución a un problema determinado, gracias a que sus diferentes tipologías ofrecen un amplio abanico de posibilidades. Como en todos los casos anteriores, la información es siempre codificada en pulsos y trasmitida usando AER.

- Se ha desarrollado e implementado un dispositivo que se ha denominado ObjectTracker. Es el elemento organizativo de más alto nivel, el cual usa una arquitectura en cascada o jerárquica de TrackCells, permitiendo el seguimiento simultáneo de tantos objetos como TrackCells lo integren.

- La implementación hardware de todas las celdas mencionadas se ha realizado utilizando elementos digitales comunes tipo FPGA. Uno de los grandes retos de este trabajo era realizar las labores de procesamiento visual en hardware utilizando elementos relativamente simples y poco costosos. La implementación que aquí se ha presentado permite demostrar que un sistema neuromórfico de procesamiento de información visual, procedente de una retina artificial, es posible sin la intervención de un computador convencional.

- Se han realizado aplicaciones reales con los sistemas implementados (seguimiento de personas y vehículos), gracias al uso de una retina artificial, que aporta al conjunto de este trabajo una dimensión realista. Si bien aquí se ha usado una retina en particular, la desarrollada por el INI, es posible la adaptación del sistema a cualquier otra con un funcionamiento similar y codificación de la información en AER.

8.2. Conclusiones

A lo largo de las páginas anteriores se han presentado diversos diseños que pretenden imitar la grandiosidad del sistema de percepción visual de los seres vivos. Las aportaciones aquí presentadas suponen un pequeño paso más, en el camino que conduce a, por una parte la construcción de sistemas inteligentes, y por otra al entendimiento de nuestra propia inteligencia. No sabemos si nunca se construirá una máquina que pueda hacerse pasar por un ser inteligente en todos los ámbitos en los que éstos se desenvuelven, sin que un observador humano se diera cuenta del engaño, pero, casi con toda seguridad, si se construirán máquinas que sean capaces de reproducir algunos de los comportamientos

inteligentes más complejos que los seres vivos demuestran, como es el caso de la visión, la audición o incluso del razonamiento abstracto.

A nuestro entender, los sistemas neuromórficos poseen un gran potencial como sistemas de procesado de información sensorial. Aunque no podemos afirmar si este potencial conseguirá que esta filosofía, la neuromórfica, se convierta en un paradigma para la construcción de sistemas inteligentes. Lo cierto es que con este trabajo, y los que seguro vendrán a continuación, se pone de manifiesto que aún queda un largo camino que recorrer.

Entendemos, por tanto, que las aportaciones aquí presentadas abren nuevas vías que permiten abordar el diseño de sistemas de visión inteligentes neuro-inspirados. Las soluciones aportadas para dos grandes problemas en los sistemas de visión artificial, como son la detección de patrones y el seguimiento de objetos, son un enfoque realista desde el punto de vista de su construcción; y que reflejan de algún modo el sistema neuronal biológico, como no podría ser de otro modo al tratarse de un sistema neuromórfico.

8.3. Trabajos futuros

- Mejorar el consumo energético de los diseños presentados, teniendo como utopía el bajísimo consumo del cerebro humano.
- Mejorar la efectividad de la localización, añadiendo redundancia en el seguimiento. El uso de varias celdas para el seguimiento de un mismo objeto podría mejorar tanto la robustez como la precisión.
- Completar la implementación del mecanismo de seguimiento permitiendo el cambio aparente de tamaño de los objetos.
- Diseñar e implementar sistemas de seguimiento de gran cantidad de objetos, en plataformas hardware de mayor capacidad. Se ha usado una FPGA de bajo coste y por tanto de baja capacidad.

- Estudiar el efecto de la arquitectura jerárquica en los sistemas con redundancia en el seguimiento y con un elevado número de celdas.
- Mejorar la detección de patrones de objetos en movimiento buscando representaciones de los mismos y añadiendo redundancia en localizaciones vecinas a la ofrecida por la CMCell, con el objetivo de minimizar el impacto de una mala localización del objeto.
- Avanzar en el diseño de redes de neuronas basadas en el modelo *Pattern Integrate and Fire*. Introduciendo mecanismos complejos de aprendizaje para neuronas pulsantes basado en la STDP.

Bibliografía y referencias

Adelson, E.H. & Bergen, J.R., 1985. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America. A, Optics and image science*, 2(2), pp.284-99. Available at: http://www.ncbi.nlm.nih.gov/pubmed/3973762.

Avis, C. et al., 2001. Report on Telluride 2001.

Barron, J.L., D.J., F. & Beauchemin, S.S., 1994. Systems and Experiment Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1), pp.43-77.

Berge, H.K.O. & Hafliger, P., 2007. High-Speed Serial AER on FPGA. *2007 IEEE International Symposium on Circuits and Systems*, pp.857-860. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4252770.

Boahen, K. a, 2000. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5), pp.416-434. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=842110.

Bohte, S.M., 2004. The evidence for neural information processing with precise spike-times: A survey. *Natural Computing*, 3(2), pp.195-206.

Botella Juan, G., 2007. *Implementación en hardware reconfigurable de un modelo de flujo óptico robusto*. Universidad de Granada.

Bredfeldt, C.E. & Ringach, D.L., 2002. Dynamics of spatial frequency tuning in macaque V1. *The Journal of neuroscience*, 22(5), pp.1976-1984.

CappoCacciaWorkshop, 2011. The 2011 CapoCaccia Cognitive Neuromorphic Engineering Workshop. Available at: http://capocaccia.ethz.ch/capo/wiki/2011.

Caviar Project, 2006. http://www2.imse-cnm.csic.es/caviar/.

Cesar Urtubia-Icario, 1996. *Nuerobiología de la Visión* 2nd ed. UPC, eds., Barcelona.

Chan, V., Liu, S.-C. & van Schaik, Andr, 2007. AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(1), pp.48-59. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4061005.

Chan, V., Schaikt, A.V. & Liu, S.-chii, 2006. Spike Response Properties of an AER EAR prelamsnary localizathon determcne uenc-hae functhons. *2006 13th IEEE International Conference on Electronics, Circuits and Systems*, pp.859-862.

Chris Eliasmith, 2003. *Neural engineering: computation, representation and dynamics in neurobiological systems* 1st ed., London: The MIT Press.

Cucchiara, R. et al., 2003. Detecting Moving Objects , Ghosts , and Shadows in Video Streams. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(10), pp.1337-1342.

Dante, V., Giudice, P.D. & Whatley, Adrian M, 2005. Interfacing with address events. *The neuromorphic engineer*, 2(1), pp.21-23.

De Valois, R.L., Albrecht, D.G. & Thorell, L.G., 1982. Spatial frequency selectivity of cells inmacaque visual cortex. *Vision Research*, 22(5), pp.545-559.

DeAngelis, G.C. et al., 1999. Functional micro-organization of primary visual cortex: receptive field analysis of nearby neurons. *The Journal of neuroscience*, 19(10), pp.4046-4064.

Deiss, S.R., Douglas, Rodney & Whatley, A. M., 1999. A pulsed-coded Communications infraestructura for Neuromorphic system. In W. Mass & C. M. Bishop, eds. *Pulsed Nueral Networks*. Cambridge; Massachusetts; London: MIT press.

Delbruck, T. & Lichtsteiner, P., 2007. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. *2007 IEEE International Symposium on Circuits and Systems*, (ISCAS), pp.845-848.

Delbruck, Tobi et al., 2010. Activity-Driven , Event-Based Vision Sensors. In *2010 IEEE International Symposium on Circuits and Systems*. Paris, France: IEEE Computer Society, pp. 2426-2429.

Dragoi, V., Sharma, J. & Sur, M., 2003. Response Plasticity in Primary Visual Cortex and its Role in Vision and Visuomotor Behaviour : Bottom-up and Top-down Influences. *IETE Journal of Research*, 49(2), pp.1-9.

Duda, R.O., Hart, P.E. & Strok, D.G., 2001. *Pattern Classification*, New York: Jonh Wiley and Sons.

Engbert, R., 2003. Microsaccades uncover the orientation of covert attention. *Vision Research*, 43(9), pp.1035-1045.

Fasnacht, D.B., Whatley, Adrian M. & Indiveri, G., 2008. A serial communication infrastructure for multi-chip address event systems. *2008 IEEE International Symposium on Circuits and Systems*, pp.648-651. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4541501.

Fernández-Domínguez, G. et al., 2010. Object Tracking on Embedded hardware. In Ahmed Nabil Belbachir, ed. *Smart Cameras*. Boston, MA: Springer US, pp. 199-223.

FPGA, 2010. http://en.wikipedia.org/wiki/Field-programmable_gate_array.

Gerstner, W et al., 1997. Neural codes: firing rates and beyond. *Proceedings of the National Academy of Sciences of the United States of America*, 94(24), pp.12740-1.

Gibson, J.J., 1950. *The Perception of the Visual World*, Houghton Mifflin Company. Available at: http://bjp.rcpsych.org/cgi/doi/10.1192/bjp.98.413.717-a [Accessed March 28, 2011].

Gomez-Rodriguez, F. et al., 2007. AER Auditory Filtering and CPG for Robot Control. In *2007 IEEE International Symposium on Circuits and Systems*. Ieee,

pp. 1201-1204. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4252858.

Gomez-Rodriguez, F. et al., 2005. Two hardware implementations of the exhaustive synthetic AER generation method. *Lecture Note in Computer Science*, 3512, pp.534 - 540.

Gomez-Rodriguez, F. et al., 2006. AER tools for communications and debugging. In *2006 IEEE International Symposium on Circuits and Systems*. IEEE, p. 4.

Gonzalez, R.C. & Woods, R.E., 2002. *Digital Image Processing* second. A. Dworkin, ed., New Jersey: Prentice-Hall, Inc.

Gu, H., Shirai, Y. & Asada, M., 1996. MDL-based segmentation and motion modeling in a long image sequence of scene with multiple independently moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1), pp.58-64. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=476012.

Hammond, P., Movat, G.S.V. & Smith, A.T., 1985. Motion after-effects in cat striate cortex elicited by moving gratings. *Experimental Brain Research*, 60, pp.411-416.

Higgins, C.M. & Koch, C., 1999. Multi-Chip Neuromorphic Motion Processing. In D. Wills & S. DeWeerth, eds. *Proc. 20th Anniversary Conference on Advanced Research in VLSI*. Los Alamitos: IEEE Computer Society, pp. 309-323.

Horn, B.K.P. & Schunck, B.G., 1981. Determining Optical Flow. *Artificial Intelligence*, 17, pp.185-203.

Hubel, D., 1995. *Eye, Brain, and Vision*, Available at: http://hubel.med.harvard.edu/book/bcontex.htm.

Hubel, D.H. & Wiesel, T.N., 1962. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *MostJournal of Physiology*, 160(1), pp.106-154.

Häfliger, P., 2007. CAVIAR Hardware Interface Standards , Version 2.01. *Interface*.

Hélène, P.-M., 2006. SPIKING NEURON NETWORKS, A survey.

Indiveri, G., Chicca, E. & Douglas, Rodney, 2006. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 17(1), pp.211-21. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16526488.

Indiveri, G. et al., 2009. Neuromorphic Systems. In L. R. Squire, ed. *Encyclopedia of Neuroscience*. Elsevier Ltd, pp. 521-528.

INE, 2011. The Institute of Neuromorphic Engineering. Available at: www.ine-web-org.

JAER, 2011. http://sourceforge.net/apps/trac/jaer/wiki.

Jain, R., Kasturi, R. & Schunck, B.G., 1995. *Machine vision*, McGraw-Hill.

Jimenez-Fernandez, A. et al., 2007. Address-event-based platform for bioinspired spiking systems P. Arena, A. Rodriguez-Vazquez, & G. Linan-Cembrano, eds. *Proceedings of SPIE*, 6592(1), pp.659206-659206-10. Available at: http://link.aip.org/link/PSISDG/v6592/i1/p659206/s1&Agg=doi [Accessed May 1, 2011].

Jimenez-Fernandez, Angel et al., 2009. From Vision Sensor to Actuators , Spike Based Robot Control through Address-Event-Representation. *Lecture Note in Computer Science*, 2, pp.797-804.

Jiménez Fernández, A.F., 2010. *Diseño y evaluación de sistemas de control y procesamiento de señales basados en modelos neuronales pulsantes*. Universidad de Sevilla.

Kandel, E.R., Schwartz, J.H. & Jessell, T.M., 2000. *Principles of Neural Science* 4th ed., New York: McGraw-Hill.

Kempter, R., Gerstner, Wulfram & van Hemmen, J., 1999. Hebbian learning and spiking neurons. *Physical Review E*, 59(4), pp.4498-4514. Available at: http://link.aps.org/doi/10.1103/PhysRevE.59.4498.

Kock, C., 1999. *Biophysics Of Computation. Information Processing in Single Neurons* 1st ed. Michael Stryker, eds., New York: Oxford University Press, Inc.

Kottke, D.P. & Sun, Y., 1994. Motion estimation via cluster matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11), pp.1128-1132. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=334394.

Laughlin, S.B. & Sejnowski, T.J., 2003. Communication in neuronal networks. *Science*, 301(5641), pp.1870-4.

Legenstein, R., Naeger, C. & Maass, Wolfgang, 2005. What can a neuron learn with spike-timing-dependent plasticity? *Neural computation*, 17(11), pp.2337-82. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16156932.

Lichtsteiner, Patrick, Posch, Christoph & Delbruck, Tobi, 2008. A 128x128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2), pp.566-576.

Linares-Barranco, A. et al., 2007. Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. In *2007 IEEE International Symposium on Circuits and Systems*. Ieee, pp. 1192-1195. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4252855.

Linares-Barranco, A. et al., 2010. On the AER convolution processors for FPGA. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, pp. 4237-4240. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5537577 [Accessed March 12, 2011].

Linares-Barranco, Alejandro et al., 2006. On algorithmic rate-coded AER generation. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 17(3), pp.771-88. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16722179.

Litzenberger, M et al., 2006. Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor. In *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*. IEEE, pp. 173-178.

Liu, S. et al., 2002. Orientation-Selective aVLSI Spiking Neurons Orientation-. *Advances in Neural Information Processing Systems*, 14.

Liu, S.-C. & Delbruck, Tobi, 2010. Neuromorphic sensory systems. *Current opinion in neurobiology*, 20(3), pp.288-95.

Maass, W., Schnitger, G. & Sontag, E., 1991. On the computational power of sigmoid versus boolean threshold circuits. In *Proc. of the 32nd Annual IEEE Symposium on Foundations of Computer Science*. pp. 767-776.

Maass, Wolfgang, 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9), pp.1659-1671.

Mahowald, M., 1994. *An analog VLSI system for stereoscopic vision* 1st ed., Boston: Klumber Academic Publisher.

Mahowald, M., 1992. *VLSI Analogs of Neuronal Visual Processing : Thesis by*. California Institute of Technology.

Martinez-conde, S., Macknik, S.L. & Hubel, D.H., 2000. Microsaccadic eye movements and firing of single cells in the striate cortex of macaque monkeys. *Nature neuroscience*, 3(4), p.409.

Martín, J.L. et al., 2005. Hardware implementation of optical flow constraint equation using FPGAs. *Computer Vision and Image Understanding*, 98(3), pp.462-490. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1077314204001638 [Accessed March 28, 2011].

Matías Casado, M., 2010. Estimación del Flujo Óptico en Sensores de Imagen AER.

Mead, C., 1990. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10), pp.1629-1636.

Miro-Amarante, L. et al., 2006. A LVDS Serial AER Link. In *2006 13th IEEE International Conference on Electronics, Circuits and Systems*. Ieee, pp. 938-941. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4263522.

Miro-Amarante, L. et al., 2007. LVDS Serial AER Link performance. In *2007 IEEE International Symposium on Circuits and Systems*. IEEE, pp. 1537-1540.

Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4252944 [Accessed August 17, 2010].

Mishkin, M., Ungerleider, L.G. & Macko, K. a, 1983. Object vision and spatial vision: two cortical pathways. *Trends in Neurosciences*, 6, pp.414-417.

Nageswaran, J.M., Dutt, N. & Delbrueck, T., 2009. Computing spike-based convolutions on GPUs. In *2009 IEEE International Symposium on Circuits and Systems*. Ieee, pp. 1917-1920. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5118157.

Nesi, P., Del Bimbo, A. & Ben-Tzvi, D., 1995. A Robust Algorithm for Optical Flow Estimation. *Computer Vision and Image Understanding*, 62(1), pp.59-68.

Ng, J., Bharath, A. a & Zhaoping, L., 2007. A Survey of Architecture and Function of the Primary Visual Cortex (V1). *EURASIP Journal on Advances in Signal Processing*, 2007, pp.1-18.

Panjwani, D.K. & Healey, G., 1995. Markov random field models for unsupervised segmentation of textured color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10), pp.939-954. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=464559.

Pardàs, M. & Salembier, P., 1994. 3D morphological segmentation and motion estimation for image sequences. *Signal Processing*, 38, pp.31-43.

Paz Vicente, R., 2008. *Una aportación al procesamiento de la información visual mediante técnicas bioinspiradas.* Universidad de Sevilla.

Paz, R. et al., 2005. Test infrastructure for address-event-representation communications. *Lecture Notes in Computer Secience*, 3512, pp.518 - 526.

Paz-Vicente, Rafael et al., 2008. Image convolution using a probabilistic mapper on USB-AER board. In *2008 IEEE International Symposium on Circuits and Systems*. IEEE, pp. 1056-1059. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4541603.

Ponulak, F. & Kasiński, A., 2010. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural*

computation, 22(2), pp.467-510. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19842989.

Potter, S.M., 2007. What Can AI Get from Neuroscience? In M. Lungarella et al., eds. *50 Years of Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 174-185.

Rivas, M. et al., 2005. Tools for Address-Event-Representation Communication. *Lecture Notes in Computer Science*, 3696, pp.289 - 296.

Rosin, P.L., 1998. Thresholding for Change Detection. In *Proceeding of the International Conference of Computer Vision*. pp. 274-279.

Saeedi, P. et al., 2002. A low-level motion and slippage controller for control of a tracked mobile robot. , 1(8), pp.1-15.

Schweitzer, H.S., 1995. Occam algorithms for computing visual motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11), pp.1033-1042. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=473229.

Senn, W., 2002. Beyond spike timing: the role of nonlinear plasticity and unreliable synapses. *Biological cybernetics*, 87(5-6), pp.344-55. Available at: http://www.ncbi.nlm.nih.gov/pubmed/12461625.

Serrano-Gotarredona, R. et al., 2006. High-speed image processing with AER-based components. In *2006 IEEE International Symposium on Circuits and Systems*. IEEE, p. 4.

Serrano-Gotarredona, R. et al., 2005. AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems. In *Advances in Neural Information Processing Systems. NIPS2005*. Vancouver, CANADA, pp. 1217-1225.

Serrano-Gotarredona, R. et al., 2008. On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing. *IEEE Transactions on Neural Networks*, 19(7), pp.1196-1219. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4490276.

Serrano-Gotarredona, Rafael et al., 2009. CAVIAR: a 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing- learning-actuating system for high-speed visual object recognition and tracking. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 20(9), pp.1417-38. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5173584 [Accessed August 17, 2010].

Serrano-Gotarredona, Rafael et al., 2006. A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(12), pp.2548-2566. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4026698.

Sivilotti, M., 1991. *Wiring considerations in Analog VLSI Systems, with application to Field-Programmable Networks*. Cal. Inst of Tech.

SRAM_Datasheet, 2010. http://www.cypress.com/?docID=25229. Available at: http://www.cypress.com/?docID=25229.

Strain, T.J. et al., 2006. A Supervised STDP Based Training Algorithm with Dynamic Threshold Neurons. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp.3409-3414. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1716565.

Szeliski, R. & Shum, H.-Y., 1996. Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12), pp.1199-1210. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=546257.

TellurideWorkshop, 2011. Telluride Neuromorphic Cognition Engineering Workshop 2011. Available at: https://neuromorphs.net/nm/wiki/2011.

Thorpe, S.J., Delorme, A. & Vanrullen, R., 2001. Spike-based strategies for rapid processing Spike-based strategies for rapid processing. *Neural Networks*, 14(33), pp.715-726.

Tovée, M.J., 2008. *An Introduction to the Visual System* Second., Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo: Cambridge University Press.

Uchiyama, T. & Arbib, M. a, 1994. Color image segmentation using competitive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12), pp.1197-1206. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=387488.

Ungerleider, L.G. & Haxby, J.V., 1994. "What" and "where" in the human brain. *Current Opinion in Neurobiology*, 4, pp.157-165.

van Rossum, M.C., Bi, G.Q. & Turrigiano, G.G., 2000. Stable Hebbian learning from spike timing-dependent plasticity. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 20(23), pp.8812-21. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16711840.

Verri, A. & Poggio, T., 1989. Motion Field and Optical Flow : Qualitative Properties. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2(5), pp.490-498.

VHDL Analysis and Standardization Group, 2010. http://www.eda.org/vhdl-200x/.

Von der Heydt, R., Peterhans, E. & Dürstler, M.R., 1992. Periodic-pattern-selective cells in monkey visual cortex. *The Journal of neuroscience :*, 12(4), pp.1416-1434.

Xilinx Inc., 2008. Spartan II datasheet. Available at: http://www.xilinx.com/support/documentation/data_sheets/ds001.pdf.

Yang, Z. et al., 2006. A neuromorphic depth-from-motion vision model with STDP adaptation. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 17(2), pp.482-95. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19695273.

Yosehp, B.-C. & Cynthia, B., 2003. *Biologically Inspired Intelligent Robots* 1st ed., Bellingham, Washington: SPIE.

Zamarreño, C., Serrano-gotarredona, R. & Serrano-gotarredona, T., 2008. LVDS interface for AER links with burst mode operation capability. *2008 IEEE International Symposium on Circuits and Systems*, pp.644-647. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4541500.

Zhang, Z., 1995. Estimation Motion and Structure from Correspondences of Line Segments Between Two Perspective Images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(12), pp.1129-1139.