Trabajo Fin de Máster
"Máster Universitario en Microelectrónica:
Diseño y Aplicaciones de Sistemas
Micro/Nanométricos"

"Tecnología de visión basada en eventos: concepto e implementación práctica"

Autor: Marcelo Fco. Simbaña Romero

Tutores: Jorge Fernández Berni y Juan Antonio Leñero Bardallo

Fecha: 5 de septiembre de 2022

RESUMEN

Este trabajo consistirá en el estudio de la tecnología de visión basada en eventos, la cual se basa en un método para la adquisición de datos visuales inspirado en la biología. Es por ello, por lo que los sensores basados en esta tecnología, aparte de llamarse sensores basados en eventos, también se conocen como sensores bio-inspirados. A lo largo de este trabajo se hará una revisión de la historia de estos sensores para entender cómo han evolucionado hasta la actualidad, así como un estudio de su principio de operación para entender su fundamento

Se verá que esta forma de adquirir imágenes ofrece múltiples ventajas sobre la forma de adquisición convencional que utilizan los sensores CMOS, que se conoce como adquisición de imágenes basadas en *frames*. Algunas de estas ventajas son una mayor resolución temporal o un mayor ancho de banda. No obstante, para explotar estas ventajas es necesario diseñar nuevos algoritmos y formas de procesado o adaptar aquellos basados en *frames* para obtener las distintas características de la escena que luego serán útiles en distintos campos, como la robótica.

Como aplicación práctica del trabajo, se estudiarán distintos ejemplos de implementación y aplicación práctica de tres algoritmos de procesado de imagen para sensores de visión de tipo DVS. Se utilizará un sensor de visión DVS comercial con un kit de desarrollo y algunas de las herramientas proporcionadas por su fabricante.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Los sensores de visión basados en eventos se caracterizan por proporcionar una salida cuando ocurre algún acontecimiento en la escena como, por ejemplo, una variación en la luminancia. Dentro de estos sensores existen aquellos que se conocen como bio-inspirados, que son el foco principal de este trabajo. Los sensores bio-inspirados reciben este nombre porque se basan en la retina del ojo humano, emulando la comunicación en el nervio óptico a través de una codificación de la información en forma de pulsos. La salida de estos sensores se conoce como eventos y, en función del tipo de sensor, estos eventos representarán distintas características de la escena.

Los sensores de visión bio-inspirados que predominan en la industria hoy en día tienen unas características concretas que son producto de la evolución de años de investigación y que, en algunos casos, difieren mucho de algunas de las características desarrolladas en sensores anteriores.

Este capítulo busca introducir al lector de forma breve cuál ha sido la historia y evolución de este tipo de sensores para así poder contextualizar las características de los sensores modernos de visión basada en eventos. Además, se explicará cual es la motivación, objetivos y organización de este trabajo.

1.1. Antecedentes

El primer modelo de retina lo desarrolló Fukushima [1] en 1970 utilizando elementos discretos, pero su impacto en la industria fue insignificante. No fue hasta 1992 cuando Mahowald and Mead desarrollaron e implementaron el primer modelo de retina en un proceso de integración a gran escala (VLSI) de silicio [2] (Ver Sección 1.1.2.A). Tanto este modelo como el anterior buscaban emular el comportamiento de la retina del ojo humano a través de componentes electrónicos, por lo que es interesante entender primero el funcionamiento de la retina humana y cómo los primeros sensores bio-inspirados se basaron en modelos de esta para desarrollarse.

1.1.1. Retina biológica

La Figura 1.1 muestra un esquema de las principales partes del ojo humano, que se compone de tres capas: la capa externa, la capa media y la capa interna. La retina se encuentra en la capa interna. Es el lugar donde se transforma la energía lumínica de la luz en energía eléctrica y se lleva al sistema nervioso a través del nervio óptico.

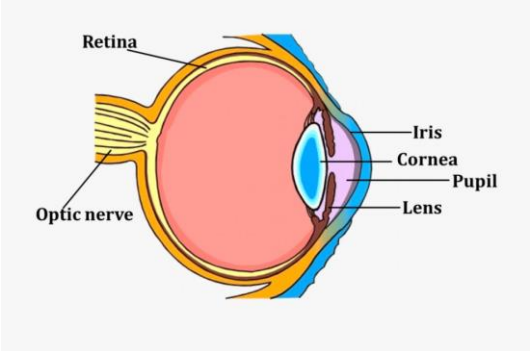

Figura 1.1. Partes del ojo humano [3].

El primer científico que estudio la retina en detalle fue Ramón y Cajal [4] en 1893 en un trabajo que recogió observaciones de trabajos previos para presentar un estudio de los tipos de células de la retina en los vertebrados, un trabajo que se considera como el punto de partida para cualquier estudio anatómico sobre la retina [5][6][7][8].

La Figura 1.2 muestra la sección transversal de la retina del ojo humano. Esta se compone principalmente de tres capas: la capa fotorreceptora, la capa plexiforme externa y la capa plexiforme interna.

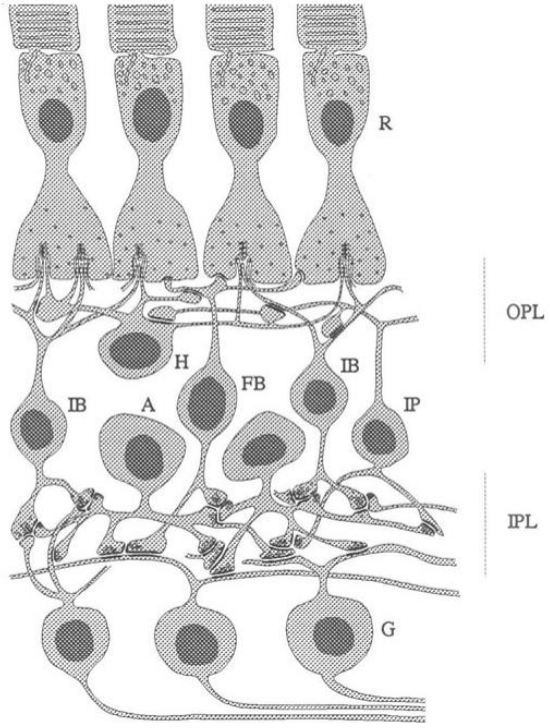

Figura 1.2. Sección transversal de la retina del ojo humano R: Capa fotorreceptora, OPL: Capa plexiforme externa, IPL: Capa plexiforme interna, H: Células horizontales, IB: Célula bipolar invaginante, FB: Célula bipolar plana, A: Célula amacrina, IP: Célula interplexiforme, G: Célula ganglionar [9].

La capa fotorreceptora está compuesta por dos tipos de células: los conos y bastones. Estos se encargan de transformar la luz entrante en una señal eléctrica que estimulan las células bipolares y horizontales de la capa plexiforme externa [10].

Existen dos clases de células bipolares: las células bipolares invaginantes y las células bipolares planas. Estas células codifican por separado los cambios de

contraste espaciotemporal, comparando las señales que recibe de la capa fotorreceptora con valores medios almacenados en las células horizontales. Para ello, las células horizontales y los terminales de las células bipolares se unen en los pedículos de los bastones y conos de los fotorreceptores, donde las células bipolares son excitadas por las diferencias entre las salidas de los fotorreceptores y las células horizontales y, a su vez, retroalimentan a los fotorreceptores.

Las células bipolares invaginantes y planas conectan con muchos tipos de células amacrinas y ganglionares a través de sinapsis químicas en la capa plexiforme interna. De forma similar a las células horizontales, las células amacrinas median el proceso de transmisión de señales entre las células bipolares y las células ganglionares.

Brevemente, las células bipolares y ganglionares se pueden dividir en grupos de células con una respuesta más sostenida y células con una respuesta más transitoria. Estas células transmiten la información a lo largo de dos caminos paralelos en la retina: la vía magnocelular, donde las células son sensibles a los cambios temporales de la escena, y la vía parvocelular, donde las células son sensibles a las formas de la escena. A través de estas y muchas más vías, la información visual llega a la corteza cerebral para ser procesada.

Las retinas biológicas tienen muchas características, salvando las distancias entre biología y tecnología, que no poseen los sensores de visión de silicio convencionales, como un gran rango dinámico de intensidad lumínica, que permite la visión en condiciones de iluminación de más de 9 décadas, desde la luz de las estrellas hasta la luz del sol. Además, las retinas biológicas eliminan información redundante, ya que las células de la capa plexiforme externa e interna codifican los cambios de contraste espaciotemporal en lugar del nivel de brillo absoluto. Como se verá posteriormente, estas características son propias de los sensores bioinspirados y son algunas de las ventajas respecto a los sensores convencionales.

1.1.2. Historia

Los sensores DVS (*Dynamic Vision Sensor*, *DVS*) y algunas de sus variantes son los que han predominado en la industria y son los más usados actualmente. No obstante, para llegar hasta este tipo de sensores se ha tenido que partir de diseños más primitivos que, en un principio, buscaban seguir una arquitectura similar a la retina humana explicada en el apartado anterior. Es por esto por lo que es interesante ver cómo históricamente los primeros sensores bio-inspirados han ido evolucionando hasta los actuales.

Desde que Misha Mahowald y Carver Mead [2] desarrollaron el primer sensor bio-inspirado se han ido desarrollando otros tipos de sensores con distintas características.

Estos sensores se pueden diferenciar en dos categorías [10]:

- **Sensores de contraste espacial**. Basados en relaciones de intensidad absoluta, son útiles para escenas estáticas, analizándolas para obtener distintas

características de la escena o para clasificación de objetos. Ejemplos de estos sensores son el desarrollado por Misha y Carver Mead (Sección 1.1.2.A) o el sensor Parvo-Magno (Sección 1.1.2.B).

- **Sensores de contraste temporal**. Basados en cambios relativos de intensidad, son útiles para escenas dinámicas en las que la iluminación no es uniforme y se utilizan para seguimiento de objetos o navegación. El máximo exponente de este tipo de sensores es el DVS (Sección 1.1.2.E).

Además, en función del tiempo de exposición y el mecanismo de reinicio, se pueden dividir en:

- **Sensores síncronos.** Los píxeles utilizan el mismo tiempo de exposición y sus salidas son leídas de forma síncrona.
- **Sensores asíncronos.** Los píxeles generan eventos independientes sin sincronización con los demás píxeles de la matriz, buscando asemejarse a la retina humana, que es totalmente asíncrona Este tipo de sensores son más complejos de implementar mediante hardware que los síncronos ya que varios píxeles pueden requerir la transmisión de forma simultánea.

Siguiendo esta clasificación, a continuación, se hará un repaso de los principales sensores bio-inspirados que se han ido desarrollando a lo largo de la historia.

A. Misha Mahowald y Carver Mead

Fue el primer sensor bio-inspirado, desarrollado por Misha Mahowald y Carver Mead entre 1986 y de Doctorado de Mahowald [2]. Era un sensor de contraste espacial y buscaba emular la detección de contraste espacio-temporal de la retina y la interacción/inhibición entre células vecinas.

Para ello, cada fotorreceptor de silicio imitaba a un cono y contenía un fotosensor de tiempo continuo y un circuito adaptativo que ajustaba su respuesta para hacer frente a los niveles de luz cambiantes. Además, una red de resistencias variables MOS imitaba la capa de células horizontales, proporcionando una retroalimentación basada en la media de luz que incidía en los fotorreceptores. Por último, un circuito que imitaba las células bipolares amplificaba la diferencia de señal entre el fotorreceptor y la media local y rectificaba esta señal amplificada en las salidas de ON y OFF.

Este sensor abrió el campo de investigación de los sensores bio-inspirados como lo conocemos hoy en día y, aunque se obtuvo un circuito con un comportamiento muy similar al de una retina biológica, el sensor tenía la desventaja de que presentaba un gran *mismatch* en los transistores, provocando un desajuste significativo en la respuesta de los píxeles, por lo que necesitaba de métodos precisos de calibración y circuitería robusta para disminuirlo. Además, el tamaño de los píxeles era demasiado grande para un uso práctico. Por último, era un sensor síncrono, por lo que requería procesar los valores de iluminación de toda la matriz de píxeles, lo que no lo hacía competitivo en términos de velocidad contra los sensores tradicionales.

B. Parvo-Magno

En el año 2004 Zaghloul y Boahen presentaron el sensor Parvo-Magno [11]. Este era una mejora del presentado por Mahowald y Mead, ya que buscaba modelar tanto la capa plexiforme externa como la interna, mientras que el propuesto por Mahowald y Mead solo modelaba la capa plexiforme externa, incluyendo circuitos que simulaban células bipolares y ganglionares de respuesta sostenida y transitoria.

Es un sensor asíncrono de contraste espacial y temporal, lo que le otorgaba las principales características adaptativas de las retinas biológicas, como la adaptación a la luz y al contraste. Este sensor tenía complejas e interesantes propiedades, pero también la desventaja de que el *mismatch* entre la respuesta de los píxeles era demasiado grande como para tener un uso práctico.

C. VISe (VIsion Sensor)

Denominado Sensor de Visión de Contraste Espacial y Orientación, es un sensor síncrono de contraste espacial diseñado por Ruedi en 2003 [12]. Permitía procesar tanto el nivel de contraste como la orientación angular del mismo.

Se diferencia de los anteriores en que estos sensores emiten una salida eventos siguiendo un orden en el tiempo. Este orden viene determinado por la magnitud del nivel contraste que experimentan los píxeles, por lo que permite reducir en gran medida la cantidad de datos. Operan de la siguiente manera: dado un tiempo de exposición para todos los píxeles, los eventos que se transmitirán al terminar este tiempo vendrán ordenados de mayor a menor contraste espacial, es decir, los píxeles que transmitirán primero sus eventos serán aquellos que hayan experimentado un mayor cambio en el nivel de contraste. Esto permite que la lectura pueda ser abortada si se tiene un tiempo de procesado limitado sin perder la información de los cambios de contraste más grandes.

Los píxeles de este sensor poseen un gran rendimiento en términos de bajo *mismatch* y alta sensibilidad, lo que le da un gran rango dinámico. No obstante, este enfoque se abandonó porque era difícil su escalado a procesos tecnológicos más pequeños, entre otros motivos.

D. Octopus

Este tipo de sensor fue desarrollado por Eugenio Culurciello en 2003 [13]. Este sensor se desmarca de los anteriores porque no modelaba a la retina biológica, sino que más bien usaban un protocolo AER (Sección 2.4) para comunicar directamente los valores de intensidad en los píxeles codificados en frecuencia a través de una señal de pulsos. Como se verá más adelante, los protocolos AER se usan para el envío y recepción de señales en sistemas neuro mórficos. Se denominó “Octopus” porque se asemeja al ojo de los pulpos y otros cefalópodos, en el que los fotorreceptores están detrás de la retina en vez de delante y parte del procesamiento de la retina se realiza en el lóbulo óptico.

Presentan algunas ventajas respecto a los sensores convencionales, como la velocidad, ya que la frecuencia de los pulsos puede alcanzar valores del orden de kHz en interiores, dándoles una resolución temporal del orden de miles de imágenes por segundo. Además, es un sensor asíncrono, ya que los píxeles no tienen que ser escaneados para obtener su valor de iluminación, sino que envían pulsos siempre y cuando estén iluminados. Esto implica también la ventaja de ahorro de energía, ya que, si no están iluminados, no envían pulsos. Otra ventaja es que poseen un gran rango dinámico, pues la frecuencia de los píxeles puede expandirse hasta en cinco décadas en escala logarítmica, lo que les da la capacidad de detectar niveles de iluminación intraescena con un rango dinámico superior a 100dB.

Por contraparte, su gran desventaja es que la calidad de la imagen es muy baja, ya que el ruido FPN (*Fixed-Pattern Noise*) asociado al *mismatch* de los transistores es muy grande en este tipo de sensores.

E. Dynamic Vision Sensor (DVS)

Fue diseñado en primer lugar por Patrick Lichtsteiner y colaboradores en 2008 [14]. Es un sensor de contraste temporal y asíncrono que tiene como salida un flujo de eventos que codifican los cambios de brillo en la escena con una resolución temporal del orden de microsegundos. Al ser asíncrono, permite reducir el flujo de datos y el consumo de potencia.

El hecho de responder de forma asíncrona ante cambios en el contraste temporal en vez de los cambios absolutos de iluminación permite mejorar a los sensores basados en imágenes en términos temporales (son más rápidos, tienen mayor resolución temporal, etc.).

Como ya se ha mencionado, el tipo y variaciones de este sensor han sido las que han predominado en la industria y será el foco principal de este trabajo (Sección 2.5.1).

F. Asynchronous Time Based Image Sensor (ATIS)

Aunque para muchas aplicaciones solo es necesario el procesamiento de los eventos, en algunas puede ser de gran ayuda también tener una salida estática de luminancia absoluta. Esto es lo que busca este sensor presentado por Christoph Posch en 2011 [15]. Los píxeles de este sensor combinan un píxel DVS con una unidad de medida de intensidad (IMU), de forma que los eventos generados por el sensor DVS se usan como *trigger* para capturar medidas en escala de grises con el sensor IMU.

G. Dynamic and Active Pixel Vision Sensor (DAVIS)

Desarrollado por C. Brandli en 2014 [16] este sensor combina un píxel CMOS convencional APS (*Active Pixel Sensor*) con un pixel DVS, con el objetivo de compartir el mismo circuito del fotodiodo. Su ventaja respecto al ATIS es que ocupa un área menor.

La Tabla 1.1 compara los principales sensores bio-inspirados que se han desarrollado a lo largo de la historia y el ojo humano.

Tabla 1.1. Comparación de los principales sensores bio-inspirados y el ojo humano [10].

Sensor	Parvo-Magno	VISe	DVS	ATIS	Ojo humano
Año	2001	2003	2006	2010	N/A
Funcionalidad	Asíncrono. Contraste espacial y temporal.	Basado en tramas. Contraste espacial y temporal. Salida ordenada.	Asíncrono. Sensor de visión dinámico de contraste temporal.	Imagen asíncrona basada en tiempo.	>50 clases de células
Clase	CE, CT, SA	CE, SS	CT, SA	CT, SA	Retina de mamíferos
Tamaño del píxel (µm)	34x40	69x69	40x40	30x30	1x1
Resolución	96x96	128x128	128x128	304x240	100M
Consumo/pixel	10 µW	18 µW	400 nW	1.3 µW	30 pW
Rango dinámico	~50dB	120dB	120dB	125dB	180dB
Latencia, frames/s (fps), events/s (Meps)	10 Meps	<2ms (desde 60 a 500 fps)	2 Meps	30 Meps	20ms

CE: Contraste espacial, CT: Contraste temporal, SA: Sensor asíncrono, SS: Sensor síncrono

Principalmente se puede observar que cualquier tipo de sensor bio-inspirado tiene unas características muy pobres respecto al ojo humano. No obstante, lo importante es que se aprecia cómo las distintas investigaciones han buscado mejorar el rango dinámico, así como reducir la potencia y aumentar la matriz de píxeles.

1.2. Motivación

Los sensores de visión basados en eventos se postulan como una alternativa a la forma en la que se obtiene la información visual. Debido a sus particulares propiedades, existe un gran interés comercial en explotarlas en distintos campos, como la robótica o la conducción autónoma. No obstante, son precisamente estas propiedades las que plantean el reto de adaptar años de desarrollo de algoritmos para cámaras de visión tradicionales o desarrollar nuevos algoritmos que permitan obtener las mismas características de una manera eficiente. El estudio de este tipo de algoritmos y flujo de desarrollo es lo que motiva a la realización de este trabajo.

1.3. Objetivos del trabajo

Los objetivos del trabajo son realizar un estudio tanto a nivel hardware como de aplicación de los sensores de visión basados en eventos. Esto implicará abordar su operación tanto a nivel de píxel como a nivel algorítmico y de software.

A nivel hardware se deberán entender los principales componentes de este tipo de sensores y cuáles son las relaciones entre sí, mientras que a nivel software se requerirá un estudio de cómo se implementan los distintos algoritmos que permiten obtener las características deseadas de la imagen.

1.4. Organización

El trabajo se ha dividido en cinco capítulos:

En primer lugar, en este capítulo introductorio se ha buscado repasar brevemente la historia de estos sensores para entender por qué tienen las características actuales y se han explicado las motivaciones y objetivos del trabajo.

En el segundo capítulo se verán las características técnicas de los sensores de visión basados en eventos y los tipos que actualmente predominan en la industria.

El tercer capítulo recoge el estado actual de esta tecnología en términos de implementación en distintos sistemas. En este capítulo se hace un repaso del paradigma actual de estos sensores, así como las distintas aplicaciones en las que pueden ser útiles. También se estudiarán algunos sistemas que se están comercializando actualmente.

En el cuarto capítulo se explicarán algunos métodos para la ejecución de distintas aplicaciones recogidas en el tercer capítulo y se harán una serie de experimentos con kit de desarrollo Gen 3.1 VGA de Prophesee, explicando el proceso de implementación de los algoritmos que permiten ejecutar la aplicación deseada.

Por último, en el quinto capítulo se expondrán las conclusiones relacionadas con este trabajo.

2. FUNDAMENTOS DE LOS SENSORES DE VISIÓN BASADOS EN EVENTOS

Los sensores de visión convencionales tienen como principio de operación adquirir imágenes completas de la escena con una tasa de muestro constante. Esto se conoce como visión basada en fotogramas (*frame-based vision*), y se basa en obtener información de todos los píxeles del sensor en cada una de estas imágenes, lo que implica que no se tiene en cuenta si la información en los píxeles sigue siendo la misma o no, procesándose igualmente. También conlleva un tiempo de procesado que va a limitar la resolución temporal, teniendo como consecuencia que se pierda información de la escena. Además, esto aumenta el volumen del flujo de datos, por lo que se requieren de más recursos computacionales para el procesado de esta información.

En cambio, los sensores de visión basados en eventos o bio-inspirados, tienen un principio de operación basado en la biología. Se puede decir que estos sensores, al igual que en la visión biológica, se rigen por los acontecimientos que ocurren en la escena, ya que la información de cada píxel se recoge de forma independiente y asíncrona, es decir, cuando hay un cambio de información en un píxel, este se recoge tan pronto como ocurre. Esto se conoce como visión basada en eventos (*event-based vision*) y resuelve algunos problemas que tiene la visión basada en frames, como el hecho de no perder información entre la toma de las imágenes o no almacenar datos redundantes.

No obstante, esta nueva forma de procesar imágenes implica que los algoritmos de procesamiento de tradicionales no puedan ser aplicados y se tenga que desarrollar una nueva forma de procesar este tipo de información que permita explotar todo su potencial.

En este capítulo, se explicará con detalle qué son las cámaras basadas de eventos, cuál es su principio de operación, qué tipos se han desarrollado a lo largo de la historia, etc. También se hará un repaso de las principales aplicaciones, así como de algunos algoritmos desarrollados hasta ahora.

2.1. ¿Qué son las cámaras basadas en eventos?

Las cámaras basadas en eventos utilizan sensores de visión bio-inspirados para proporcionar una imagen de salida que se forma a través de los cambios de brillo locales a nivel de píxel. Estas cámaras no capturan imágenes usando un disparador como las cámaras convencionales, sino que cada píxel opera de forma independiente y asíncrona, reportando cambios en el brillo del píxel tan pronto como van ocurriendo y permaneciendo inactivos cuando no hay cambios [17].

2.1.1. Diferencias respecto a las cámaras convencionales

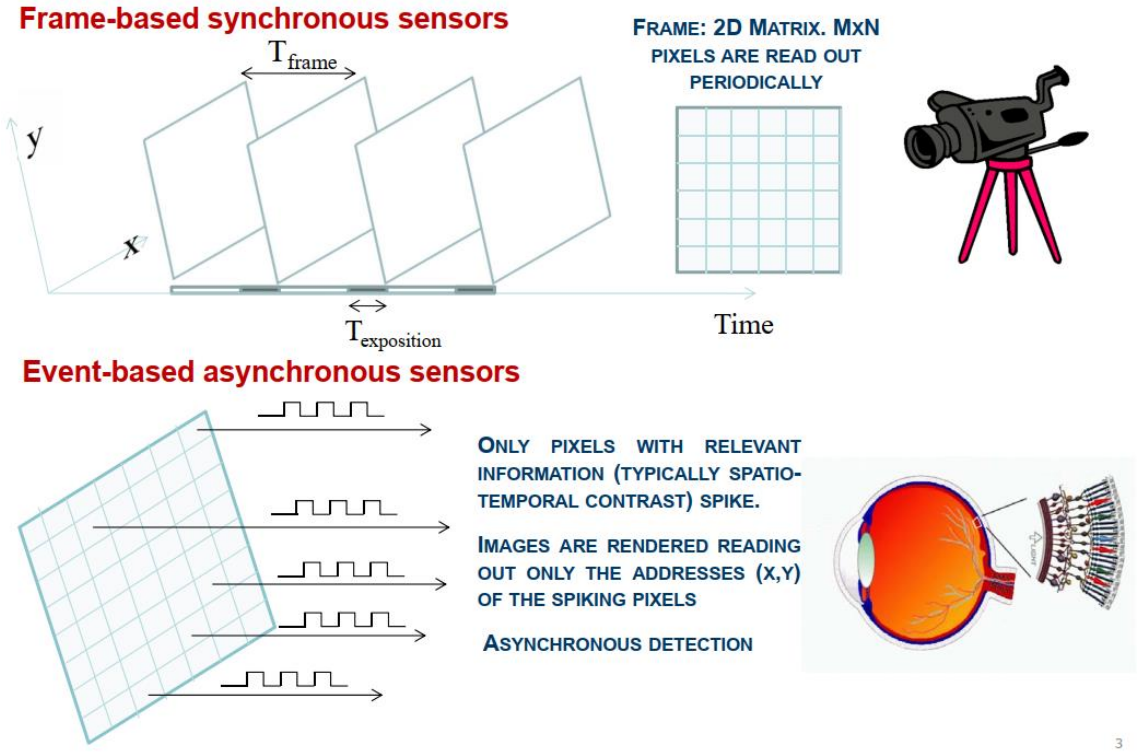

Figura 2.1. Comparación del principio de operación entre la visión basada en frames (arriba) y la visión basada en eventos (abajo) [18].

Para entender cómo funcionan los sensores de visión basados en eventos, primero se debe entender cómo funcionan los sensores de visión basados en fotogramas.

Los sensores de visión basados en fotogramas adquieren imágenes completas de la escena midiendo el nivel de brillo de todos los píxeles de la matriz, y notificándolo a la periferia del sensor para su procesamiento. El número de estas imágenes viene dado por un reloj externo, que dicta el número de imágenes por segundo o *frames per second (fps)*. En el caso de la Figura 2.1, se puede observar cómo cada imagen se toma cada intervalo de tiempo T_{frame} dando lugar a que se pueda perder información importante entre las imágenes. Este tiempo T_{frame} determina el número de *frames per second* de la grabación. Por lo general, este número suele ser del orden de 25 o 30, lo que da lugar a una resolución temporal del orden de milisegundos. Por otro lado, cada una de estas imágenes tarda en tomarse un tiempo $T_{exposure}$, por lo que es posible que la información cambie cuando se están leyendo los píxeles, lo que daría lugar a lo que se conoce como desenfoque de movimiento (*motion blur*).

Sin embargo, los sensores de visión basados en eventos, de forma análoga a la visión biológica, monitorean continuamente el nivel de brillo en cada uno de los píxeles y, cuando ha habido un cambio determinado en el nivel de brillo, notifican a la periferia del sensor que ha sucedido este cambio. Este cambio en el nivel de brillo se denomina *evento*. En el caso de que el nivel de brillo se haya incrementado se dice que el evento es positivo y, en caso contrario, se dice que el evento es negativo. Una vez se ha notificado un evento a la periferia, el nivel de brillo actual se almacena en

cada píxel para seguir monitoreando cambios en este, por lo que los sensores de visión basados en eventos, al contrario que los sensores basados en fotogramas reportan y miden cambios relativos en el brillo.

Cabe destacar que la detección de estos eventos se hace de forma asíncrona y no está impuesta por un reloj externo, solucionando el problema de la pérdida de información. Además, los eventos se envían con una resolución temporal del orden de microsegundos y el hecho de que midan cambios relativos en el nivel de brillo les permite obtener un mayor rango dinámico.

Otro ejemplo se muestra en la Figura 2.2, donde se aprecia que la salida de los sensores basados en fotogramas son imágenes formadas por la lectura de todos los píxeles, mientras que la salida de un sensor bio-inspirado es una serie de eventos que indican si ha habido un incremento o decremento en el brillo de un píxel.

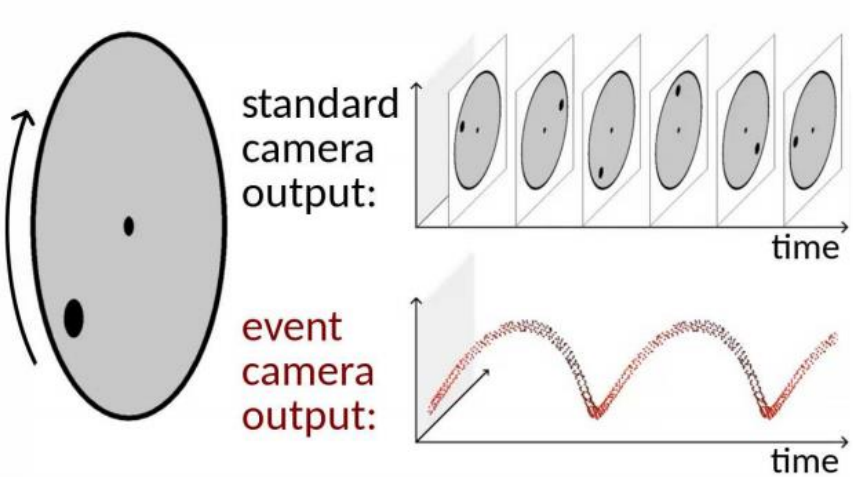

Figura 2.2. Salida de una cámara convencional comparada con la de una cámara de eventos [19].

El hecho de que los sensores basados en fotogramas requieran de una lectura completa de toda la matriz de píxeles implica un gasto de energía que a veces es innecesario, pues se puede dar el caso de que el nivel de brillo de cada píxel no haya cambiado. Esto también se soluciona con los sensores de visión bio-inspirados, reduciendo el consumo de potencia en aproximadamente tres órdenes de magnitud.

Las principales diferencias entre las cámaras convencionales y cámaras basadas en eventos se recogen en la Tabla 2.1.

Tabla 2.1. Características de las cámaras convencionales y cámaras de eventos [20].

Característica	**Cámaras convencionales**	**Cámaras de eventos**
Resolución temporal	~ ms	~ µs
Rango dinámico	60 dB	140 dB
Desenfoque de movimiento	Sí	No
Consumo de potencia (*die level)*	10 mW	10 µW

Estas características las hacen muy interesantes en campos donde la velocidad del procesado de las imágenes es importante, como la robótica, pero también en campos donde interesa tener un gran rango dinámico y bajo consumo de potencia, como las imágenes panorámicas 3D, escaneo 3D o reconstrucción de imágenes de gran rango dinámico.

2.2. Ventajas y desventajas

Algunas de las ventajas de las cámaras de eventos respecto a las cámaras convencionales tienen origen en las diferencias explicadas anteriormente:

- **Alta resolución temporal**. En la parte analógica, se puede decir que el monitoreo de los cambios de brillo es bastante rápido, mientras que, en la parte digital, la lectura se hace con relojes digitales del orden de megahercio, por lo que los eventos se detectan y se leen en un tiempo del orden de los microsegundos. Esto permite que las cámaras de eventos no sufran de desenfoque de movimiento típico de las cámaras convencionales.
- **Baja latencia**. Dado que cada píxel trabaja de forma independiente, un evento se transmite tan pronto como se detecta, lo que permite que las cámaras de eventos tengan una latencia mínima, por debajo de los milisegundos.
- **Bajo consumo**. Las cámaras de eventos solo transmiten cambios en el nivel de brillo del píxel, por lo que la potencia consumida recae en el procesado de los cambios de brillo en el píxel. En términos absolutos, el consumo de sistemas embebidos con cámaras de eventos puede ser incluso menor que 100 mW [20].
- **Gran rango dinámico**. Los fotorreceptores de los píxeles trabajan en escala logarítmica y de forma independiente, sin esperar a un disparador global. Esto permite obtener un rango dinámico mayor que 120 dB, notablemente superior a los 60 dB de las cámaras convencionales de alta calidad. Por lo tanto, los píxeles de las cámaras de eventos pueden adaptarse tanto a estímulos con muy baja luminosidad como aquellos con muy alta luminosidad.

No obstante, cabe recalcar que para aprovechar estas características aún existen algunos retos en el paradigma actual que deben resolverse (Sección 3.1.2).

2.3. Principio de operación de los sensores DVS

Las cámaras de eventos responden a los cambios en el brillo de la escena de forma asíncrona e independiente por cada píxel, dando como salida una secuencia de pulsos digitales que representan cambios de brillo en los píxeles de una determinada magnitud y un determinado tiempo, ambos ya predefinidos. Estos pulsos se conocen como eventos. Esta codificación está inspirada en la naturaleza de vías visuales biológicas, de ahí lo del término “sensores de visión bio-inspirados”.

Cada píxel memoriza su nivel de intensidad cada vez que envía un evento y continúa monitoreándola para enviar otro evento cuando haya un cambio suficientemente grande de este valor memorizado. Cuando esto ocurre, la cámara envía un evento y se transmite al chip junto a las coordenadas x e y del píxel, el tiempo t, y la polaridad p del evento. Esta polaridad tiene dos valores: positivo, en el caso de que el brillo haya incrementado o negativo, en el caso de que haya decrementado.

Los sensores de visión que integran este tipo de píxeles se denominan DVS (*dynamic vision sensor*). Un ejemplo se muestra en la Figura 2.3.

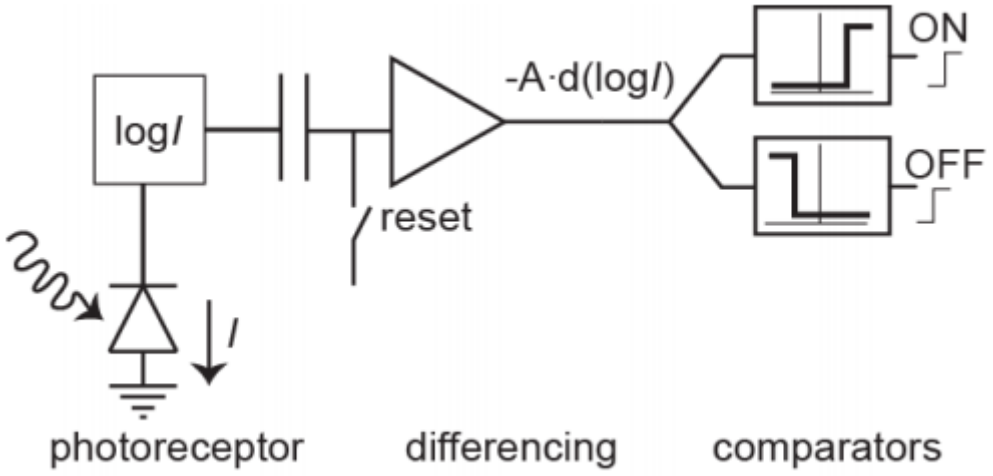

Figura 2.3. Esquema de un sensor DVS [14].

Por lo general, estos sensores están compuestos de tres etapas:

1. La etapa del fotorreceptor, que incluye el fotorreceptor en sí y una primera etapa de poca ganancia, pero logarítmica, proporcionando a la salida un nivel de tensión que cambia ante valores logarítmicos de la intensidad *I* generada por el fotorreceptor.
2. Un circuito diferenciador de poca ganancia, que amplifica la señal que viene del fotorreceptor. Aunque tiene poca ganancia es suficiente para detectar el contraste temporal. Este amplificador se reinicia cada vez que se genera un evento para guardar el valor de la tensión que generó el evento.
3. Por último, un par de comparadores que evalúan la señal amplificada que viene del circuito amplificador. Cuando la señal toma un valor determinado, los comparadores hacen de disparadores para generar el evento. Cuando se activan, reinician el amplificador y comunican a la periferia del chip que ha ocurrido un evento en el píxel correspondiente. Ahí, la circuitería correspondiente se encarga de capturar la polaridad del evento, la localización y el tiempo correspondiente.

Cabe destacar que las cámaras de eventos son sensores cuya salida, por lo general, dependerá de la cantidad de movimiento en la escena, ya que este cambio de movimiento se traducirá en cambios en la intensidad. Esto se debe a que la luz incidente en un píxel es un producto de la iluminación de la escena y la reflectancia de la superficie. Como la iluminación suele ser constante, un cambio de intensidad logarítmica en la escena suele indicar un cambio de reflectancia debido al movimiento en el campo de visión de la cámara.

Cuanto más rápido sea este movimiento, más eventos por segundo se generarán, ya que cada píxel adapta la tasa de muestreo del modulador delta encargado de transformar la señal analógica a digital en función del valor del cambio de intensidad del píxel que monitoriza. Los eventos se capturan con una precisión de

microsegundos y se transmiten con una latencia de décimas de milisegundos, lo que hace que estos sensores reaccionen rápidamente ante estímulos visuales.

2.4. Protocolo de comunicación AER (*Address Event Representation*)

AER (*Address Event Representation*) es un protocolo para la comunicación de eventos neuronales entre chips neuro mórficos. Este protocolo nace de la necesidad encontrar una forma alternativa de simular la comunicación de pulsos entre las neuronas del cerebro en sistemas neuro mórficos, es decir, comunicaciones punto a punto entre neuronas [21][22].

El problema reside en que en el cerebro humano se pueden implementar comunicaciones punto a punto ya que contiene alrededor de 10^{11} neuronas, cada una de las cuales contiene entre 1000 y 10000 conexiones con otras neuronas. La implementación de esta cantidad de conexiones no es posible en un chip, ya que los chips están formados en un plano de dos dimensiones, mientras que las neuronas del cerebro humano se despliegan en las tres dimensiones del espacio [23].

El principio en que se basa este protocolo de comunicación reside en aprovechar la ventaja de la velocidad proporcionada por los sistemas digitales para compensar la desventaja en densidad y número de conexiones respecto a los sistemas neuro mórficos reales. Como se verá a continuación, esto se consigue multiplexando las salidas de los píxeles en el tiempo mediante el uso de un bus compartido para transmitir sus coordenadas.

2.4.1. Encoders y decoders

Por lo general, el objetivo de este protocolo es comunicar un grupo de neuronas con otras, es decir, se partirá de un grupo emisor y un grupo receptor, tal y como se muestra en la Figura 2.4. Para ello, el grupo emisor usará circuitos AER que multiplexen/codifiquen los pulsos que provienen de su matriz de neuronas, mientras que el grupo receptor usará circuitos AER que demultiplexen/decodifiquen estos pulsos. Estos circuitos se conocen como *encoders* y *decoders*.

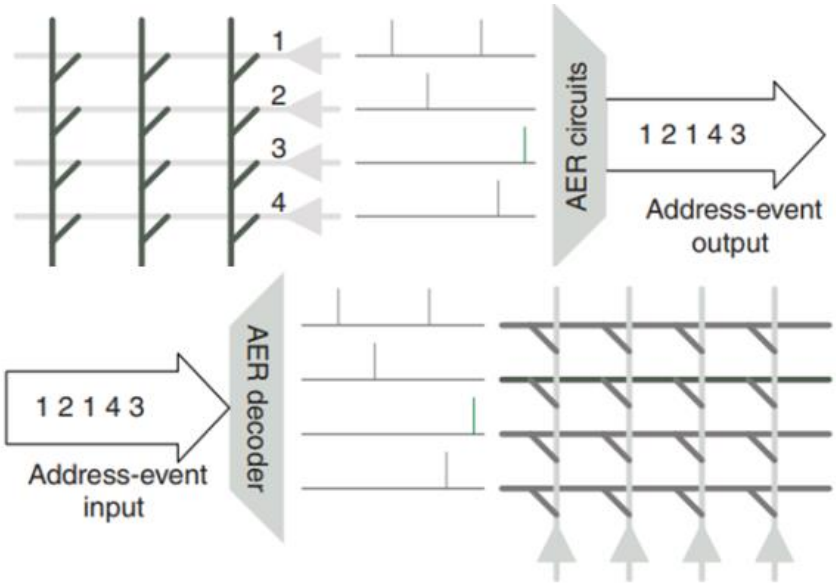

Figura 2.4. Representación del bus de datos compartido para el grupo emisor (arriba) y receptor (debajo) [24].

En el lado emisor, la secuencia de valores que se envían por el bus de datos indica qué neurona ha enviado un pulso. El momento en que se genera este identificador corresponde al momento en que la neurona produjo el pulso, más un pequeño retraso debido al proceso de codificación. En su forma más simple, N diferentes neuronas se enrutan usando $\log N$ bits. En el lado receptor, los valores de entrada recibidos por el bus indican los identificadores de las neuronas que reciben el pulso. Estos identificadores se decodifican y el pulso llega correctamente a su destino con un tiempo de retraso respecto al momento entró en el circuito. Cabe destacar que las direcciones de las neuronas emisoras y receptoras no tienen por qué coincidir. En sistemas más complejos, los identificadores de las neuronas emisoras se transforman en identificadores o grupo de identificadores de neuronas receptoras.

Es necesario mencionar que también se deben implementar mecanismos para la resolución de conflictos tanto como por el lado emisor como el receptor para evitar colisiones. Hay una gran variedad de mecanismos como, por ejemplo, *Full Arbitration, Collision Discarding,* o *Aging versus Loss Trade Off* [23]. Cada uno tiene sus ventajas y desventajas y la elección de este depende del sistema neuromórfico en el que se va a implementar.

2.4.2. Comunicación *On-Chip*

Hay dos métodos distintos que utilizan los circuitos AER emisores para enviar los pulsos. El primero, y más simple, se basa en organizar linealmente las neuronas transmisoras; un único *encoder* recoge todos los pulsos y los canaliza a un bus de salida. Este método es muy útil para sistemas con pocas neuronas. Cuando el número de neuronas se incrementa, se utiliza el segundo método que consiste en formar las neuronas en una matriz de dos dimensiones y usar dos *encoders* para identificar la fila y la columna de la neurona que envía el pulso. Este método es el que se usa para los sensores bio-inspirados y es el que se va a explicar a continuación.

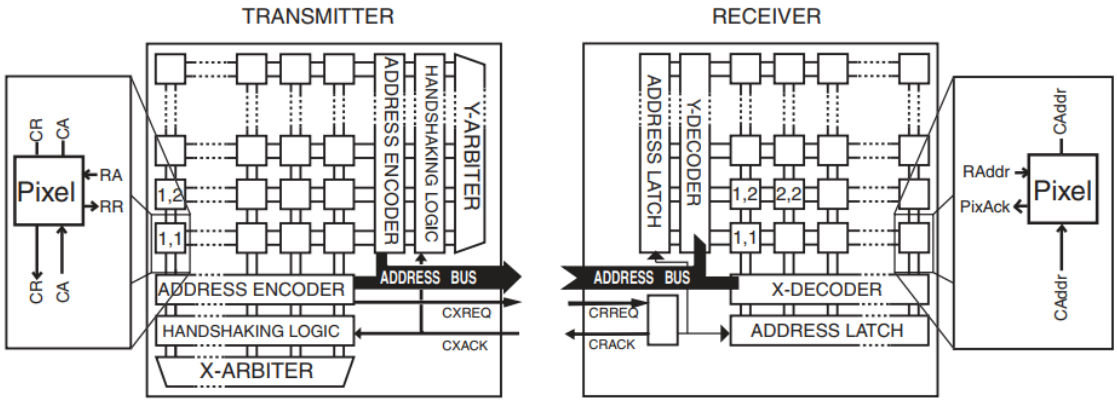

Figura 2.5. Diagrama de bloques de un circuito AER tanto para un bloque 2D emisor como un bloque 2D receptor [24].

En la Figura 2.5 se puede observar la organización a nivel de bloques de los circuitos AER encargados de codificar y decodificar los pulsos que se envían entre un chip emisor y uno receptor.

Centrándonos en el chip emisor, que es el que simularía a una matriz de píxeles de un sensor bio-inspirado, cada una de las neuronas va a tener dos señales de salida denominadas *Column Request* y *Row Request* (CR y RR) y dos señales de entrada llamadas *Column Acknowledge* y *Row Acknowledge* (CA y RA). Las señales RR y RA se comparten para todas las neuronas de una fila, mientras que las CR y CA se comparten para todas las de una columna. Además, cada conjunto de filas y columnas tiene tres bloques principales:

- **Codificador de dirección (*Address encoder*).** Se encargan de codificar la dirección de la columna o fila de la neurona que manda el pulso.
- **Lógica de *handshaking*.** Se encargan de gestionar la comunicación *off-chip* y evitar colisiones.
- **Árbitro.** El árbitro se encarga de gestionar las señales de *request* y *acknowledge* de cada una de las filas (árbitro Y) y columnas (árbitro X).

El proceso es el siguiente: cuando una neurona quiere enviar un pulso, activa su señal RR. El árbitro Y recibe la petición y cuando es el turno de la fila de la neurona en cuestión para enviar el pulso, activa la señal RA. En ese momento, la neurona activa la señal CR. Al igual que el árbitro Y, el árbitro X activará la señal CA cuando sea el turno de esa columna para transmitir. Cuando la neurona recibe tanto la señal RA como CA, retira la petición y da por enviado el pulso. Además, los árbitros se comunican con los codificadores de dirección de las filas y con la lógica de *handshaking* para codificar la fila y columna de la neurona y gestionar la comunicación fuera del chip a través de las señales CXREQ y CXACK.

2.4.3. Comunicación *Off-Chip*

La comunicación *off-chip* se gestiona a través de los circuitos AER que implementan la lógica de *handshaking*. En el caso descrito en la Figura 2.5, el emisor usará dos rutas de comunicación, una para los datos (dirección de la neurona emisora) y otra para el control (envío de las señales de *request* y *acknowlege*). Los datos deben ser

válidos antes de que el emisor active la señal de *request* CXREQ*.* Cuando el lado receptor recibe esta señal a través de la señal CRREQ, activa su señal de *acknowledge* CRACK y captura la dirección de la neurona. De forma similar al lado emisor, el lado receptor posee decodificadores que se encargan de estimular la neurona a través de su fila y columna correspondiente a con el uso de las señales RAddr y CAddr. Finalmente, una vez la señal CRREQ se desactiva por parte del chip emisor, el receptor desactivará la señal CRACK cuando reciba la señal de *akcnowledge* por parte de la neurona PixAck, que indica que la neurona ha recibido el pulso correctamente.

También existen variaciones del protocolo que permiten acelerar el ciclo de comunicación, por ejemplo, no esperar a la inactivación de la señal CRACK para enviar otro pulso.

2.5. Tipos

Como ya se ha mencionado, el píxel DVS ha sido el que ha predominado en la industria. Variaciones de este píxel son las que se comercializan hoy en día. En esta sección entraremos un poco más en profundidad en el diseño del píxel, así como el de alguna de sus variaciones.

2.5.1. Dynamic Vision Sensor (DVS)

En la Figura 2.6 (a) se puede apreciar una realización de píxel DVS. De forma análoga a lo representado en la Figura 2.3, se puede observar lo siguiente:

1. El circuito del fotorreceptor usa una configuración de transimpedancia para obtener un valor de tensión que varía logarítmicamente con la corriente del fotorreceptor. El fotodiodo PD está alimentado a través de la fuente del transistor M_{fb}, el cual, tiene su puerta conectada a la salida de un amplificador inversor formado por los transistores M_{pr}, M_{cas} y M_n, cuya entrada está conectada al fotodiodo. Esta configuración mantiene al fotodiodo sujeto a una tierra virtual, lo que provoca que se multiplique el ancho de banda por el factor de ganancia del bucle, siendo beneficioso para aplicaciones de alta velocidad, especialmente en condiciones de baja luminosidad. Además, la suma de las fotocorrientes se encuentra en el drenador del transistor M_{fb}, lo que permite adaptar los valores de alimentación.
2. A continuación, la salida del fotorreceptor V_P se conecta a un seguidor de tensión formado por los transistores M_{b2} y M_{sf} para aislar el fotorreceptor de transitorios rápidos en el circuito diferencial. La salida del seguidor de tensión V_{sf} se conecta a través del condensador C_1 a la entrada del circuito diferencial.
3. La entrada y la salida del circuito diferencial se encuentran realimentadas a través del condensador C_2, el cual puede ser cortocircuitado a través de la señal *reset*. Esto permite que la salida V_{diff} se resetee a un valor a V_{dd} menos

la tensión de drenador polarizada en inversa del transistor M_{dp}. Por lo tanto, el cambio desde V_{diff} hasta su valor de reseteo representa la amplificación de la señal de entrada V_{sf}. La ganancia de esta etapa viene dada por lo bueno que sea el *matching* de los condensadores ya que se ve afectada por la relación C_1/C_2.

4. Los comparadores formados por los transistores M_{ON_n}, M_{Onp}, M_{OFF_n} y M_{OFF_p} comparan la salida del amplificador diferencial ante valores mayores y menores diferenciados en un determinado *offset* desde la tensión de reseteo, generando eventos cuando se alcanzan estos valores.

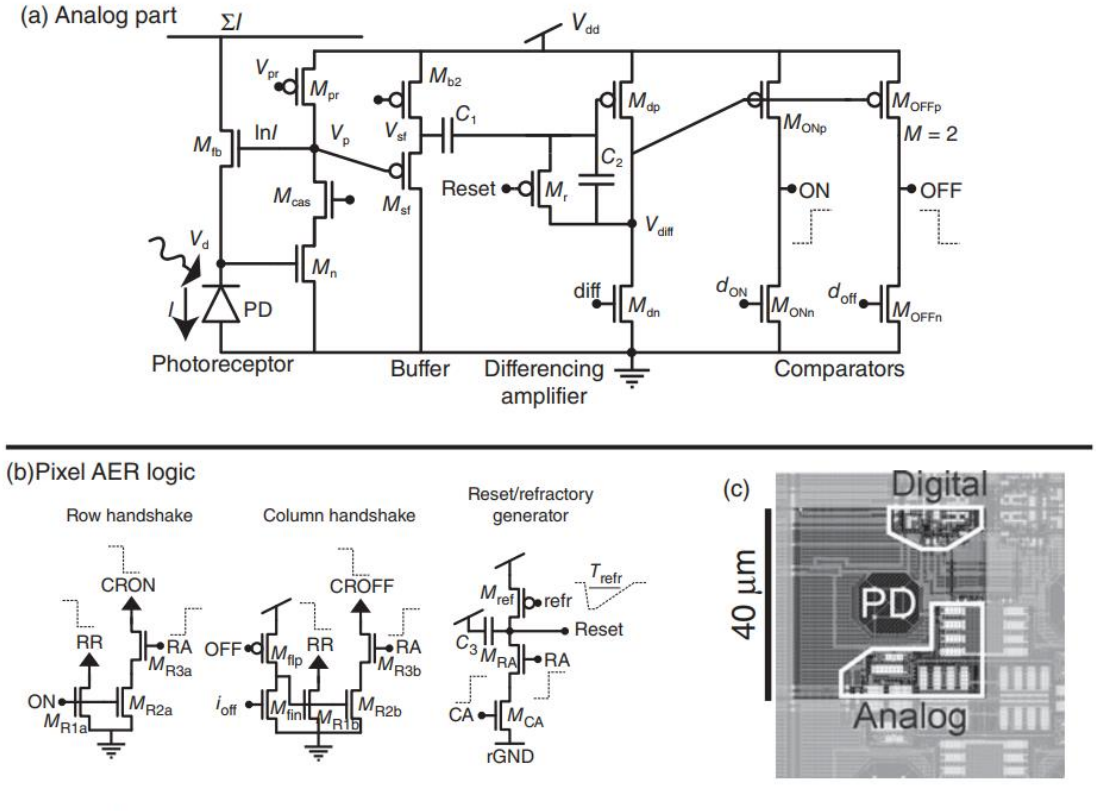

Figura 2.6: (a) Esquema de un píxel DVS, (b) circuitos AER y (c) *layout* del píxel [10].

La principal desventaja de este píxel es que el efecto del *mismatch* en el comparador es inevitable, aunque puede reducirse aumentando la precisión de la ganancia del circuito diferencial.

En la Figura 2.6 (b), se puede apreciar los circuitos AER que implementan la comunicación de los eventos. De forma similar a lo explicado en el apartado 2.4.2, cada píxel tiene tres señales de *request* de salida (RR, CRON y CROFF) y dos señales de *acknowledge* de entrada (RA y CA). El ciclo de comunicación comienza con la señal RR a nivel alto cuando alguno de los comparadores cambia de estado desde su condición de reset y finaliza con la activación de la señal Reset a través del transistor M_r.

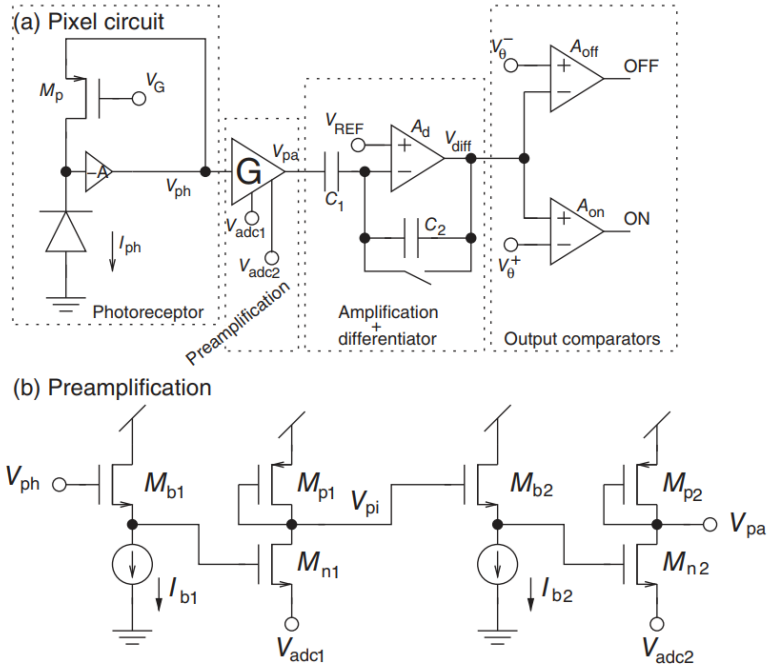

Figura 2.7. Píxel DVS con gran sensibilidad. (a) Esquema del píxel. (b) Esquemático de la etapa de preamplificación [25].

Existen modificaciones de este píxel que permiten aumentar la sensibilidad al contraste. En la Figura 2.7 se presenta una propuesta recogida en [25] donde se añade un amplificador extra a la salida del receptor, en el nodo V_{ph}. Esta implementación añade algunos transistores para implementar un amplificador de dos etapas. Los transistores de amplificación están alimentados en el límite entre inversión fuerte a inversión débil e introduce una ganancia de voltaje de en torno a 25, lo que permite reducir la capacidad de la etapa original de amplificación en torno a 1/5. Como resultado, el área del píxel se reduce en un 30%, se mejora la sensibilidad al contraste y el *mismatch* general se reduce en torno a la mitad, aunque el consumo de potencia se multiplica por un valor entre 5 y 10.

2.5.2. Asynchronous Time-Based Image Sensor (ATIS)

Como se comentó en el capítulo anterior, este sensor incorpora un sensor IMU a un sensor DVS para obtener una salida estática de brillo absoluto.

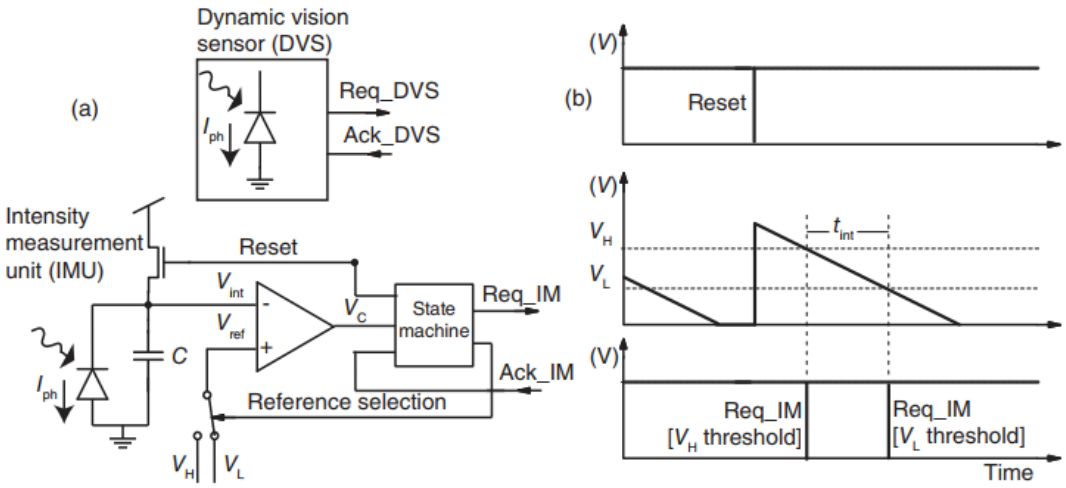

Figura 2.8. Píxel ATIS (a) y ejemplo de señales (b) [10].

En la Figura 2.8 (a) se aprecia un esquema del píxel ATIS, donde cabe destacar que la señal de resteo del píxel DVS activa la lectura del sensor IMU, el cual usa una señal modulada en pulso (PWM) para codificar el nivel absoluto de intensidad, como se observa en la señal Figura 2.8 (b).

Para obtener el nivel de brillo absoluto, la señal PWM se compara con dos umbrales de tensión V_H y V_L, generando un pulso cada vez que se cruzan y midiendo el tiempo transcurrido. Este tiempo se conoce como tiempo de integración t_{int} y es inversamente proporcional a la corriente generada por el fotodiodo. Tras cada ciclo, el fotodiodo se resetea y la fotocorriente se descarga.

Este enfoque permite un mayor rango dinámico y mejora la relación señal a ruido ya que el tiempo de integración para la medida del nivel absoluto de brillo se optimiza para cada píxel por separado en vez ser fijo para toda la matriz como en los sensores convencionales. Por lo tanto, el rango dinámico está limitado por la corriente de oscuridad (*dark current*) y el tiempo de integración disponible. La intensidad podrá ser obtenida siempre que haya más fotocorriente que corriente de oscuridad y el usuario pueda esperar el tiempo suficiente para cubrir todo el rango de tensión de integración. Además, cada píxel incluye una circuitería que permite implementar una máquina de estados para gestionar las señales de entrada y salida del sensor IMU. Por último, la lectura de los valores de los píxeles DVS y de cada sensor IMU se realiza a través de circuitos AER distintos alojados a los cuatro lados del sensor.

Sus principales ventajas son que el sensor DVS como el IMU logran un rango dinámico mayor que 120 dB y que ambos se basan en eventos, ya que la salida del sensor IMU se proporciona por cada lectura del sensor DVS.

2.5.3. Dynamic and Active Pixel Vision Sensor (DAVIS)

La Figura 2.9 muestra de forma simplificada el píxel DAVIS (*Dynamic and Active Pixel Vision Sensor*), que combina un píxel CMOS convencional APS (*Active Pixel*

Sensor) con un pixel DVS, con el objetivo de compartir el mismo circuito del fotodiodo.

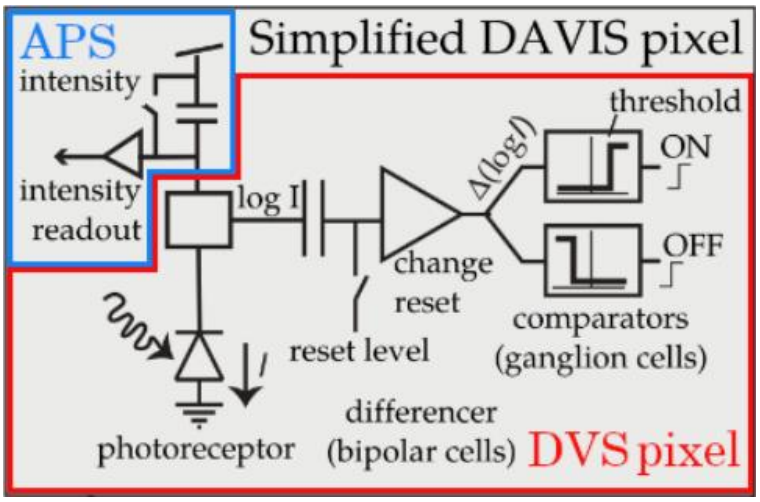

Figura 2.9. Esquemático del píxel DAVIS [16].

Las lecturas del píxel APS pueden tener una ratio fija o bajo demanda, tras un análisis de las lecturas del DVS, aunque esto implica un cierto tiempo de procesado.

La ventaja de este píxel sobre el ATIS es que es mucho más pequeño ya que el fotodiodo se comparte entre los dos píxeles y el circuito del píxel APS añade solo un 5% más de área respecto al píxel DVS. Además, la lectura de la intensidad absoluta puede omitirse si el usuario así lo prefiere. No obstante, esta lectura en este caso tiene un menor rango dinámico (55 dB) y es redundante si el brillo en el píxel no cambia.

3. IMPLEMENTACIÓN EN SISTEMAS COMERCIALES

Como se ha visto en capítulos anteriores, el campo de la visión basada en eventos comenzó en 1991 con la búsqueda de crear un nuevo paradigma en la computación de imágenes inspirado en arquitecturas biológicas. Se podría decir que las dos primeras décadas de la carrera particular de esta tecnología han sido más bien de pura investigación, no siendo hasta los inicios de la década de 2010 cuando este tipo de cámaras se empezó a comercializar y a tener un uso práctico en distintas aplicaciones. Esta comercialización coincide con el desarrollo del píxel DVS, ya que la forma de representar la escena de estos sensores es la que más potenciales aplicaciones ha tenido, por lo que es el cual han apostado los fabricantes.

En este capítulo se va a hacer un análisis del estado actual de los sensores de visión basados en eventos, centrándose en las variaciones del píxel DVS. También se va a hacer un análisis de las aplicaciones que tienen estos sensores, así como de los algoritmos que incorporan los sistemas de visión basados en eventos para procesar la información que les proporcionan.

3.1. Paradigma actual

3.1.1. Características actuales de los sensores

La Tabla 1.1 reúne las principales características de los sensores bio-inspirados comerciales más populares de los últimos años. Cabe destacar que estas características no están estandarizadas y puede que las condiciones para la medida de estos valores difieran respecto a un fabricante u otro. En cualquier caso, comparando esta tabla con los primeros sensores basados en eventos de la Tabla 1.1, se puede observar que ha existido una clara tendencia en aumentar la resolución, la velocidad de lectura y disminuir el tamaño del píxel. Además, algunos modelos presentan otras características como salida en escala de grises o sensores IMU que pueden ser muy útiles para distintas aplicaciones (Sección 3.3).

Tabla 3.1. Comparación de propiedades de distintas cámaras de eventos.

Vendedor	Prophesee		IniVation				Samsung
Modelo	Gen3.1 [26]	Gen4.1(1) [26]	DAVIS346 [27]	DVXplorer Lite [27]	DVXplorer [27]	DVXplorer Mini [27]	DVS Gen4 [28]
Año	2019	2021	2017	2019	2020	2021	2020
Resolución (pixels)	640x480	1280 x 720	346 x 260	640 x 480	640 x 480	640 x 480	1280 x 960
Rango dinámico (dB)	> 120	> 86	120	90 - 110	90 - 110	90 - 110	-
Tamaño del píxel (μm^2)	15	4.86	18.5	18	9	9	4.95
Tamaño del chip (mm)	13 x 15	6.22 × 3.5	-	-	-	-	8.37 x 7.64

Latencia (µs)	40-200	220 - 1000	< 1000	< 1000	< 1000	< 1000	
Máx. ancho de banda (Meps)	66	1060	12	100	165	450	1300
Fill Factor (%)	25	> 77	22	-	-	-	22
Consumo	26 – 176 mW DC	60 – 120 mW DC	<180 mA @ 5 VDC (USB)	<140 mA @ 5 VDC (USB)	<140 mA @ 5 VDC (USB)	<140 mA @ 5 VDC (USB)	150 mW
Salida en escala de grises	No	No	Si	No	No	No	No
Rango dinámico (escala de grises) (dB)	N/A	N/A	55dB	N/A	N/A	N/A	N/A
Tecnología CMOS (nm)	180 1P6M CIS	90 BI CIS	180 1P6M MIM	90 BI CIS	90 BI CIS	90 BI CIS	65 nm 1P6M BSI
Otros	-	-	Salida IMU de 6 ejes (giroscopio y acelerómetro) de hasta 8kHz de frecuencia de muestreo.				-

1. Desarrollado en colaboración con Sony

Además, el desarrollo de estos sensores ha llevado a compañías como IniVation, Samsung o Prophesee a ofrecer kits de evaluación y desarrollo, así como librerías software que permiten procesar los eventos de forma eficiente y extraer la información de estos.

3.1.2. Retos del paradigma actual

Las propiedades de los sensores de visión basados en eventos le dan una serie de ventajas respecto a las cámaras convencionales que pueden ser muy útiles en campos como la robótica o sistemas embebidos. No obstante, a pesar de sus ventajas y de que ya existen fabricantes que ofertan este tipo de cámaras en el mercado, todavía no están integradas en las industrias en las que se presupone que se pueden aprovechar de sus ventajas. El hecho de que no estén completamente integradas se debe a varios factores. Desde un punto de vista del procesamiento, se presentan los siguientes problemas:

1. **Extracción de la información.** La salida cámaras de eventos es binaria, ya que únicamente se indica si el brillo aumenta o disminuye. Por otro lado, los eventos no dependen sólo del brillo de la escena, sino también de movimientos pasados de la misma y el movimiento de la propia cámara. Esto implica que se necesita procesar la información de forma distinta a la tradicional, planteando la pregunta de cuál es el mejor método para extraer información de los eventos para una tarea concreta.
2. **Nuevos algoritmos.** En concordancia con lo anterior, ya que la salida espaciotemporal de las cámaras de eventos es, en esencia, diferente respecto a la de las cámaras convencionales, no es posible aplicar algoritmos utilizados con cámaras convencionales de forma directa, lo que abre multitud de caminos a explorar para el desarrollo de nuevos algoritmos o la necesidad de encontrar una mejor relación entre imágenes y eventos que permite aprovechar décadas de desarrollo sin perder las ventajas de los eventos.
3. **Ruido y efectos dinámicos.** Como en todos los sensores de visión, en las cámaras de eventos también existe el ruido *shot* debido a los fotones y el ruido de los transistores, así como no idealidades. No obstante, este

problema se acentúa en las cámaras de eventos, dado que el proceso de cuantización de los cambios de brillo es complejo y aún no ha sido completamente caracterizado. Por lo tanto, se necesita de un modelado de los efectos no ideales para extraer información correctamente de los eventos.

4. **Retraso provocado por el árbitro.** El árbitro en sistemas AER es el principal componente que retrasa la transmisión de eventos a la periferia del array [29]. Los protocolos AER no pueden gestionar de forma correcta todas las peticiones de los píxeles para enviar eventos si se dan muchos en un corto periodo de tiempo, lo que provoca que los píxeles no respondan en el orden correcto o no respondan. El desarrollo de procesadores dedicados a la precisión temporal y la diversidad de los eventos resolvería este problema.

Por otro lado, en términos de disponibilidad para producción en masa se refiere, existen varios obstáculos que los fabricantes deben solventar para que se impulse la producción de este tipo de cámaras:

- **Coste**. Este es el mayor obstáculo por solventar, ya que el coste de una de estas cámaras suele ser de varios miles de euros, por lo que no son muy accesibles por el momento. Este elevado coste se debe principalmente a los costes de ingeniería no recurrentes (investigación, desarrollo, diseño, prueba del prototipo, etc.).
- **Tamaño del píxel.** El coste de las cámaras se ve limitado por el tamaño del *die,* ya que el silicio cuesta entre 5 y 10 dólares por cm^2 en producción en masa [20] y el diseño de una nueva óptica miniaturizada de producción en masa para adaptarse a un formato de sensor diferente puede costar decenas de millones de dólares. No obstante, lo que ha tenido más prioridad en el desarrollo de estas cámaras ha sido incrementar la velocidad de lectura, resolución, añadir características como salida en escala de grises, etc., pero no se le ha dado prioridad a reducir el tamaño del píxel. Solo en los últimos modelos se puede ver una clara mejoría en la relación resolución/tamaño del píxel, logrando resoluciones de 1280 x 960 en 4.95 μm^2, lo cual es una reducción considerable teniendo en cuenta que estos píxeles son circuitos de señal mixta.
- **Fill Factor**. Se define como *fill factor* o factor de relleno a la relación entre el área sensible a la luz de un píxel y su área total. Aunque es un problema que se ha ido solucionando recientemente, las primeras cámaras de eventos tenían un *fill factor* en torno al 10-20%, lo que reducía drásticamente el rendimiento.

Por otra parte, se está trabajando en conseguir otras características para las cámaras de eventos que también pueden ser muy interesantes, como obtener una salida en color o mayor sensibilidad al contraste.

3.2. Representación de eventos

Los sistemas que procesan eventos constan de varias etapas: el preprocesado, que puede considerarse como una adaptación inicial de la entrada, el procesamiento

principal, donde se analizan los eventos y se extraen las distintas características, y el postprocesado, donde se crea la salida y se actúa donde el sistema determine, si es necesario. Dado que el preprocesado puede entenderse como una forma de representar los eventos, en este apartado se verán las distintas formas de representarlos.

Para muchos de los ejemplos mostrados en este apartado, se han desarrollado librerías en C++ y en MATLAB, que se pueden encontrar en [30]. También se pueden ejecutar las reproducciones de los ejemplos y ver la secuencia completa.

3.2.1. Representación espaciotemporal

A nivel de software, los eventos vienen caracterizados de forma:

$$e_k = \{t_k, p_k, \{x_k, y_k\}\} \tag{3.1}$$

Siendo k el índice del evento, t_k representa la marca de tiempo (*timestamp*) del evento, p_k es la polaridad del mismo (si es positivo o negativo) y el par $\{x_k, y_k\}$ representa las coordenadas del píxel.

De acuerdo con la Ecuación (3.1), los eventos se pueden representar directamente como se muestra en la Figura 3.1. En ella, se muestran los eventos espaciotemporales recogidos en la grabación de un número 7 parpadeando en la pantalla.

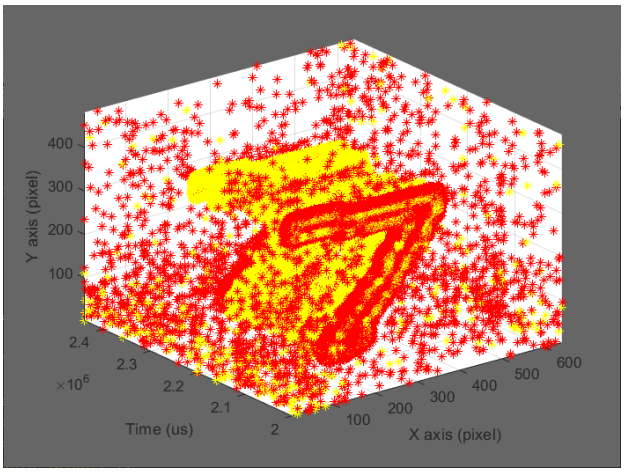

Figura 3.1. Representación de un número 7 intermitente en MATLAB. Los eventos negativos se representan en rojo y en amarillo los positivos.

Un evento aislado contiene poca información, por lo que este tipo de representación se usa para algoritmos que procesan la información evento a evento. Un ejemplo serían filtros tales como los probabilísticos (filtro de Kalman) [31] o redes neuronales de impulso [19] (*Spiking Neural Networks, SNNs*). Estos filtros o redes poseen información adicional provenientes de eventos pasados o de sensores adicionales, que se actualizan de forma asíncrona con el nuevo evento para producir una salida. Este método de representación respeta la naturalidad de los eventos. Principalmente, la disparidad de los eventos y baja latencia, pero son muy sensibles a los parámetros de ajuste (por ejemplo, pesos del filtro) y consumen muchos

recursos computacionales, dado que el sistema se debe actualizar con la llegada de un evento.

También se pueden utilizar para algoritmos que procesan eventos en paquetes [32], por ejemplo, paquetes de eventos producidos en una determinada localización. Cuando se usa este segundo método, se pierde latencia a favor de eficiencia computacional.

3.2.2. Imagen de eventos

De forma similar a los sensores de visión basados en fotogramas, estas imágenes se forman acumulando eventos durante un periodo T_1 determinado. Al final de dicho periodo, en una imagen 2D se muestra la localización y la polaridad del último evento de cada píxel. Esta imagen 2D se puede representar como una matriz M de tres dimensiones de acuerdo con la ecuación (3.2).

$$M\left(x, y, \frac{t}{\Delta}\right) = p\left(x_k, y_k, \frac{t}{\Delta}\right) \tag{3.2}$$

Para $t = k\Delta$ siendo Δ el periodo escogido.

Un ejemplo se muestra en la Figura 3.2. En ella, se puede apreciar un medallón representado como imagen de eventos, durante un periodo de 40 ms, es decir, a 25 *fps*.

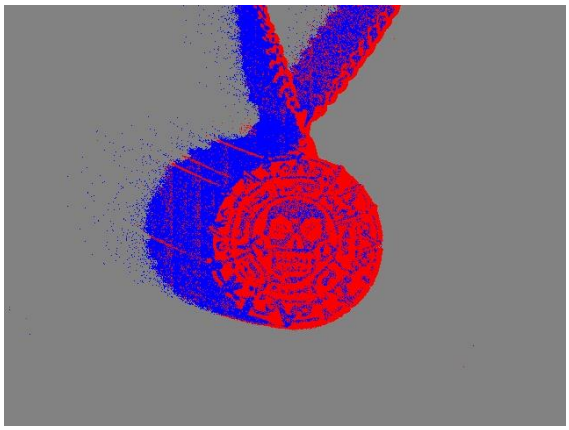

Figura 3.2. Medallón representado como imagen de eventos. La imagen es una de las tramas generadas a 25 *fps* a través de una de las grabaciones recogidas en [30]. Los eventos positivos se representan en azul, mientras que los negativos se representan en rojo.

Una forma de hacerlo más eficiente en términos computacionales es definiendo un segundo periodo $T_2 < T_1$ durante el cual los eventos se ignorarán, de forma que se solo los últimos eventos contenidos en el intervalo $T_1 - T_2$ serán los que se muestren.

Este tipo de representación tiene las siguientes ventajas:

- Es una representación con la que se está familiarizado y se puede entender de forma intuitiva.
- Se destaca en gran medida los bordes de la escena, que son las regiones con más información

- Informa tanto de la existencia de eventos como de su inexistencia.
- Es compatible con algoritmos de visión convencionales.

No obstante, también presenta las siguientes desventajas:

- Esta práctica no es deseable ya que se cuantiza la temporalidad de los eventos, lo que limitaría algunas de las ventajas que tienen este tipo de cámaras.
- Se descarta información relativa a la polaridad de los eventos, ya que el mostrar el último evento descarta si ha habido o no eventos anteriores y de qué tipo.
- La imagen representada tiene una gran dependencia de los periodos usados.

Un ejemplo de uso de este tipo de representación para aplicar algoritmos tradicionales se da en [33], donde se aplican técnicas de compensación de movimiento a las imágenes de eventos para usar algoritmos de odometría visual e inercial basado en fotogramas clave y optimización no lineal para estimar los 6 grados de libertad de la cámara.

3.2.3. Histogramas

Están compuestos por dos imágenes 2D, una para cada polaridad, ya que se basa en almacenar el número de eventos del mismo tipo que ha ocurrido en un píxel en un tiempo determinado. Por lo tanto, cada píxel se representará un valor de intensidad, donde los valores más altos corresponderán a mayor actividad. En este caso, cada conjunto de imágenes se puede definir como una matriz M de cuatro dimensiones que se actualiza con la llegada de un evento e_k y se representa con una periodicidad Δ de acuerdo con la ecuación (3.3)

$$M_j\left(x, y, \frac{t}{\Delta}, p\right) = M_j\left(x_k, y_k, \frac{t}{\Delta}, p_k\right) + 1 \tag{3.3}$$

Para $t = j\Delta$ siendo Δ el periodo escogido.

Un ejemplo se muestra en la Figura 3.3, donde se puede observar que el mayor número de eventos positivos se dio en la parte izquierda, mientras que el mayor número de eventos negativos se dio a la derecha.

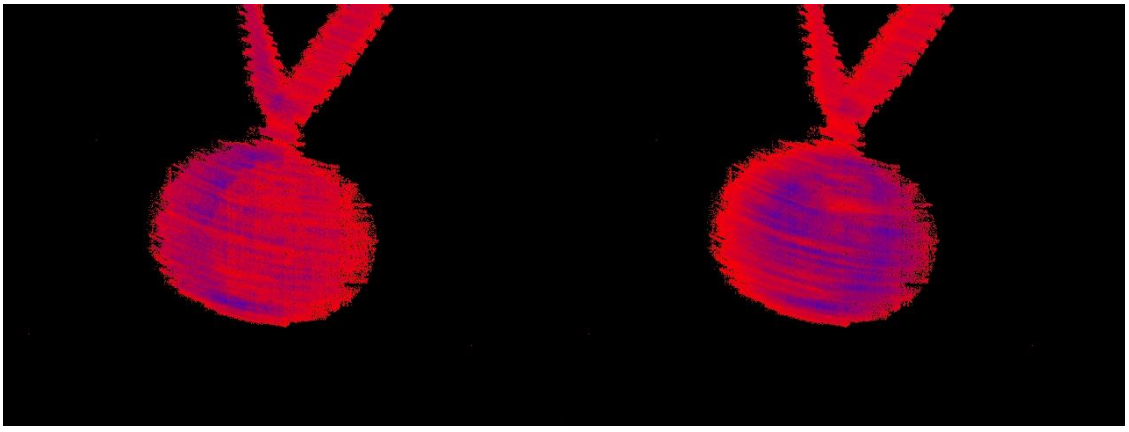

Figura 3.3. Histograma de eventos positivos (izquierda) y de eventos negativos (derecha). Cuanto más azul es el píxel, mayor es el número de eventos almacenados.

Hay ocasiones en las que existe una cantidad importante de actividad en una escena y puede ocurrir que un mismo píxel reciba varios eventos positivos y negativos. Considerando que un evento positivo y uno negativo en la misma celda se anulan mutuamente, se puede obtener el histograma de la diferencia representando la suma total de eventos en cada píxel para un periodo Δ como se expresa en la ecuación (3.4).

$$D_j\left(x, y, \frac{t}{\Delta}\right) = M_j\left(x_k, y_k, \frac{t}{\Delta}, 1\right) - M_j\left(x_k, y_k \frac{t}{\Delta}, 0\right) \tag{3.4}$$

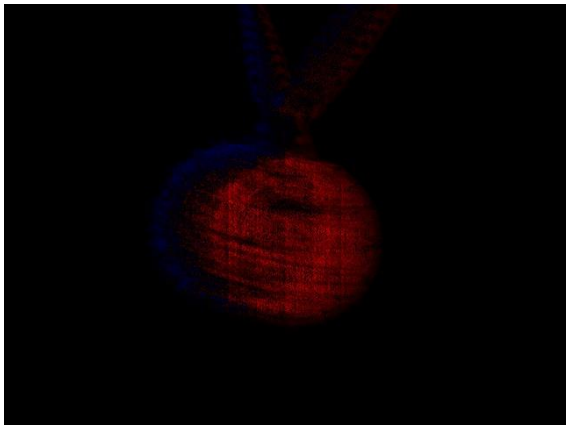

Figura 3.4. Histograma de la diferencia. La suma total de eventos se representa en azul, si es positiva, y en rojo, si es negativa.

La Figura 3.4 muestra un ejemplo de ello, donde se observa, en concordancia con la Figura 3.3, que el mayor número de eventos positivos se da en la parte izquierda, mientras que el de eventos negativos se da a la derecha. También se observa como los eventos se anulan en la zona media, ya que el movimiento del objeto provoca que se den incrementos y decrementos de brillo en los mismos píxeles.

Aunque no se pierde información relativa a la polaridad de los eventos, esta práctica también cuantiza la temporalidad de los eventos y se pierde la asincronía característica de los mismos.

En [34], se usa este tipo de representación como entrada de una CNN para predecir el ángulo de giro del volante de un vehículo.

3.2.4. Superficies temporales

Al igual que los histogramas, las superficies temporales están formadas por dos imágenes 2D donde en cada píxel se almacena un valor del tiempo, por ejemplo, el tiempo del último evento. Cada píxel representará un valor de intensidad y los valores más altos corresponderán a la actividad más reciente. Las superficies temporales se actualizan de forma asíncrona y, si se requiere, se pueden reiniciar de forma síncrona, es decir, descartando los eventos que se produjeron un tiempo determinado atrás. No obstante, para poder representarlas se debe definir un periodo Δ durante el cual se acumulará las temporalidades de los eventos a mostrar.

Cada conjunto de imágenes estará definido por una matriz T de 3 dimensiones. Esta se actualizará con la llegada de un evento e_k conforme a la siguiente ecuación:

$$T_j(x, y, p) = t(p_k) - j\Delta \tag{3.5}$$

Un ejemplo se muestra en la Figura 3.5, de la que se puede extraer que la pieza se estaba moviendo de izquierda a derecha.

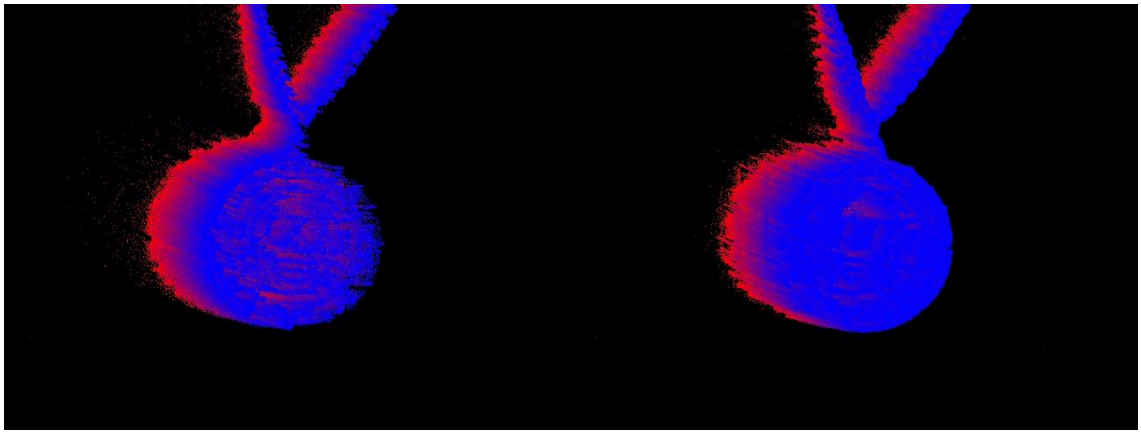

Figura 3.5. Superficie temporal de eventos positivos (izquierda) y de eventos negativos (derecha). Los eventos más recientes se representan en azul, mientras que los más lejanos en el tiempo se representan en rojo.

De forma similar a los histogramas, es posible que la polaridad de los eventos no sea necesaria y solo sea necesario conocer si ha habido actividad o no, por lo que se pueden formar superficies temporales absolutas de un solo canal, de acuerdo con la siguiente expresión:

$$T_j(x, y) = t_k - j\Delta \tag{3.6}$$

Un ejemplo se muestra en la Figura 3.6 donde, de forma análoga a la figura anterior, se puede apreciar que el medallón se estaba desplazando de izquierda a derecha.

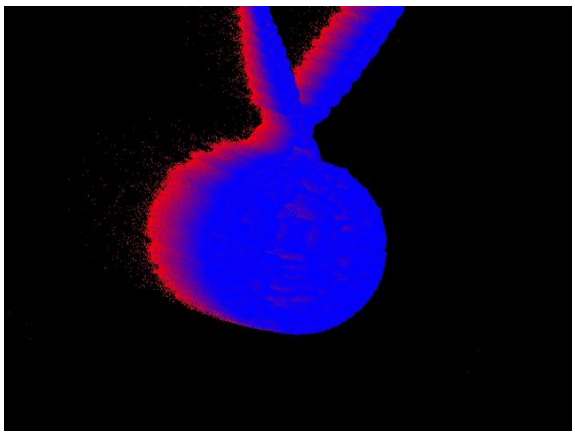

Figura 3.6. Superficie temporal absoluta.

Cabe destacar que superficies temporales no solo se pueden resetear de forma lineal como en la ecuación (3.6), también pueden resetearse de forma cuadrática, cúbica, exponencial, etc.

Las superficies temporales tienen la ventaja de que se comprime mucho la información, ya que solo se almacena un valor de tiempo por cada píxel, y que son

asíncronas, pues se actualizan cada vez que llega un nuevo evento. No obstante, estas dos ventajas son, a su vez, su principal desventaja, ya que es por ellas por lo que pierden eficacia en escenas donde los píxeles se actualicen con mucha frecuencia.

Un ejemplo del uso de esta forma de representación se presenta en [35], donde las superficies temporales se utilizan como única entrada de una red neuronal de aprendizaje auto supervisado para la obtención del flujo óptico (Sección 3.3.2).

3.2.5. Otras

Existen otras formas de representación de eventos también muy comunes que se deben mencionar, por ejemplo:

- **Cuadrícula de vóxeles**. Es una representación espaciotemporal donde cada vóxel representa un píxel, un intervalo de tiempo, y la suma total de eventos acumulados en ese tiempo, aunque también se pueden representar las dos polaridades por separado. Este tipo de representación preserva mejor la información temporal que las superficies temporales ya que no se comprime en una representación 2D. En [36], se usa este tipo de representación en dos canales para alimentar una Red Generativa Antagónica (*GAN, Generative Adversarial Network*) para construir imágenes de gran rango dinámico, usando también el píxel APS de una cámara DAVIS (Sección 2.5.3)
- **Imágenes compensadas**. Son imágenes que no solo dependen de los eventos sino también de hipótesis sobre el movimiento de la escena. Se basan en que, cuando existe movimiento en la escena, se desencadenan eventos en los píxeles que captan ese movimiento. Si se estima este movimiento (Sección 3.3.2) y se asignan un tiempo de referencia inicial, se puede producir una imagen nítida de la escena. En [37], se usa una cámara convencional para extraer características de la imagen que luego son perseguidas a través de las imágenes compensadas obtenidas de la cámara de eventos.
- **Imágenes reconstruidas**. Son imágenes de brillo que se reconstruyen a través de los eventos y sus temporalidades y se pueden interpretar como imágenes independientes del movimiento. Puede ser usada para aplicaciones como reconocimiento de líneas [38].

3.3. Algoritmos y aplicaciones

Existen diversas aplicaciones en las que el uso de que la visión basada en eventos proporciona ventajas significativas respecto a las cámaras convencionales. Aunque algunas están más desarrolladas que otras, todas plantean diversos retos que aún están por resolver.

3.3.1. Detección de características y seguimiento.

Esta es una de las aplicaciones fundamentales para tareas de visión como segmentación de movimiento (Sección 3.3.7) u odometría visual (Sección 3.3.5). Las

características de las cámaras basadas en eventos permiten monitorear la escena de forma asíncrona y con poca latencia, lo que permite obtener información de la escena cuando las cámaras convencionales no pueden, es decir, en el tiempo entre tramas.

Principalmente, existen dos problemas que se plantean con esta aplicación:

1. La variabilidad en la apariencia de la escena que provoca los eventos. Por ejemplo, un mismo objeto provocará eventos distintos si se mueve en dirección horizontal y vertical. Esto es un problema porque para seguir objetos se deben establecer una relación entre eventos en diferentes momentos, lo que es más complicado si la escena presenta mayor variabilidad.
2. El ruido y las no-linealidades provocadas por el propio sensor, ya que este tipo de sensores suelen ser bastante ruidosos.

Los primeros métodos para esta aplicación eran muy simples y se basaban en demostrar la baja latencia y los pocos requisitos de procesamiento de los sistemas de visión basados en eventos, asumiendo un escenario con cámara estática y siguiendo objetos conocidos como círculos [39] y líneas [40].

No obstante, ya existen métodos que permiten distinguir los objetos que se mueven en la escena, como el propuesto en [41], donde construyen grafos del vecino más cercano (k-NN) a partir de representaciones espaciotemporales para posteriormente detectar agrupaciones espaciales y diferenciar los distintos objetos. También se pueden encontrar trabajos que adaptan métodos basados en computación visual tradicional para detectar características como esquinas [42]. Por último, es común combinar eventos con imágenes tradicionales para mejorar el rendimiento de la aplicación [43].

3.3.2. Estimación del flujo óptico

El flujo óptico puede definirse como el movimiento aparente de los patrones de intensidad luminosa dentro de un fotograma, suponiendo que las variaciones de luminancia dentro del *frame* se deben únicamente al desplazamiento de dichos patrones. Se trata de un movimiento relativo entre los objetos de la escena visual y un observador [44][45]. El cálculo de este permite obtener la velocidad de objetos en el campo visual, sin conocimiento previo de la geometría de la escena o del movimiento de la cámara.

Con cámaras convencionales, el flujo óptico se puede obtener con tres tipos de algoritmos [46]:

1. Técnicas diferenciales. Basadas en el procesado de derivadas espaciotemporales de la intensidad de la imagen.
2. Métodos basados en frecuencia y fase. Se fundamentan en aplicar filtros de velocidad tuneados a las imágenes.
3. Coincidencia basada en la región. Las que se basan en la búsqueda del vector de velocidad que minimiza el error entre dos fotogramas sucesivos

Sin embargo, los eventos se encuentran con los siguientes problemas a la hora de intentar aplicar estas técnicas:

1. No proporcionan ni nivel de brillo ni datos continuos en el espacio.
2. Un evento no proporciona suficiente información.
3. Calcular el flujo en todo el plano de la imagen no es posible, por lo que se calcula en puntos específicos, como la localización de los eventos o con imágenes formadas por ellos.

Los primeros métodos [47] trataban de adaptar enfoques de computación visual convencional, pero para calcular las derivadas espaciales con los eventos se deben hacer demasiadas suposiciones que daban lugar a estimaciones inconclusas. Se ha trabajado también en métodos que tienen en cuenta la distribución local de los eventos [48] o métodos biológicamente inspirados [49], entre otros. No obstante, los últimos trabajos utilizan alguno tipo de red neuronal, como SNNs [50] o CNNs [51], pero también emplean el uso de nuevas formas de representar los eventos, como superficies de distancia exponenciales inversas [52].

3.3.3. Reconstrucción 3D.

La reconstrucción 3D implica estimar la profundidad de la escena, como se muestra en la Figura 3.7. Como se verá a continuación, es un problema complejo que requiere considerar el escenario en cuestión, la configuración y el movimiento de la cámara. En función de estos parámetros, se puede abordar el problema de las siguientes formas:

Figura 3.7. Ejemplo del cálculo de la profundida de la escena [20].

1. **Estéreo e instantáneo**. Hace referencia al cálculo de la profundidad de la escena usando dos o más cámaras de eventos completamente fijas en un periodo corto de tiempo. Los últimos trabajos emplean el uso de cámaras tradicionales con cámaras de eventos que no están integradas en el mismo píxel y así evitar los problemas de ruido que se generan al combinar píxeles APS y DVS [53]. También se han publicado métodos basados en una CNN para obtener la profundidad a través de los eventos e imágenes reconstruidas [54] o que usan métodos de aprendizaje profundo (*deep learning*) [55].
2. **Panorama multi-perspectiva**. Cuando el problema anterior se aborda usando dos cámaras de eventos no simultáneas, es decir, las cámaras rotan con un movimiento conocido para recuperar la intensidad absoluta de la imagen

[56] o hacer coincidir eventos usando métricas temporales [57], lo que implica una serie de limitaciones en la parte hardware del sistema.

3. **Estimación de profundidad monocular**. Hace referencia a cuando el problema se aborda con una sola cámara. Esto provoca que hace que el problema sea muy distinto al anterior ya que no se puede establecer ningún tipo de correlación entre múltiples planos. Es importante aclarar que la cámara que se use puede incluir un píxel APS y DVS. Los últimos trabajos emplean el uso de redes neuronales recurrentes (RNNs, *Recurrent Neural Networks*) adaptadas para trabajar con datos asíncronos y de múltiples sensores [58]. También se han publicado trabajos que permiten obtener la profundidad de todos los píxeles usando una sola cámara a través de redes CNNs entrenadas [59].
4. **Estimación del flujo usando luz estructurada**. Su diferencia con los anteriores es que se interfiere en la escena emitiendo luz y midiendo su reflexión con las cámaras de eventos [57][60].

3.3.4. SLAM

El objetivo del SLAM (*Simultaneous Localization and Mapping*) es que un robot o vehículo pueda construir un mapa de navegación de su entorno y, a su vez, localizarse en el mismo para navegar por él. Por lo tanto, es una aplicación que abarca otras aplicaciones, como detección de características o estimación de profundidad. Dado que los métodos para imágenes convencionales no son directamente aplicables a estas aplicaciones, el objetivo es diseñar nuevas técnicas que hagan uso de las ventajas de las cámaras basadas en eventos.

El problema de la localización se abarcó primero, ya que este presenta menos grados de libertad que el problema del mapeo. Aun así, las investigaciones han partido desde lo más simple a lo más complejo. Por ejemplo, el primer trabajo [61] que abordó este problema consistía en fijar una cámara que veía una escena plana con patrones blancos o negros, la cual era paralela a su plano de movimiento. Tras esto, la estimación de la orientación 3D de la cámara se abarcó en trabajos como [62], que introdujo técnicas de compensación de movimiento para estimar la velocidad angular de la cámara. Finalmente, trabajos como [63] abordaron el seguimiento de la cámara optimizando no-linealmente el error fotométrico entre imágenes reconstruidas, usando predicciones dadas por un mapa de la escena. Un trabajo más reciente [64] hace uso de la teoría de Lie aplicada a filtros de Kalman para seguir tanto el movimiento de la cámara como el de un objeto en frente de ella.

Si se suma el problema del mapeo, los primeros trabajos desarrollaron sistemas SLAM en dos dimensiones, como el propuesto en [65], que fue una extensión de [61]. Al ser en dos dimensiones, los movimientos estaban restringidos a un solo plano, además de que las escenas presentaban un gran contraste. El mismo caso en tres dimensiones se expone en [66] usando una cámara RGB para estimar la profundidad. El movimiento rotacional se consigue a través de filtros probabilísticos en trabajos como [67][68]. El problema en 6 dimensiones se aborda en [69] como una extensión de [67], usando filtros probabilísticos intercalados en el seguimiento

de la cámara, estimando la profundidad y reconstruyendo imágenes. El problema que presenta este trabajo es que es computacionalmente costoso y necesita una tarjeta gráfica para operar en tiempo real. No obstante, la propuesta de [70] muestra que no es necesaria la reconstrucción de imágenes para estimar la profundidad y el seguimiento de la cámara y no necesita de tarjeta gráfica para operar en tiempo real. Por último, cabe mencionar que algunos trabajos más recientes emplean el uso de sensores IMU para mejorar el rendimiento y la robustez de los sistemas SLAM [71].

3.3.5. Odometría visual inercial

La odometría visual inercial (VIO, *Visual Inertial Odometry*) hace referencia al problema de estimar la posición y orientación de un objeto. El sensor IMU que poseen algunas cámaras (Tabla 3.1) es muy útil para este problema.

Los primeros trabajos buscaron adaptar métodos para cámaras convencionales convirtiendo los eventos en información geométrica antes de ser procesados junto a los datos provenientes del sensor IMU. Estos trabajos se dividen en:

1. **Basados en características**. Constan de dos etapas, primero se extraen las características buscadas de los eventos (Sección 3.3.1) y posteriormente los puntos estimados de estas se fusionan con datos del sensor IMU usando distintos algoritmos [33][72].
2. **Basados en la reproyección de errores**. En este caso se fusionan los eventos y los datos provenientes del sensor IMU en un marco de trabajo de tiempo continuo, con el objetivo de optimizar el procesamiento [73].

No obstante, los últimos trabajos no han usado el sensor IMU, lo que se conoce como *stereo VIO*. En [74] se propone acumular eventos y transformarlos en imágenes para aplicar algoritmos tradicionales. Esta propuesta ha ofrecido buenos resultados, aunque limita alguna de las ventajas que ofrecen las cámaras basadas en eventos frente a los sensores tradicionales (Sección 3.2.2). En [75] se usan dos cámaras DVS para determinar el movimiento de la cámara, además de un mapa semidenso (que no abarca todo su conjunto) 3D de la escena.

3.3.6. Reconstrucción de imagen

Dado que los eventos representan cambios de brillo, en un escenario ideal sin ruido y con una respuesta óptima del sensor, la integración de los eventos en un espacio de tiempo determinado permitiría obtener intensidad de brillo absoluto. Es más, debido a la alta resolución temporal de los eventos, las imágenes pueden ser reconstruidas a una velocidad bastante alta o incluso en tiempo continuo [76].

El paradigma actual hace uso de redes neuronales para esta aplicación. En [77] se hace uso de una red neuronal de dos fases, compuesta por una CNN optimizada para predicción de Laplace y una SNN optimizada para integración de Poisson. De forma similar, en [78] se hace uso de dos redes neuronales de aprendizaje autosupervisado (SLL, *self-supervised learning*), una para estimación de flujo y otra para la reconstrucción de imagen, la cual utiliza la relación flujo-intensidad para

reconstruir las imágenes que mejor satisfacen los eventos de entrada y el flujo estimado. Otros trabajos que siguen la misma línea son [79] y [80].

3.3.7. Segmentación del movimiento

La segmentación del movimiento es un problema que se plantea cuando existe la presencia de eventos desordenados que pueden tener origen en el movimiento de objetos o movimiento de la cámara. Por lo tanto, para aplicaciones como reconocimiento de objetos, es necesario separar o segmentar los eventos causados por el objeto a reconocer, el resto de los objetos y el movimiento de la cámara.

Aunque la segmentación de movimiento es un tema de mucha importancia en la visión basada en fotogramas, está bastante inexplorada en la visión basada en eventos [20]. No obstante, se han desarrollado diversas técnicas para atajar el problema. En [81], propone un método basado en esquemas multi-modelos de ajuste, donde se busca optimizar una determinada función de energía, y en una forma de representación de los eventos usando la triangulación de Delaunay. Por otro lado, en [82] se busca separar los grupos de eventos de los distintos movimientos de la escena para después unificarlos basándose en medidas que dependen del contraste y la distancia entre el grupo y el centroide de los grupos. También se han presentado trabajos basados en el uso de CNNs, como en [83], donde se diseña una red neuronal convolucional supervisada que utiliza como entrada representaciones espaciotemporales con un largo periodo de tiempo.

3.3.8. Reconocimiento

El problema de reconocimiento es muy amplio y puede llegar a ser muy complejo, pues depende del objeto a reconocer. Los algoritmos dedicados a este tipo de problema han ido creciendo en complejidad, partiendo desde la posibilidad de reconocer formas simples conocidas a través del uso de modelos, como manchas [84], círculos [85] o líneas [40], hasta obtener pequeñas características bajo aprendizaje no supervisado, agrupando los eventos en pequeños grupos que se asocian en función de su cercanía [38].

Para objetos sencillos, también se puede hacer uso de plantillas de coincidencia (*template matching*), lo que permite no describir la geometría del objeto [85]. Estas plantillas también pueden usarse para objetos más complejos, buscando las coincidencias en características de más bajo nivel que el objeto entero en sí, para que posteriormente un clasificador como el *Nearest Neighbor* que tome la decisión si el objeto es el que se busca o no [38]. Este enfoque se puede implementar de forma jerárquica usando SNNs, de forma que cada capa detecte progresivamente características más complejas. También se puede emplear el uso de técnicas de aprendizaje profundo como retro propagación, con la ventaja de que no se necesita un clasificador a la salida, pero sí muchísimos más datos etiquetados para el aprendizaje.

La mayoría de los enfoques basados en el aprendizaje transforman eventos en una forma de representación para modelos jerárquicos basados en imágenes (ANNs),

como tensores [86] o superficies temporales [87]. El problema de estos enfoques, similar a casos anteriores, es que limitan la característica asíncrona de los eventos y cuantifican su temporalidad.

Los últimos métodos hacen uso de representaciones gráficas compactas que luego se emplean en CNNs [88], pero aun así deben integrar un cierto número de eventos o durante un determinado periodo de tiempo. En [89] se busca resolver este problema utilizando convoluciones de grafos incrementales que se actualizan evento a evento.

3.3.9. Control neuromórfico

Este tipo de algoritmos se desarrolla para implementar controles basados en la neuro morfología, dado que estos sensores emulan la respuesta biológica neuro mórfica basada en pulsos.

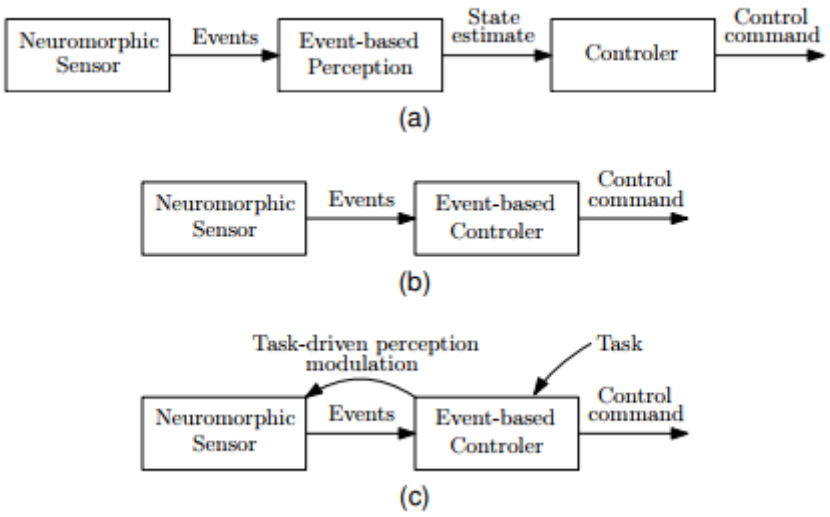

Figura 3.8. Distintos tipos de arquitectura de controladores neuro mórficos [20].

En la Figura 3.8 (a) se puede apreciar una arquitectura de control basada en la visión neuro mórfica, donde se tiene un sensor neuromórfico que recibe las entradas y produce los eventos, un sistema de percepción basado en eventos que estima el estado del sistema, y un controlador tradicional que se llama de forma asíncrona para producir la señal salida. En (b) se puede apreciar una arquitectura más nativa, donde el controlador está basado en eventos, por lo que son los propios eventos los que se usan como entradas del controlador para generar la salida. Por último, en (c) se muestra una arquitectura donde se recibe información extra de la tarea que se está ejecutando y se retroalimenta para ajustar la señal de salida.

Los primeros trabajos basados en estos controladores fueron el proyecto CAVIAR [85] y el Robot Goalie [84]. Los trabajos más recientes son [90], donde se presenta un robot para sistemas de perforación con precisión submilimétrica, y [91], que utiliza algoritmos de visión implementados como una SNN y los usa para controlar un dron.

3.4. Métodos de procesamiento de eventos

En función de cuántos eventos son procesados simultáneamente, se pueden distinguir dos categorías: métodos que operan en función de cada evento (*event-by-event basis*), donde el estado del sistema puede cambiar con la llegada de un solo evento y aquellos que operan con grupos de eventos (*group of events*).

El uso de filtros y ANNs (*Artificial Neural Networks*) está más ligado al procesamiento evento a evento, mientras que las DNNs (*Deep Neural Networks*) se asocian más a procesamiento con grupos de eventos.

También existen modelos que implementan una arquitectura inspirada en la visión biológica a través del uso de neuronas artificiales.

3.4.1. Event-by-event

Los métodos de procesamiento evento a evento se usan de la mano de filtros, que pueden ser determinísticos o probabilísticos, y ANNs.

Los filtros determinísticos como convoluciones se han utilizado para reducción de ruido [31] o reconocimiento de gestos [92]. Por otro lado, los filtros probabilísticos están más enfocados para distintos tipos de reconstrucción de imágenes [93] o en sistemas SLAM (*Simultáneos Location And Mapping*) [33][94][95] aunque estos últimos tienen una fuerte dependencia con información adicional como la imagen en escala de grises, que puede reconstruirse con eventos pasados o sensores IMU. El uso de filtros es muy común debido a que, de forma natural, procesan los datos de forma asíncrona, lo que preserva las características de los sensores, y agregan información de fuentes de información pequeñas, como los eventos [20].

No obstante, la tendencia de los últimos años es usar arquitecturas neuronales multicapa que contienen parámetros que se obtienen de los eventos. Esto también se está empleando en sistemas SLAM [96], pero también para otras aplicaciones como estimación de flujo [97], reconstrucción de imágenes [77], reconstrucción 3D [58] o extracción de características [98][99].

3.4.2. Group of events

Estos métodos buscan procesar varios eventos para alcanzar una suficiente relación señal a ruido y dar solución a un problema dado. Suelen usar los eventos para construir distintas formas de representación (Sección 3.2.1) y extraer la información necesaria para resolver el problema. Por lo tanto, los eventos se procesan de forma diferente dependiendo de su representación. Por ejemplo, algunos trabajos recientes han usado las imágenes de eventos (Sección 3.2.2) como entrada de métodos de aprendizaje basado en imágenes como DNNs [100] o SVM (*Support Vector Machine*) [101]. Otro ejemplo es el uso de superficies temporales como parte de extractores de características jerárquicos [38][87] y para detectores de esquinas [102]. Un sistema completo que utiliza este método de procesamiento se muestra en [103], donde se propone un sistema para la detección de eventos anormales denominado NeuroAED. En este sistema, se utilizan representaciones

espaciotemporales para extraer el flujo óptico en un periodo determinado. Posteriormente, esa porción de la representación espaciotemporal se divide en lo que denominan "cubos espaciotemporales", descartando aquellos que no tienen información del flujo óptico o la densidad de eventos no supera cierto umbral. Estos cubos espaciotemporales se usan como entradas de un descriptor visual basado en eventos, cuyas características se usan como entrada de un modelo de representación esparcida (SR, *Sparse Representation*) para detectar si se ha dado un evento anormal o no.

El uso de DNNs con grupos de eventos también se ha estado usando para otras aplicaciones como predicción del ángulo de dirección [104][100] o estimación del flujo óptico [105][106], entre otras. Estos métodos se diferencian principalmente en la representación de los eventos y en las funciones de pérdida optimizada durante el entrenamiento.

3.4.3. Procesamiento Visual Biológicamente Inspirado

Dado que este tipo de sensores fueron diseñados inspirándose en la función biológica de la visión, se han desarrollado algunos modelos de procesamiento de eventos que imitan la forma en que la visión procesa toda la información que llega desde la retina. Para ello, se usan neuronas artificiales como *Leaky-Integrate and Fire* o *Adaptive Exponential*, que son bloques básicos de una red neuronal SNN y están inspirados en la corteza visual de los mamíferos. Brevemente, se podría decir que estas neuronas reciben eventos como impulsos de entrada de alguna de las regiones del campo de visión y modifican el estado interno del sistema, ofreciendo una salida distinta a la anterior. Algunos modelos se usan para reconstrucción 3D estéreo [107][108] o control de la vergencia binocular de robots [109].

3.5. Ejemplos

En esta sección se van a explicar algunos de los sistemas comerciales que utilizan este tipo de sensores y así tener una visión de real de cuáles son sus posibilidades.

3.5.1. Automoción

Hoy en día los sistemas basados en visión artificial están basados en tecnología basada en *frames*, adquiriendo imágenes de toda la escena con una frecuencia determinada. Este tipo de tecnología, por su naturaleza, tiene el problema de que una gran parte de los datos que se recogen son innecesarios, lo que aumenta el tamaño global de los datos a transmitir y procesar. Es aquí donde la tecnología de visión basada en eventos tiene una oportunidad para desarrollarse en el mercado de la automoción, ya que los sensores basados en esta tecnología funcionan de forma asíncrona, enviando únicamente los cambios de brillo que se producen en cada píxel, reduciendo el número de datos innecesarios. Dentro de este ámbito, este tipo de tecnología puede tener un gran protagonismo en distintos sistemas ADAS (*advanced driver-assistance systems*) (sistemas de detección de peatones, freno autónomo de

emergencia, aviso de salida de carril, etc.) o sensores LIDAR (Laser Imaging Detection and Ranging) para conducción autónoma.

Algunos ejemplos del uso de este tipo de tecnología puede ser el método patentado por Luminar Technologies [110], donde se correlacionan los eventos de un sensor DVS con la nube de puntos tridimensional de un sensor LIDAR. El sensor DVS se encarga de seleccionar una región de interés de la escena para que el sensor LIDAR escanee solo las partes que están cambiando en tiempo real, reduciendo la latencia y el gasto computacional que conllevan los métodos de detección y clasificación de objetos. En la misma línea, la empresa Volkswagen patentó un sistema de visión neuromórfico para la generación y el procesamiento de imágenes con el fin de respaldar los sistemas ADAS y las funciones de conducción automatizada [111]. Otro ejemplo más reciente es el de VoxelFlow [112], que es un sistema desarrollado por Terranet con tres cámaras basadas en eventos y un escáner láser que a través de triangulación genera representaciones 3D en un tiempo del orden de nanosegundos.

3.5.2. Sistemas de vigilancia

Los sensores bio-inspirados son muy útiles para sistemas de vigilancia ya que trabajan de forma asíncrona comunicando únicamente los cambios de brillo que suceden en cada píxel, por lo que por su naturaleza eliminan información redundante, tienen poca latencia y consumo.

Las aplicaciones como detección de características y seguimiento, reconocimiento o estimación de profundidad son esenciales para la implementación de sistemas de vigilancia inteligentes, lo que se traduce en características como detección y análisis de movimiento, detección de intrusos o gestión de multitudes. Todo ello sin preocuparse por la privacidad de las personas.

El ejemplo comercial de estos sistemas es el producto SmartThings Vision de Samsung [113], que es un sistema de vigilancia diseñado para detectar intrusos y se puede comprar en algunos países como Suecia. También puede detectar si alguna persona se ha caído y envía una alarma de atención médica.

4. EJEMPLOS Y ESTUDIO DE APLICACIONES

Como se ha visto en capítulos anteriores, las cámaras basadas en eventos son muy distintas a las cámaras CMOS convencionales. Estos sensores proporcionan una secuencia asíncrona de eventos que discretizan cambios en el brillo de un píxel y registran el tiempo en que este evento ocurrió.

En este apartado se explicará cómo se puede aprovechar este comportamiento para obtener distintos tipos de información con el uso de las cámaras basadas en eventos.

Para ello, se explicarán las principales características del kit de desarrollo Gen 3.1 de Prophesee, que es el que se usará para realizar los experimentos. Además, se explicará el método por el cual se obtiene la información de las características del estudio en cuestión y como consecuencia se entenderá por qué, a diferencia de las cámaras CMOS convencionales, este tipo de cámaras requiere de un procesado distinto de la información para aprovechar sus ventajas.

4.1. Kit de desarrollo EVK - Gen 3.1 VGA de Prophesee

El kit de desarrollo EVK – Gen 3.1 VGA es el que se ha usado para realizar los distintos experimentos. Sus características se listan en la Tabla 3.1.

4.1.1. Herramientas proporcionadas

Metavision Intelligence es como denomina Prophesee a su software *framework* para trabajar con cámaras de visión basadas en eventos. Este *framework* se divide en módulos. Cada módulo se encarga de abordar una aplicación en concreto y contiene un conjunto concreto de APIs, ejemplos o tutoriales.

Existen dos grandes grupos de módulos: módulos *Open* y módulos *Advance*.

Los módulos *Open* son los siguientes:

- HAL. Acceso genérico al hardware de la cámara.
- Base. Clases básicas y utilidades usadas por otros módulos.
- Core. Bloques de procesamiento comúnmente usados.
- Core ML. Funciones genéricas para *Machine Learning*.
- Driver. APIs para facilitar la interacción con sistemas basados en eventos.
- UI. Clases para mostrar imágenes de eventos por pantalla.

Mientras que los módulos *Advance* son:

- CV. Algoritmos relacionados con visión computacional: filtros, transformadas, etc.
- CV3D. Algoritmos para localizar la cámara en 3D y reconstruir su entorno.
- Analytics. Algoritmos para supervisar y analizar el flujo de eventos, por ejemplo, recuento, seguimiento, medición de vibraciones, etc.

- Calibration. Algoritmos para calibrar una cámara basada en eventos.
- ML. Módulos de Python para manipular conjuntos de datos basados en eventos y diseñar redes neuronales basadas en eventos.

Algunos de estos módulos serán los que se usen a lo largo de este apartado.

4.2. Detector de luz intermitente

Detectar luces intermitentes es un ejemplo de la ventaja de baja latencia que tienen estas cámaras respecto a las basadas en sensores CMOS convencionales. Estas capturan imágenes en una frecuencia entre 15-30 Hz con una latencia entre 50-250 ms. En cambio, las cámaras basadas en eventos tienen una latencia del orden de microsegundos, por lo que prácticamente no tienen latencia respecto a las cámaras basadas en sensores CMOS, las cuales tienen una latencia dos órdenes de magnitud más alta.

Esta ventaja, unida a un procesamiento distinto de la información, es la que permite detectar luces intermitentes a una frecuencia notablemente superior a la de las cámaras convencionales.

4.2.1. Método

Este método se describe en [114] y, aunque data de 2013, se ha usado en varios trabajos como en [115] y [116]. Un esquema del proceso que realiza el algoritmo se puede apreciar en la Figura 4.1.

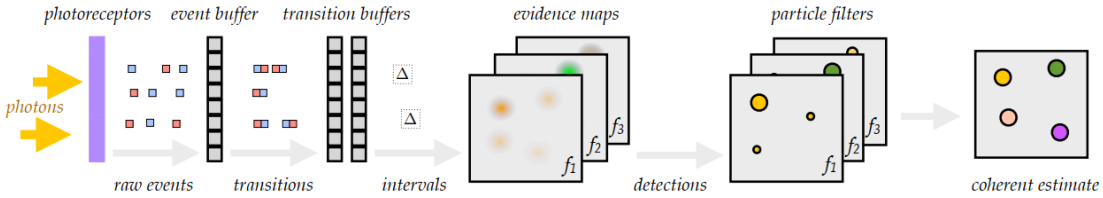

Figura 4.1. Esquema del método propuesto por D. Scaramuzza y colaboradores para la detección de luces intermitentes [114].

La entrada del algoritmo es una secuencia de eventos que representan cambios en el brillo de un píxel, obteniendo como salida una estimación de la frecuencia de parpadeo de este.

A. Eventos

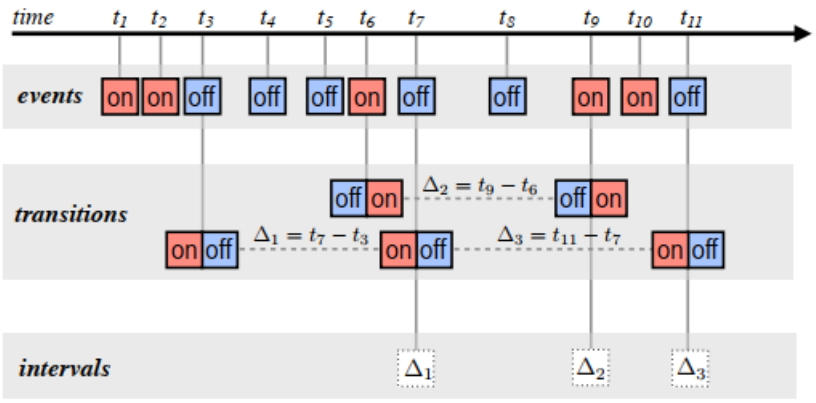

Figura 4.2. Representación de eventos, transiciones e hipertransiciones [114].

Los eventos generados vendrán caracterizados de forma:

$$\{t_k, p_k, \{x_k, y_k\}\} \tag{4.1}$$

siendo k el índice del evento, t_k representa cuándo ocurrió el evento, p_k indica si el evento es positivo y el par $\{x_k, y_k\}$ las coordenadas del píxel.

B. Transiciones

last event	*current event*	*transition event*
$\langle t_{k-1},$ on, $\langle x, y\rangle\rangle$	$\langle t_k,$ on, $\langle x, y\rangle\rangle$	none
	$\langle t_k,$ off, $\langle x, y\rangle\rangle$	$\langle t_k,$ off on, $\langle x, y\rangle\rangle$
$\langle t_{k-1},$ off, $\langle x, y\rangle\rangle$	$\langle t_k,$ on, $\langle x, y\rangle\rangle$	$\langle t_k,$ on off, $\langle x, y\rangle\rangle$
	$\langle t_k,$ off, $\langle x, y\rangle\rangle$	none

Figura 4.3. Cómo pasar de eventos a transiciones [114].

Considerando un único píxel de coordenadas $\{x, y\}$ el tiempo t_{k-1}, la polaridad p_{k-1} se almacena en todo momento. Cuando la polaridad del evento actual p_k es distinta de la polaridad previa p_{k-1} , se crea una transición. Estas transiciones se caracterizan de forma:

$$\{t_k, q_k, \{x_k, y_k\}\} \tag{4.2}$$

donde t_k es el tiempo cuando ocurrió el segundo evento que dio lugar a la transición, q_k representa la polaridad de la transición, que puede ser positiva-negativa o negativa-positiva y $\{x_k, y_k\}$ son las coordenadas del píxel.

C. Hipertransiciones

A continuación, de forma análoga a la obtención de las transiciones, el algoritmo busca lo que se denomina como “hipertransiciones”. Estas hipertransiciones se

generan tras dos transiciones sucesivas del mismo tipo y vienen caracterizadas como:

$$\{t_k, \Delta_k, \{x_k, y_k\}\} \tag{4.3}$$

donde t_k es el tiempo cuando ocurrió la hipertranisición que dio lugar a la transición, el valor Δ_k es el intervalo de tiempo entre las dos transiciones del mismo tipo y $\{x_k, y_k\}$ son, una vez más, las coordenadas del píxel. Nótese que ya no hace falta la polaridad de las transiciones, pues por cada parpadeo de un píxel, se va a generar una pareja de transiciones positiva-negativa y negativa-positiva, en un intervalo de tiempo que, si parpadea a una frecuencia estable, debe ser el mismo.

D. Mapa de probabilidad

Una vez obtenidas las hipertranisiciones, se procede a formar un "mapa de evidencia" o "mapa de probabilidad". Para ello, se definen las n frecuencias $\{f_i\}$, donde $i \in \{1, n\}$ denota el índice de las n frecuencias que se quieren detectar. Para cada una de estas frecuencias se forma el mapa de probabilidad $I_i(\langle x, y\rangle, t)$ sobre la imagen, y se asigna a cada píxel una probabilidad de que la frecuencia a detectar con índice i se encuentre sobre ese píxel. Este mapa de probabilidad se forma con las hipertransiciones, de modo que una hipertransición con intervalo $\Delta_k$contribuye al mapa de probabilidad de la frecuencia f_i con un peso proporcional a $p(\Delta_k|f_i)$. Este es el valor que se le da a la probabilidad de que una frecuencia f_i produzca una hipertransición con intervalo Δ_k. Experimentalmente, se encontró que la distribución de $p(\Delta_k|f_i)$ se asemeja a una Gaussiana, de modo que

$$p(\Delta_k|f_i) = \mathrm{N}(\frac{1}{\Delta_k} - f_i, \sigma^2) \tag{4.4}$$

La desviación aproximadamente es 30 Hz, pero puede variar en función de las condiciones del experimento.

El mapa de probabilidades recoge eventos dentro de un intervalo $1/f_i$. Por lo tanto, el mapa de probabilidades $I_i(\langle x, y\rangle, t)$ para un píxel de coordenadas x, y y un tiempo t viene dado por la suma de todas las contribuciones de las hipertransiciones provocadas por los eventos en el píxel dentro del intervalo $\left[t - \frac{1}{f_i}, t\right]$, de forma que:

$$I_i(\langle x, y\rangle, t) = \sum_{t_k \in \left[t-\frac{1}{f_i}, t\right] \wedge \langle x_k, y_k\rangle = \langle x, y\rangle} \mathrm{N}(\frac{1}{\Delta_k} - f_i, \sigma^2) \tag{4.5}$$

Al final del intervalo $\left[t - \frac{1}{f_i}, t\right]$, el mapa de probabilidades $I_i(\langle x, y\rangle, t)$ puede ser interpretado como la probabilidad de que un píxel con frecuencia de parpadeo f_i esté en la posición $\langle x, y\rangle$. El mapa, por tanto, tendrá un alto valor en los píxeles que parpadeen a f_i y un valor menor en los píxeles que parpadeen a otra frecuencia, ya que todos los eventos contribuyen a todos los mapas con distinto peso, como se indicó en la ecuación (4.4).

Por último, del mapa de probabilidades se extraen m locales máximos con al menos δ píxeles de separación entre ellos. Los valores locales máximos se usan como peso w para finalmente, detectar qué píxeles están parpadeando y a qué frecuencia. Las detecciones se clasifican con un tiempo t , unas coordenadas $\langle x, y \rangle$ y el peso w_i^j, de forma que:

$$\{\langle t, \langle x, y \rangle, w_i^j \rangle\}, \ j \in \{1, \dots, m\} \tag{4.6}$$

E. Filtrado y reconstrucción

Esta etapa se introduce con el objetivo de proporcionar robustez al algoritmo como en cualquier problema de detección.

Se utiliza un filtro de partículas para desarrollar qué puntos parpadean y a qué frecuencia. Cada partícula tiene unas coordenadas $\langle x, y \rangle$, un peso w (obtenido del paso anterior), y una incertidumbre espacial isotrópica r, que comienza en 1 píxel.

Con un filtro de partículas para cada frecuencia se elige la combinación de partículas de todos los filtros con el mayor peso combinado de forma que no haya dos marcadores demasiado cercanos entre sí.

4.2.2. Experimentos

En esta sección se va a poner a prueba esta característica con el kit Gen 3.1 VGA de Prophesee. En primer lugar, se va a medir su respuesta en frecuencia, con el objetivo de valorar en qué rango de frecuencia de parpadeo es capaz el sensor de detectar eventos y, posteriormente, se hará uso de la librería Python de proporcionada por su fabricante para el cálculo de esta frecuencia y valorar si se puede aprovechar el máximo potencial del sensor.

A. Condiciones

La Figura 4.4 representa la configuración que se ha empleado para el desarrollo del experimento. Como se puede apreciar consta de 3 elementos: una placa de evaluación del microcontrolador MSP430 de Texas Instruments que se ha usado para generar una señal PWM, un diodo LED con una resistencia en serie y el Gen 3.1 VGA de Prophesee. El microcontrolador emite una señal PWM con una amplitud de 5 V y un *duty cycle* del 50%, que va conectada al diodo LED, con la idea de recorrer un gran rango de frecuencias y observar hasta qué punto el sensor es capaz de capturar este parpadeo y si calcula la frecuencia correctamente.

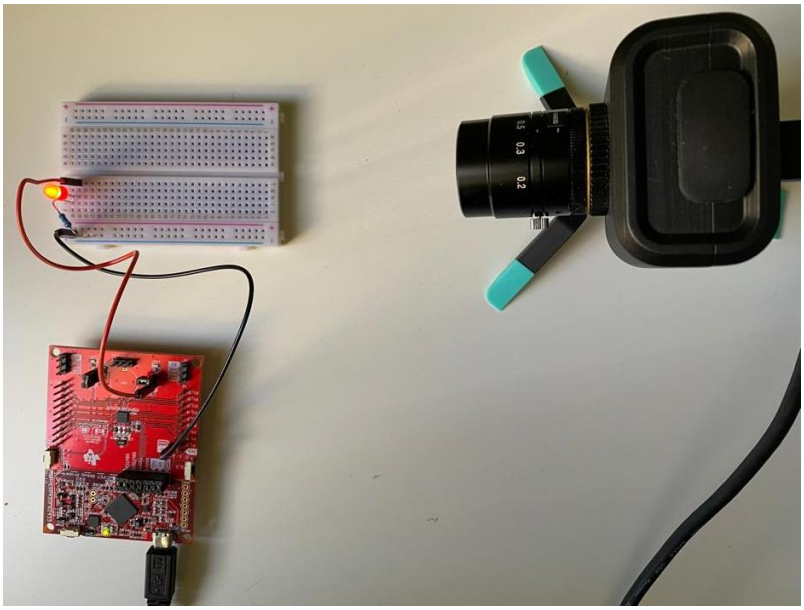

Figura 4.4. Microcontrolador, LED y sensor de Prophesee.

Además, para evitar el ruido *Flicker* de pantallas o lámparas, el experimento se ha realizado en una habitación con la luz totalmente apagada. También, para descartar eventos que se produzcan por otros efectos de ruido, se ha tomado como región de interés únicamente el LED parpadeando, como se observa en la Figura 4.5.

Figura 4.5. Imagen de la ROI seleccionada en la que el LED parpadea.

Por último, cada una de las frecuencias muestreadas se ha grabado durante 10 segundos, tiempo más que suficiente para obtener el número de datos necesarios y que el algoritmo de Prophesee pueda obtener el valor de la frecuencia.

B. Respuesta en frecuencia

Para caracterizar la respuesta en frecuencia del sensor, se han calculado tanto los eventos por segundo generados para cada frecuencia, así como los eventos por ciclo y se han representado en escala logarítmica en función de la frecuencia. Los resultados se pueden observar en la Figura 4.6.

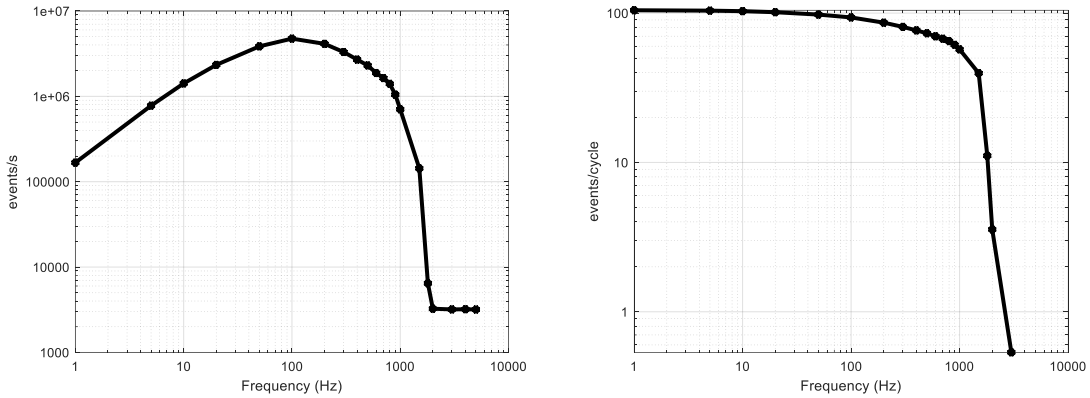

Figura 4.6. Eventos por segundo (izquierda) y eventos por ciclo (derecha).

En primer lugar, se puede observar que en ambos casos que el sensor presenta un descenso significativo en la capacidad para captar eventos a partir de los 1000 Hz. Por otro lado, se observa que el punto de máximo rendimiento se encuentra en torno a los 100 Hz, ya que es cuando el sensor capta más eventos en un tiempo dado. Siguiendo el esquema de la Figura 2.6, la parte que limita el ancho de banda del píxel es el fotorreceptor, ya que el ancho de banda es proporcional a la fotocorriente a bajos niveles de iluminación. Esto se debe a que, a baja iluminación, del fotorreceptor es proporcional a la fotocorriente porque el ancho de banda está determinado por la constante de tiempo RC formada por la capacitancia parásita del fotodiodo y la resistencia de la fuente del transistor de retroalimentación. Como esta conductancia subumbral es proporcional a la fotocorriente, el ancho de banda es proporcional a la fotocorriente a bajas intensidades [14].

C. Resultados

La librería de Prophesee proporciona cuatro parámetros para la configuración del algoritmo que obtiene la frecuencia de parpadeo: valor mínimo de la frecuencia, valor máximo, tamaño mínimo del *cluster* y tiempo máximo de diferencia. Los dos primeros indican cual deben ser los valores máximos y mínimos de frecuencia que el algoritmo debe filtrar, el tamaño mínimo del *cluster* indica cuál debe ser el tamaño mínimo de la agrupación de eventos que deben parpadear a la misma frecuencia, mientras que el tiempo máximo de diferencia fija el tamaño máximo de la ventana temporal en la que el algoritmo trabajará.

Los valores mínimos y máximos de frecuencia se han fijado en 0.1 Hz y 10000 Hz, respectivamente. El tamaño mínimo del *cluster* se ha ajustado a 5 y el tiempo máximo de diferencia en 1s.

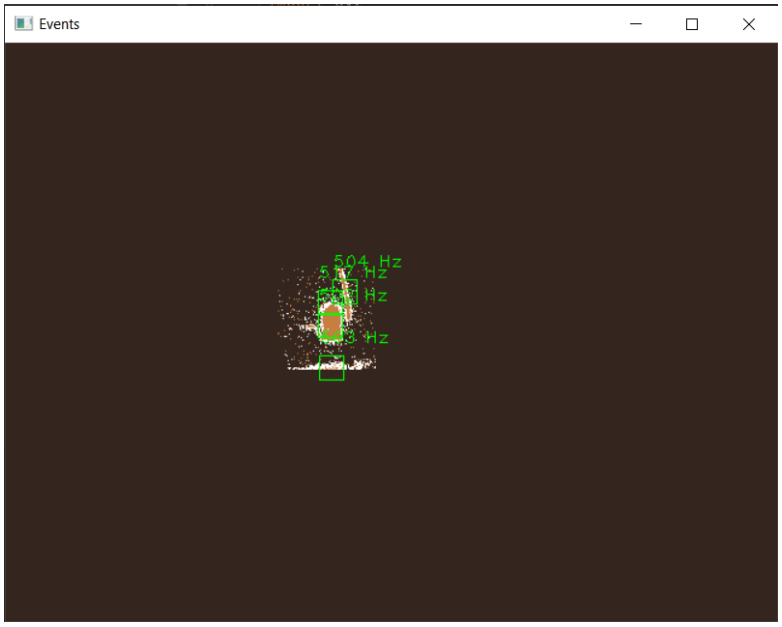

Figura 4.7. Salida del algoritmo para la grabación a 500 Hz.

La Figura 4.7 muestra la salida del algoritmo para la frecuencia de 500 Hz, donde se observa que se han remarcado los distintos *clusters* para mostrar la frecuencia capturada. Los resultados para todas las frecuencias muestreadas se pueden observar en la Figura 4.8.

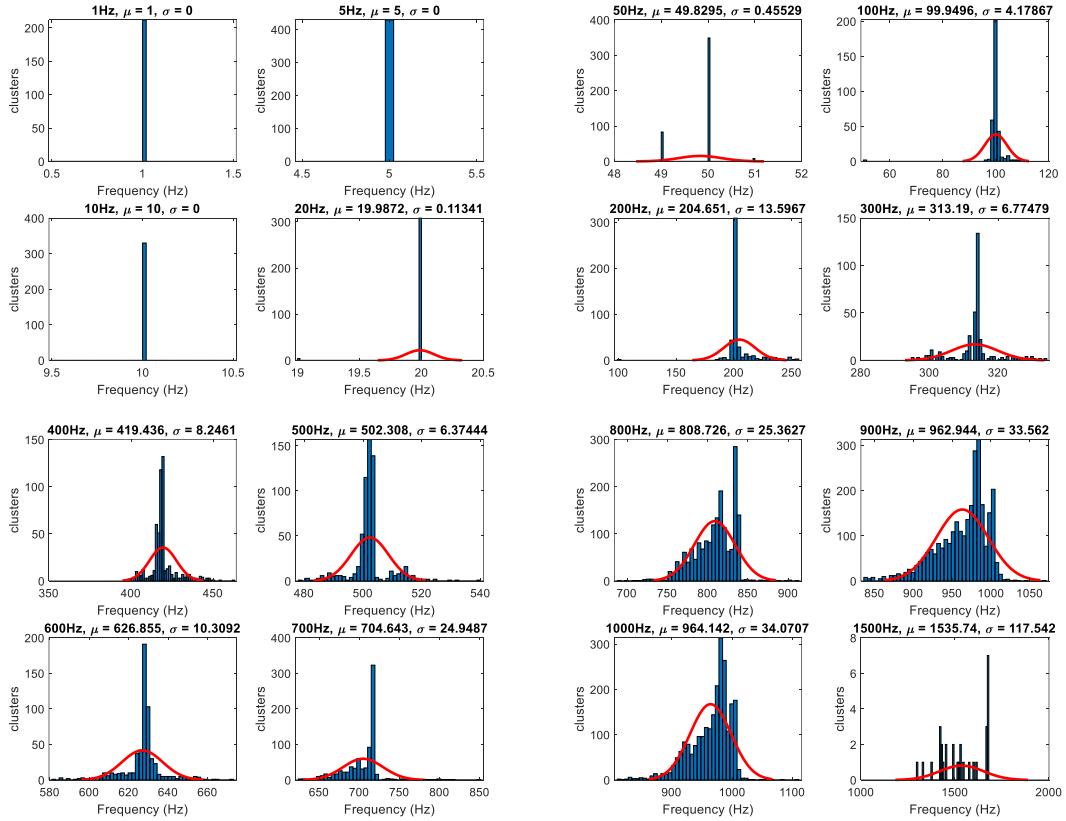

Figura 4.8. Resultados de los valores de frecuencia obtenidos con la librería de Prophesee.

Para caracterizar el rendimiento del algoritmo, se han modelado los resultados como una distribución de probabilidad estadística normal. La media, la desviación típica y el número de *clusters* totales se puede observar en la Tabla 4.1.

Tabla 4.1. Media, desviación típica y número de *clusters* de todas las frecuencias muestreadas.

Frecuencia (Hz)	μ (Hz)	σ (Hz)	Nº Clusters
1	1	0	212
5	5	0	429
10	10	0	330
20	19.98	0.11	312
50	49.82	0.45	440
100	99.94	4.17	337
200	204.65	13.59	490
300	313.19	6.77	363
400	420.43	8.24	548
500	502.30	6.37	623
600	626.85	10.33	585
700	704.64	24.94	824
800	808.72	25.36	1794
900	962.94	33.56	2833
1000	964.14	34.07	2383
1500	1535.74	117.54	31
1800	N/A	N/A	0
2000	N/A	N/A	0
3000	N/A	N/A	0
4000	N/A	N/A	0

Observando la Figura 4.8 y Tabla 4.1, se puede decir que el rendimiento del algoritmo es muy bueno hasta la frecuencia de 100 Hz, a partir de la cual se empiezan a notar diferencias respecto a la media de los valores obtenidos, así como un aumento de la desviación típica. No obstante, el rendimiento sigue siendo aceptable hasta que alcanza los 900 Hz, frecuencia que coincide con una caída de los eventos por segundo y eventos por ciclo de la Figura 4.6, lo que da lugar a una caída de los *clusters* obtenidos, siendo prácticamente nulo para valores mayores que 1500 Hz.

4.2.3. Utilidades

Esta característica y algoritmo se podría utilizarse para la localización entre robots en maniobras acrobáticas de alta velocidad. Además, en aplicaciones como la robótica de rescate o la calibración de este tipo de sensores.

4.3. Detector de esquinas

La aplicación que se presenta a continuación consiste en detectar las esquinas de los objetos, filtrando aquellos eventos que son generados por ellas.

Las esquinas en una escena son características que poseen mucha información, están bien localizadas y son menos susceptibles al enfoque de la cámara. Además, permite reducir considerablemente el número de eventos en escenas muy texturizadas donde se generan millones de ellos, lo que reduce la latencia en posteriores etapas para aplicaciones como la odometría visual (Sección 3.3.5) y o seguimiento de objetos (Sección 3.3.1), ya que estas pueden funcionar bien utilizando solo las esquinas de la escena.

En esta sección se van a estudiar dos métodos para el filtrado de esquinas. Ambos métodos utilizan superficies temporales (Sección 3.2.4) como preprocesamiento de

los eventos, a las cuáles se les aplican operaciones de comparación para obtener los eventos generados por las esquinas.

4.3.1. Método

Se van a estudiar dos métodos para ello, uno propuesto por Mueggler et al. [102] y uno propuesto por Ignacio Alzugaray y Margarita Chli [117].

A. Mueggler et al.

Este método, que denominaremos a partir de ahora como *FastDetector*, está inspirado en el método FAST [118], el cual es un detector de esquinas para cámaras tradicionales. FAST considera un píxel como esquina si n píxeles contiguos a lo largo de un círculo con centro en el píxel de interés tienen mayor o menor brillo que el píxel central más un umbral. Por lo general, se toma como $n = 9$ y un círculo de radio 3, con 16 píxeles.

Dado que las cámaras de eventos no representan intensidades absolutas de brillo y la información visual recibida por ellas se actualiza de forma asíncrona, los autores proponen trabajar con superficies temporales, separando los eventos por polaridad y procesándolos de forma independiente.

El método se basa en el análisis de las temporalidades de los eventos alrededor del último evento que se ha reportado. Una esquina en movimiento creará un mapa local de eventos, tal como se presenta en la Figura 4.9, donde se observan dos regiones claramente diferenciadas: aquellas con una temporalidad mayor (eventos más recientes) y aquellas con una temporalidad menor (eventos más antiguos). De forma análoga a FAST, el método buscará segmentos de píxeles contiguos con una mayor temporalidad que el resto de los píxeles del círculo.

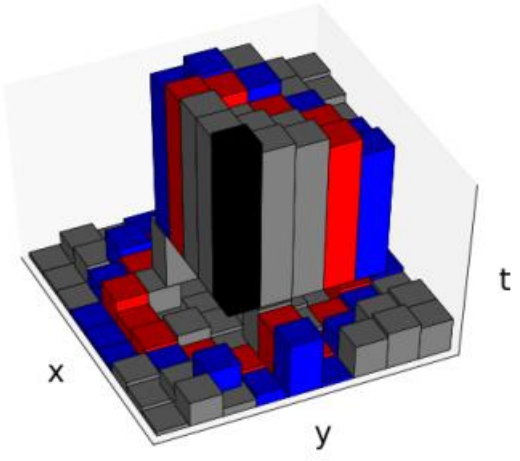

Figura 4.9. Representación de una esquina como superficie temporal alrededor del último evento reportado (negro) y los círculos usados para la comparación de temporalidades [102].

De forma más específica, se definen dos círculos de radio 3 y 4, tal como se muestra en Figura 4.10, en los cuáles se harán las comprobaciones.

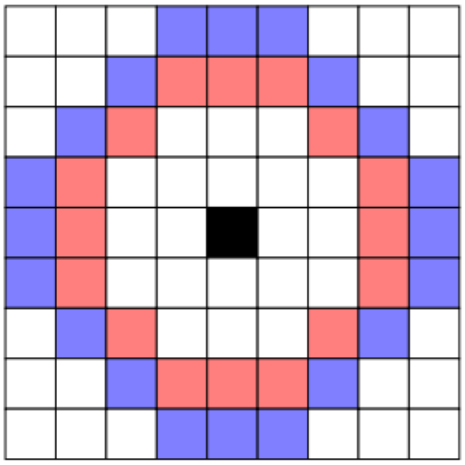

Figura 4.10. Círculos de radio 3 y 4 [102].

Para el círculo interior (círculo rojo), se buscarán segmentos con una longitud entre 3 y 6 píxeles. Para el círculo exterior (círculo azul), las longitudes deberán ser entre 4 y 8 píxeles. Finalmente, si en ambos círculos se encuentran segmentos que cumplan la determinada longitud, el píxel central se considerará como esquina.

B. Ignacio Alzugaray y Margarita Chli

Este método, que denominaremos a partir de ahora como *ArcStarDetector*, está inspirado en el anterior, basándose también en el uso de superficies temporales y analizando las temporalidades de los eventos alrededor del último evento reportado. De forma similar, se buscarán segmentos de píxeles contiguos con una mayor temporalidad que el resto.

No obstante, este método solventa alguno de sus problemas del método anterior. El principal problema que se presenta en el método anterior es que las temporalidades alrededor del círculo en cuestión dependen de la dirección del movimiento de la esquina, tal como se representa en la Figura 4.11, donde se puede observar que la misma esquina puede generar dos segmentos complementarios en función de la dirección del movimiento. En el caso de la izquierda podrá ser detectada por el método anterior, ya que la longitud de los segmentos entra dentro de lo establecido, pero la del caso de la derecha no, ya que la longitud es mayor.

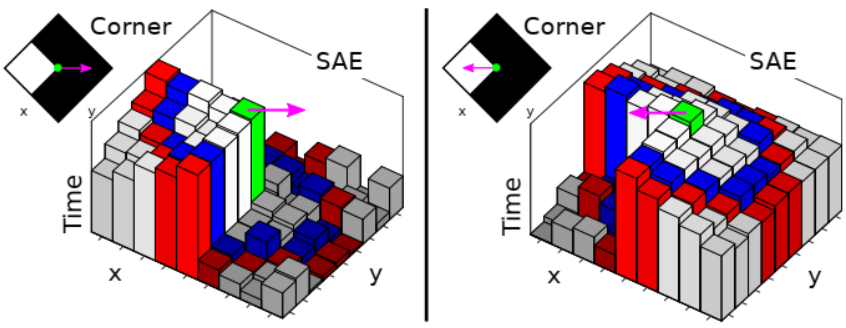

Figura 4.11. Representación de una esquina como superficie temporal para direcciones de movimiento opuestas [117].

Para resolver este problema, la localización de un nuevo evento define una máscara circular de un radio predefinido tal y como se muestra en la Figura 4.12, la cual

define los eventos del círculo C en el que se buscarán los segmentos. Dentro de este círculo, se inicializa un arco A_{new} con el evento de temporalidad más alta, así como el par de eventos adyacentes E_{CCW} y E_{CW} a ese evento. En cada una de las iteraciones se selecciona el evento adyacente al arco con temporalidad más alta E_{CCW} o E_{CW}. Si el evento más antiguo del arco A_{new} es también más viejo que el evento adyacente seleccionado, el arco A_{new} se expande hasta ese elemento y se actualiza la posición del evento adyacente E_{CCW} o E_{CW} seleccionado a la siguiente posición en el círculo. Inicialmente, aunque la condición previamente mencionada no se cumpla, el arco A_{new} se expande hasta una longitud mínima L_{min}. Este proceso se repite hasta que la posición de E_{CCW} y E_{CW} coinciden. En ese momento, el evento se clasificará como esquina si la longitud del arco A_{new} o su complementario en el círculo $C \backslash A_{new}$ tiene un valor entre L_{min} y L_{max}.

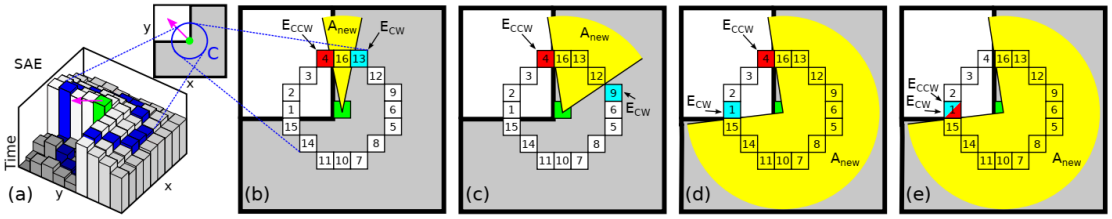

Figura 4.12. (a) Superficie temporal local del nuevo evento (verde) y eventos del círculo C (azul). (b) Arco A_{new} inicializado y eventos adyacentes E_{CCW} (rojo) o E_{CW} (cian). (c) Arco A_{new} expandido hasta una longitud $L_{min} = 3$ hacia el evento adyacente más antiguo, en este caso, E_{CW}. (d) La posición de E_{CW} se actualiza en cada iteración hasta alcanzar la posición 15 del círculo C, en ese momento, el arco A_{new} se expande hasta dicha posición. (e) La posición E_{CCW} se actualiza hasta que coincide con la de E_{CW}. El evento se clasifica como corner ya que la longitud del arco complementario es 4 y cumple con el rango predefinido [117].

Al igual que el método anterior, se emplea el uso de círculos de radio 3 y 4, y las longitudes mínimas para detectar si el evento es o no una esquina también será entre 3 y 6 para el círculo de radio 3, y entre 4 y 8 para el círculo de radio 4.

Además, este método también hace uso de un filtrado previo de los eventos para reducir aquellos que se procesan por el algoritmo. Este filtrado se basa en la implementación de una ventana temporal k. Tras la llegada de un nuevo evento con temporalidad t_r, este se almacenará en la superficie temporal, pero los eventos reportados en el mismo píxel cuya temporalidad sea $t < t_r + k$ serán rechazados, tal como se muestra en la Figura 4.13.

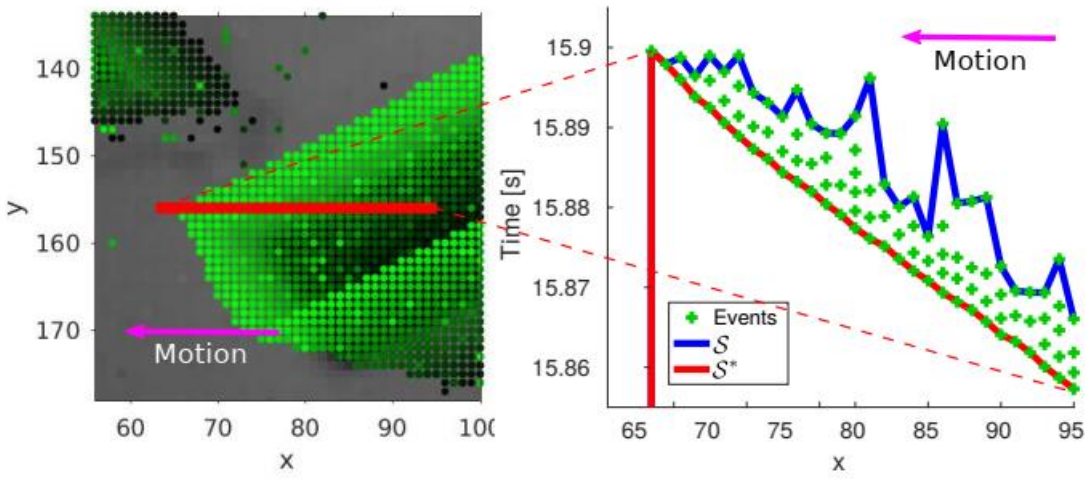

Figura 4.13. Eventos generados debidos al movimiento de un rectángulo (izquierda). Representación de la temporalidad de los eventos generados por la línea roja en el eje x (derecha). La superficie temporal $\mathcal{S}$ se representa en azul, mientras que la superficie temporal filtrada $\mathcal{S}^*$ se representa en rojo [117].

Este filtrado permite representar de forma precisa aquellas regiones con un gran contraste de eventos, así como reducir el número de datos a ser procesados, reduciendo también el tiempo de procesamiento y haciendo el algoritmo más eficiente.

4.3.2. Experimentos

A. Adaptación al entorno de Prophesee

Los trabajos explicados en el apartado anterior fueron testeados en un entorno ROS Kinetic. Para la Gen 3.1 (Tabla 3.1) Prophesee proporciona un *framework* y un set de herramientas y librerías en C++ que permiten procesar los eventos adquiridos por la cámara de la forma más eficiente posible.

La clase *Pipeline* de la librería es la que se usa para organizar el procesado de los datos. Estos se encadenan en unidades de procesamiento, llamadas etapas. Cada una de estas etapas realiza una función distinta, permitiendo que en su conjunto se produzca un procesamiento completo de los eventos.

Para este experimento, se ha construido un *Pipeline* que tiene como entrada los eventos generados por la cámara o una grabación, y muestra como salida dos pantallas, una con los eventos sin filtrar y otra con las esquinas detectadas. El esquema de este *Pipeline* se puede apreciar en la Figura 4.14.

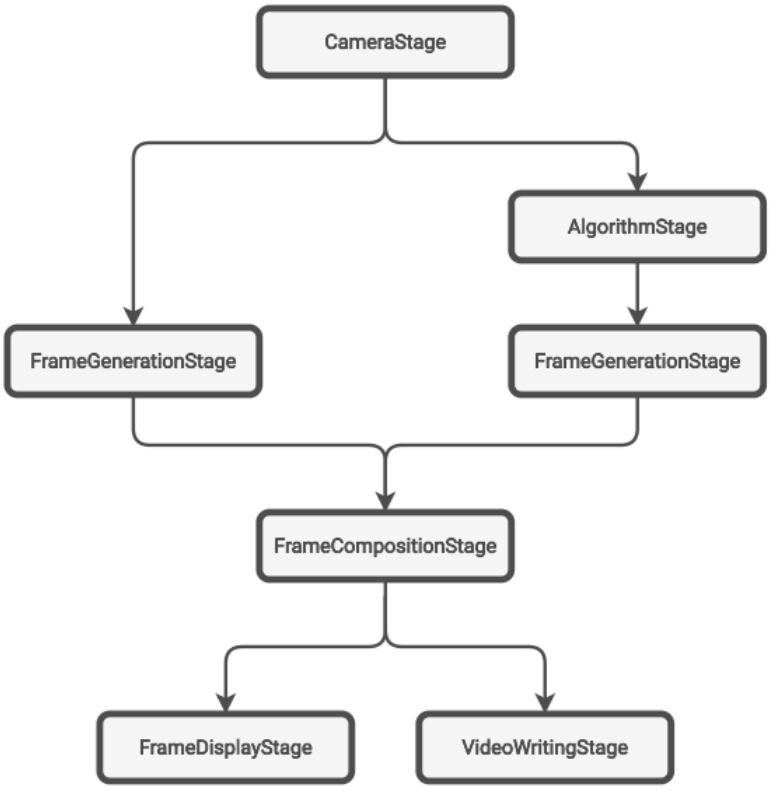

Figura 4.14. Pipeline empleado para el testeo de los algoritmos.

La función de cada una de las etapas es la siguiente:

- *CameraStage*: se encarga de adquirir eventos de una cámara o de una grabación (archivo .raw).
- *AlgorithmStage:* aplica algún tipo de algoritmo a los eventos. En este caso, es una etapa personalizada donde se han implementado los detectores de esquina.
- *FrameGenerationStage:* genera una imagen de eventos (Sección 3.2.2) a partir de los eventos adquiridos de las etapas previas.
- *FrameCompositionStage:* se usa para componer una pantalla con distintas imágenes.
- *FrameDisplayStage:* muestra en pantalla la imagen.
- *VideoWritingStage:* se encarga de grabar un video de las imágenes.

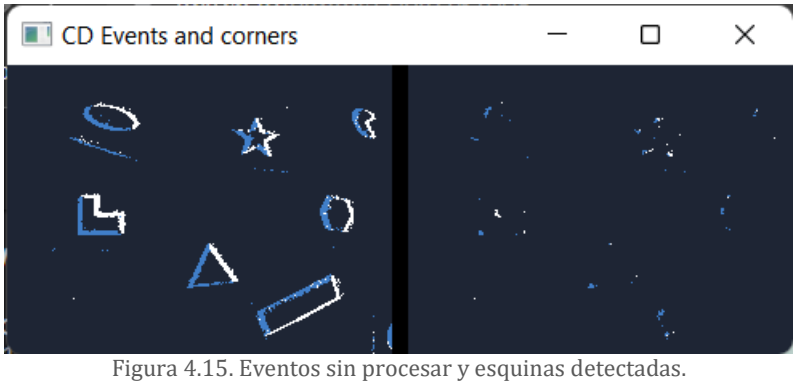

Figura 4.15. Eventos sin procesar y esquinas detectadas.

Resumidamente, la bifurcación de la izquierda de la Figura 4.14 no realiza ningún procesamiento de los eventos y los muestra en pantalla, mientras que la bifurcación de derecha se encarga detectar las esquinas y también mostrar los eventos que se han definido como tales por pantalla. La Figura 4.15 muestra la salida del *Pipeline* durante su ejecución.

B. Adaptación del Event-Camera Dataset

Los métodos fueron probados en la base de datos Event-Camera Dataset [119], la cual es una colección de conjuntos de datos grabados con un sensor DAVIS en una variedad de entornos sintéticos y reales, con el objetivo de que sirva como base de datos sólida para comparar algoritmos destinados a la robótica, alto rango dinámico y aplicaciones de visión por computador de alta velocidad.

Este conjunto de datos se proporciona en dos formatos, archivos .txt y archivos .rosbag. En ambos casos, este tipo de datos no es compatible con las librerías de Prophesee, por lo que para hacer uso de ellos se tuvo que programar un *script* en Python que pasara del formato .txt al formato .raw.

En el formato .txt se codifica un evento por línea tal y como se muestra en la Tabla 4.2:

Tabla 4.2. Formato .txt de la base de datos Event-Camera Dataset.

Temporalidad (µs)	Eje X (Píxel)	Eje Y (Píxel)	Polaridad

Sin embargo, los archivos .raw son archivos binarios *Little Endian* que pueden ser de dos formatos distintos, EVT 2.0 y EVT 3.0. Para el caso del sensor Gen3.1, el formato que se debe usar es el EVT 2.0.

El formato EVT 2.0 está compuesto por palabras de 32 bits (4 bytes). Los 4 bits más significativos (MSB) de cada palabra se usan para definir el tipo de esta. Los distintos tipos y sus valores se muestran en la Tabla 4.3.

Tabla 4.3. Tipos de palabras para el formato EVT 2.0.

Tipo	**Descripción**	**Valor**
CD_OFF	Evento con polaridad 0.	‘0000’
CD_ON	Evento con polaridad 1.	‘0001’
EVT_TIME_HIGH	Bits más significativos de la temporalidad de los eventos. Los bits menos significativos (LSB) se adjuntan a las palabras CD_OFF / CD_ON.	‘1000’
EXT_TRIGGER	Salida de activación externa.	‘1010’
OTHERS	Se usa para extensiones en los tipos de eventos.	‘1110’
CONTINUED	Datos extra usados por eventos que llegan en varias palabras. Son específicas de cada vendedor.	‘1111’

Los 28 bits restantes dependen del tipo de palabra. Para este caso, solo se van a emplear los tipos de palabra CD_OFF, CD_ON y EVT_TIME_HIGH. El EVT_TIME_HIGH se muestra en la Tabla 4.4 y se emplea para establecer los 28 bits más significativos de la temporalidad de los eventos siguientes, ya que esta se codifica con 34 bits.

Tabla 4.4. Formato de palabra tipo EVT_TIME_HIGH.

Rango	Longitud	Campo	Descripción	Valor
31-28	4	Tipo	Tipo de palabra	'1000'
27-0	28	Temporalidad	Bits más significativos de la temporalidad de los siguientes eventos.	-

Los tipos de palabra CD_OFF y CD_ON se muestran en las tablas Tabla 4.5 y Tabla 4.6, respectivamente. Como se puede observar, los 28 bits restantes incluyen los 6 bits menos significativos de la temporalidad y las coordenadas X e Y, que se codifican con 11 bits cada una.

Tabla 4.5. Formato de palabra tipo CD_OFF.

Rango	Longitud	Campo	Descripción	Valor
31-28	4	Tipo	Tipo de palabra	'0000'
27-22	6	Temporalidad	Bits menos significativos de la temporalidad.	-
21-11	11	Coordenada X	Coordenada X del píxel.	-
10-0	11	Coordenada Y	Coordenada Y del píxel.	-

Tabla 4.6. Formato de palabra CD_ON.

Rango	Longitud	Campo	Descripción	Valor
31-28	4	Tipo	Tipo de palabra	'0001'
27-22	6	Temporalidad	Bits menos significativos de la temporalidad.	-
21-11	11	Coordenada X	Coordenada X del píxel.	-
10-0	11	Coordenada Y	Coordenada Y del píxel.	-

El *script* de Python que permite pasar de un formato a otro también se encuentra en el repositorio [30] y se deja a disposición del lector.

C. Condiciones

Los métodos se han comparado de tres formas distintas en el *framework de* Prophesee.

1. Comparación con un ejemplo del conjunto de datos del sensor DAVIS, pero en el entorno de Prophesee.
2. Comparación con una grabación del Gen 3.1, que tiene una resolución mayor que el sensor DAVIS.
3. Adaptación a la misma grabación del Gen 3.1 pero adaptando el código a una resolución mayor. La adaptación propuesta ha sido utilizar máscaras circulares de radio 7 y 8, así como buscar segmentos entre 3 y 12 píxeles para la máscara de radio 7, y 4 y 16 para la máscara de radio 8.

En los tres casos, se va a comparar el tiempo de procesado de los eventos como su nivel de detección de esquinas.

D. Resultados

Para la comparación con un ejemplo de la base de datos del sensor DAVIS se ha usado el conjunto *shapes_6dof.*, mientras que para la comparación para la comparación con la grabación del Gen 3.1 se ha usado la denominada *filled_figures*.

La Tabla 4.7 muestra los resultados del primer experimento. Se puede apreciar que tanto el tiempo de la grabación como el tiempo de ejecución son similares, por lo que el algoritmo puede ejecutarse en tiempo real. No obstante, el número de esquinas detectadas por el ArcStarDetector es tres veces superior a las del FastDetector, con un tiempo medio de ejecución inferior. Esto no quiere decir que estos eventos detectados sean realmente esquinas, pero si deja una clara evidencia de que el ArcStarDetector es computacionalmente mucho más eficiente que el FastDetector.

Figura 4.16 muestra una imagen de la grabación del experimento. Se observa que el FastDetector es capaz de detectar esquinas que se mueven en una dirección concreta mientras que el ArcStarDetector permite detectarlas todas.

Tabla 4.7. Resultados del primer experimento, en el entorno de Prophesee con el conjunto de datos *shape_6dof.*

Parámetro	FastDetector	ArcStarDetector
Tiempo de grabación (s)	59.73	
Tiempo de ejecución (s)	59.69	59.69
Eventos	17962477	
Eventos/s	300701	
Nº de esquinas	722603	2094390
Tiempo medio por iteración (µs)	204	168

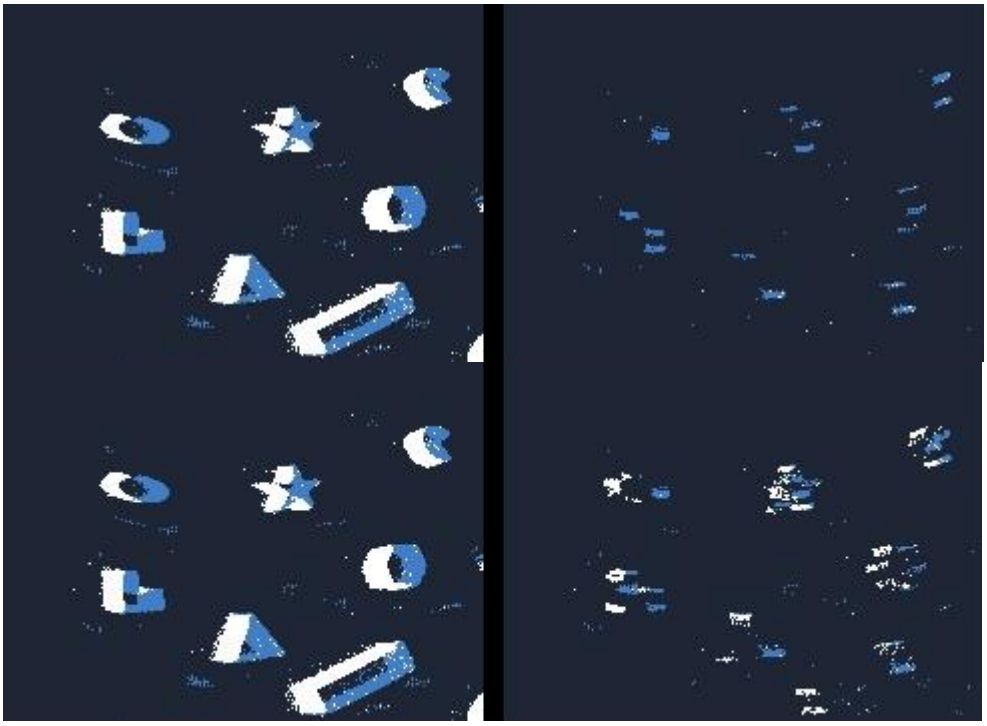

Figura 4.16. Imagen del primer experimento. Arriba, FastDetector. Abajo, ArcStarDetector.

La Tabla 4.8 muestra los resultados del segundo experimento en el entorno de Prophesee con la grabación de la Gen 3.1 *filled_figures*. Se puede observar que, a pesar de que esta grabación tiene el triple de eventos por segundos que la anterior, el tiempo de grabación y de ejecución coinciden, por lo que ambos algoritmos pueden ser procesados en tiempo real. No obstante, como se puede apreciar en la Figura 4.17, los eventos detectados como esquinas no pueden realmente considerarse como tales, ya que se detecta como esquina prácticamente cualquier borde. Esto se debe a que el tamaño de las máscaras circulares no es el apropiado para el tamaño de estas figuras.

Tabla 4.8. Resultados del segundo experimento, en el entorno de Prophesee con la grabación de la Gen 3.1 *filled_figures*

Parámetro	FastDetector	ArcStarDetector
Tiempo de grabación (s)	16.47	
Tiempo de ejecución (s)	16.49	16.50
Eventos	14918672	
Eventos/s	905595	
Nº de esquinas	148486	378326
Tiempo medio por iteración (µs)	198	105

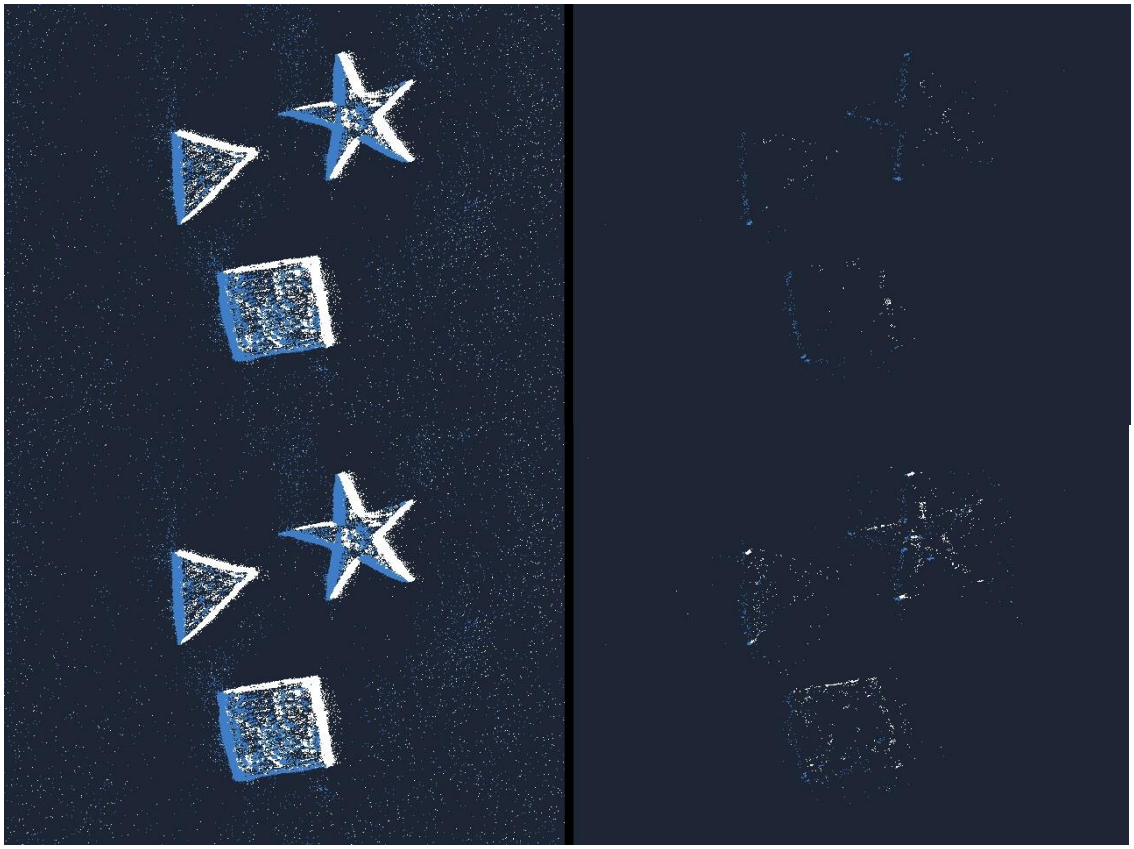

Figura 4.17. Imagen del segundo experimento. Arriba, FastDetector. Abajo, ArcStarDetector.

Por último, la Tabla 4.9 recoge los datos del tercer experimento, en el entorno de Prophesee con la grabación de la Gen 3.1 *filled_figures* y la modificación propuesta. En este caso, se observa como el FastDetector no se puede ejecutar en tiempo real y el tiempo medio por iteración es casi cinco veces mayor que el del ArcStarDetector, que sí se puede ejecutar en tiempo real. Además, el ArcStarDetector detecta cinco veces más esquinas que el FastDetector. Como se comentó en el primer experimento, esto muestra que el ArcStarDetector es mucho más eficiente en términos computacionales. Aun así, en la Figura 4.18 se aprecia que las esquinas son más pronunciadas en ambos casos, pero no tanto como se desearía. Esto se debe a que el tamaño de las máscaras circulares no es realmente el apropiado para una grabación tan texturizada, aunque aumentarlo implicaría un mayor tiempo de ejecución y la posibilidad de que el ArcStarDetector no se pueda ejecutar en tiempo real.

Tabla 4.9. Resultados del tercer experimento en el entorno de Prophesee con la grabación de la Gen 3.1 *filled_figures* y la modificación propuesta.

Parámetro	FastDetector	ArcStarDetector
Tiempo de grabación (s)	16.4739	
Tiempo de ejecución (s)	20.91	16.48
Eventos	14918672	
Eventos/s	905595	
Nº de esquinas	43504	215299
Tiempo medio por iteración (µs)	1302	292

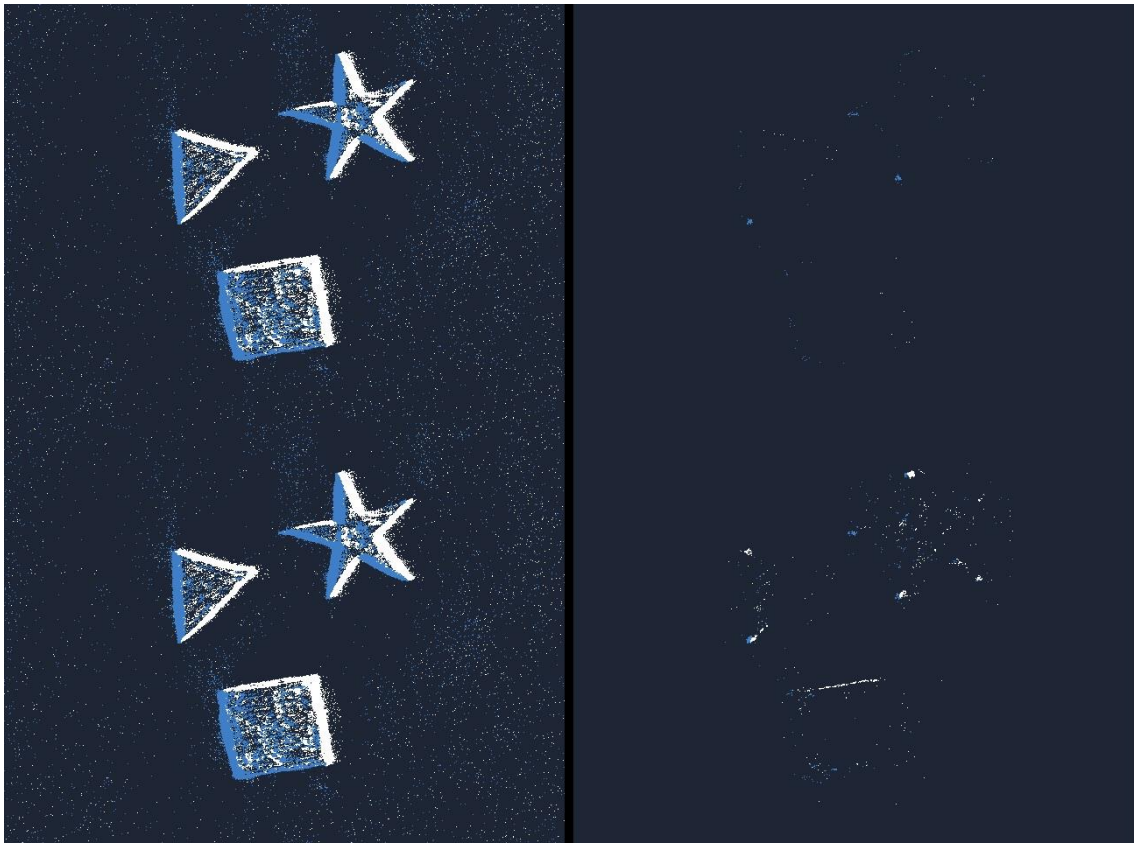

Figura 4.18. Imagen del tercer experimento. Arriba, FastDetector. Abajo, ArcStarDetector.

4.3.3. Conclusiones

Como ya se ha comentado en el apartado anterior, el ArcStarDetector es computacionalmente mucho más eficiente que el FastDetector y también es capaz de detectar más esquinas. Aun así, el hecho de identificar más eventos como esquinas en menos tiempo no implica que estos eventos sean realmente esquinas y habría que establecer un mecanismo de comparación para ambos métodos. De hecho, en ambas publicaciones se establecen mecanismos de ”ground truth” distintos, cada uno con distintas ventajas e inconvenientes, por lo que la comparación entre ambos algoritmos realmente no puede ir más allá de su eficiencia computacional.

Además, se observa que la detección de las esquinas depende mucho del tamaño de la figura, ya que el tamaño de las máscaras y de las longitudes de los segmentos a buscar deberá ser mayor para figuras mayores, por lo que la adaptabilidad a distintos entornos es limitada.

Por último, cabe comentar que el gasto computacional es muy elevado en ambos casos. Procesar millones de eventos a través de bucles puede ser el cuello de botella de un sistema basado en eventos, ya que se debe tener en cuenta que esta sería una de las muchas etapas de procesado.

4.3.4. Utilidades

El principal uso de este algoritmo sería en una de las primeras etapas de un sistema SLAM, ya que las esquinas proporcionan información muy relevante de la imagen. También puede usarse para reconstrucción 3D o reconocimiento de objetos.

4.4. Estimación del flujo óptico

La última aplicación que se va a estudiar será la estimación del flujo óptico. Como se explicó en el apartado 3.3.2, la estimación del flujo óptico se puede definir como la representación del desplazamiento de los píxeles en un periodo corto de tiempo. La estimación de este desplazamiento es fundamental para otras aplicaciones de mayor complejidad como detección de objetos y seguimiento, estimación de movimiento, odometría visual, etc. No obstante, no existe una solución directa a este problema utilizando algoritmos para las cámaras convencionales debido a la disparidad y asincronía natural de los eventos, por lo que a lo largo de la historia se han propuesto distintos enfoques para la resolución de este problema.

4.4.1. Método

En este apartado se revisará el enfoque propuesto por Brebion et al. [120]. También se harán pruebas con las herramientas proporcionadas por Prophesee.

A. Brebion et al.

El método propuesto por Brebion et al. está basado en el trabajo de Almatrafi et al. [121] donde, a grandes rasgos, se buscaba acumular eventos durante un periodo de tiempo para crear imágenes de estos y usar algoritmos tradicionales para estimar el flujo óptico. Brebion et al. han conseguido hacer este método más rápido y robusto, haciéndolo compatible con sensores de mayor resolución, como el Gen 4 (Tabla 3.1).

Un esquema de la arquitectura propuesta se muestra en la Figura 4.19. En ella, se pueden observar cuatro etapas que se encargan de todo el procesado para la obtención del flujo óptico.

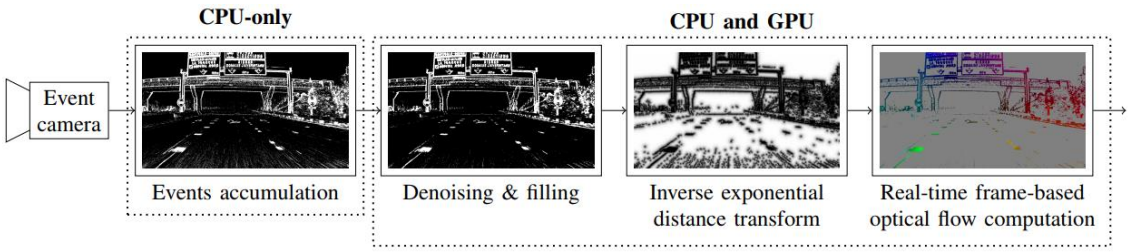

Figura 4.19. Arquitectura propuesta por Brebion y colaboradores [120].

Las cuatro etapas son las siguientes:

1. **Acumulación de eventos**. El primer componente de la arquitectura se encarga de recibir y acumular los eventos de una cámara, formando imágenes de eventos con ellas (Sección 3.2.2) de un determinado periodo ΔT, pero sin tener en cuenta la polaridad de los eventos para ahorrar en gasto

computacional. La ventaja de usar imágenes de eventos en lugar de superficies temporales o imágenes reconstruidas es que únicamente se recoge la información necesaria para usar un enfoque basado en imágenes, lo que computacionalmente hablando se traduce en, una vez más, eficiencia computacional.

2. **Eliminación de ruido y rellenado**. Dado que las cámaras de eventos generan una gran cantidad de ruido, es necesario implementar una etapa se deshaga de este ruido para no afectar a la calidad de la estimación del flujo óptico. Algunos métodos de eliminación de ruido se implementan en la etapa de acumulación de eventos, pero estos son computacionalmente muy exigentes, por lo que los autores proponen aplicar la eliminación de ruido una vez se ha creado la imagen de eventos. Además, también se implementa una etapa de rellenado para estabilizar la imagen. Ambos procesos se realizan por separado para evitar crear inconsistencias. La idea para la eliminación de ruido es descartar píxeles aislados, mientras que para el rellenado se busca añadir eventos en aquellas regiones donde hay suficientes píxeles como para estar seguros de que los píxeles en cuestión deberían haber generado un evento. Un ejemplo de este procesado se muestra en la Figura 4.20.

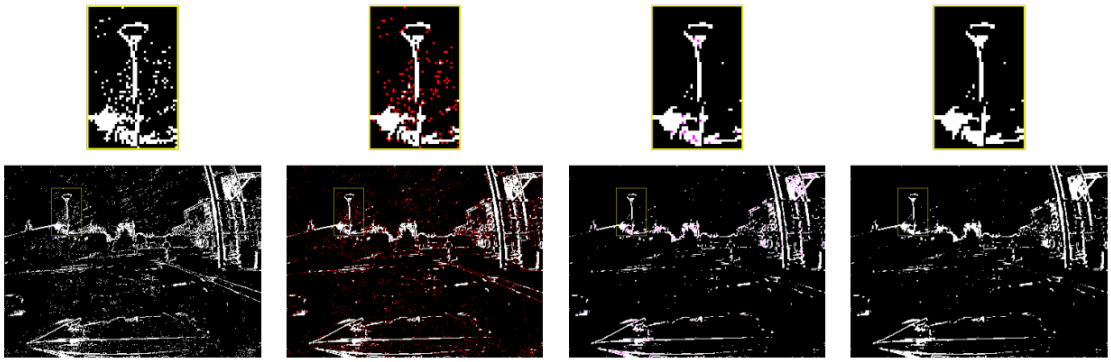

Figura 4.20. De izquierda a derecha: imagen original; imagen original con los píxeles identificados como ruido en rojo; imagen sin ruido con los píxeles que se van a rellenar en rosa; la imagen totalmente procesada [120].

3. **Superficie de distancia exponencial inversa**. La superficie de distancia temporal inversa es una variación de la superficie de distancia propuesta por Almatrafi et al. [121], la cual se trata de una representación en escala de grises de una imagen de eventos. Esta imagen en escala de grises (Figura 4.21) se forma asignando un valor a cada uno de los píxeles en función de su distancia a los píxeles que han generado un evento. Estos valores representan las formas de los objetos mediante las posiciones relativas de sus bordes, que satisfacen las ecuaciones de flujo óptico basadas en imágenes

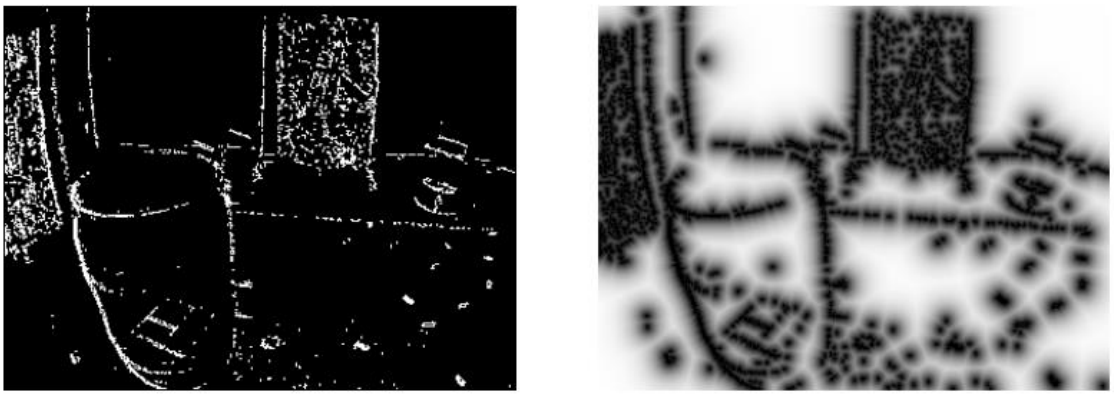

Figura 4.21. Imagen de eventos procesada (izquierda) y superficie de distancia temporal inversa (derecha) [120].

4. **Estimación del flujo óptico**. Por último, la última etapa de la arquitectura consiste en la estimación de flujo óptico en sí misma. Para ello usan el mismo método que Adarve et al.[122], que predice el flujo óptico utilizando un proceso de deformación de la imagen, y propaga temporalmente las estimaciones de flujo óptico utilizando una arquitectura incremental. Lo más interesante de este método es que permite calcular el flujo para todos los eventos y no solo para determinadas partes de la imagen, aunque los autores lo restringen a la imagen una vez se ha eliminado el ruido y se ha rellenado, ya que el uso de la superficie de distancia exponencial inversa se utiliza solo para crear transiciones suaves, que son necesarias para determinar el flujo óptico. El resultado se muestra en la Figura 4.22 a modo de ejemplo, donde se aprecia que los píxeles con tonos azules se están moviendo hacia la izquierda y aquellos con tonos rojos se mueven a la derecha.

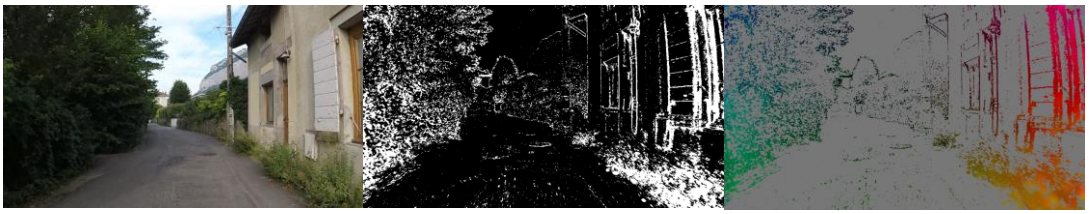

Figura 4.22. Ejemplo del cálculo de flujo óptico [120].

B. Librería de Prophesee

La librería de Prophesee proporciona la clase en C++ *SparseOpticalFlowAlgorithm* para la obtención del flujo óptico, que se calcula de forma dispersa, lo que quiere decir que la información de flujo se genera en grupos de eventos y no para cada evento. No obstante, lo cierto es que la clase no es muy parametrizable y el tamaño de los grupos de eventos no puede ser modificado. Lo único que se puede modificar es si el cálculo se desea para objetos que se mueven rápida o lentamente, pero no se define realmente qué es rápido o lento en este caso.

Como se explicó en el apartado (Sección 4.3.2.A) se usará la librería *Pipeline* para implementar las distintas etapas en las que se calculará el flujo óptico. Estas etapas se pueden observar en la Figura 4.23.

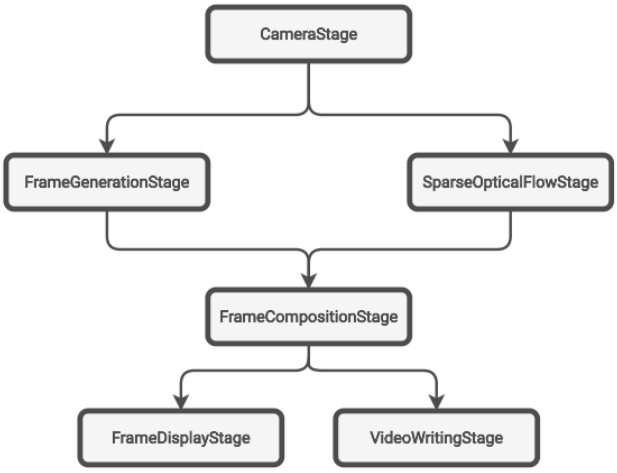

Figura 4.23. *Pipeline* para el ejemplo del cálculo del flujo óptico.

Las etapas son las mismas que en la sección 4.3.2.A, a excepción de la siguientes:

- SparseOpticalFlowStage. Etapa que se encarga de estimar el flujo óptico. Los eventos se agrupan en función de su velocidad y dirección, por lo que eventos con dirección y velocidad similares se agrupan y el flujo óptico se calcula para ellos. Se configuró para detectar objetos rápidos.

El resultado final se puede apreciar en la Figura 4.24.

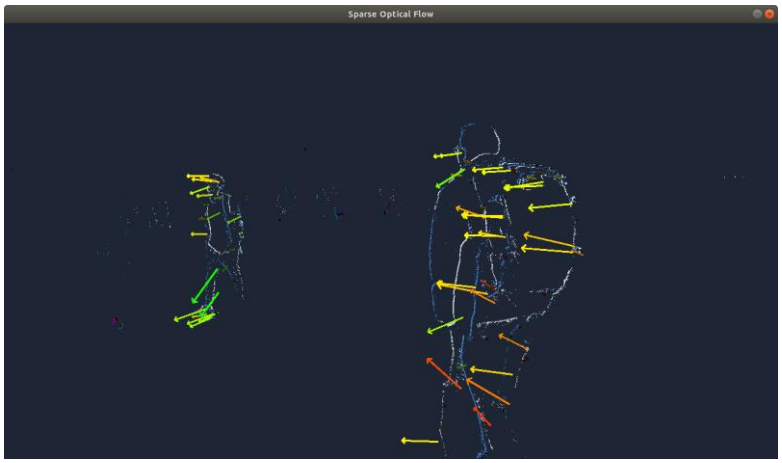

Figura 4.24. Estimación del flujo óptico con la librería de Prophesee.

La mayor diferencia, como ya se ha comentado, es que el flujo óptico no se calcula para todos los píxeles, además de que el resultado se representa con flechas y colores.

4.4.2. Experimentos

Para la comparación de ambas herramientas se van a usar tres grabaciones distintas, dos proporcionadas por Prophesee y una grabada con la Gen 3.1. Las proporcionadas por Prophesee se denomina *80_balls* y *traffic_monitoring* y ambas se pueden encontrar en su documentación oficial. La grabada con la Gen 3.1 se denomina *pirates_medallion* y se puede encontrar en el repositorio GitHub de este proyecto.

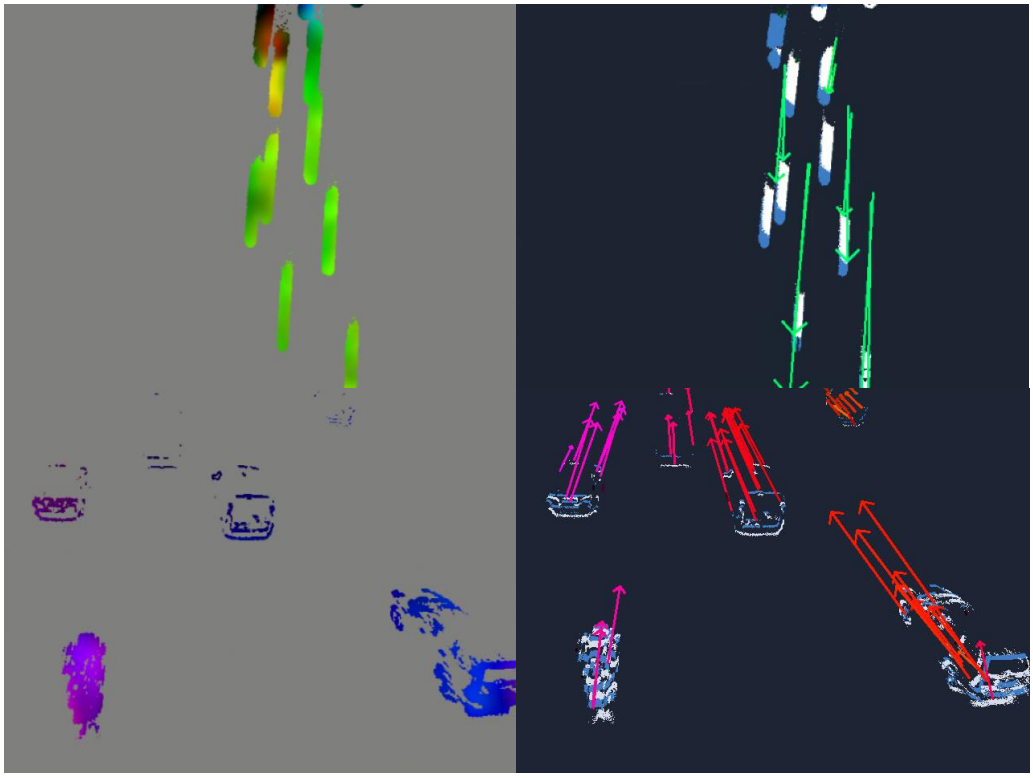

Figura 4.25. Comparación con las grabaciones de Prophesee: *80_balls* (arriba) y *traffic_monitoring* (abajo).

Con respecto a las grabaciones de la Figura 4.25, el flujo óptico se pudo calcular en tiempo real para ambas. En las dos grabaciones los resultados son similares, aunque el hecho de que el enfoque propuesto por Brebion et al. calcule el flujo óptico para todos los eventos permite obtener más información de este, aunque la utilidad dependerá de la aplicación para la que se va a usar.

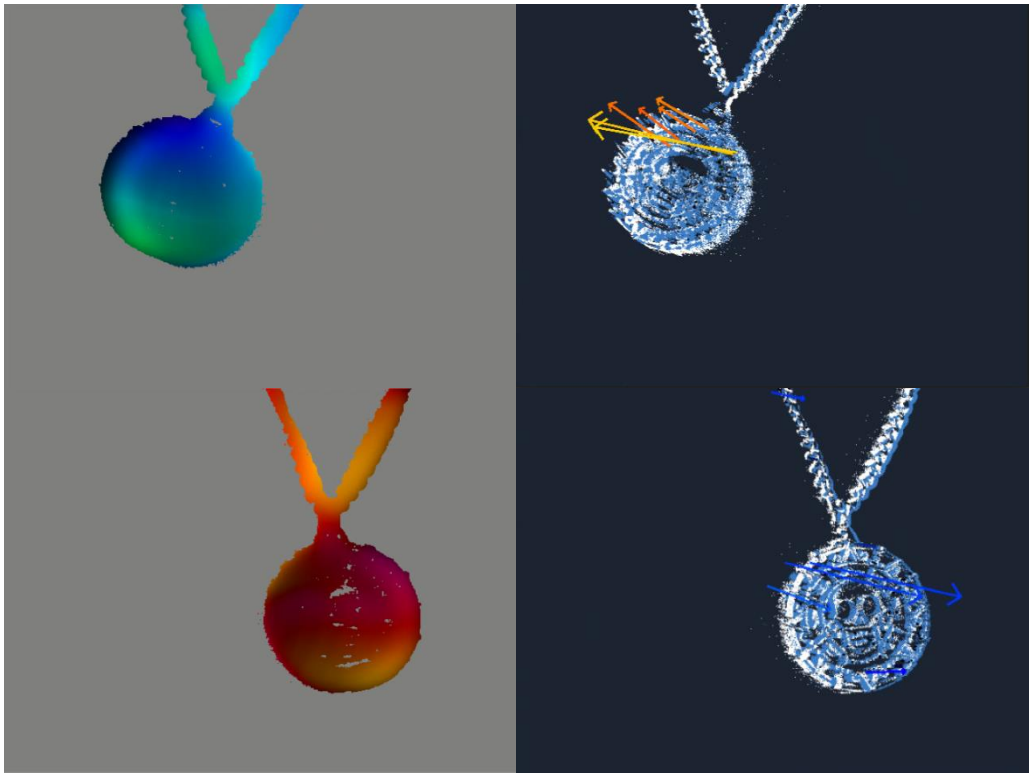

Figura 4.26. Comparación con la grabación *pirates_medallion*. En la figura de arriba, el medallón se está moviendo hacia la izquierda y la de abajo se mueve hacia la derecha.

Sin embargo, con la grabación *pirates_medallion,* el rendimiento del enfoque de Brebion et al. no funcionó en tiempo real debido a que es una grabación muy texturizada, no se hizo uso del tipo de datos originales (los autores usan archivos .rosbag mientras que el usado en este trabajo es .dat) y a la gran cantidad de recursos computacionales que los autores tenían a su disposición (portátil HP ZBook 17 G, con procesador Intel i9-9880H CPU, tarjeta gráfica NVIDIA Quadro RTX 5000 GPU y 64 GB RAM, en su caso, y portátil Legion 5-15ARH05, con procesador AMD Ryzen 7 4800H, tarjeta gráfica GeForce GTX 1650 4GB y 16 GB de RAM, en el caso de este trabajo).

En el caso de Prophesee, el programa rinde a tiempo real con estos recursos, pero no detecta el medallón como un objeto completo y hay bastantes zonas en las que no se calcula el flujo óptico.

4.4.3. Conclusiones

Ambos métodos no están basados en aprendizaje automatizado, lo que les permite funcionar a tiempo real, un aspecto fundamental para un sistema embebido. El método propuesto por Brebion et al. es muy preciso pero necesita de recursos que encarecerían el producto final. En el caso de Prophesee, la precisión es limitada y depende mucho de la escena.

4.4.4. Utilidades

El cálculo del flujo óptico tiene muchísimas aplicaciones. Es un bloque fundamental de la odometría visual [123], reconocimiento de acciones [124], conducción autónoma [125] o navegación de robots [126], por lo que obtener el flujo óptico sin un gran gasto computacional y de manera eficiente produciría una mejora significativa en la integración definitiva de esta tecnología en el ámbito comercial.

5. CONCLUSIONES Y LÍNEAS FUTURAS

El propósito de este Trabajo Fin de Máster era adquirir un conocimiento general de la tecnología de visión basada en eventos tanto a nivel hardware como software.

Para ello se ha hecho un estudio de la evolución histórica de estos sensores, con la idea de conocer el camino que los ha llevado hasta sus características actuales, llegando a la conclusión de que difieren significativamente de sus características iniciales. Los sensores de visión basados en eventos actuales capturan imágenes de forma asíncrona, midiendo los cambios de brillo a nivel de píxel y emitiendo estos como un evento que codifica el tiempo, la ubicación y el signo del cambio en sí. La propia naturaleza de la adquisición de imágenes de estos sensores les permite una serie de ventajas respeto a las cámaras convencionales, como alta resolución temporal, bajo consumo de energía y un gran ancho de banda. Se ha visto que estas características le dan una gran ventaja en diversos campos como en la robótica. No obstante, es esta forma de adquirir imágenes la que plantea la necesidad de adaptar los algoritmos de visión tradicionales al tipo de procesamiento que utilizan estas cámaras para que su uso predomine en la industria.

El hecho de querer entender los distintos algoritmos y aplicaciones de estos sensores ha motivado a la creación de unas librerías en C++ para representar eventos. La forma de representar eventos es muy importante ya que, como se ha comprobado, suele ser el primer paso para la implementación de muchos algoritmos que ejecutan las distintas aplicaciones. Aun así, se ha comprobado que casi todas las aplicaciones están muy lejos de llegar a su máximo potencial y, salvo en casos excepcionales, su implementación en sistemas comerciales es limitada dada la gran cantidad de recursos que necesitan.

De todas las aplicaciones que tienen estos sensores, se seleccionaron tres desarrolladas por distintos grupos de investigación y su rendimiento se comparó con las herramientas propuestas por Prophesee con el kit de desarrollo Gen 3.1. Se observó que la mayoría de los investigadores usan ROS como entorno para el desarrollo de aplicaciones, mientras que Prophesee propone un entorno multihilo que se implementa a través de una librería denominada *Pipeline.* Esto provoca que la integración de gran parte de los algoritmos no sea inmediata con los sensores de esta compañía y requiera de una adaptación de las librerías para funcionar correctamente.

Aparte de lo ya mencionado, a lo largo de este proyecto se ha visto que esta tecnología tiene mucho potencial, pero también se ha hecho frente a muchos problemas que se deben solucionar para que los sensores de visión se terminen de imponer en la industria:

- **Los datos ocupan mucho espacio**. El tamaño de las grabaciones depende del fabricante, pero por lo general pequeñas grabaciones pueden ocupar varios gigabytes de memoria y, en el caso de bases de datos, varios terabytes. Esto

es un inconveniente para aplicaciones como la videovigilancia, además de suponer un problema para el uso de estas bases de datos, ya que no es común encontrar equipos de trabajo con memorias del orden de varios terabytes.

- **Recursos computacionales elevados**. En concordancia con el punto anterior, la mayoría de las aplicaciones que funcionan en tiempo real son para sensores de muy baja resolución por lo que el número de datos a procesar es menor. Cuando la resolución aumenta, el número de datos a procesar aumenta exponencialmente, lo que requiere el uso de más recursos computacionales para ejecutar en tiempo real la aplicación (mayor número de CPUs, aumentar el tamaño de las memorias, etc.) lo que encarecería en exceso un producto comercial.
- **Mercado hermético**. Aunque cada vez más los investigadores se abren a compartir el código fuente con el que han podido desarrollar las distintas aplicaciones, lo cierto es que es complicado acceder a este en la gran mayoría de los casos. Además, las respuestas a dudas sobre la investigación o a la solicitud de ciertos recursos o información a las empresas que comercializan con estos sensores suele ser esquiva y difusa. Por otro lado, los repositorios de código relacionado con el procesado de los eventos solo contienen librerías para tareas muy sencillas. Por lo tanto, la apertura generalizada de este conocimiento impulsaría el desarrollo de esta tecnología.
- **Precio de los kits de desarrollo**. El hecho de que los kits de desarrollo estén disponibles para ser adquiridos ya es un progreso importante, pero, aun así, el hecho de que valgan varios miles de euros es un impedimento para que las universidades o empresas de sistemas embebidos se adentren en esta tecnología para impulsarla.

Como se ha podido comprobar, gracias a sus propiedades los sensores de visión basados en eventos pueden imponerse como la tecnología predominante respecto a la adquisición de información visual para diversos sistemas, pero, tras más de treinta años de desarrollo, aún existen barreras tanto tecnológicas como financieras que deben superar para entrar definitivamente en este mercado.

REFERENCIAS

[1] K. Fukushima et al., "An electronic model of the retina", *Proceedings of the IEEE*, vol. 58, no. 12, pp. 1950-1951, 1970.

[2] M. Mahowald, "VLSI analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, California Institute of Technology, Pasadena, California, 1992.

[3] M. Gonzalez. (s.f). Las 11 partes del ojo humano (y sus características). *EstiloNext*. Recuperado el 10 de marzo de 2022. https://estilonext.com/salud-y-bienestar/partes-ojo-humano.

[4] S. Ramón y Cajal, "La rétine des vertébrés", 1893.

[5] S.W. Kuffler, "Discharge patterns and functional organization of mammalian retina". *J Neurophysiol*. 1953.

[6] R. Masland, "The fundamental plan of the retina", *Nat Neurosci*, vol. 4, pp. 877–886, 2001.

[7] P. R. Martin, "Colour processing in the primate retina: recent progress". *J Physiol*, 1998.

[8] F. S. Werblin, J. E. Dowling, "Organization of the retina of the mudpuppy, Necturus maculosus. II. Intracellular recording", *J Neurophysiol*, 1969.

[9] M. Mahowald, "M. Mahowald, "An Analog VLSI System for Stereoscopic Vision", 1994", 1994.

[10] S. Liu et al. "Event-Based Neuromorphic Systems", 2015.

[11] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip I: Outer and inner retina models", *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 4, pp. 657-666, 2004.

[12] P. -. Ruedi et al., "A 128 × 128 pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction", *IEEE Journal of Solid-State Circuits*, vol. 38, no. 12, pp. 2325-2333, 2003.

[13] E. Culurciello, R. Etienne-Cummings and K. A. Boahen, "A biomorphic digital image sensor", *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 281-294, 2003.

[14] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[15] C. Posch, D. Matolin and R. Wohlgenannt, "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS," in IEEE Journal of Solid-State Circuits, vol. 46, no. 1, pp. 259-275, 2011.

[16] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[17] Event Camera. (27 de septiembre de 2021). En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Event_camera&oldid=1046720027.

[18] J.A. Leñero et al., "Pipeline AER arbitration with event aging", *IEEE International Symposium on Circuits and Systems (ISCAS) (2017), p 8050977-*, 2017.

[19] E. Mueggler, B. Huber and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers", *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014

[20] G. Gallego et al., "Event-based Vision: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154-180, 2022.

[21] M. Silvilotti, "Wiring considerations in analog VLSI systems with application to field programmable networks," Ph.D. dissertation, Cal. Inst. of Tech., Pasadena, California, 1991

[22] A. Mortara, E. Vittoz, P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, 1995.

[23] P. Häflige et al., "Neuromorphic Electronics: Lecture Notes", *Compedium for INF5470M*, 2012.

[24] R. Vogelstein et al., "Spike Timing-Dependent Plasticity in the Address Domain", *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002.

[25] J. A. Leñero-Bardallo, T. Serrano-Gotarredona and B. Linares-Barranco, "A 3.6µs latency asynchronous frame-free event-driven dynamic-vision-sensor," *IEEE Journal of Solid-State Circuits*, vol. 46, No. 6, pp. 1443-55, 2011.

[26] (s.n). (s.f). Sensor. *Prophesee.* Recuperado el 12 de julio de 2022. https://www.prophesee.ai/event-based-sensor-packaged/

[27] (s.n). (s.f). Specification – Current models. *Inivation*. Recuperado el 12 de julio de 2022. https://inivation.com/wp-content/uploads/2021/08/2021-08-iniVation-devices-Specifications.pdf

[28] Y. Suh et al., "A 1280×960 Dynamic Vision Sensor with a 4.95-µm Pixel Pitch and Motion Artifact Minimization", *2020 IEEE International Symposium on Circuits and Systems (ISCAS),* 2020.

[29] R. Benosman, "Event Computer Vision 10 years Assessment: Where We Came From, Where We Are and Where We Are Heading To", *Third International Workshop on Event-based Vision*. 2021.

[30] S. R. Marcelo F., Display-Events (2022), *Repositorio de GitHub*, https://github.com/marcelosr97/Display-Events

[31] S. Guo et al., "A Noise Filter for Dynamic Vision Sensors based on Global Space and Time Information*", arXiv*, 2020.

[32] A. Zihao, Y. Chen, K. Daniilidis, "Realtime time synchronized event-based stereo", *Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[33] H. Rebecq, T. Horstschaefer, D. Scaramuzza, "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization"*, British Machine Vision Conf. (BMVC),* 2017.

[34] A. I. Maqueda et al., “Event-based vision meets deep learning on steering prediction for self-driving cars”, *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 5419–5427, 2018.

[35] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems (RSS)*, 2018.

[36] S. Mostafavi et al., “Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks”, *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR),* 2019.

[37] D. Gehrig et al., “EKLT: Asynchronous photometric feature tracking using events and frames”, *Int. J. Comput. Vis.*, 2019.

[38] X. Lagorce et al, "HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346-1359, 2017.

[39] Z. Ni et al., “Asynchronous event-based high-speed vision for microparticle tracking”, *J. Microscopy*, vol. 245, no. 3, pp. 236–244, 2012.

[40] J. Conradt et al., “A pencil balancing robot using a pair of AER dynamic vision sensors,” in *IEEE Int. Symp. Circuits Syst. (ISCAS),* 2009, pp. 781–784.

[41] A. Mondal et al., “Moving Object Detection for Event-based Vision using Graph Spectral Clustering”*, IEEE Int. Conf. Computer Vision Workshop (ICCVW)*, 2021.

[42] Liu, Z., Fu, Y., “e-ACJ: Accurate Junction Extraction for Event Cameras”, *arXiv*, 2021.

[43] Y. Dong, T. Zhang, “Standard and Event Cameras Fusion for Feature Tracking”, *Int. Conf. on Machine Vision and Applications (ICMVA)*, 2021.

[44] B. K. P. Horn, B. G. Schunck, “Determining optical flow”, *Artif. Intell.*, vol. 17, nos. 1–3, pp. 185–203, 1980.

[45] J. L. Barron, N. A. Thacker, “Tutorial: Computing 2D and 3D optical flow”, *Imag. Sci. Biomed. Eng. Division*, Med. School, Univ. Manchester, Manchester, U.K., Tech. Rep. 2004, pp. 1–12, 2005.

[46] R. de la Rosa-Vidal et al., "A mobile platform for movement tracking based on a fast-execution-time optical-flow algorithm", *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[47] R. Benosman et al., “Asynchronous frameless event-based optical flow,” *Neural Netw.*, vol. 27, pp. 32–37, 2012.

[48] R. Benosman et al., “Event-based visual flow,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, 2014.

[49] T. Brosch, S. Tschechne, H. Neumann, “On event-based optical flow detection,” *Front. Neurosci.*, vol. 9, Apr. 2015.

[50] M. Gehrig et al., "Event-Based Angular Velocity Regression with Spiking Networks”, *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[51] M. Gehrig et al., “E-RAFT: Dense Optical Flow from Event Cameras”, *IEEE Int. Conf. on 3D Vision (3DV)*, 2021.

[52] V. Brebion, J. Moreau, F. Davoine, "Real-Time Optical Flow for Vehicular Perception with Low- and High-Resolution Event Cameras", *IEEE Trans. Intell. Transp. Syst. (T-ITS)*, 2021

[53] Z. Wan et al., "Stereo Hybrid Event-Frame (SHEF) Cameras for 3D Perception", *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021

[54] S. H. Ahmed et al., "Deep Event Stereo Leveraged by Event-to-Image Translation", *AAAI Conf. Artificial Intelligence*, 2021.

[55] S. Tulyakov et al., "Learning an event sequence embedding for dense event-based deep stereo", *IEEE Int. Conf. Computer Vision (ICCV)*, 2019.

[56] S. Schraml, A. N. Belbachir, H. Bischof, "An event-drivenstereo system for real-time 3-D 360 panoramic vision", *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 418–428, 2016.

[57] S. Schraml, A. N. Belbachir, H. Bischof, "Event-Driven Stereo Matching for Real-Time 3D Panoramic Vision", *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015.

[58] D. Gehrig et al., "Combining Events and Frames Using Recurrent Asynchronous Multimodal Networks for Monocular Depth Prediction", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2822-2829, 2021.

[59] J. Hidalgo-Carrió, D. Gehrig, D. Scaramuzza, "Learning Monocular Dense Depth from Events", *IEEE Int. Conf. on 3D Vision (3DV)*, 2020.

[60] J.N.P Martel, et al., "An Active Approach to Solving the Stereo Matching Problem using Event-Based Sensors", *IEEE Int. Symp. Circuits and Systems (ISCAS)*, 2018.

[61] D. Weikersdorfer, J. Conradt, "Event-based particle filtering for robot self-localization", *IEEE Int. Conf. Robot. Biomimet*ics (ROBIO), 2012.

[62] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 632–639, 2017.

[63] S. Bryner et al., "Eventbased, direct camera tracking from a photometric 3D map using nonlinear optimization," *IEEE Int. Conf. Robot. Autom.* (ICRA), 2019.

[64] W. Chamorro, J. Andrade-Cetto, J. Solà, "High-Speed Event Camera Tracking", *British Machine Vision Conf.* (BMVC), 2020.

[65] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous localization and mapping for event-based vision systems", *Int. Conf. Comput. Vis. Syst. (ICVS)*, pp. 133–142, 2013.

[66] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor", *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 359– 364. 2014.

[67] H. Kim et al., "Simultaneous mosaicing and tracking with an event camera", *British Mach. Vis. Conf. (BMVC)*, 2014.

[68] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," in *IEEE Int. Conf. Comput. Photography (ICCP)*, pp. 359–364.2017.

[69] H. Kim, S. Leutenegger, A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera", *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[70] H. Rebecq et al., “EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time”, IEEE Robot. Autom. Lett., vol. 2, no. 2, pp. 593–600, 2017.

[71] A. R. Vidal et al., "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios”, *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994-1001, 2018.

[72] A. Zhu, N. Atanasov, K. Daniilidis, “Event-based Visual Inertial Odometry”, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.

[73] E. Mueggler et al., “Continuous-time visual-inertial odometry for event cameras,” *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1425–1440, 2018.

[74] K. Xiao et al., “Research on Event Accumulator Settings for Event-Based SLAM”, *arXiv*, 2021.

[75] Y. Zhou, G. Gallego, S. Shen, “Event-based Stereo Visual Odometry”, *IEEE Trans. Robot. (TRO)*, 2021.

[76] C. Scheerlinck, N. Barnes, R. Mahony, “Continuous-time intensity estimation using event cameras”, *Asian Conf. Comput.Vis. (ACCV)*, 2018.

[77] H. C. Duwek, A. Shalumov, E. E. Tsur, "Image Reconstruction from Neuromorphic Event Cameras using Laplacian-Prediction and Poisson Integration with Spiking and Artificial Neural Networks”, *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[78] F. Paredes-Valles, G. de Croon, G. C. H. E., “Back to Event Basics: Self-Supervised Learning of Image Reconstruction for Event Cameras via Photometric Constancy”, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021.

[79] X. Zhang et al., “Event-Based Synthetic Aperture Imaging with a Hybrid Network”, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021.

[80] Y. Zou et al., “Learning to Reconstruct High Speed and High Dynamic Range Videos from Events”, *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021.

[81] Y. Zhou et al., “Event-based Motion Segmentation with Spatio-Temporal Graph Cuts”, IEEE Trans. Neural Netw. Learn. Syst. (TNNLS), 2021.

[82] C. M. Parameshwara et al., “MOMS with Events: Multi-Object Motion Segmentation with Monocular Event Cameras”, *arXiv*, 2020.

[83] A. Mitrokhin et al., “Learning visual motion segmentation using event surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”*, 2020.

[84] T. Delbruck, P. Lichtsteiner, “Fast sensory motor control based on event-based hybrid neuromorphic-procedural system”, *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 845–848, 2007.

[85] R. Serrano-Gotarredona et al., "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*, vol. 20, no. 9, pp. 1417-1438, 2009.

[86] J. A. Perez-Carrasco et al., “Mapping from frame-driven to frame-free event-driven vision systems by low-rate coding and coincidence processing–application to feedforward ConvNets”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–2719, 2013.

[87] A. Sironi et al., “HATS: Histograms of averaged time surfaces for robust eventbased object classification”, *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018

[88] Y. Bi et al., “Graph-based object classification for neuromorphic vision sensing”, *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 491–501, 2019.

[89] Y. Li et al., “Graph-based Asynchronous Event Processing for Rapid Object Recognition”, *IEEE Int. Conf. Computer Vision (ICCV)*, 2021.

[90] A. Ayyad et al., “Neuromorphic Vision Based Control for the Precise Positioning of Robotic Drilling Systems”*, arXiv*, 2021

[91] A. Vitale et al., "Event driven Vision and Control for UAVs on a Neuromorphic Chip”, *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 103-109, 2021.

[92] R. Ghosh et al., “Spatiotemporal Filtering for Event-Based Action Recognition*”, arXiv*, 2019.

[93] Z. Wang, Y. Ng, C. Scheerlinck and R. Mahony, "An Asynchronous Kalman Filter for Hybrid Event Cameras”, *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[94] C. Gu et al., "The Spatio-Temporal Poisson Point Process: A Simple Model for the Alignment of Event Camera Data”, *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[95] Y. Zhou, G. Gallego and S. Shen, "Event-Based Stereo Visual Odometry," in *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1433-1450, 2021.

[96] R. Kreiser et al., “An On-chip Spiking Neural Network for Estimation of the Head Pose of the iCub Robot”, *Front. Neurosci.*, 2020.

[97] F. Paredes-Valles et al., “Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks”, *arXiv*, 2021.

[98] Y. Wang et al., “Event-Stream Representation for Human Gaits Identification Using Deep Neural Networks”, *IEEE Trans. Pattern Anal. Machine Intell. (TPAMI)*, 2021.

[99] A. Samadzadeh et al., “Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction”, *arXiv*, 2020.

[100] A. I. Maqueda et al., “Event-based vision meets deep learning on steering prediction for self-driving cars”, *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018.

[101] A. Nguyen et al., “Realtime 6DOF pose relocalization for event cameras with stacked spatial LSTM networks”, *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2019.

[102] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection”, British Mach. Vis. Conf. (BMVC), 2017.

[103] G. Chen et al., "NeuroAED: Towards Efficient Abnormal Event Detection in Visual Surveillance with Neuromorphic Vision Sensor", *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 923-936, 2021.

[104] J. Binas, D. Neil, S.-C. Liu, T. Delbruck, "DDD17: End-to-end DAVIS driving dataset", *ICML Workshop on Machine Learning for Autonomous Vehicles*, 2017.

[105] C. Ye et al., "Unsupervised learning of dense optical flow and depth from sparse event data," *arXiv e-prints*, 2018.

[106] A. Z. Zhu et al., "Unsupervised event-based learning of optical flow, depth, and egomotion", *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019.

[107] M. Osswald et al., "A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems", *Scientific Reports*, vol. 7, 2017

[108] G. Dikov et al., "Spiking cooperative stereo-matching at 2ms latency with neuromorphic hardware", *Conf. Biomimetic and Biohybrid Systems*, pp. 119–137, 2017.

[109] V. Vasco et al., "Vergence control with a neuromorphic iCub", *IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, 2016.

[110] Luminar Technologies. (2018). *Dynamic vision sensor to direct lidar scanning*, https://patents.justia.com/patent/10516876, EEUU.

[111] Volkswagen AG. (2019). *Digital Neuromorphic (nm) Sensor Array, Detector, Engine and Methodologies,* https://patents.justia.com/patent/10387741, EEUU.

[112] (s.n). (25 de febrero de 2021). Terranet AB Demonstrates Ultra-Fast VoxelFlow™ Sensor at STARTUP AUTOBAHN in joint Presentation with Mercedes-Benz. *Automotive World*. Recuperado el 12 de julio de 2022. https://www.automotiveworld.com/news-releases/terranet-ab-demonstrates-ultra-fast-voxelflow-sensor-at-startup-autobahn-in-joint-presentation-with-mercedes-benz/

[113] (s.n). (s.f). SmarThings Vision. *Samsung*. Recuperado el 12 de julio de 2022. https://www.samsung.com/se/smartthings/camera/smart-things-vision-gp-u999gteeaea/

[114] A. Censi et al., "Low-latency localization by active LED markers tracking using a dynamic vision sensor," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[115] G. Chen et al., "A Novel Illumination-Robust Hand Gesture Recognition System With Event-Based Neuromorphic Vision Sensor", *IEEE Transactions on Automation Science and Engineering*, vol.18, no.2, pp.508-520, 2021.

[116] H. Asgari et al., "Digital Multiplier-Less Spiking Neural Network Architecture of Reinforcement Learning in a Context-Dependent Task", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol.10, no.4, pp.498-511, 2020.

[117] Alzugaray, M. Chli, "Asynchronous Corner Detection and Tracking for Event Cameras in Real Time", *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[118] E. Rosten, T. Drummond, "Machine learning for high-speed corner detection", Eur. Conf. Comput. Vis. (ECCV), pp. 430–443, 2006.

[119] E. Mueggler et al. "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM". *Int. J. Robot. Research*, 2017.

[120] V. Brebion, J. Moreau, F. Davoine, "Real-Time Optical Flow for Vehicular Perception with Low- and High-Resolution Event Cameras", *IEEE Trans. Intell. Transp. Syst. (T-ITS)*, 2021.

[121] M. Almatrafi et al. "Distance surface for event-based optical flow", *IEEE TPAMI*, vol. 42, pp. 1547– 1556, 2020.

[122] J. Adarve, R. Mahony, "A filter formulation for computing real time optical flow", *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, pp. 1192–1199, 2016.

[123] T. Qin, P. Li, S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator", *IEEE Transactions on Robotics*, 2018.

[124] K. Simonyan, A. Zisserman, "Two-stream convolutional networks for action recognition in videos", *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2014.

[125] J. Janai et al., "Computer vision for autonomous vehicles: Problems, datasets and state of the art". Foundations and Trends® in Computer Graphics and Vision, 2020.

[126] K. McGuir, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1070–1076, 2017.