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

Estudio de la técnica de adquisición de datos Quanta imaging en sensores de imagen asíncronos / Study of the data acquisition technique Quanta imaging with asynchronous image sensors

Roberto José Méndez Romero
Juan Antonio Leñero Bardallo
6 de septiembre de 2021

Declaración personal de autoría

Roberto José Méndez Romero con DNI 49566090D estudiante del Máster Universitario en Microelectrónica: Diseño y Aplicaciones de Sistemas Micro/Nanométricos de la Universidad de Sevilla, como autor de este documento académico titulado *Estudio de la técnica de adquisición de datos Quanta imaging en sensores de imagen asíncronos* y presentado como Trabajo de Fin de Máster

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Sevilla, a 6 de septiembre de 2021

Fdo: Roberto José Méndez Romero

Resumen

En este Trabajo de Fin de Máster se realiza un estudio de viabilidad sobre la aplicación de la técnica de adquisición de datos *Quanta Imaging* en sensores de luminancia asíncronos o de tipo *Octopus*. Para ello, se comienza presentando las características de estos sensores y también la de los sensores que ponen nombre a la técnica, conocidos como *Quanta Imaging Sensors*. Por otra parte, se exponen los principios en los que se basa la técnica y las ventajas que se podrían derivar de su uso.

Posteriormente, se detallan los dos modelos que se han desarrollado para simular la adquisición y generación de imágenes mediante un sensor de luminancia asíncrono. El primero, basado en una arquitectura reportada en la bibliografía, se utilizará como referencia. El segundo, que añade al anterior la implementación de la técnica *Quanta Imaging*, servirá como objeto de estudio del trabajo. Por otro lado, se explican las métricas que se han definido para comparar las imágenes generadas por cada modelo.

Puesto que la técnica *Quanta Imaging* requiere del uso de ventanas temporales de exposición, se definen una serie de parámetros para generar diferentes tipos y se caracterizan para analizar los artefactos que producirían sobre las imágenes generadas.

Por último, la validación de la técnica se realiza en dos pasos. En primer lugar, se compara numérica y visualmente los resultados obtenidos por simulación a partir de una batería de imágenes reales para distintas tasas de adquisición. En segundo lugar, se reconstruye una imagen aplicando la técnica de *Quanta Imaging* con datos experimentales obtenidos por un sensor de luminancia asíncrono real en un entorno de laboratorio. Como resultado, se llega a la conclusión de que la técnica cumple su cometido y resulta beneficiosa en este tipo de sensores.

Abstract

In this Master's Thesis, a feasibility study on the application of the Quanta Imaging data acquisition technique in asynchronous or Octopus type luminance sensors is carried out. To this end, the characteristics of these sensors are presented, as well as those of the sensors that give name to the technique, known as *Quanta Imaging Sensors*. The principles on which the technique is based and the advantages that could be derived from its use are presented.

Subsequently, the two models that have been developed to simulate the acquisition and generation of images using an asynchronous luminance sensor are detailed. The first one, based on an architecture reported in the literature, will be used as a reference. The second one, which adds to the previous one the implementation of the Quanta Imaging technique, will serve as the object of study of this work. The metrics that have been defined to compare the images generated by each model are explained.

Since the Quanta Imaging technique requires the use of exposure temporal windows, a series of parameters are defined to generate different types and characterization in terms of these parameters is performed to analyze the artifacts they would produce on the generated images.

Finally, the validation of the technique is performed in two steps. Firstly, we compare numerically and visually the results obtained by simulation from a battery of real images for different acquisition rates, and secondly, an image is reconstructed by applying the Quanta Imaging technique with experimental data obtained by a real asynchronous luminance sensor in a laboratory environment. As a result, it is concluded that the technique fulfils its purpose and it is beneficial for this type of sensors.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se presentan las características principales y el principio de funcionamiento de las dos familias de sensores en las que se basa la realización de este trabajo. Por un lado, la familia de los *Asynchronous Image Sensors*, y por otro la de los *Quanta Image Sensors*. También, se expone el problema principal detectado en la bibliografía y la solución que se propone a este. Finalmente, se detallan las razones que motivan la realización de este trabajo y se describen los objetivos que se persiguen en el mismo.

1.1. Estado del arte

1.1.1. *Asynchronous Image Sensors*

El desarrollo de sensores de visión asíncronos basados en eventos (también conocidos como *"event-driven"*) ha aumentando significativamente desde que se publicara a principios de los años noventa el primer sensor integrado asíncrono bioinspirado [1]. Así, a lo largo de los años han ido apareciendo diferentes tipos de sensores que pueden ser clasificados en tres categorías principales atendiendo a la información que procesan de la escena visual [2].

En la primera, se incluyen los sensores que detectan el contraste espacial (*CD*, del inglés *"Contrast Detection"*) dentro de la escena, cuya primera implementación fue realizada en 1991 por Mahowald [1]. Estos sensores tratan de emular el procesamiento temprano de la escena visual que realiza la retina humana, de ahí que en la literatura se conozcan como retinas de silicio. Aunque actualmente el cálculo del contraste espacial basado en eventos no es muy competitivo, debido a que requiere operaciones no lineales que son difíciles de implementar con precisión en el dominio analógico, estos sensores fueron importantes ya que motivaron el desarrollo de los sensores de visión basados en eventos englobados en las categorías que siguen a esta primera.

La segunda categoría, corresponde a los sensores de visión dinámica (*DVS*, del inglés "*Dynamic Vision Sensors*"). El primer sensor de visión de tipo *DVS* funcional fue publicado en 2008 por Lichtsteiner [3]. Los píxeles de este tipo de sensores se caracterizan por disparar solamente cuando detectan variaciones transitorias en la iluminación, lo que les permite ser capaces de rastrear objetos que se mueven muy rápido con un consumo de energía y ancho de banda muy reducidos. De acuerdo a [4], los *DVS* responden de forma autónoma a los cambios relativos de intensidad con una resolución temporal del orden de microsegundos a lo largo de seis décadas de iluminación. Por ello, su uso se ha extendido a muchos campos de aplicación como son el seguimiento y detección de partículas en fluidos [5], la monitorización del tráfico de vehículos [6] y el entrenamiento de redes neuronales [7, 8], entre otros.

Por último, a la tercera categoría pertenecen los sensores tomados como objeto de estudio en este trabajo y que en la literatura se conocen como *Spiking Luminance Sensors* (en español, "Sensores de luminancia") u *Octopus Retinas*. En este caso, fue Culurciello quien en 2003 propuso el primer sensor de este tipo [9]. Estos sensores se caracterizan por realizar una conversión de luz a frecuencia para codificar los niveles de iluminación. Su principio de operación es el que se muestra en la figura 1.1. En primer lugar, mediante el transistor M_{p1} se realiza un *reset* global de la matriz cargando el condensador C_{ph} de cada píxel a la tensión V_{DD}. Una vez realizado el *reset*, debido a la fotocorriente (I_{ph}) dicho condensador comienza a descargarse, de modo que su tensión disminuye aproximadamente de manera lineal con una pendiente proporcional a I_{ph}.

Cada vez que la tensión en el nodo V_{ph} alcanza un valor igual a la tensión de referencia V_{bot} y el comparador dispara, se envía un evento fuera del *chip* con las coordenadas del píxel que ha alcanzado el valor de tensión V_{bot}. Inmediatamente después, el píxel se auto-resetea a través del transistor M_{p2} y comienza a integrar la carga de nuevo. El envío del evento al exterior se realiza mediante un circuito que implementa un protocolo de comunicación conocido como *AER* (del inglés *"Address Event Representation"*) y que será descrito más adelante.

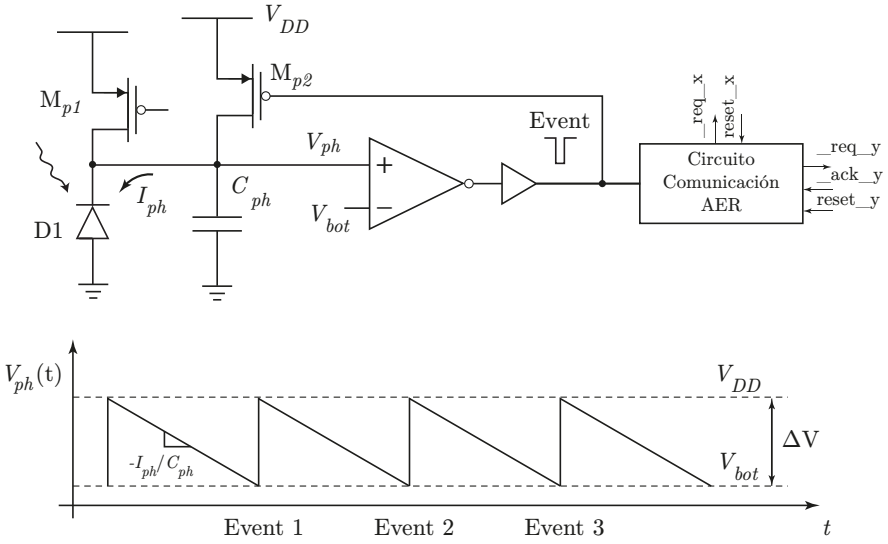


FIGURA 1.1: Oscilador astable para realizar una conversión de luz a frecuencia.

De acuerdo a [10, 11], la frecuencia de oscilación de los píxeles se puede aproximar mediante la siguiente expresión:

$$f_{osc} \approx \frac{I_{ph}}{C_{ph} \cdot (V_{DD} - V_{bot})} \tag{1.1}$$

Estas frecuencias de los pulsos que emiten los píxeles pueden abarcar más de cinco décadas [9, 12], lo que supone la capacidad de detectar niveles de iluminación dentro de la escena con un rango dinámico (*DR*, del inglés "*Dynamic Range*") superior a 100 dB [2].

A pesar de que estos sensores no implementan ningún tipo de procesamiento, resultan muy atractivos en aplicaciones donde se requiere detectar los niveles de iluminación de ciertas regiones con un consumo bajo de recursos. Además, tienen ventajas inherentes sobre los sensores de imagen convencionales basados en fotogramas. La primera es la velocidad, ya que la frecuencia a la que pueden pulsar los píxeles alcanza fácilmente tasas del orden de los kilohercios en entornos interiores [9,10]. Esto implica una resolución temporal equivalente del orden de los kiloframes por segundo. La segunda ventaja es su funcionamiento autónomo. Los píxeles no necesitan ser escaneados periódicamente para medir sus valores de iluminación, en su lugar, emiten pulsos continuamente siempre que están iluminados. En caso contrario, no producen ningún evento. Esto también es una ventaja para las aplicaciones en las que los píxeles oscuros no tienen sentido. Por otra parte, su consumo de energía se reduce con respecto a los sensores basados en fotogramas. Algunos ejemplos de aplicaciones en las que han tenido éxito se pueden encontrar en las referencias [13–15], donde son propuestos un sensor capaz de detectar la posición del sol, un sensor para monitorizar llamas y un sensor para la detección rápida de colores, respectivamente.

Sin embargo, tal y como se ha reportado en la bibliografía [10, 16] el rendimiento de estos sensores está limitado por la circuitería lógica requerida para implementar el protocolo de comunicación *AER* que permite mandar al exterior los eventos que producen los píxeles. Para comprender la naturaleza del problema, se tomará de referencia el núcleo común de las arquitecturas de sensor propuestas por Leñero et al. en [10, 11] y que se muestra en la figura 1.2.

En el centro, se encuentra la matriz de píxeles del sensor. En la parte superior y en el lado derecho, está situada la lógica asíncrona necesaria para implementar el protocolo de comunicación *AER*. En este caso particular, los emisores serían los diferentes píxeles que comparten el bus *AER*, mientras que el receptor consistiría en un periférico externo (FPGA) encargado de manejar las señales de comunicación del protocolo, de almacenar las direcciones (coordenadas X e Y) de los píxeles que han producido un evento en una memoria SDRAM y de enviar/recibir datos desde un PC para la depuración del sistema y la representación de datos en tiempo real.

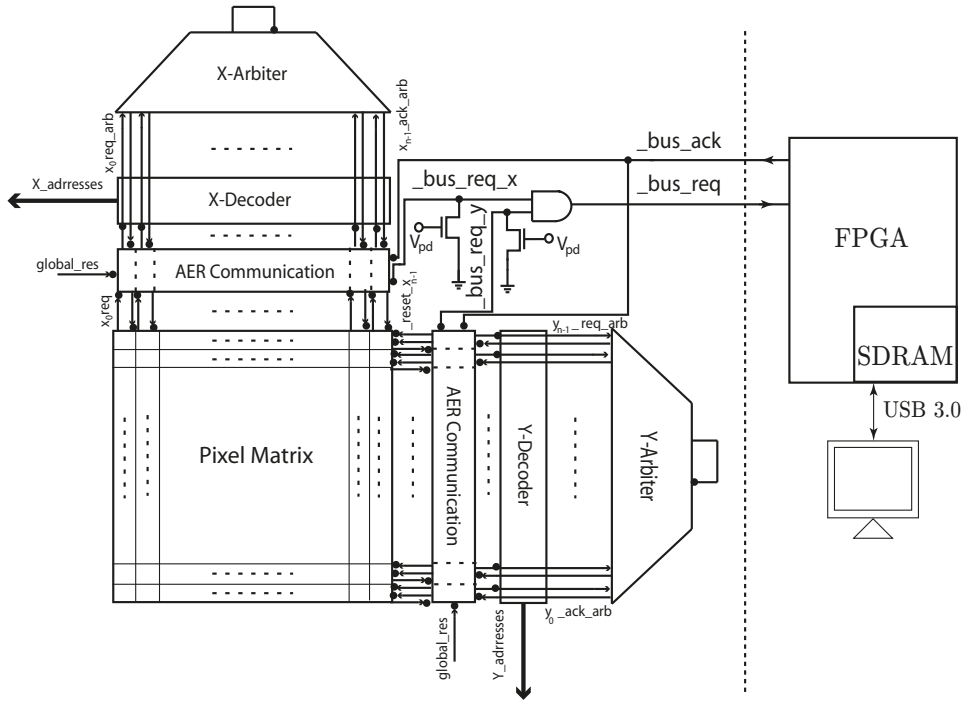


FIGURA 1.2: Arquitectura del sensor a nivel de sistema.

En la figura 1.3, se representan las señales de comunicación intercambiadas entre dos dispositivos que utilizan el protocolo *AER* para comunicarse. Cada vez que un píxel produce un evento de manera asíncrona, el sensor solicita acceso a un bus compartido. Para ello, se activa la señal REQ y se envían las coordenadas del píxel a través del bus. El periférico externo, almacenará la dirección del píxel que se ha disparado y activará la señal ACK para indicar que ha recibido la dirección. Cada vez que se reciba un evento, el número de eventos asociado al píxel se incrementa una unidad. Tras el reconocimiento, el píxel desactiva la señal REQ y retira su dirección del bus. Detalles específicos sobre la lógica AER utilizada en las arquitecturas propuestas por Leñero et al. se pueden encontrar en [17]. Para una explicación detallada de esquemas de comunicación AER más modernos se pueden consultar las referencias [15, 18].

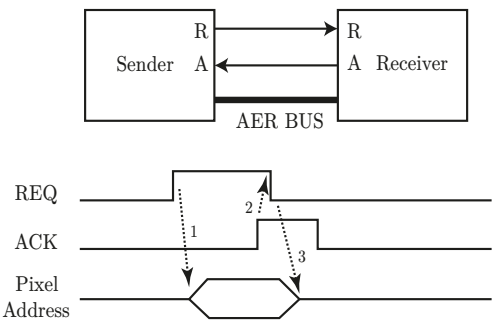


FIGURA 1.3: Esquema de comunicación AER punto a punto entre un emisor y un receptor.

Dado que toda la matriz comparte un mismo bus, es necesario un mecanismo de control que gestione las colisiones en caso de que haya congestión en el bus *AER*, es decir, que varios píxeles intenten enviar información simultáneamente. En tal situación, dichos píxeles tendrán que esperar a tener acceso al bus para poder enviar sus direcciones fuera del *chip*. El bloque encargado de dar acceso al bus y de evitar las colisiones entre píxeles que disparan simultáneamente es el sistema de arbitraje [18, 19] (*X-Arbiter* e *Y-Arbiter* en figura 1.2). Estos bloques reciben peticiones de las filas y columnas que corresponden a los píxeles que han producido un evento, y tras un proceso de arbitraje conceden acceso al bus a una única petición para que solamente un píxel pueda transmitir su dirección al exterior. En la implementación de Leñero et al, dicho proceso consta de dos pasos. En primer lugar, se arbitran las peticiones por fila con los circuitos periféricos del lado derecho. A continuación, se arbitran las peticiones por columnas. Al final, sólo la fila y columna de un determinado píxel tendrán acceso al bus compartido. Por último, se envía la señal externa REQ (*_bus_req* en figura 1.2) y la dirección del píxel fuera del chip, hasta que se recibe la señal ACK (*_bus_ack* en figura 1.2). A partir de ese instante, se atiende el siguiente evento. Entre la lógica *AER* y los bloques de arbitraje, existe un multiplexor para codificar las coordenadas X e Y de los píxeles que han transmitido un evento fuera de chip. El sistema de arbitraje utilizado en estos sensores fue propuesto por Häfliger [17] y ha sido utilizado en implementaciones anteriores de sensores de visión basados en eventos, pudiendo manejar tasas de eventos de entorno 10 Meps para píxeles de la misma fila, y 2 Meps para píxeles de diferentes filas [12, 15].

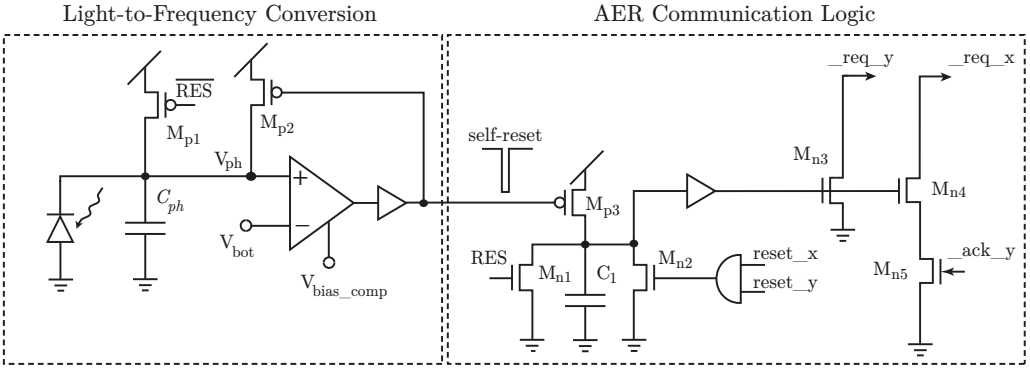


FIGURA 1.4: Esquemático del píxel propuesto.

La figura 1.4 muestra la arquitectura del sensor a nivel de píxel [11]. A la izquierda, se encuentra el oscilador astable encargado de realizar la conversión luz a frecuencia. A la derecha, la lógica asíncrona necesaria para la transmisión de los eventos fuera del *chip* mediante el protocolo AER (ver figura 1.2). Las entradas/salidas digitales son compartidas entre las filas de píxeles (Y) y las columnas de píxeles (X).

El funcionamiento global del sensor aparece representado en la figura 1.5. Inicialmente, todos los píxeles se reinician simultáneamente mediante la señal de control RES. A continuación, integran carga durante un tiempo T_{int}. Durante este período, los píxeles cuya tensión en el nodo de la capacidad de integración alcancen el valor V_{bot} enviarán eventos a través del bus *AER* compartido. El tiempo total para generar una imagen aparece indicado como T_{frame}.

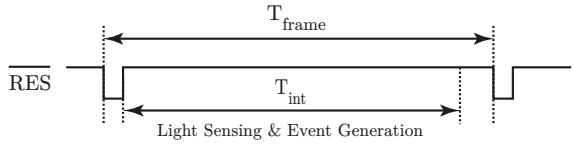


FIGURA 1.5: Cronograma del funcionamiento global.

En la figura 1.6 se muestra un cronograma de las señales digitales que intervienen en la comunicación asíncrona cuando un píxel necesita transmitir un evento fuera del *chip* con las coordenadas de su dirección. En la parte superior, se han representando las señales internas de la lógica *AER* durante el proceso de arbitraje. Tal y como se explicó anteriormente, en primer lugar se arbitran las filas de píxeles y posteriormente las columnas. En la parte inferior, se muestran las señales externas del protocolo de comunicación *AER* descrito en la figura 1.3. Una vez que el píxel obtiene el acceso al bus *AER*, envía su dirección hacia el exterior.

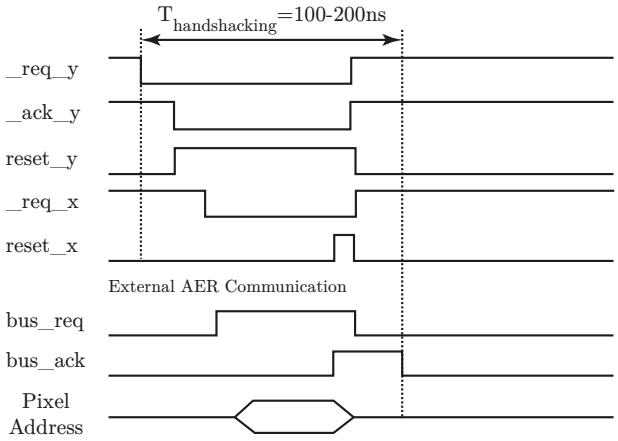


FIGURA 1.6: Cronograma de las señales que intervienen en la comunicación de eventos durante T_{int}.

El tiempo necesario para transmitir un evento ($T_{handshacking}$) se encuentran entorno a los 100-200 ns [10], lo cual es despreciable frente a las frecuencias de los píxeles que se encuentran en el orden de los *kHz*. Sin embargo, si la tasa de eventos supera el rendimiento del bus *AER*, este tiempo comienza aumentar y puede llegar a ser comparable al período de los píxeles. En tal situación, el sistema de arbitraje no es capaz de atender todos los eventos que recibe y comienza a saturar. Esta saturación es un gran problema ya que produce la pérdida de nuevos eventos, y en consecuencia, una degradación de la información que se adquiere de la escena visual.

Una posible solución que se propone en este trabajo para solventar la limitación que impone la tasa máxima de eventos que puede atender el sistema de arbitración, consistiría en adquirir un flujo de datos reducidos tal y como ocurre en los sensores conocidos como *Quanta Image Sensors*.

1.1.2. *Quanta Image Sensor*

Tras la invención del sensor de imagen CMOS, los píxeles han seguido en general la tendencia de escalado del proceso CMOS llegando a reducir su tamaño en más de 30 veces hasta el rango actual de 1-2μm, que se encuentra en el régimen *SDL* (del inglés "*Sub-Diffraction Limit*") [20]. De acuerdo a [21], los píxeles *SDL* no pueden mejorar la resolución efectiva de la imagen, y suelen dar lugar a una calidad de imagen inferior debido a un rango dinámico menor y a una peor relación señal/ruido (SNR) en condiciones de baja iluminación. El concepto *Quanta Image Sensor* (*QIS*) fue introducido en 2005 por Eric Fossum [21–23] como un cambio de paradigma para mejorar la flexibilidad de la captura de imágenes y mitigar los efectos asociados a la reducción del tamaño de los píxeles. Así, estos sensores fueron ideados para utilizar un gran número de píxeles especializados basados en unos fotodetectores binarios, de tamaños por debajo del límite de subdifracción (<900nm) y alta ganancia de conversión, conocidos como "*jots*" (del griego "lo más pequeño") que permitirían la detección individual de fotoelectrones [24, 25].

Para lograr detectar los fotoelectrones de manera individual, el nivel de ruido de un "*jots*" tendría que ser de 0.15e- r.m.s o inferior de acuerdo a [26,27]. Al ser sensible a un único fotoelectrón y tener un *pitch* pequeño (500 nm en [28]), el *QIS* tiene potencial para lograr una alta resolución espacial (cámaras de giga-píxeles [29]), una alta tasa de fotogramas (103 fotogramas/seg. [30]) y un alto rango dinámico [27, 31]. En un *QIS*, la obtención de una imagen se consigue utilizando una ventana temporal de exposición (periodo de integración) durante la que cada píxel puede ser alcanzado por uno o más fotoelectrones. Una vez finalizada la ventana, se lee el estado de cada píxel. Así, si durante el tiempo que dicha ventana se encuentra activa un píxel es alcanzado por al menos un fotoelectrón su valor se asocia con un "1" lógico, mientras que si no se produce la detección de ningún fotoelectrón su valor es asociado a un "0" lógico. A continuación, los píxeles se resetean y se vuelven a exponer durante una nueva ventana temporal, normalmente de igual duración que la anterior. Representando los estados de los píxeles como un plano binario, son necesarios al menos $2^N - 1$ planos para generar una imagen de N bits, lo que da lugar a un cubo de bits (x, y, t) como el que se muestra en la figura 1.7 [32]. Nótese que cada plano implica una nueva ventana temporal de exposición. Finalmente, para reconstruir la imagen el método más simple consiste en codificar el nivel de iluminación de cada píxel como la suma total de sus valores lógicos en cada uno de los planos binarios, esto es, realizando una suma sobre la tercera dimensión del cubo para cada píxel [33–35].

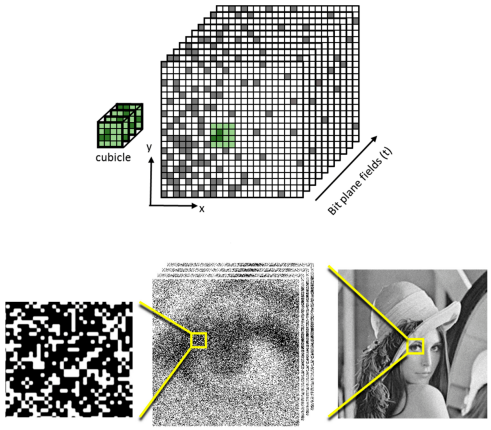


FIGURA 1.7: Ilustración conceptual de la obtención de una imagen mediante un *QIS*.

Lo expuesto anteriormente explica los principios básicos de la técnica de adquisición de datos *quanta imaging*. Sin embargo, para realizar un análisis más profundo y comprender como los planos binarios, que se obtienen mediante la aplicación de las ventanas temporales, contienen información relevante de la escena es necesario recurrir al modelado teórico de este tipo de sensores.

De acuerdo a [27], en escalas de tiempo lo suficientemente pequeñas la emisión de fotones desde una fuente de luz es estocástica y en la mayoría de estas se puede aproximar a un proceso de Poisson, tanto para el flujo de fotones que entran en el fotodetector como para los fotoelectrones recogidos en el mismo. Esto quiere decir que el número medio de fotones que llegan al sensor durante una ventana de tiempo τ solamente depende del producto del flujo ϕ [fotones/(s·fotodetector)] por la duración τ de dicha ventana. El producto $H = \phi \cdot \tau$ se conoce como exposición, y representa el número de fotones esperados a observar en una ventana de exposición τ. En un proceso de Poisson [36], la probabilidad $P[k]$ de observar k fotones para un nivel de exposición H viene dada por:

$$P[k] = \frac{e^{-H} H^k}{k!} \tag{1.2}$$

En la figura 1.8 se muestra el número de fotones observados en un total de 64 ventanas para un valor de exposición $H = 1$ [27]. Se puede apreciar que en muchas ventanas se observan múltiples fotones mientras que en otras ninguno.

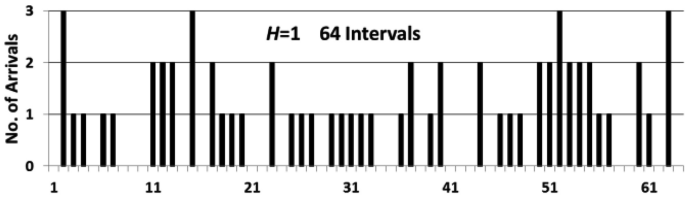


FIGURA 1.8: Número de fotones observados en un total de 64 ventanas para un valor de exposición $H = 1$.

A partir de la expresión 1.2 se puede obtener la probabilidad de que un píxel no observe, o similarmente, no sea alcanzado por ningún fotón ($k = 0$) en un determinado plano binario:

$$P[0] = e^{-H} \tag{1.3}$$

Por tanto, la probabilidad de observar, o de detectar, al menos un fotón ($k > 0$) se puede expresar como:

$$P[k > 0] = 1 - P[0] = 1 - e^{-H} \tag{1.4}$$

En el caso de una matriz de M píxeles uniformemente iluminados, el número esperado de píxeles que no observarían ningún fotón en un determinado plano binario, M_0, vendría dado por:

$$M_0 = M \cdot e^{-H} \tag{1.5}$$

mientras que el número esperado de píxeles que observarían al menos un fotón, M_1, se puede expresar como:

$$M_1 = M \cdot \left[1 - e^{-H}\right] \tag{1.6}$$

Definiendo la densidad de bits, D, como la relación $D = M_1/M = P[k > 0]$ y realizando un barrido de la exposición H se obtiene la característica de este tipo de sensores [27], tal como se muestra en la figura 1.9.

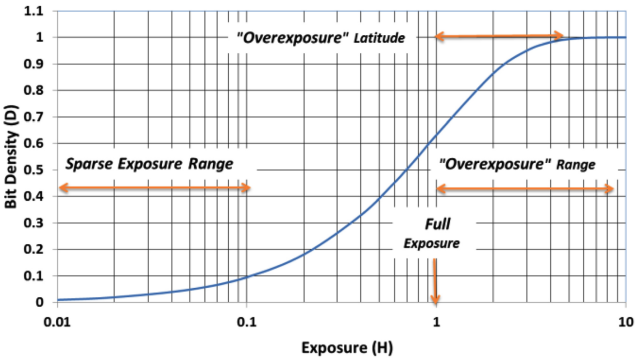


FIGURA 1.9: Característica teórica de un *Quanta Image Sensor*. La exposición H aparece representada en escala logarítmica.

Se puede observar que para una ventana de exposición de tiempo τ, cada valor de flujo ϕ tiene asociada una probabilidad $P[k > 0]$ de observar al menos un fotón (recordemos que $H = \phi \cdot \tau$ y $D = P[k > 0]$). Por tanto, a cada píxel del sensor le corresponderá una determinada probabilidad, $P[k > 0]$, de acuerdo a su nivel de iluminación, ϕ. En base a esta probabilidad se pueden diferenciar tres regiones en la gráfica. En primer lugar, una región (*Sparse Exposure Range*) con H perteneciente al intervalo [0.01, 0.1) donde tras aplicar una ventana de exposición τ existe una probabilidad muy baja de observar al menos un fotón. Esto quiere decir que en la mayoría de los planos binarios generados (ver figura 1.7) estos píxeles tendrán asociado un "0" lógico, por lo que estarán relacionados con los tonos más oscuros de la escena. En segundo lugar, una región con H perteneciente al intervalo [0.1, 1) donde la probabilidad de observar al menos un fotón es mayor conforme aumenta el valor de exposición. Así, la probabilidad de que los píxeles presenten un "1" lógico en un determinado plano binario será mayor o menor según su nivel de iluminación, por lo que estarán relacionados con los tonos grises. Y en tercer lugar, una región (*Overexposure Range*) con H perteneciente al intervalo [1, 10] donde tras aplicar una ventana de exposición τ existe una probabilidad muy alta de observar al menos un fotón. En este caso, en la mayoría de los planos binarios generados estos píxeles tendrán asociado un "1" lógico, por lo que estarán relacionados con los tonos más claros de la escena.

Esta curva en forma de "S" es bastante similar al famoso gráfico que reportaron Hurter y Driffield en 1890 [37] para las densidades de las placas fotográficas después de la exposición a la luz y el revelado. Este tipo de respuesta no lineal debe interpretarse como que existe una compresión de los niveles de iluminación, lo cual permite a los *QIS* tolerar los reflejos mejor que los sensores de imagen de estado sólido convencionales.

Aunque parte de esa respuesta no lineal puede codificarse mediante post-procesamiento en imágenes de sensores con respuesta lineal, en el caso de los *QIS* es intrínseca a sus características de funcionamiento. Entre las posibles aplicaciones del *QIS* se encuentran la obtención de imágenes científicas con poca luz (por ejemplo, la microscopía), la defensa y el sector aeroespacial, la fotografía y cinematografía profesional y de consumo, la obtención de imágenes con múltiples aperturas, la criptografía (generación de números aleatorios cuánticos) y la detección directa de partículas cargadas de baja energía, entre otras [32].

Sin embargo, para convertir al *QIS* en el sucesor real de los sensores de imagen CMOS actuales, es necesario seguir investigando varias cuestiones teóricas y tecnológicas. Las principales son: (1) algoritmos de generación de imágenes que produzcan imágenes de alta calidad; (2) comprensión de las características de formación de imágenes de los dispositivos *QIS*; (3) implementación de píxeles ("*jots*") que permitan el recuento de fotoelectrones individuales; (4) lectura de bajo consumo de grandes volúmenes de datos (la lectura de un sensor de 1 Gjot a 1000 fps produciría una tasa de datos de 1 Tb/s) y (5) procesamiento en el plano focal para reducir el volumen de datos. Detalles más específicos de este tipo de sensor se pueden encontrar en las referencias [38–42].

1.2. Motivación y Objetivos

Los sensores de luminancia asíncronos tienen numerosas ventajas. Son rápidos, presentan un consumo bajo y ofrecen un alto rango dinámico. Sin embargo, su rendimiento se ve limitado por la tasa máxima de eventos que puede manejar el sistema de arbitraje. Un posible enfoque para solucionar esta limitación podría ser aumentar la frecuencia máxima de operación del sistema de arbitraje. Sin embargo, un enfoque alternativo más interesante por los beneficios adicionales que tendría asociado, consistiría en intentar adquirir las imágenes mediante un flujo de datos reducido.

De poderse realizar, en primer lugar, evitaría la saturación del sistema de arbitraje, y por lo tanto, la pérdida de información que esta produce. Adicionalmente, permitiría reducir su consumo de energía, relajar las especificaciones reduciendo el ancho de banda, aumentar el tamaño de la matriz utilizada y/o elevar la frecuencia máxima a la que podrían pulsar los píxeles. Esto último puede ser especialmente útil en aplicaciones donde se requiera utilizar matrices basadas en dispositivos sensibles a un único fotón como podrían ser los SPADs, cuyas frecuencias se encuentran en el orden de los *MHz*.

Por tanto, la motivación principal de desarrollar algún método que permita reconstruir las imágenes con un número menor de eventos vendría dada por el hecho de que permitiría paliar la limitación que presentan actualmente este tipo de sensores, y además, podría lograr extender aún más su uso dado las ventajas adicionales que supone.

La razón que invita a apostar por un enfoque que reduzca el flujo de datos se apoya en el hecho de que es el adoptado en el caso de los *Quanta Imaging Sensors*. Se debe tener en cuenta que estos sensores, al utilizar pequeñas ventanas de exposición, generan un flujo de datos reducido en comparación con el flujo de fotones que reciben los píxeles. Sin embargo, pese a esta reducción, sigue siendo posible reconstruir las imágenes.

En el caso de los *QIS*, tal y como se expuso con anterioridad, esto es posible porque los pulsos de los píxeles siguen una distribución de Poisson, de manera que cada nivel de iluminación queda definido por un determinado valor de probabilidad. Sin embargo, en el caso de los sensores de luminancia asíncronos los píxeles pulsan de manera periódica. Esto implica que será necesario analizar tanto las características que deberán tener las ventanas, como la forma en que se deberán aplicar para preservar la información de las imágenes pese a reducir el número de eventos.

En base a lo comentado, el objetivo principal de este trabajo consistirá en realizar un estudio de la viabilidad de la técnica *Quanta Imaging* para adquirir imágenes en sensores de luminancia o de tipo *octopus* a partir de un flujo de datos reducido. Para lograr este objetivo, el alcance del trabajo incluirá el desarrollo de las siguientes tareas:

- Revisión del estado del arte y análisis del problema, realizado en este capítulo.
- Modelado de un sensor de luminancia asíncrono y de una versión que implemente *Quanta Imaging*, abordado en el capítulo 2.
- Definición de distintos tipos de ventanas, sus parámetros de configuración y la caracterización de las mismas, capítulo 3.
- Validación de la técnica mediante una batería de imágenes y datos experimentales, capítulo 4.

Capítulo 2

Modelado de un sensor de luminancia asíncrono

En este capítulo se expone el modelo que se ha desarrollado para simular la adquisión de datos y posterior generación de imagen de un sensor de luminancia asíncrono. A continuación, se muestra un procedimiento para realizar de manera optimizada la reconstrucción de imágenes mediante *Quanta Imaging* y se presenta un nuevo modelo del sensor que incluye la implementación de la técnica. Por último, se presentan las métricas de comparación que se utilizarán en capítulos posteriores para comparar las imágenes generadas por cada modelo.

2.1. Generación de imagen en sensor de luminancia asíncrono.

Tomando como referencia la arquitectura propuesta por Leñero et al. en [11], se desarrolla un modelo para simular la adquisión de datos con un sensor de luminancia asíncrono. La reconstrucción de una imagen en este tipo de sensores sigue la siguiente secuencia:

1. Se recopilan las posiciones de los píxeles que pulsan durante un determinado intervalo de tiempo, denominado tiempo de integración (T_{int}), y se almacena el número de veces (N_{events}) que ha pulsado cada uno.

2. Con los valores de dicha cuenta (N_{events}) y considerando el tiempo de integración, se estima la frecuencia de los píxeles aplicando directamente N_{events}/T_{int}.

3. La estimación de la frecuencia se codifica en un número entero con tantos *bits* como resolución se quiera para representar la imagen. El valor máximo y mínimo de frecuencia se hace coincidir con el entero más alto y más bajo respectivamente, de forma que al píxel que más ha pulsado se le asigne el tono más brillante, y al que menos, el más oscuro.

En la figura 2.1 se ha representado un diagrama que explica de manera gráfica el funcionamiento del modelo. Su implementación a nivel de código se encuentra disponible en el Anexo A.3.

En primer lugar, la imagen original se convierte a escala de grises y se reescala a un tamaño de *array* de $M \times N$ píxeles. Utilizando la función *im2frequency()* los niveles de intesidad de los píxeles se convierten a frecuencia realizando una interpolación lineal, obteniéndose la matriz de frecuencias F_rs. El rango de frecuencias para realizar esta conversión se especifica como parámetro de entrada al modelo. A continuación, mediante la función *pulses()*, ecuación 2.1, se calcula el número de pulsos o eventos generado por cada píxel durante el tiempo de integración T_{int}, obteniéndose la matriz de pulsos S.

$$S = floor(T_{int} \cdot F\_rs) \tag{2.1}$$

Dividiendo el resultado entre el mismo tiempo de integración T_{int} se calcula la matriz de frecuencias medidas, *F_MEA*. Finalmente, mediante la función *frequency2im()* se genera la imagen codificando en n *bits* los valores de frecuencia.

Los resultados de este modelo servirán de referencia a la hora de evaluar el modelo que implementará *Quanta Imaging* y que se presenta en la siguiente sección.

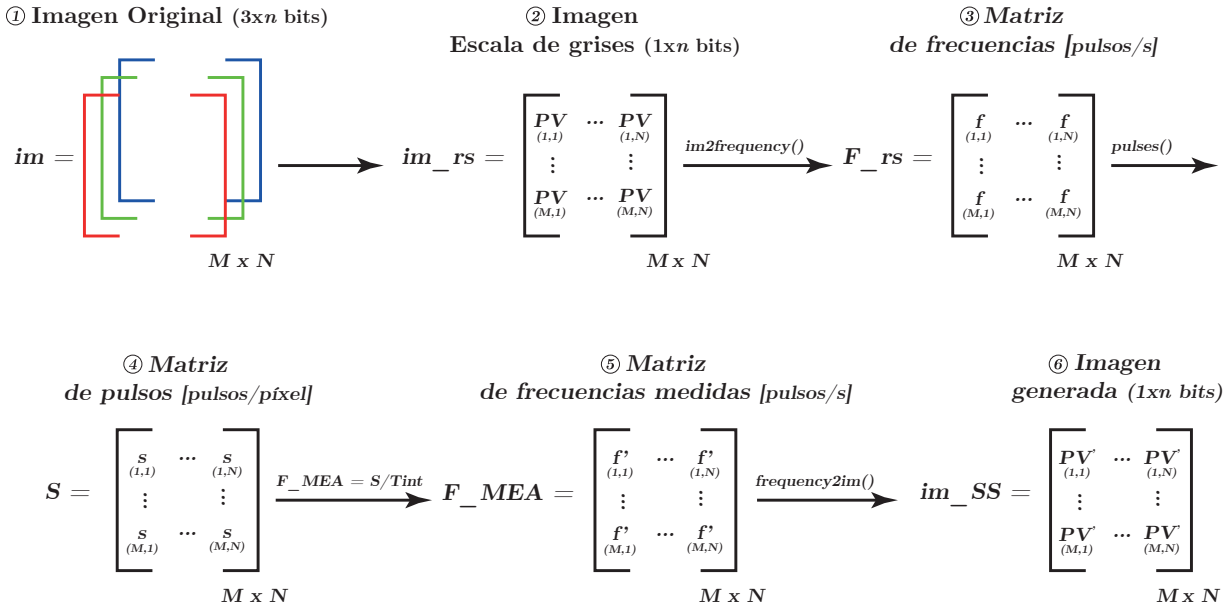


FIGURA 2.1: Generación de imagen en sensor de luminancia asíncrono sin *Quanta Imaging*.

2.1.1. Modelo de saturación del sistema de arbitraje

Tal como se detalló en 1, estos sensores pierden información cuando la tasa de eventos generados por el *array* de píxeles supera la frecuencia máxima de operación del sistema de arbitraje. En consecuencia, solamente aquellos píxeles más iluminados, los que emiten más pulsos, son los que se atienden más veces. Esto provoca un efecto muy llamativo en las imágenes generadas ya que aparecen zonas muy brillantes o muy oscuras sin tonos intermedios. Un sistema de arbitraje para sensores de alto rango dinámico se puede encontrar en [10]. El sistema descrito se caracteriza por ser de tipo *greedy*, lo que significa que no tiene en cuenta que existan peticiones anteriores de píxeles de otras filas mientras que en la fila que está atendiendo se produzcan nuevas peticiones. Esto en la práctica supone un problema ya que una pequeña cantidad de píxeles puede acaparar todo el ancho de banda del sistema.

Una estimación precisa del orden de atención de los píxeles cuando existe saturación en un sistema de este tipo escapa de los objetivos de este trabajo, ya que implicaría modelar en profundidad los detalles de la arquitectura. Además, resultaría poco práctico debido al volumen de eventos o peticiones generados por una matriz cuyos píxeles pueden llegar a pulsar a frecuencias en el orden de los kilohercios. Por ello, se opta por simplificar el problema desarrollando un modelo estadístico bajo la hipótesis de que la probabilidad de que el sistema de arbitraje atienda un determinado píxel, en caso de que exista saturación, será mayor cuanto mayor sea su frecuencia. En la figura 2.2, se muestra un diagrama del algoritmo que utiliza el modelo para seleccionar las peticiones que serán atendidas.

En primer lugar, se calcula el número máximo de eventos (*MER*, del inglés "*Max Event Rate*") que el sistema de arbitraje puede atender mediante el producto de su frecuencia máxima de operación y el tiempo de imagen dados. A continuación, se comprueba si el número de eventos o peticiones a atender es igual o inferior al *MER*. En caso de no serlo, se asocia un peso a cada píxel inversamente proporcional a su frecuencia. De esta forma, cuanto mayor es el peso de un determinado píxel más probable es que el sistema de arbitraje no lo atienda. Generado los pesos, se realiza un proceso de eliminación de eventos en función de estos. Una vez finalizado, se comprueba si el número de eventos es igual al *MER*. En caso afirmativo, el algoritmo finaliza. En caso contrario, seguirá realizando procesos de eliminación hasta lograr igualarlo. La implementación a nivel de código del modelo se encuentra disponible en el Anexo A.3.

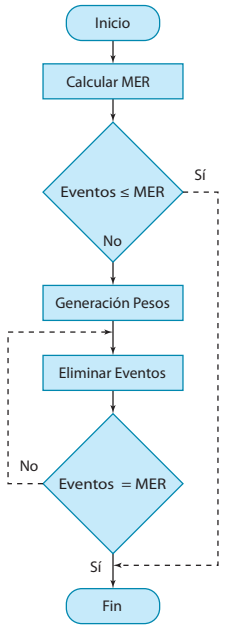


FIGURA 2.2: Diagrama de funcionamiento del modelo para simular la saturación del sistema de arbitraje.

2.2. Reconstrucción de imágenes mediante *Quanta Imaging*

Antes de desarrollar el modelo del sensor que implementará *Quanta Imaging*, es necesario realizar una implementación de la técnica que pueda tratar de manera optimizada el elevado volumen de datos que supone la simulación de eventos de un array de píxeles. Una implementación típica basada en el tratamiento de los datos mediante bucles anidados provocaría tiempos de simulación excesivamente largos (ver Fig. 2.3) que no serían prácticos para la depuración y el estudio de la técnica que se pretende en este trabajo. Por ello, se opta por realizar una implementación utilizando matrices bidimensionales y tridimensionales que permita procesar los datos en paralelo para acelerar el modelo.

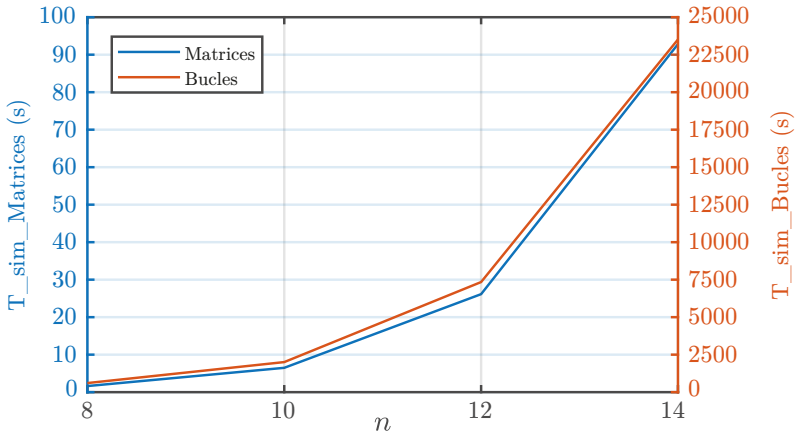


FIGURA 2.3: Tiempo de simulación utilizando bucles (naranja) y matrices (azul) para reconstruir una imagen con 2^n-1 planos. Tamaño del array: 256×256. Procesador: Intel® Core™ i7-1165G7.

En la figura 2.4, se muestra un diagrama de funcionamiento de la implementación realizada en MATLAB® para reconstruir imágenes mediante *Quanta Imaging* de manera optimizada.

Por simplicidad se ha asumido para generar la información de los planos binarios que los pulsos de los píxeles, k, siguen una determinada distribución de Poisson de parámetro H, esto es, $k \sim \text{Poisson}(H)$. Más adelante, se mostrará como generar la información de los planos para el caso de un sensor de luminancia asíncrono donde los pulsos de los píxeles son periódicos. El código completo de ambas implementaciones se adjunta en los Anexos A.1 y A.2. A continuación, se describen los pasos que aparecen representados en la figura.

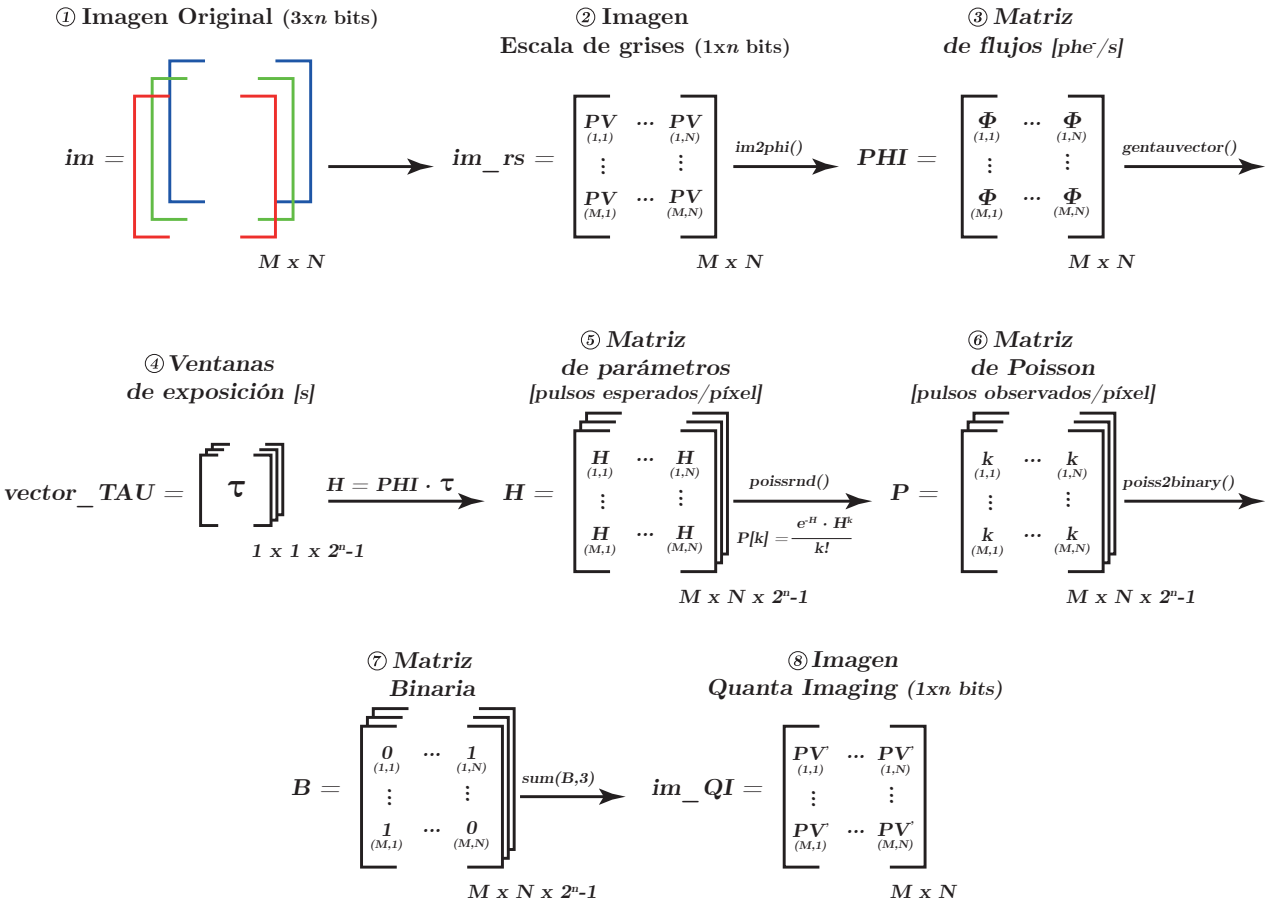


FIGURA 2.4: Reconstrucción de una imagen de n *bits* mediante *Quanta Imaging*

Partiendo de la imagen original RGB (n *bits*/color), se convierte a escala de grises con una profundidad de n *bits* y se reescala a un tamaño de *array* de $M \times N$ píxeles. Así, la imagen queda almacenada como una matriz (*im_rs*) donde cada elemento se corresponde con un determinado píxel. A continuación, utilizando la función *im2phi()* se obtiene la matriz de flujos *PHI*. Esta función realiza una interpolación lineal para obtener a partir de los valores (PV) de la matriz *im_rs* el flujo (fotones/s) que estaría recibiendo cada píxel. Llegados a este punto, mediante la función *gentauvector()* se genera un vector (*vector_TAU*) cuyos elementos se corresponden con las ventanas temporales de exposición, τ, de cada plano binario (por ejemplo, 255 para n = 8 *bits*). Realizando el producto elemento a elemento de la matriz *PHI* con este vector, se obtiene la matriz tridimensional H que contendría el parámetro de la distribución de Poisson que seguiría cada píxel para generar la información de cada uno de los planos. En este caso, por simplicidad, se considera que tanto la matriz de flujos *PHI* como las ventanas de exposición son constantes entre los distintos planos, por lo que el parámetro H de cada píxel también lo sería. Utilizando esta matriz H y la función *poissrnd()* de MATLAB®, la cual genera números aleatorios siguiendo una determinada distribución de Poisson, se genera la matriz tridimensional P que contiene el número de pulsos observados, k, por cada píxel en cada uno de los planos. Una vez obtenida la matriz P, se utiliza la función *poiss2binary()* para asignar en cada uno de los planos un "1" lógico a todos aquellos píxeles cuyo número de pulsos observados haya sido mayor o igual a uno, obteniéndose la matriz B que se correspondería con el cubo de *bits* mostrado en la figura 1.7. Finalmente, para reconstruir la imagen se calcula el valor (PV') de cada píxel como la suma en la tercera dimensión de la matriz B para cada uno de sus elementos.

En la figura 2.5 se muestran los resultados de la reconstrucción con diferentes números de planos binarios de varias imágenes de 8 *bits*.

FIGURA 2.5: (a) Imagen en escala de grises. (b,c,d) Imágenes de 8 *bits* reconstruidas mediante *Quanta Imaging* con $2^n - 1$ planos binarios : b) $n = 8$, c) $n = 10$ y d) $n = 14$. Ventanas temporales de exposición, τ, en todos los casos de 1µs.

2.3. Generación de imagen en sensor de luminancia asíncrono aplicando *Quanta Imaging*.

El objetivo principal del Trabajo Fin de Máster es estudiar la viabilidad de la técnica de adquisición de imagen *Quanta Imaging* en sensores de luminancia asíncronos. La base de la propuesta consiste en comprobar si es posible reconstruir la imagen cuando los tiempos de llegada de los eventos, en lugar de seguir una distribución de Poisson, ocurren de manera periódica en función del nivel de iluminación de los píxeles.

En la figura 2.6 se ha representado el proceso para generar la información de los planos binarios cuando la señal de los píxeles tienen una frecuencia fija. Mediante la aplicación de $2^n - 1$ ventanas temporales de observación, se asocia un 1 a cada píxel si pulsa al menos una vez durante dicho tiempo y un 0 en el caso de que no lo haga. Tal como se hace en *Quanta Imaging*, la información generada en cada uno de las ventanas temporales se almacena en la matriz tridimensional de planos binarios, donde las filas y columnas se corresponden con las posiciones de los píxeles en el *array* y la profundidad con el número de ventanas que se han utilizado. Finalmente, utilizando el cubo de *bits* que se ha generado, se realiza la suma en la tercera dimensión para reconstruir la imagen.

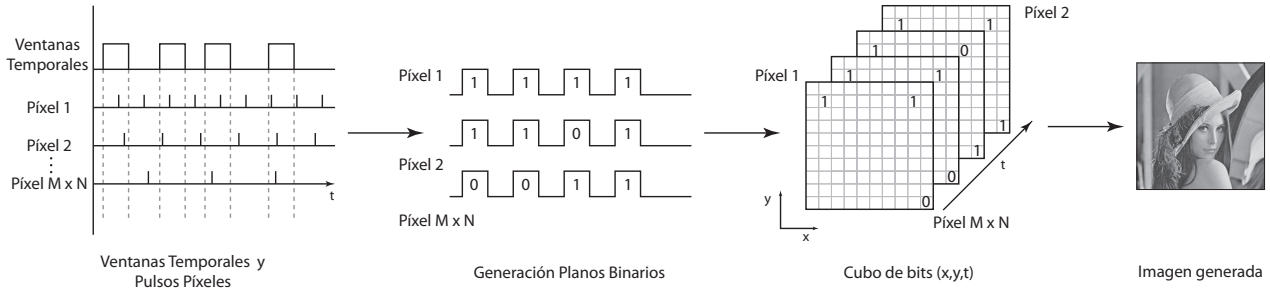


FIGURA 2.6: *Quanta Imaging* en sensor de luminancia asíncrono.

En la figura 2.7 se muestra el modelo desarrollado para generar imágenes mediante un sensor de luminancia asíncrono con *Quanta Imaging*. Puesto que en el capitulo siguiente se explica en detalle el proceso de generación de las ventanas de observación, en este apartado se asume que ya se han fijado las posiciones de las ventanas, de forma que se tienen los tiempos de inicio y final de cada ventana almacenados en los vectores t_1 y t_2, respectivamente.

Tal como se hace en el modelo anterior, la imagen original se convierte a escala de grises y se redimensiona al tamaño del *array*. De esta forma, se puede calcular la frecuencia de los pulsos de cada uno de los píxeles por medio de la función *im2frequency*, obteniéndose la matriz *F_rs*.

Tras esto, se aplican las ventanas temporales de observación a dicha matriz, donde se cuentan el número de veces que ha pulsado el mismo píxel durante el intervalo que la define. Para dicho cálculo se utiliza la función *pulses_in_window()* que realiza la operación de la ecuación 2.2.

$$S_z = floor(t\_2_i \cdot F\_rs) - floor(t\_1_i \cdot F\_rs) \tag{2.2}$$

Donde ahora S es una matriz tridimensional que almacena en z el número de veces que ha pulsado cada píxel en el plano i-ésimo. A partir de esta matriz, se genera la mencionada matriz de planos binarios (B) mediante la función *spiking2binary()* que realiza la operación lógica $B = S > 0$, asociando un 1 en cada plano donde el píxel ha pulsado al menos una vez. A continuación, realizando la suma en la tercera dimensión de B, se genera una matriz ($COUNT$) con el número de veces que ha pulsado en total cada píxel. Finalmente, se utiliza la función *count2im()* que asocia a cada valor de $COUNT$ un código binario en función de la resolución de la imagen que se quiera generar. El código completo se encuentra disponible en el Anexo A.4.

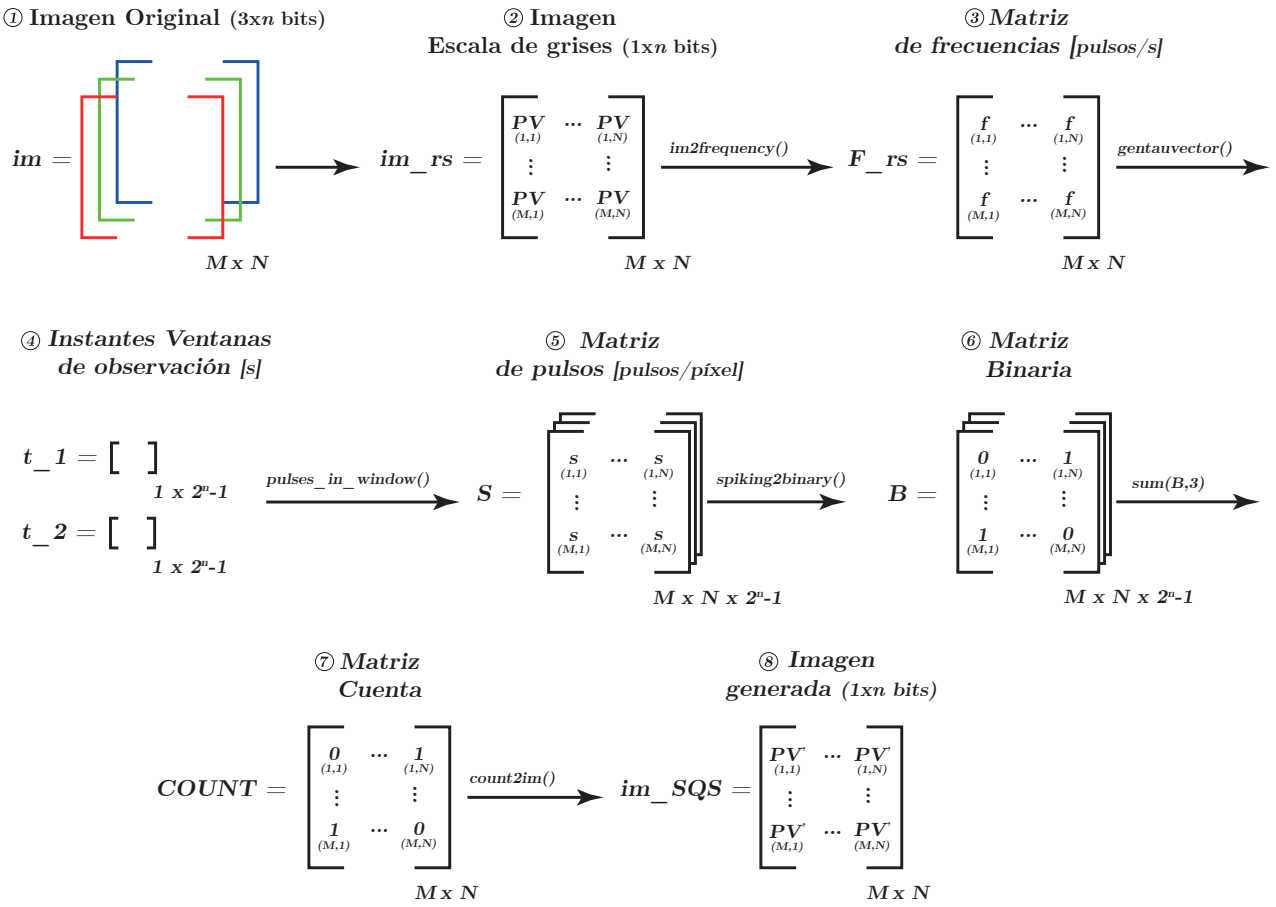


FIGURA 2.7: Generación de imagen en sensor de luminancia asíncrono con *Quanta Imaging*.

2.3.1. Modelo de saturación del sistema de arbitraje utilizando ventanas

A la hora de modelar la saturación cuando tenemos ventanas de observación se deben tener en cuenta algunos aspectos que no se consideraban en el caso anterior.

En primer lugar debemos diferenciar dos estados distintos, uno en el que la ventana de observación está activa y otro cuando la ventana está desactivada y se está esperando la apertura de la siguiente, en adelante a este intervalo se le llamará tiempo muerto. En el caso de que se produzca saturación durante el tiempo que la ventana se encuentra activa, el número de peticiones pendientes irá creciendo hasta que finalice la ventana. A partir de dicho instante, el sistema de arbitraje dispone del tiempo muerto (en el que no recibe nuevos eventos) para atender las peticiones que se quedaron pendientes. Dentro de la ventana los píxeles con mayor frecuencia seguirían teniendo mayor probabilidad de ser atendidos, mientras que durante el tiempo muerto la lectura de los píxeles por parte del sistema de arbitraje se realizaría de acuerdo a la propagación de las peticiones a través del mismo. Determinar este orden con exactitud requeriría de simulaciones temporales que tengan en cuenta los retrasos de la lógica del sistema de arbitraje. Por tanto, tal como se hizo en el caso anterior, este fenómeno se modelará como un caso estadístico donde los eventos se eliminarán seleccionándolos de forma aleatoria.

El segundo punto a tener en cuenta está referido a la relación entre el tiempo de observación y el tiempo muerto. Si bien es cierto que estrictamente deberían aplicarse dos modelos distintos para cada situación, se debe tener en cuenta que carece de sentido que el tiempo de observación sea del mismo orden de magnitud que el tiempo muerto, ya que entonces se obtendría el mismo efecto que en la saturación descrita anteriormente donde solo se atienden los píxeles más iluminados. Por esta razón, el modelo que se describe en este apartado asume que la ventana de observación es lo suficientemente pequeña como para considerar que los eventos que se producen ocurren de manera instantánea y que solamente puede existir un evento por píxel en cada ventana. De esta forma, se puede asumir una lectura equiprobable de los píxeles por parte del sistema de arbitraje.

Así, en la figura 2.8 se ha representado el diagrama de flujo que se corresponde con el modelo de saturación de un sensor de luminancia asíncrono cuando se aplica la técnica de *Quanta Imaging*.

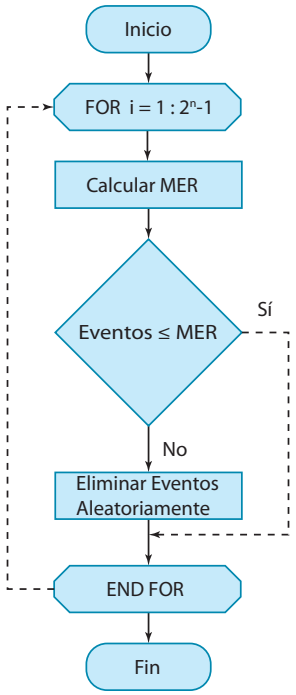


FIGURA 2.8: Diagrama de funcionamiento del modelo para simular la saturación del sistema de arbitraje.

En este caso la saturación se debe realizar plano a plano, teniendo que calcular la tasa máxima de eventos que el sistema de arbitración puede asumir para cada uno de ellos. Si dicha tasa de eventos es inferior al número de eventos registrados en dicho plano no se produciría saturación y se avanzaría al siguiente. En caso contrario, se tendría que eliminar eventos de forma aleatoria hasta que se cumpliera la condición de salida ($Eventos \leq MER$). Tal como se ha dicho anteriormente, esta eliminación se hace considerando que la lectura de cada píxel es equiprobable y que solo se tiene un evento por píxel en cada plano.

2.4. Métricas de comparación: *RMSE* y *NMI*

Para poder realizar una comparación objetiva de las imágenes generadas por los sensores presentados anteriormente y cuantificar su similitud con respecto a la imagen original, se opta por utilizar dos métricas ampliamente extendidas en el mundo del procesamiento de imágenes [43]. La primera de ellas, es el error cuadrático medio (*MSE*, del inglés "*Mean Square Error*"), el cual viene definido por la siguiente expresión:

$$MSE = \frac{1}{N} \sum_i \sum_j \left(Im_1(i,j) - Im_2(i,j) \right)^2 \tag{2.3}$$

donde Im_1 e Im_2 son las dos imágenes en escala de grises a comparar e (i, j) abarcarían el ancho y alto de la imagen.

Aunque el *MSE* puede ser más atractivo desde el punto de vista del procesamiento de imágenes por su coste computacional, en este trabajo se utilizará la raíz cuadrada de este (*RMSE*, del inglés "*Root Mean Square Error*") ya que la interpretación de su valor resulta más intuitiva.

Esta medida al tener en cuenta únicamente la diferencia de niveles de intensidad entre ambas imágenes, presenta una serie de limitaciones. Por ejemplo, una imagen obtenida dividiendo por un factor de 2 todos los niveles de intensidad de la imagen original tendrá normalmente un *RMSE* mucho mayor que una imagen obtenida asignando a cada píxel el nivel de intensidad medio de la imagen original. Sin embargo, la primera contiene toda la información presente en la imagen original, mientras que la segunda no contiene prácticamente ninguna información. Para solventar este problema, se puede utilizar otra métrica conocida como información mutua (*MI*, del inglés "*Mutual Information*") [44] que no requiere que los niveles de intensidad sean los mismos en las dos imágenes para determinar su grado de similitud. Esta segunda métrica supone que cada una de las imágenes es una variable aleatoria (X e Y). Así, la información mutua sería la información media que las observaciones de Y proporcionan sobre X, y al contrario. Es decir, una medida de lo bien que se pueden predecir los niveles de intensidad en la segunda imagen dados los de la primera. En el caso de que X e Y sean variables aleatorias discretas la *MI* vendría definida por la siguiente expresión:

$$MI(X,Y) = \sum_i \sum_j p(x_i, y_j) \log\left(\frac{p(x_i, y_j)}{p(x_i)p(y_j)}\right) \tag{2.4}$$

donde (i, j) abarcarían el ancho y alto de la imagen, $p(x)$ y $p(y)$ son las funciones de distribución de probabilidad marginal de X e Y y $p(x, y)$ es la función de distribución de probabilidad conjunta de X e Y.

La información mutua (*MI*) entre dos imágenes puede considerarse como la información media en *bits* que un píxel de una de las imágenes proporciona sobre el píxel correspondiente de la otra. Por ejemplo, la *MI* entre una imagen y ella misma de una imagen en escala de grises de 8 bits es 8, si y sólo si, la imagen contiene realmente 256 niveles de gris diferentes y están distribuidos por igual. Aunque la *MI* es una medida más precisa de la información relativa entre dos imágenes que el *RMSE*, no tiene en cuenta las diferencias de nivel de gris. En un caso extremo, una de las imágenes podría ser el negativo exacto de la otra, y sin embargo la *MI* resultaría máxima. Por tanto, una determinación justa de qué sensor genera una imagen más similar a la original debe realizarse teniéndose en cuenta cual de ellos obtiene mejores resultados tanto en términos de *RMSE* como de *MI*.

A la hora de interpretar los valores de la *MI* resulta más intuitivo emplear una versión normalizada de esta (*NMI*, del inglés "*Normalized Mutual Information*"). En el caso de este trabajo, la *NMI* se obtiene normalizando con respecto al valor de *MI* que obtendría la imagen original consigo misma, esto es:

$$NMI = MI(X,Y) \cdot \frac{1}{MI(X,X)} \tag{2.5}$$

donde se ha considerado que X sería la variable aleatoria asociada a la imagen original e Y la variable aleatoria correspondiente a la imagen generada por un determinado sensor.

El cálculo del *RMSE* se puede intuir que es bastante directo. Sin embargo, la estimación de la *MI* puede resultar más compleja debido a los cálculos de probabilidades que requiere. Aunque existen diferentes técnicas para estimar la *MI* [45], existen dos métodos que son los más utilizados. Uno de ellos se basa en la *KDE* (del inglés "*Kernel Density Estimation*"), mientras que el otro en los histograma de las imágenes [46]. En el caso de este trabajo, el método utilizado sería este último. La implementación a nivel de código de las métricas presentadas se encuentra disponible en el *script* que se adjunta en el Anexo A.5.

Capítulo 3

Caracterización de *Quanta Imaging* en un sensor de luminancia asíncrono

En este capítulo se comienza definiendo los parámetros de interés de las ventanas temporales que se utilizarán para simular la aplicación de *Quanta Imaging* en un sensor de luminancia asíncrono. A continuación, se presentan los diferentes tipos de ventanas desarrollados, así como la configuración de parámetros que las caracterizan. Posteriormente, se exponen las especificaciones a considerar para la generación de cada ventana. Por último, se propone un esquema tentativo a nivel de píxel para aplicar *Quanta Imaging* en sensores de luminancia asíncronos.

3.1. Ventanas de observación

La implementación de las ventanas temporales necesarias para simular la aplicación de *Quanta Imaging* en un sensor de luminancia asíncrono requiere establecer una serie de parámetros que permitan su generación. En la figura 3.1 aparecen indicados sobre el par de ventanas representadas. El primero de ellos, *T_binary_plane*, se corresponde con la duración del periodo de los planos binarios. En el caso de t_1, t_2 y t_3, se definen para determinar la posición que ocupa la ventana dentro del periodo de un determinado plano. El parámetro t_1, establece el instante de tiempo donde comienza la ventana, t_2 el instante donde termina, y t_3 el instante donde finaliza el periodo del plano binario. Esto permitirá generar diferentes tipos de ventanas. Al tiempo que la ventana se encuentra activa $(t_2 - t_1)$ se le denominará τ_{on}, mientras que al ciclo de trabajo de los planos binarios, dado por la relación τ_{on}/T_binary_plane, D.

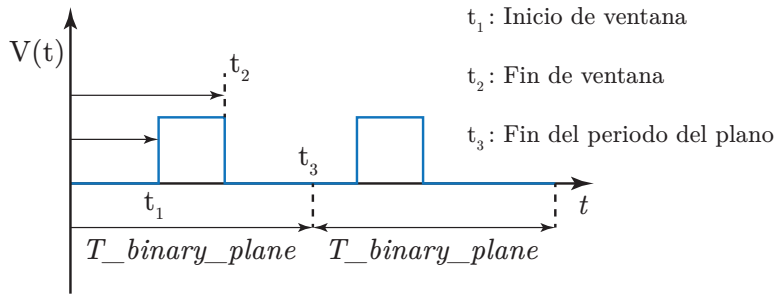


FIGURA 3.1: Instantes característicos de las ventanas temporales de observación.

Dado un determinado τ_{on}, el desplazamiento de la ventana dentro del periodo del plano podría realizarse manteniendo constante t_3 y variando t_1, o bien, fijando $t_1 = 0$ y variando t_3. Sin embargo, esta última opción tiene la ventaja de que el tiempo de lectura de eventos (tiempo desde t_1 de una ventana hasta t_1 de la siguiente) coincide con *T_binary_plane*. Por ello, es la elegida a fin de facilitar la implementación de las ventanas.

3.2. Tipos de ventanas de observación y característica.

En las secciones que siguen se ilustran las formas de ventanas que se han desarrollado estableciendo diferentes configuraciones de los parámetros presentados previamente. También, se muestra la característica de 8 bits que obtendría un sensor de luminancia asíncrono que implementara cada tipo de ventana. Esta característica se obtiene a partir del modelo presentado en la sección 2.3 utilizando como imagen un degradado en escala de grises como el que se muestra en la figura 3.2. Este degradado al presentar una profundidad de 16 bits permite que exista una cantidad de 255 frecuencias diferentes para cada código de la característica.

En cuanto al número de filas (M) y columnas (N) de la matriz del sensor vendría impuesto por el tamaño del propio degradado, en este caso $M = N = 256$. Por otra parte, el número de planos binarios podría ser de 255, ya que la característica que se pretende obtener se va a codificar en 8 bits. Sin embargo, se opta por tomar un número mayor. En concreto, se establece en 1023 planos a fin de hacer más evidente sobre la característica, el efecto que provoca las propiedades de cada tipo de ventana frente al efecto o ruido que puede producir una codificación en 8 bits con únicamente 255 muestras.

Con respecto al valor de frecuencia máxima a la que podrían pulsar los píxeles (f_{max}), se elige para ser de 55kHz de acuerdo a los datos reportados por Leñero et al. en la referencia [10]. No obstante, como se indica posteriormente este valor podría ser superior o inferior.

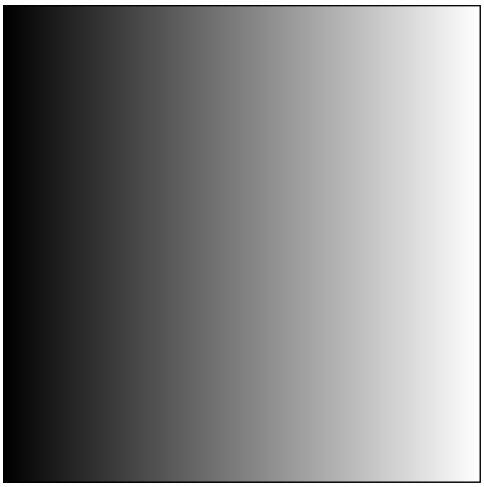


FIGURA 3.2: Degradado de 16 bits utilizado para obtener la característica de cada tipo de ventana.

El tiempo que las ventanas se encuentran activas (τ_{on}) se fija según la relación 3.1. Esto se hace así porque los planos binarios que se utilizan para reconstruir las imágenes mediante *Quanta Imaging* se obtienen asociando un 1 lógico a todos aquellos píxeles que hayan pulsado al menos una vez. Por tanto, carece de sentido observar más de 1 pulso de cada píxel para determinar su estado lógico ya que no aportaría información a la hora de reconstruir la imagen pero sí supondría un aumento en el número de eventos a leer.

$$\tau_{on} = \frac{1}{f_{max}} \tag{3.1}$$

De aquí se deduce que fijar τ_{on} según la expresión superior tiene varias implicaciones:

1. Al ser los pulsos de los píxeles periódicos, permite que f_{max} se corresponda directamente con el código más alto de la característica, en este caso 255.

2. Idealmente, dentro de una misma ventana cada píxel solamente puede pulsar una única vez, por lo que se reduce el número de eventos generados.

3. Permite que la técnica sea fácilmente escalable a otras frecuencias de operación, ya que el tamaño de las ventanas se ajusta de acuerdo a f_{max}.

3.2.1. Ventanas *Periodic*

El primer tipo de ventana de observación desarrollado, *Periodic*, consiste en aplicar las ventanas de manera periódica. La figura 3.3, donde se ha representado en azul la señal de las ventanas junto con una señal periódica en naranja de frecuencia $1/T\_binary\_plane$, permite apreciar fácilmente la periodicidad de estas.

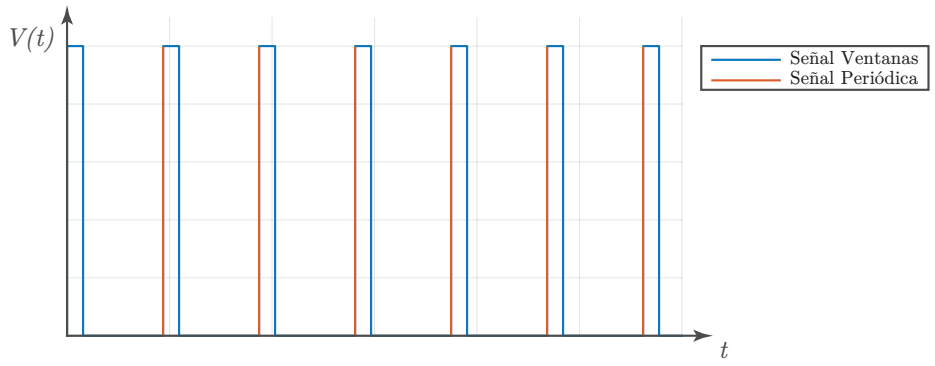


FIGURA 3.3: Ventanas temporales de observación de tipo *Periodic*.

En este caso tanto el tiempo que la ventana se encuentra activa (τ_{on}) como el periodo de los planos binarios (T_binary_plane) permanecen constantes. La configuración de los parámetros necesaria para su generación quedaría como se muestra en la expresión 3.2. Nótese que tanto en este tipo de ventana, como en las que se presentan posteriormente, siempre es necesario cumplir la condición $t_3 \geq t_2$.

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \tau_{on} \\ t_3 &= T\_binary\_plane \end{aligned} \tag{3.2}$$

En la figura 3.4 se muestra la característica que tendría un sensor de luminancia asíncrono que implementara este tipo de ventanas. Se puede observar que existen frecuencias inferiores a f_{max} (55kHz) que obtienen el código máximo de la característica, y al contrario, frecuencias superiores a la mínima que son asociadas con el código mínimo. Esto vendría explicado por el hecho de que tanto los pulsos de los píxeles como las ventanas son periódicos. Por tanto, existen determinadas frecuencias que se sincronizan con estas siendo muy observadas (en el caso extremo siempre), y frecuencias que se sincronizan de modo que son menos observadas (en el caso extremo nunca). Esto aparece reflejado en el degradado, donde algunos píxeles aparecen totalmente iluminados o más de lo que deberían, mientras que otros aparecen menos iluminados o totalmente oscuros.

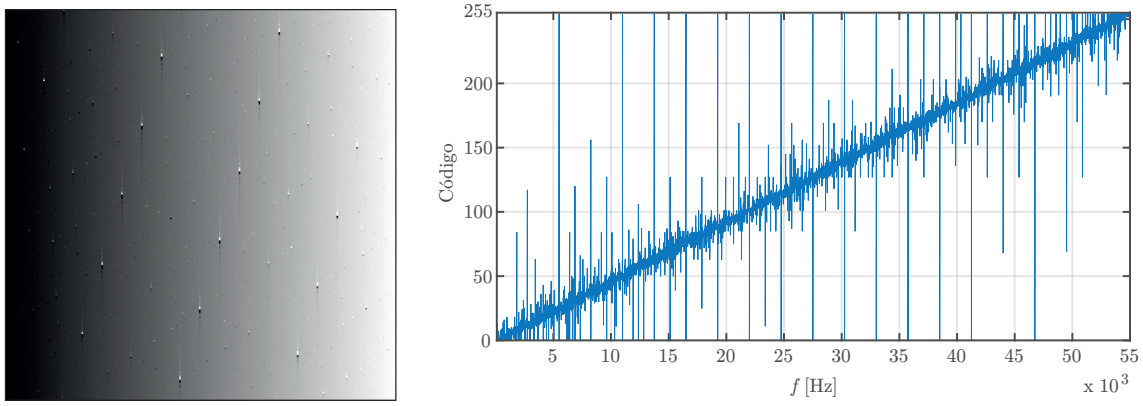


FIGURA 3.4: Característica de un sensor de luminancia asíncrono aplicando *Quanta Imaging* con ventanas de tipo *Periodic*.

3.2.2. Ventanas *Progressive-Shift*

El segundo tipo de ventana de observación desarrollado, *Progressive-Shift*, consiste en aplicar las ventanas realizando un desplazamiento progresivo entre estas. Para ello, la ventana siguiente se desplaza con respecto a la anterior una cantidad igual a una fracción del período de los planos (T_binary_plane) cada vez mayor, tal como se muestra en la figura 3.5.

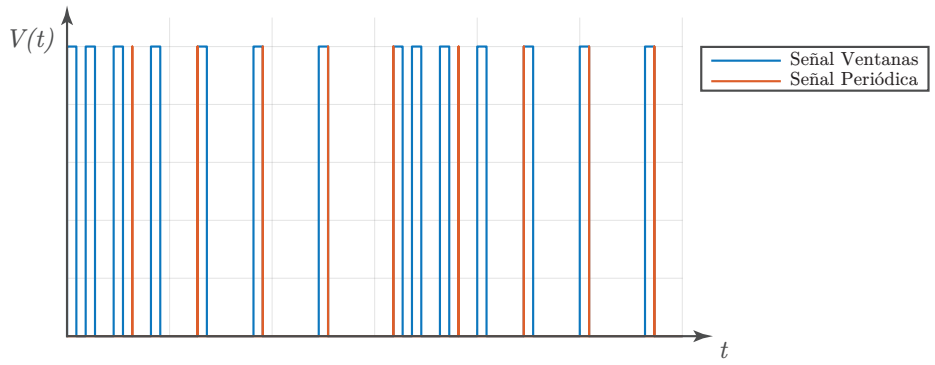


FIGURA 3.5: Ventanas temporales de observación de tipo *Progressive-Shift*.

El número total de desplazamientos diferentes (*divisions*) se toma como el entero que resulta de la siguiente expresión:

$$divisions = floor\left(\frac{T\_binary\_plane}{\tau_{on}}\right)$$

El tiempo que la ventana permanece activa (τ_{on}) es constante, mientras que el período de los planos varía de acuerdo al valor del desplazamiento. La configuración de los parámetros para su generación viene dada por la ecuación 3.3, donde m es un número entero de valor inicial igual a 1 que se incrementa progresivamente en una unidad hasta alcanzar el valor de *divisions*. Una vez alcanzado, vuelve a valer 1 y se repite la secuencia.

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \tau_{on} \\ t_3 &= t_2 + \frac{m}{divisions} \cdot T\_binary\_plane \end{aligned} \tag{3.3}$$

En la figura 3.6 se muestra la característica que tendría un sensor de luminancia asíncrono que implementara este tipo de ventanas. Se puede observar que la amplitud máxima de los picos se reduce con respecto a las ventanas periódicas. Como resultado, en el degradado las variaciones en los niveles de iluminación de los píxeles con respecto al valor que deberían tener, son ahora más suaves.

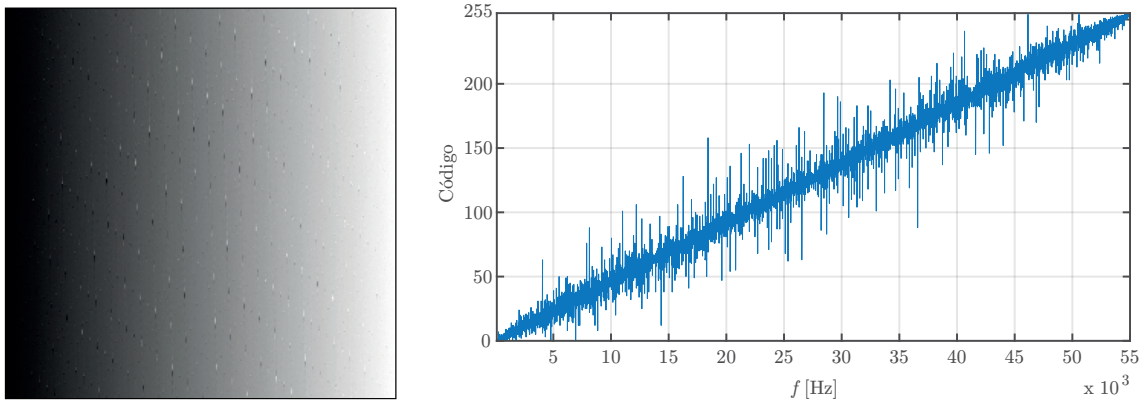


FIGURA 3.6: Característica de un sensor de luminancia asíncrono aplicando *Quanta Imaging* con ventanas de tipo *Progressive-Shift*.

3.2.3. Ventanas *Random-Plane*

El tercer tipo de ventana de observación desarrollado, *Random-Plane*, consiste en aplicar las ventanas de manera aleatoria con el objetivo de evitar la sincronización de estas con las señales periódicas de los píxeles. Para ello, el periodo de los planos se hace depender de una variable que sigue una distribución aleatoria uniforme. El comportamiento que presenta este tipo de ventana se muestra en la figura 3.7, donde se puede apreciar que no exhibe ningún patrón sistemático como ocurría en las ventanas presentadas anteriormente.

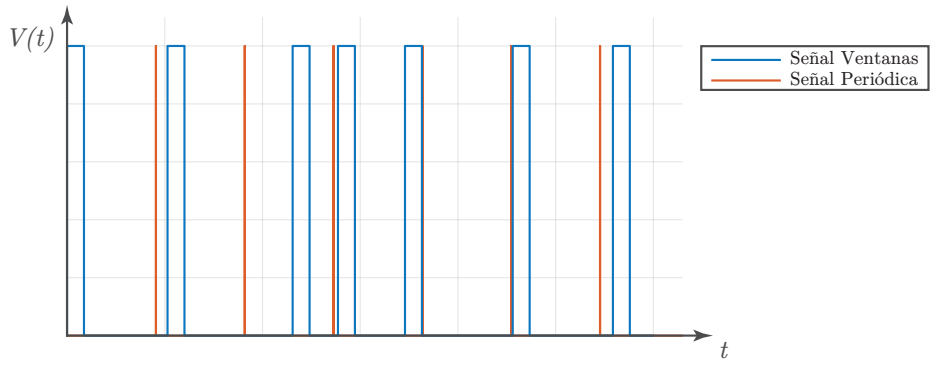


FIGURA 3.7: Ventanas temporales de observación de tipo *Random-Plane*.

El tiempo que la ventana permanece activa (τ_{on}) se mantiene constante, mientras que el período de los planos variaría según el valor de la variable. La configuración de los parámetros para generar este tipo de ventana se muestra en la ecuación 3.4, donde *rand* es una variable pseudoaleatoria de media cero que sigue una distribución uniforme y que se encuentra acotada en el intervalo $(0.5, 1.5)$. Esto último permite colocar el inicio de la siguiente ventana en cualquier punto del intervalo $(0, T\_binary\_plane)$.

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \tau_{on} \\ t_3 &= T\_binary\_plane \cdot rand \end{aligned} \tag{3.4}$$

La figura 3.8 muestra la característica que tendría un sensor de luminancia asíncrono que implementara este tipo de ventanas. Se puede observar que la aleatoriedad introducida en el periodo de los planos consigue evitar la sincronización con las frecuencias de los píxeles, y en consecuencia, no aparecen picos en la característica. No obstante, esta dependencia con una variable aleatoria introduce un cierto ruido aleatorio sobre la característica que resulta en un degradado más borroso.

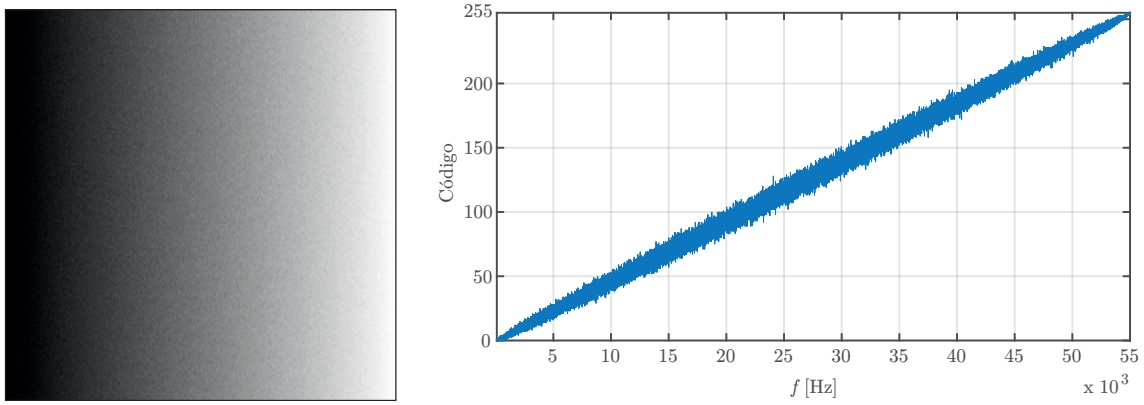


FIGURA 3.8: Característica de un sensor de luminancia asíncrono aplicando *Quanta Imaging* con ventanas de tipo *Random-Plane*.

3.2.4. Ventanas *Arrivaltime-Poisson*

El cuarto y último tipo de ventana desarrollado, *Arrivaltime-Poisson*, también consiste en aplicar la ventanas de manera aleatoria, tal como se muestra en la figura 3.9. Sin embargo, en este caso los periodos de los planos vienen dados por los tiempos de llegada (conocidos como *Interarrival Times*) de eventos que siguen una determinada distribución de Poisson. Para ello, el periodo de los planos se hace depender de una variable aleatoria que sigue una distribución exponencial. Esta distribución se puede interpretar como el lapso de tiempo que transcurre hasta el primer evento de Poisson. Una explicación más detallada de la relación entre la distribución exponencial y la distribución de Poisson se puede encontrar en la referencia [47].

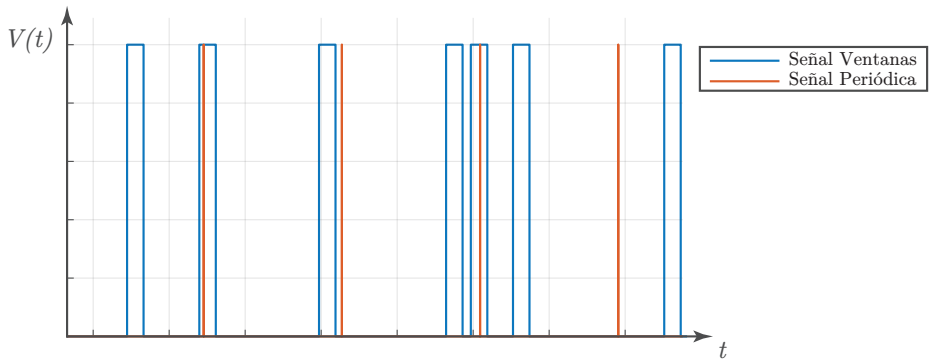


FIGURA 3.9: Ventanas temporales de observación de tipo *Arrivaltime-Poisson*.

El tiempo que la ventana permanece activa (τ_{on}) se mantiene constante, mientras que el período de los planos variaría según el valor de la variable. La configuración de los parámetros para generar este tipo de ventana se muestra en la expresión 3.5, donde *rand* es un número pseudoaleatorio perteneciente al intervalo $(0, 1)$ y λ el parámetro de la distribución de Poisson (número medio de eventos por unidad de tiempo).

$$\begin{aligned} t_1 &= 0 \\ t_2 &= \tau_{on} \\ t_3 &= -\frac{\ln(1 - rand)}{\lambda} \end{aligned} \tag{3.5}$$

Para que los instantes t_3 sean mayores que los instantes t_2, se ha encontrado que una buena aproximación es tomar λ como:

$$\lambda = \frac{1}{T\_binary\_plane}$$

En la figura 3.10 se ha representado la característica que tendría un sensor de luminancia asíncrono que implementara este tipo de ventanas. De nuevo, la aleatoriedad de las ventanas evita que aparezcan picos muy acentuados sobre la característica. Sin embargo, el aspecto del degradado que se obtiene es algo más borroso debido al ruido aleatorio.

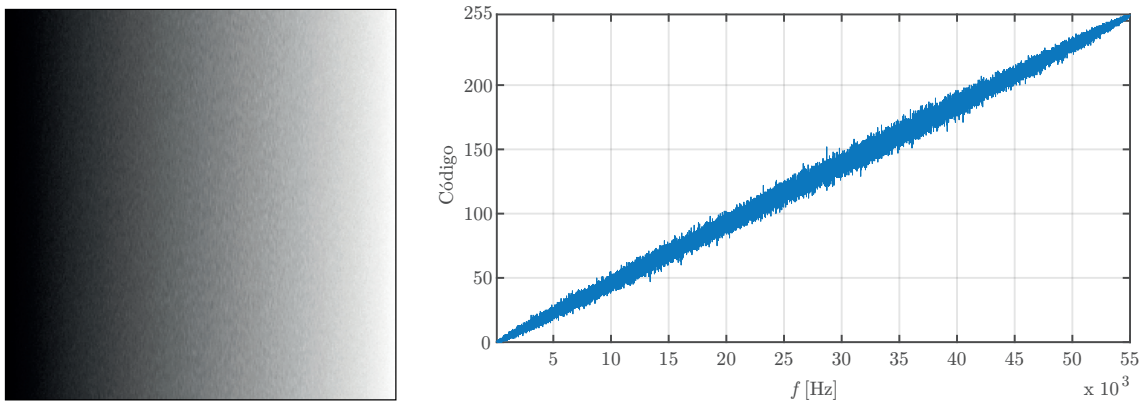


FIGURA 3.10: Característica de un sensor de luminancia asíncrono aplicando *Quanta Imaging* con ventanas de tipo *Arrivaltime-Poisson*.

3.3. Especificaciones para la generación de ventanas

En esta sección se explicará en detalle como determinar el valor de *T_binary_plane* según las especificaciones que se deseen alcanzar dentro de los límites que ofrece cada tipo de ventana.

Todas las gráficas aquí mostradas se obtienen a partir del modelo presentado en 2.3 realizando un barrido del ciclo de trabajo. De nuevo, se utiliza el degradado de 16 bits anterior como imagen de entrada para generar con cada tipo de ventana una imagen codificada en 8 bits. La frecuencia máxima a la que se espera que pulsen los píxeles se mantiene igual ($f_{max} = 55\,kHz$). En el caso del número de planos, que vendría dado por $2^n - 1$, se considera suficiente tomar unos valores de n iguales a 8, 10 y 12 para mostrar lo que se pretende.

3.3.1. Periodic

La figura 3.11 muestra los valores de las métricas *RMSE* y *NMI* en función del ciclo de trabajo (D) que se obtienen para los diferentes valores de n utilizando ventanas de tipo *Periodic*.

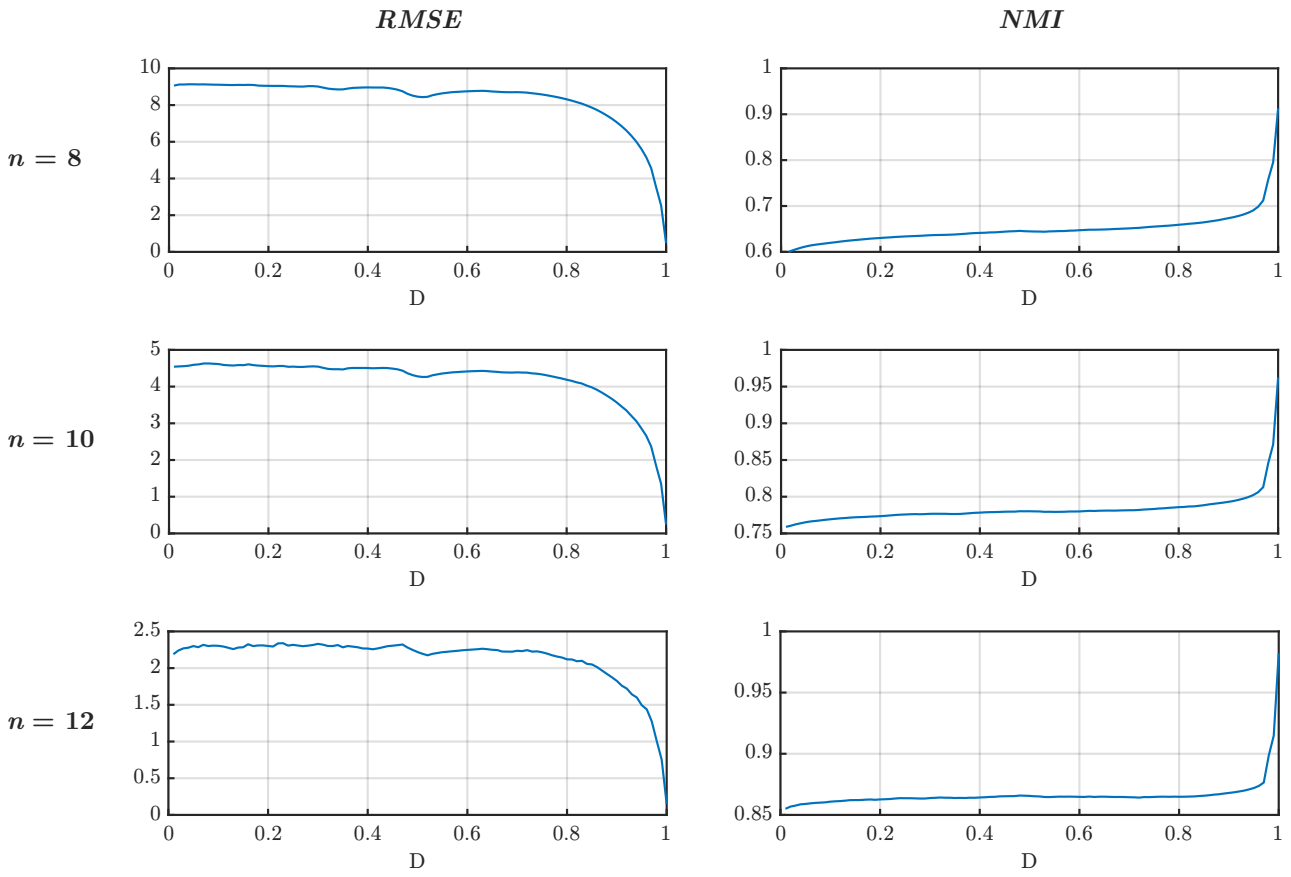


FIGURA 3.11: Métricas en función del ciclo de trabajo para diferentes valores de n utilizando ventanas de tipo *Periodic*.

Si se analizan las gráficas correspondientes al *RMSE* para un determinado valor de n, se puede apreciar que su valor disminuye lentamente hasta alcanzar aproximadamente un 0.8 en el ciclo de trabajo. A partir de ese punto, comienza a disminuir significativamente hasta alcanzar un valor mínimo cercano a 0 para $D = 1$. Esto último debe ser así, ya que aplicar ventanas periódicamente con un ciclo de trabajo igual a la unidad resulta en una sucesión de ventanas contiguas que equivale a observar constantemente los pulsos de los píxeles. En el caso de la *NMI*, el comportamiento es similar pero al contrario. Comienza creciendo lentamente hasta que entorno a un 0.8 de ciclo de trabajo empieza a aumentar notablemente alcanzando su valor máximo en $D = 1$.

Si el análisis se realiza entre gráficas con distinto valor de n, se puede observar que ambas métricas mejoran al aumentar el número de ventanas ($2^n - 1$). En el caso del *RMSE* se obtiene un máximo menor, mientras que en el caso de la *NMI* aumenta el valor mínimo obtenido.

En la figura 3.12 se ha representado en forma de porcentaje la reducción, *ReducEventsEXP*(%), que permite en el número de eventos la aplicación de ventanas de tipo *Periodic* con respecto a un sensor de luminancia asíncrono que no implemente *Quanta Imaging*.

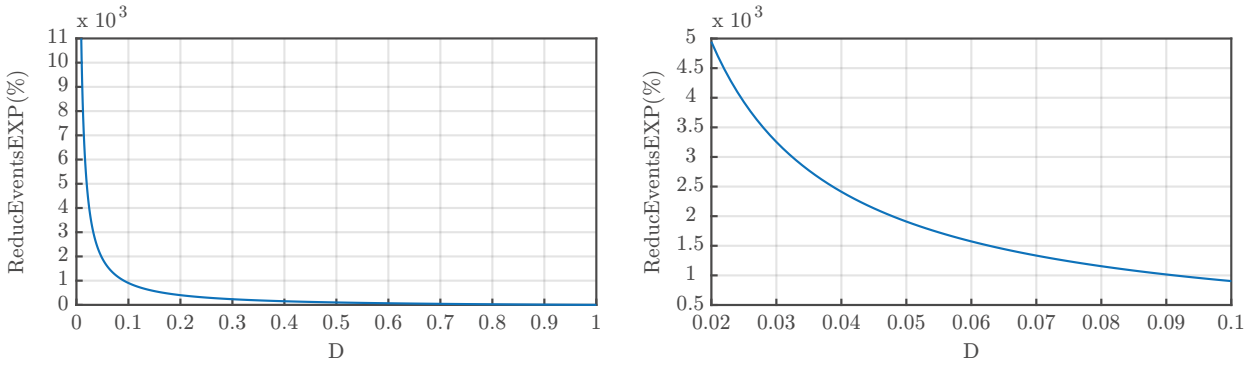


FIGURA 3.12: a) Porcentaje de reducción del número de eventos en función del ciclo de trabajo para ventanas de tipo *Periodic*. b) *Zoom*.

Este porcentaje de reducción, se puede obtener mediante la siguiente expresión:

$$ReducEventsEXP(\%) = \left[\frac{1}{D} - 1\right] \times 100 \tag{3.6}$$

En el caso del número de *FPS*, su valor vendría dado por la expresión 3.7. En la figura 3.13 se ha representado su valor para los distintos valores de n considerando una $f_{max} = 55\,kHz$.

$$FPS = \frac{f_{max} \cdot D}{2^n - 1} \tag{3.7}$$

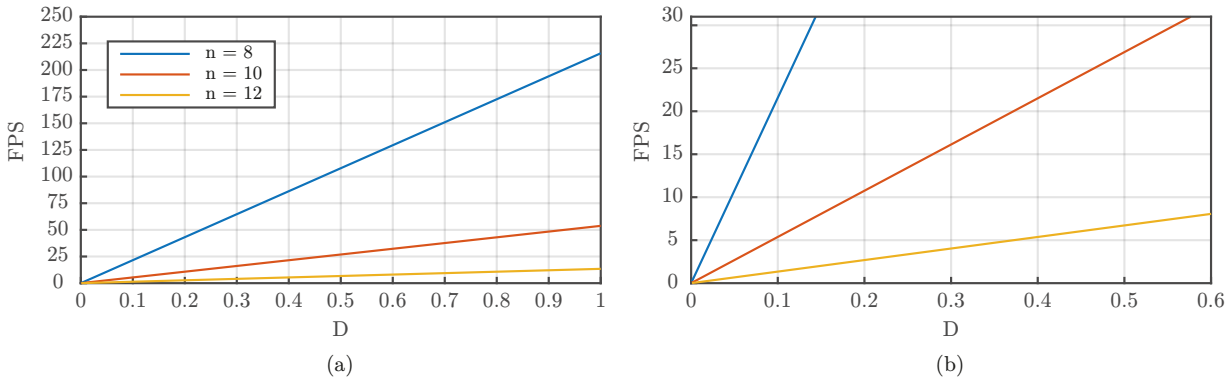


FIGURA 3.13: a) Número de FPS en función del ciclo de trabajo y de n para ventanas de tipo *Periodic*. b) *Zoom*.

Aunque este tipo de ventana puede establecer teóricamente un ciclo de trabajo en el intervalo (0,1], se debe tener en cuenta las consideraciones de diseño que existen para elegir un valor adecuado dada una f_{max} y determinado número de planos. Por una parte, valores excesivamente pequeños de D permitirían un porcentaje de reducción de eventos muy elevado como muestra la figura 3.12, pero provocaría tasas de *FPS* bajas como puede apreciarse en la figura 3.13. Por otra parte, valores elevados del ciclo de trabajo mejorarían las métricas (*RMSE* y *NMI*) y permitirían una mayor tasa de *FPS*. Sin embargo, haría que la diferencia de eventos con respecto a un sensor de luminancia que no implemente *Quanta Imaging* fuera prácticamente nula.

En cuanto al número de planos, se ha visto que aumentar n permite mejorar los valores de *RMSE* y *NMI*. Sin embargo, un aumento de n suponen nuevas consideraciones de diseño. Mantener el mismo ciclo de trabajo para que no disminuya el porcentaje de reducción de eventos, provocaría una bajada en la tasa de *FPS*. En cambio, aumentar el ciclo de trabajo para que no disminuya la tasa de *FPS*, provocaría una disminución en el porcentaje de reducción de eventos.

Las consideraciones de diseño que se han comentado aquí aplican a todo los tipos de ventanas, y es importante destacar que no existe una única configuración que sea óptima en todas las situaciones. Dependerá de los requisitos de cada aplicación.

Una vez seleccionado el ciclo de trabajo de interés, determinar la especificaciones en términos de τ_{on} y *T_binary_plane* de la ventana para operar a una determinada frecuencia máxima (f_{max}) se realiza siguiendo los siguientes pasos:

1. Se calcula el tiempo que la ventana estará activa (τ_{on}) de acuerdo a la frecuencia máxima a la que se espera que pulsen los píxeles:
$$\tau_{on} = \frac{1}{f_{max}} \tag{3.8}$$
2. Se calcula el período de los planos (*T_binary_plane*) necesario para obtener el ciclo de trabajo requerido:
$$T\_binary\_plane = \frac{\tau_{on}}{D} \tag{3.9}$$

3.3.2. Progressive-Shift

En la figura 3.14 se muestran los valores de las métricas *RMSE* y *NMI* en función del ciclo de trabajo (D) que se obtienen para los diferentes valores de n utilizando ventanas de tipo *Progressive-Shift*.

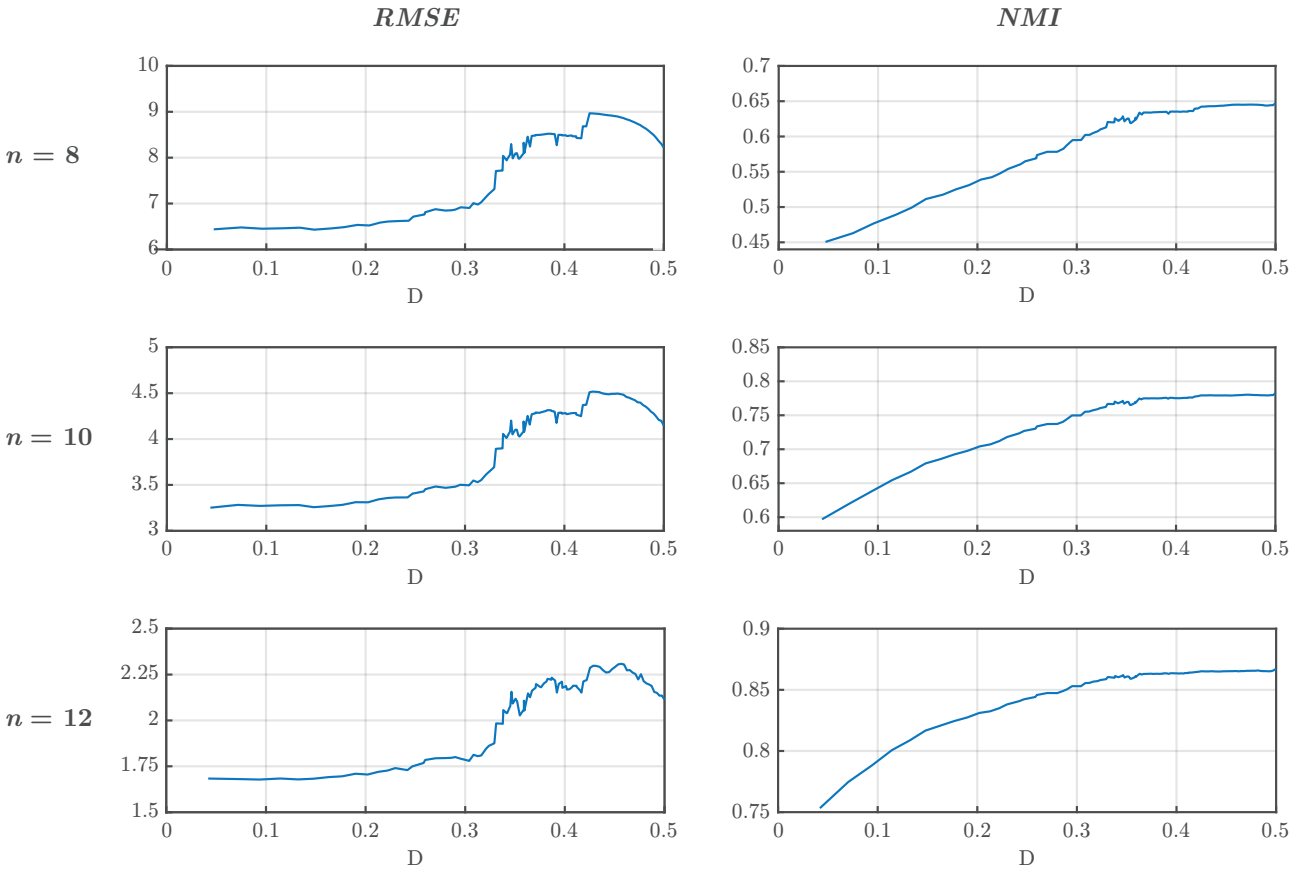


FIGURA 3.14: Métricas en función del ciclo de trabajo para ventanas de tipo *Progressive-Shift*.

Se puede observar que para valores del ciclo de trabajo inferiores a 0.2 el *RMSE* permanece relativamente estable mientras que la *NMI* aumenta progresivamente. Sin embargo, para valores superiores el *RMSE* comienza aumentar y frena la tendencia creciente de la *NMI*. Nótese que el ciclo de trabajo de este tipo de ventana, debido a su implementación (ver subsección 3.2.2), esta limitado a 0.5 como valor máximo.

Si se superpone a la gráfica anterior de las métricas la obtenida por las ventanas *Periodic* como se muestra en la figura 3.15, se puede apreciar que conforme aumenta el ciclo de trabajo la *Progressive-Shift* tiende a tener la

respuesta de la *Periodic*. Esto es así porque conforme aumenta el ciclo de trabajo, disminuye el *T_binary_plane*. Como el número de desplazamientos viene dado por la relación T_binary_plane/τ_{on}, también disminuye conforme *T_binary_plane* se hace más pequeño. Cuanto menor sea el número de desplazamientos, más periódica se vuelven las ventanas. En el caso extremo, cuando *T_binary_plane* iguala a τ_{on}, solo existe un desplazamiento y las ventanas se vuelven totalmente periódicas.

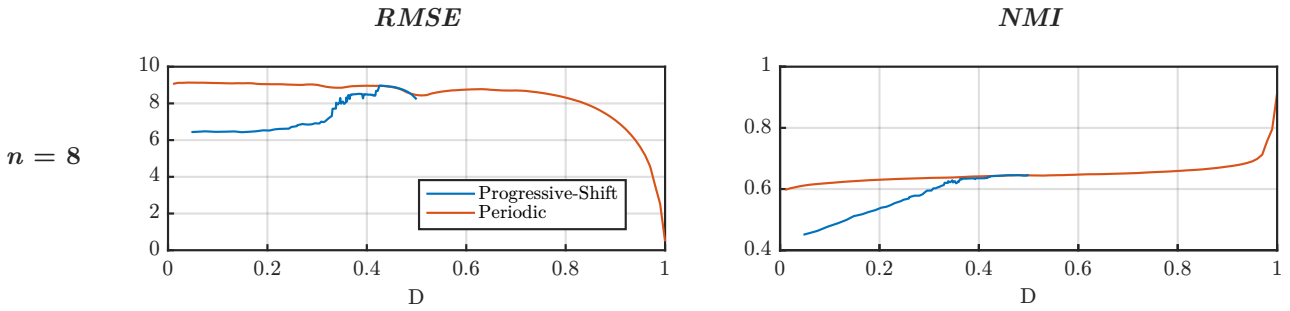
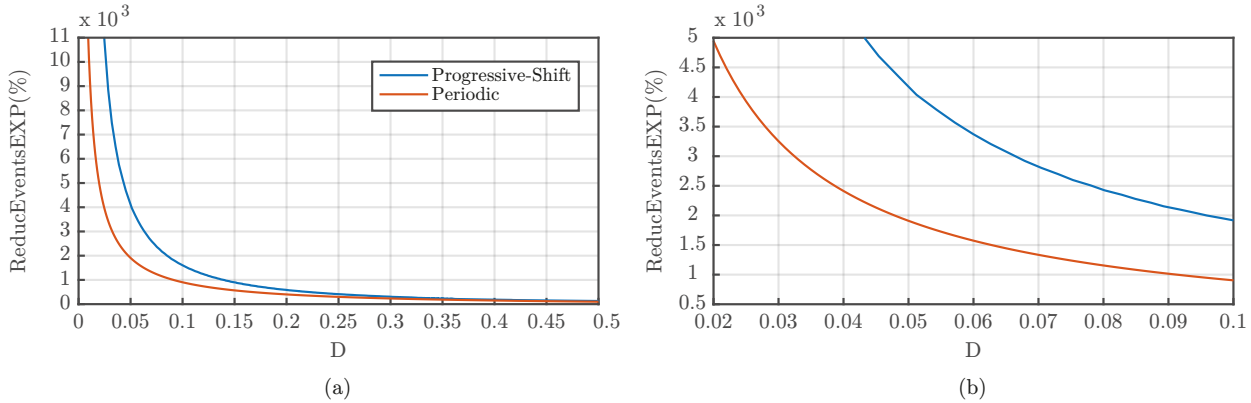


FIGURA 3.15: Comparación de métricas entre *Progressive-Shift* y *Periodic* para $n = 8$.

En la figura 3.16 se ha representado la reducción que produce este tipo de ventana en comparación con la de tipo *Periodic*. Se puede observar que para un ciclo de trabajo dado, la reducción es mayor en el caso de la *Progressive-Shift*. Esto es así porque el valor del ciclo de trabajo representado para esta última es una media, por lo que en determinados planos será mayor, menor o igual a la media.



(a) (b)

FIGURA 3.16: a) Comparación en la reducción de eventos entre *Progressive-Shift* y *Periodic*. b) *Zoom*

En este caso, la reducción del número de eventos se puede aproximar como:

$$ReducEventsEXP(\,\%) \geq \left[\frac{1}{D} - 1\right] \times 100 \approx \left[\frac{1 + divisions \cdot D}{D} - 1\right] \times 100 \tag{3.10}$$

En el caso de los *FPS* ocurre justo lo contrario. En la figura 3.17 se muestra la tasa de *FPS* de este tipo de ventana en comparación con la de tipo *Periodic* para una $f_{max} = 55\,kHz$ y $n = 8$. Se puede apreciar que para ciclos de trabajos pequeños el número de *FPS* es menor que en el caso de la *Periodic*, mientras que se va igualando conforme aumenta este. De manera aproximada, la tasa de *FPS* se puede obtener como:

$$FPS \leq \frac{f_{max} \cdot D}{2^n - 1} \approx \frac{f_{max} \cdot D}{(2^n - 1) \cdot (D + 1)} \tag{3.11}$$

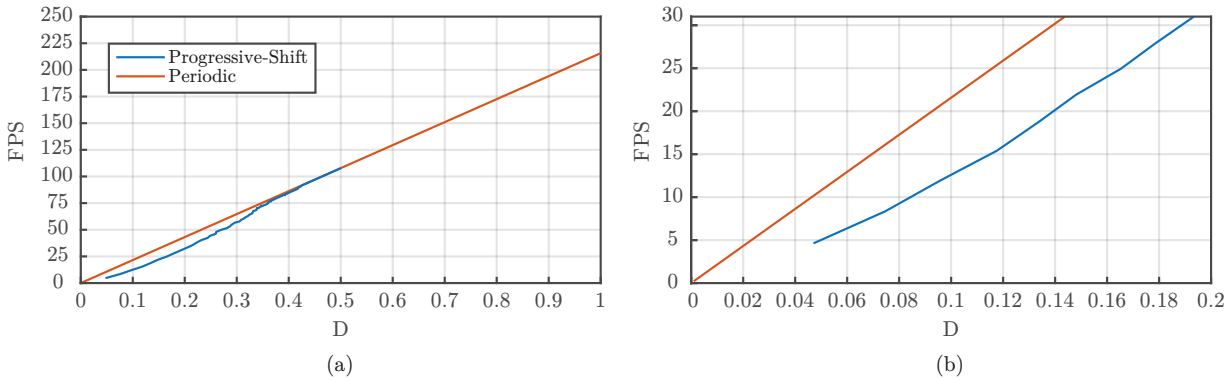


FIGURA 3.17: a) Comparación de la tasa de *FPS* entre *Progressive-Shift* y *Periodic*. b) *Zoom*

Una vez seleccionado el ciclo de trabajo de interés, determinar la especificaciones en términos de τ_{on} y *T_binary_plane* de la ventana para operar a una determinada frecuencia máxima (f_{max}) se realiza siguiendo los siguientes pasos:

1. Se calcula el tiempo que la ventana estará activa (τ_{on}) de acuerdo a la frecuencia máxima a la que se espera que pulsen los píxeles:

$$\tau_{on} = \frac{1}{f_{max}}$$

2. Se calcula el período de los planos (*T_binary_plane*) necesario para obtener el ciclo de trabajo requerido:

$$T\_binary\_plane = \frac{\tau_{on}}{D}$$

3. Se calcula el número de desplazamiento posibles para ese *T_binary_plane* mediante la expresión:

$$divisions = floor\left(\frac{T\_binary\_plane}{\tau_{on}}\right) \tag{3.12}$$

4. Se calcula el ciclo de trabajo medio que se obtendría realmente para ese *T_binary_plane* mediante la expresión:

$$D' = \sum_{m=1}^{divisions} \frac{1}{divisions + m \cdot T\_binary\_plane \cdot f_{max}} \tag{3.13}$$

5. Si $D' \neq D$ sería necesario iterar. Tomar un valor mayor de *T_binary_plane* y volver al paso 3.

3.3.3. Random-Plane

La figura 3.18 muestra los valores de las métricas *RMSE* y *NMI* en función del ciclo de trabajo (D) que se obtienen para los diferentes valores de n utilizando ventanas de tipo *Random-Plane*. En este caso, debido al carácter aleatorio de este tipo de ventanas, el ciclo de trabajo representa el valor medio que se obtiene para un total de 50 iteraciones utilizando un mismo valor de *T_binary_plane*. La desviación típica del ciclo de trabajo con respecto a la media presenta un máximo de 2.7 % ($n = 8$) y un mínimo de 0.15 % ($n = 12$).

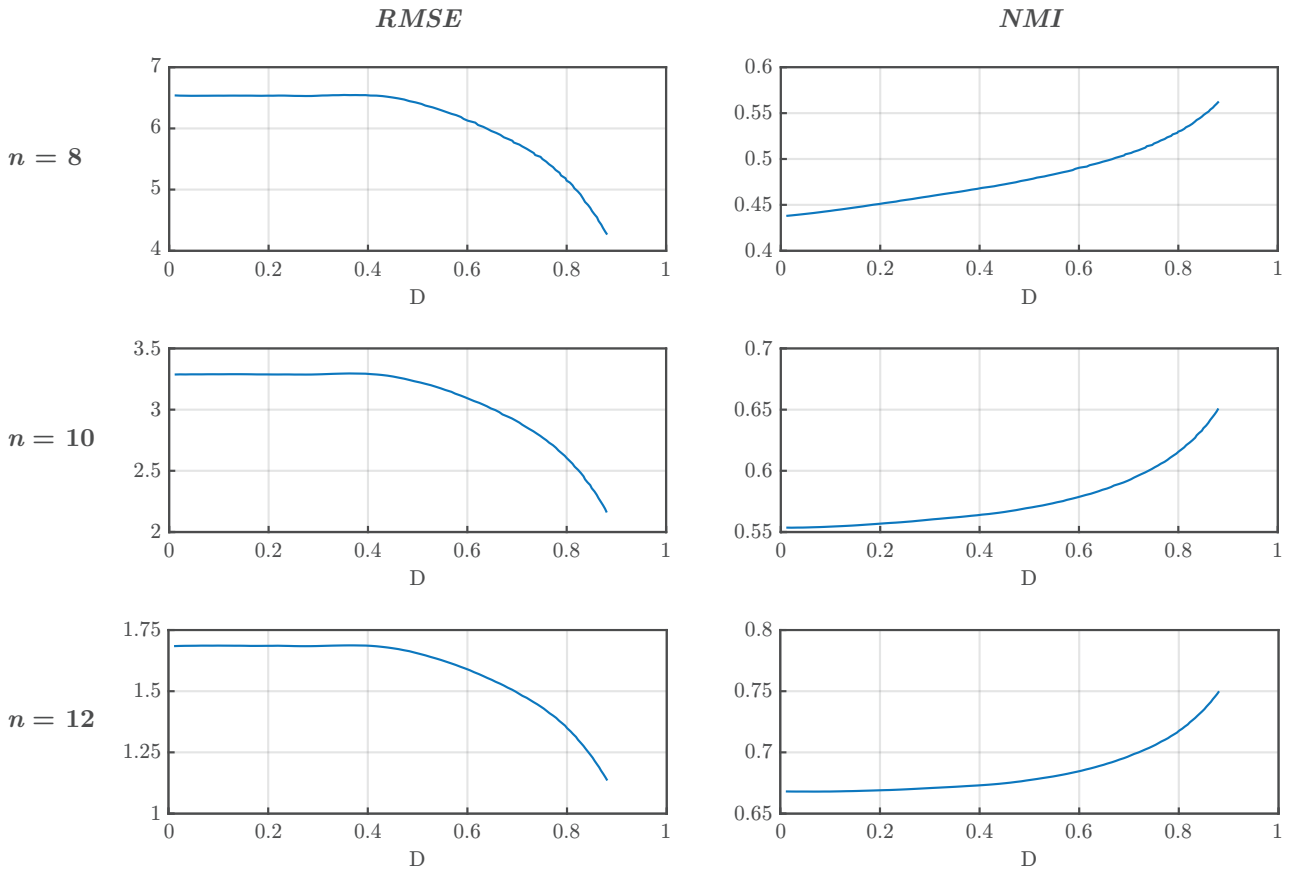


FIGURA 3.18: Métricas en función del ciclo de trabajo para ventanas de tipo *Random-Plane*.

Se puede observar que el *RMSE* permanece muy constante hasta un ciclo de trabajo de 0.4. A partir de ese punto, comienza a disminuir hasta alcanzar un valor mínimo entorno a un 0.85 de ciclo de trabajo. Para $n = 12$ se puede apreciar que se obtiene un valor cercano a la unidad. El hecho de que el ciclo de trabajo se encuentre limitado a un valor máximo de 0.85 se debe a las propiedades de este tipo de ventana (ver subsección 3.2.3). No obstante, intentar forzar ciclos de trabajos superiores, además de disminuir la reducción de eventos, podría derivar en la obtención de periodos de planos inferiores a τ_{on}. En el caso de la *NMI*, se observa que presenta siempre una tendencia creciente que se vuelve más abrupta conforme disminuye el *RMSE*. En este caso, para $n = 12$ llega alcanzar un 0.75 como valor máximo.

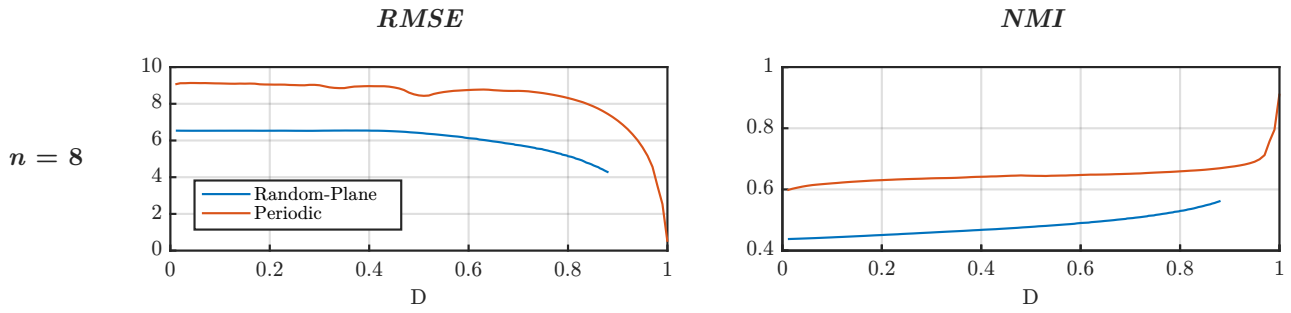


FIGURA 3.19: Comparación de métricas entre *Random-Plane* y *Periodic* para $n = 8$.

Si se superpone a la gráfica anterior de las métricas la obtenida por las ventanas *Periodic* como se muestra en la figura 3.19, se puede apreciar que para un mismo valor de n los resultados del *RMSE* son mejores en la *Random-Plane*, mientras que en el caso de la *NMI* los mejores resultados los obtiene la *Periodic*.

Esto último de acuerdo a la definición de las métricas (ver sección 2.4) tiene sentido. Los picos acentuados que aparecen en la característica de la *Periodic* (ver subsección 3.2.1) provocan un *RMSE* mayor en comparación con la característica de la *Random-Plane* (ver subsección 3.2.3) donde esos picos ya no están presentes. En cambio, el degradado que obtiene esta última es más borroso debido al ruido aleatorio, provocando que la información, y en consecuencia la *NMI*, sean menores que en el caso de la *Periodic*.

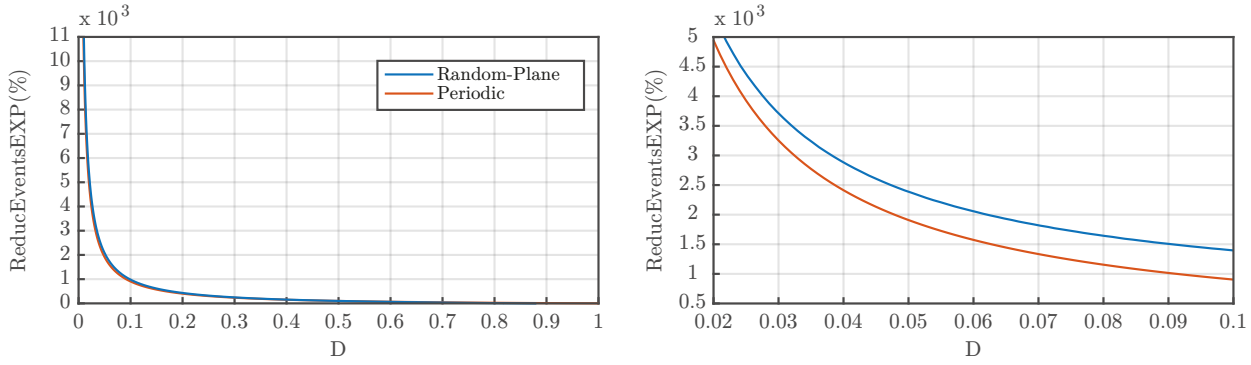


FIGURA 3.20: a) Comparación en la reducción de eventos entre *Random-Plane* y *Periodic*. b) *Zoom*

En la figura 3.20 se muestra la reducción de eventos en forma de porcentaje que produce este tipo de ventana en comparación con la de tipo *Periodic*. Se puede observar que los resultados son bastantes similares. Esto se debe a que la variable pseudoaleatoria asociada a los planos binarios se definió con media cero (ver subsección 3.2.3), por lo que si el número de planos es lo suficientemente grande, los resultados deben coincidir. Por tanto, la reducción del número de eventos vendrá dada aproximadamente por:

$$ReducEventsEXP(\,\%) \approx \left[\frac{1}{D} - 1\right] \times 100 \tag{3.14}$$

En el caso de los *FPS* ocurre algo similar, la figura 3.21 obtenida para una $f_{max} = 55\,kHz$ y $n = 8$ muestra que los resultados son parecidos. Por tanto, la tasa de *FPS* se puede aproximar a:

$$FPS \approx \frac{f_{max} \cdot D}{2^n - 1} \tag{3.15}$$

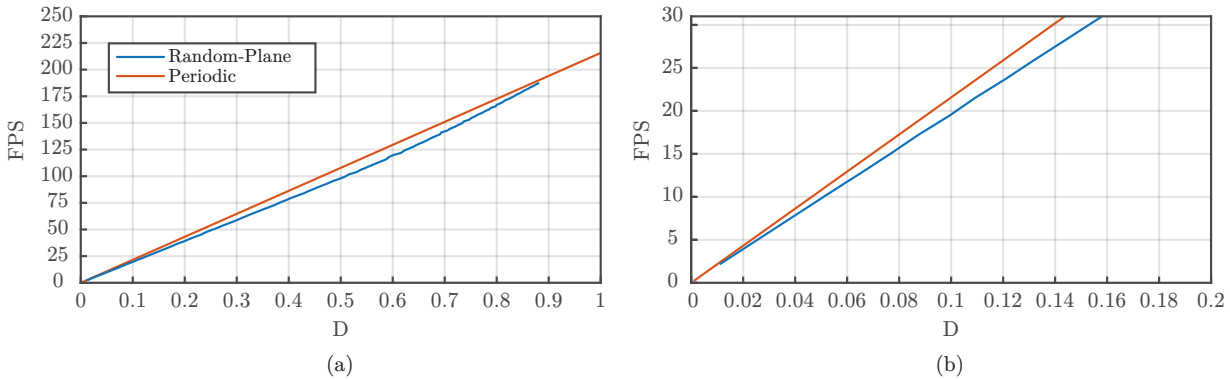


FIGURA 3.21: a) Comparación de la tasa de *FPS* entre *Random-Plane* y *Periodic*. b) *Zoom*

Una vez seleccionado el ciclo de trabajo de interés, el cálculo de las especificaciones en términos de τ_{on} y *T_binary_plane* de la ventana para operar a una determinada frecuencia máxima (f_{max}), se realizaría siguiendo los mismos pasos que se indicaron para el caso de las ventanas *Periodic*.

3.3.4. Arrivaltime-Poisson

En la figura 3.22 se muestran los valores de las métricas *RMSE* y *NMI* en función del ciclo de trabajo (D) que se obtienen para los diferentes valores de n utilizando ventanas de tipo *Arrivaltime-Poisson*. De manera análoga al caso de la *Random-Plane*, debido al carácter aleatorio de este tipo de ventanas, el ciclo de trabajo representa el valor medio que se obtiene para un total de 50 iteraciones utilizando un mismo valor de *T_binary_plane*. La desviación típica del ciclo de trabajo con respecto a la media presenta un máximo de 9.3 % (n = 8) y un mínimo de 0.28 % (n = 12). En cuanto a su valor máximo, ocurre igual que con la *Random-Plane*, se ve limitado a un 0.8 aproximadamente por las propiedades de este tipo de ventana.

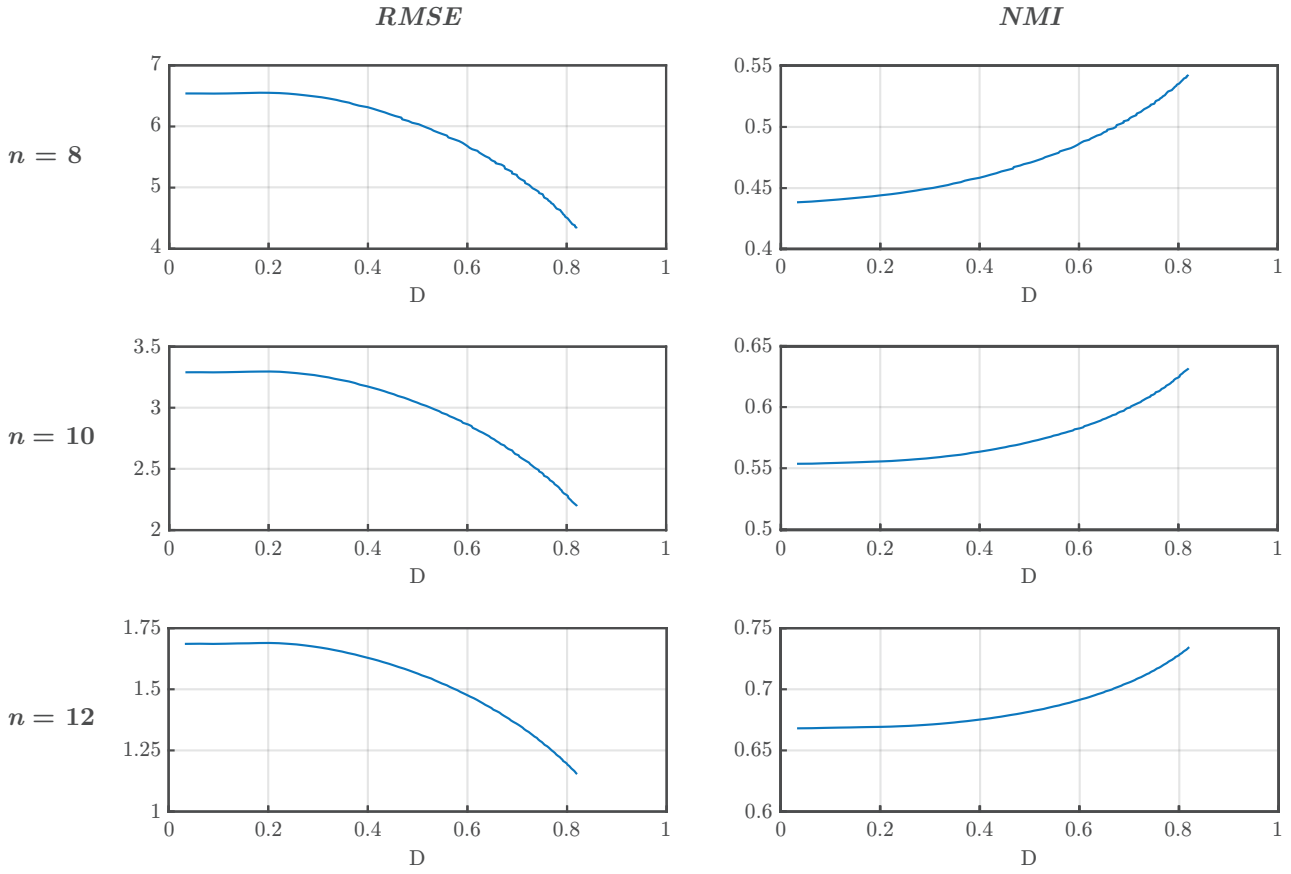


FIGURA 3.22: Métricas en función del ciclo de trabajo para ventanas de tipo *Arrivaltime-Poisson*.

Superponiendo a la gráfica anterior de las métricas las obtenidas por las ventanas *Random-Plane* y *Periodic* se obtienen los resultados que se muestran en la figura 3.23. Se puede apreciar que el comportamiento de la *Arrivaltime-Poisson*, en lo que a métricas se refiere, es muy similar a la *Random-Plane* llegando a obtener valores ligeramente mejores en el *RMSE*.

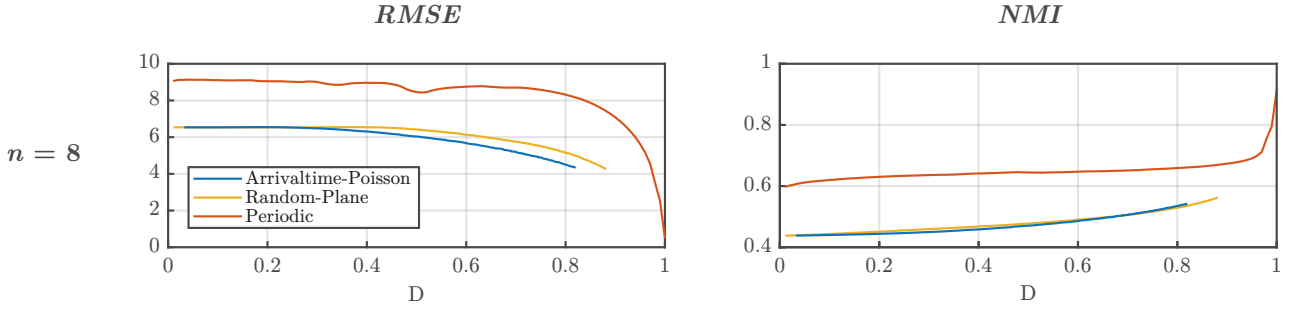


FIGURA 3.23: Comparación de métricas entre *Arrivaltime-Poisson*, *Random-Plane* y *Periodic* para n = 8.

En cambio, como se muestra en la figura 3.24, la reducción de eventos que produce este tipo de ventana dista bastante de la obtenida por la *Periodic* y la *Random-Plane* para un mismo ciclo de trabajo. Realizando un ajuste polinómico, se obtiene que la reducción del número de eventos vendrá dada aproximadamente por:

$$ReducEventsEXP(\%) \geq \left[\frac{1}{D} - 1\right] \times 100 \approx \left[\frac{1}{D'} - 1\right] \times 100 \tag{3.16}$$

siendo $D' = 2.649D^3 - 1.669D^2 + 0.898D - 0.002$

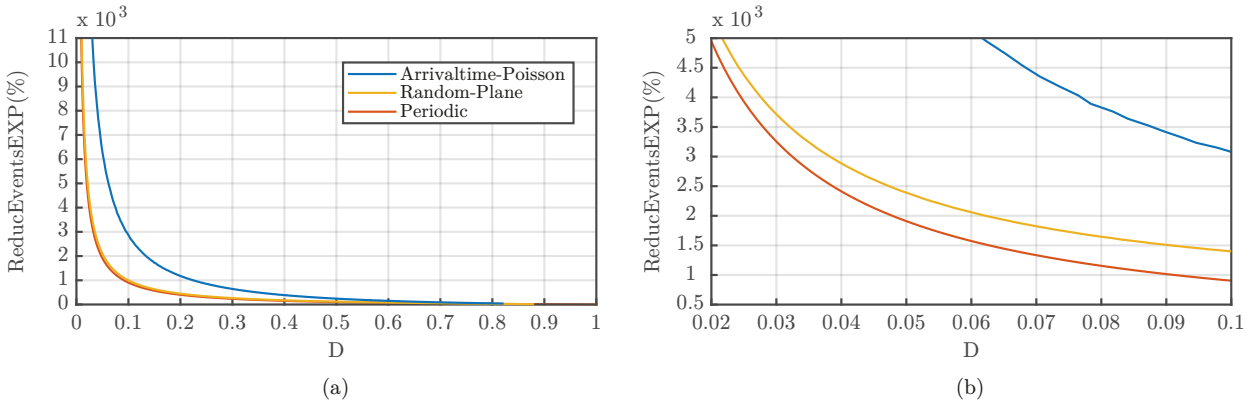


FIGURA 3.24: a) Comparación en la reducción de eventos entre *Arrivaltime-Poisson*, *Random-Plane* y *Periodic*. b) *Zoom*

En el caso de los *FPS* ocurre algo similar, la figura 3.25 obtenida para una $f_{max} = 55\,kHz$ y $n = 8$ muestra que los resultados de la *Arrivaltime-Poisson* se alejan bastante de la *Periodic*. En este caso, la tasa de *FPS* se puede aproximar como:

$$FPS \leq \frac{f_{max} \cdot D}{2^n - 1} \approx \frac{f_{max} \cdot D'}{2^n - 1} \tag{3.17}$$

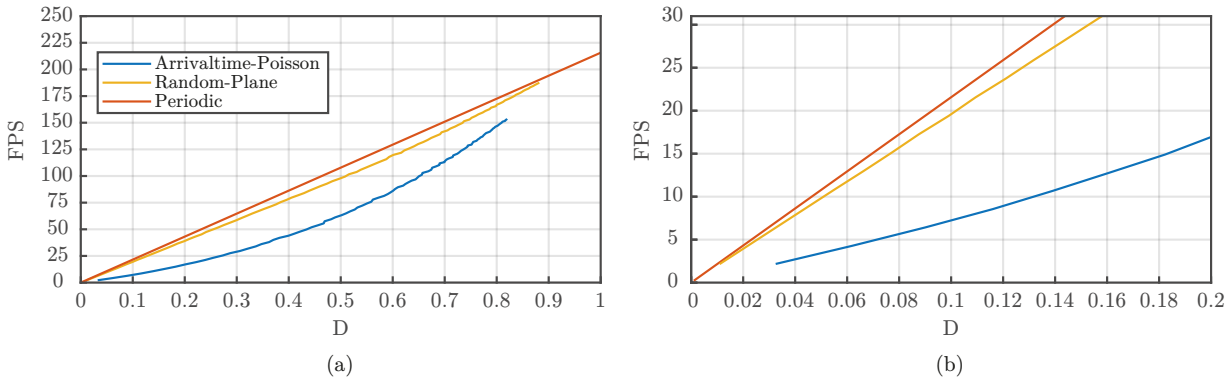


FIGURA 3.25: a) Comparación de la tasa de *FPS* entre entre *Arrivaltime-Poisson*, *Random-Plane* y *Periodic*. b) *Zoom*

Una vez seleccionado el ciclo de trabajo de interés, el cálculo de las especificaciones en términos de τ_{on} y *T_binary_plane* de la ventana para operar a una determinada frecuencia máxima (f_{max}), se realizaría siguiendo los siguientes pasos:

1. Se calcula el tiempo que la ventana estará activa (τ_{on}) de acuerdo a la frecuencia máxima a la que se espera que pulsen los píxeles:

$$\tau_{on} = \frac{1}{f_{max}} \tag{3.18}$$

2. Se calcula el período de los planos (*T_binary_plane*) necesario para obtener el ciclo de trabajo requerido:

$$T\_binary\_plane = \frac{\tau_{on}}{D} \tag{3.19}$$

3.4. Arquitectura de píxel propuesta.

Tomando como punto de partida la arquitectura presentada por Leñero et al. en [11], se introducen los cambios que serían necesarios a nivel de píxel para aplicar *Quanta Imaging* en un sensor de luminancia asíncrono. El resultado es el esquemático que se muestra en la figura 3.26. En azul se destacan las modificaciones realizadas con respecto al esquemático original.

Cabe mencionar que aunque aquí la técnica se ha aplicado a una arquitectura en particular, es independiente de la arquitectura de los píxeles y de la lógica de lectura. El único requisito es que los píxeles utilizados realicen una conversión luz-frecuencia para que sus salidas sean un tren de pulsos.

Se puede observar que solamente es necesario añadir dos llaves, M_{n6} y M_{p4}), gobernadas por la señal de las ventanas *WS* (del inglés, "*Windows Signal*"). De esta forma, solamente cuando la señal de las ventanas se encuentra activa se permite la conexión con la lógica AER. El transistor M_{p4} se ha añadido para asegurar el corte del transistor M_{p3} cuando las ventanas se encuentran a nivel bajo.

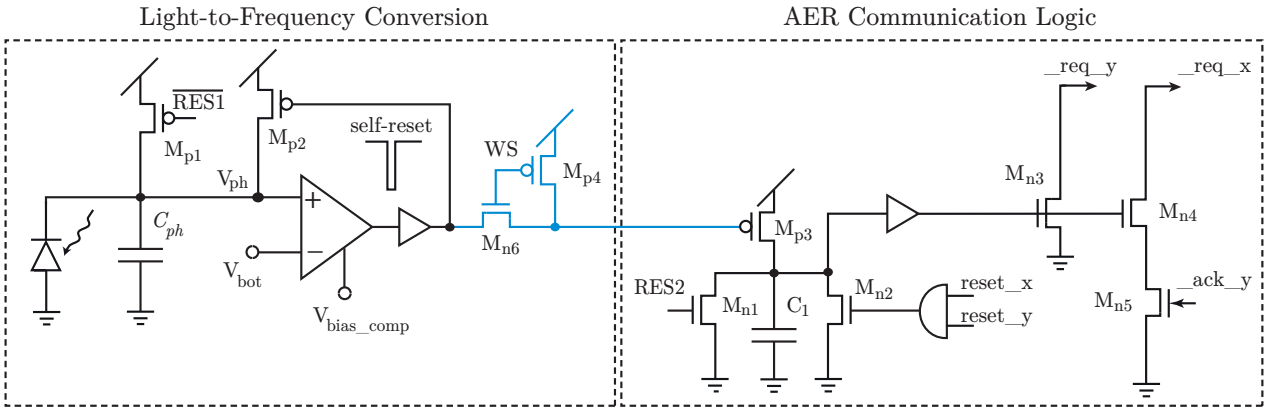


FIGURA 3.26: Esquemático para implementar *Quanta Imaging* en un sensor de luminancia asíncrono.

Por simplicidad, en la figura 3.27 solamente se detalla el cronograma de las señales afectadas por la implementación de *Quanta Imaging*. El funcionamiento de la lógica AER seguiría siendo el mismo al presentado en la introducción.

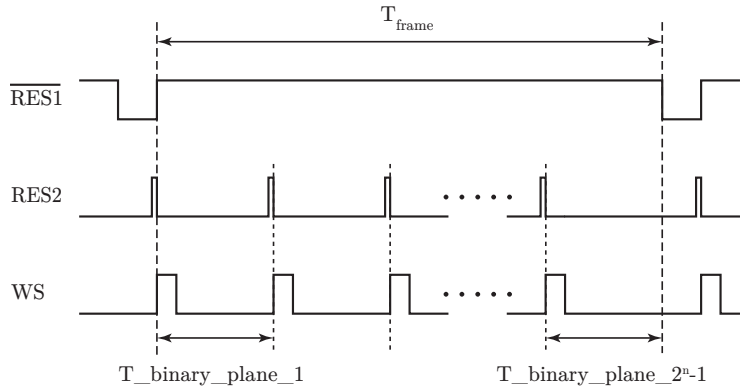


FIGURA 3.27: Cronograma de las señales que intervienen en el funcionamiento del píxel de un sensor de luminancia asíncrono con *Quanta Imaging*.

La operación del píxel comienza con un *reset* global de todos los integradores mediante la señal de control $\overline{RES1}$. Esto permite sincronizar las señales periódicas de los píxeles, premisa que se ha considerado a la hora de desarrollar la función del modelo encargada de contar los pulsos observados en las ventanas para cada píxel. Una vez realizado el *reset* de la matriz, mediante la señal *WS* se aplicarían tantas ventanas como planos binarios se requieran para realizar la reconstrucción de la imagen. El uso de la señal $RES2$ no es imprescindible. Sin embargo, como se mostrará en el capítulo siguiente, los resultados de simulación sugieren que la realización de un *reset* plano a plano de la lógica AER es muy beneficioso cuando se produce la saturación del sistema de arbitraje.

Capítulo 4

Resultados de simulación y validación con datos experimentales

En este capítulo se comienza exponiendo los resultados obtenidos por los cuatro tipo de ventanas utilizando una batería de 40 imágenes reales. Se considera un primer caso de 1 FPS orientado a obtener imágenes visualmente más atractivas, y un segundo caso de 30 FPS que prioriza la velocidad de captura. En ambos casos, se muestran los resultados ideales y los resultados considerando el efecto de la saturación del sistema de arbitraje, tanto a nivel numérico como visual. Por último, se muestra la reconstrucción de una imagen mediante *Quanta Imaging* a partir de los datos experimentales obtenidos por un sensor de luminancia asíncrono real.

4.1. Caso 1: Simulación 1 FPS

En este caso la generación de imágenes a partir de una batería se realiza a una tasa de 1 FPS utilizando 4095 planos ($n = 12$) y considerando que los píxeles pulsan a una frecuencia máxima $f_{max} = 55\,kHz$. Las 40 imágenes utilizadas junto con su índice se encuentran disponibles en el Anexo B. Dichas imágenes tienen una resolución de 16 bits, igual que el degradado del capítulo anterior, por lo que las imágenes generadas serán codificadas en 8 bits. En cuanto a las especificaciones para cada tipo de ventana, obtenidas mediante las expresiones presentadas en 3.3, son las recogidas en la tabla 4.1.

Tabla 4.1: Especificaciones de cada ventana para 1 FPS considerando $n = 12$ y una $f_{max} = 55\,kHz$.

Tipo de ventana	D	τ_{on} [µs]	T_binary_plane [µs]
Periodic	0.075	18.182	244.200
Progressive-Shift	0.080	18.182	736.018
Random-Plane	0.075	18.182	244.200
Arrivaltime-Poisson	0.101	18.182	179.746

Por otra parte, los resultados que se obtienen por simulación en las condiciones anteriores utilizando como imagen un degradado de 16 bits se resumen en la tabla 4.2. Estos valores servirán para contrastar los resultados obtenidos por cada tipo de ventana utilizando la batería de imágenes reales.

Tabla 4.2: Valores de *RMSE*, *NMI* y *ReducEventsEXP*(%) utilizando un degradado de 16 bits con $n = 12$.

Tipo de ventana	*RMSE*	*NMI*	*ReducEventsEXP*(%)
Periodic	1.961	0.859	1234
Progressive-Shift	1.716	0.803	1242
Random-Plane	1.685	0.668	1235
Arrivaltime-Poisson	1.687	0.670	1228

4.1.1. Comparativa Numérica.

En la figura 4.1 se muestran los resultados numéricos ideales obtenidos a partir de la batería por cada tipo de ventana. El eje "x" se corresponde con los índices asociados a cada una de las imágenes, mientras que el eje "y" representa las métricas (*RMSE* y *NMI*, respectivamente) y la reducción del número de eventos en forma de porcentaje (*ReducEventsEXP*(%)). Atendiendo a los gráficos, se puede observar que los resultados, en general, se mantienen en unos valoras similares a los obtenidos en la caracterización (Tabla 4.2). El *RMSE* se encuentra en el orden de 2 para los cuatro tipos de ventanas, a excepción de las imágenes 26 y 32 donde alcanzan valores más altos. Sin embargo, se puede deducir que estos picos en el *RMSE* no altera considerablemente la información en las imágenes, ya que el gráfico del *NMI* no presenta variaciones notables con respecto a lo esperado en esos dos mismos puntos. Finalmente, se puede observar que no existe prácticamente diferencia en la reducción de eventos que produce un tipo de ventana y otro. Esto era de esperar ya que los ciclos de trabajo tiene valores similares y se mantienen iguales entre las distintas imágenes para conseguir una tasa de 1 FPS en cada caso.

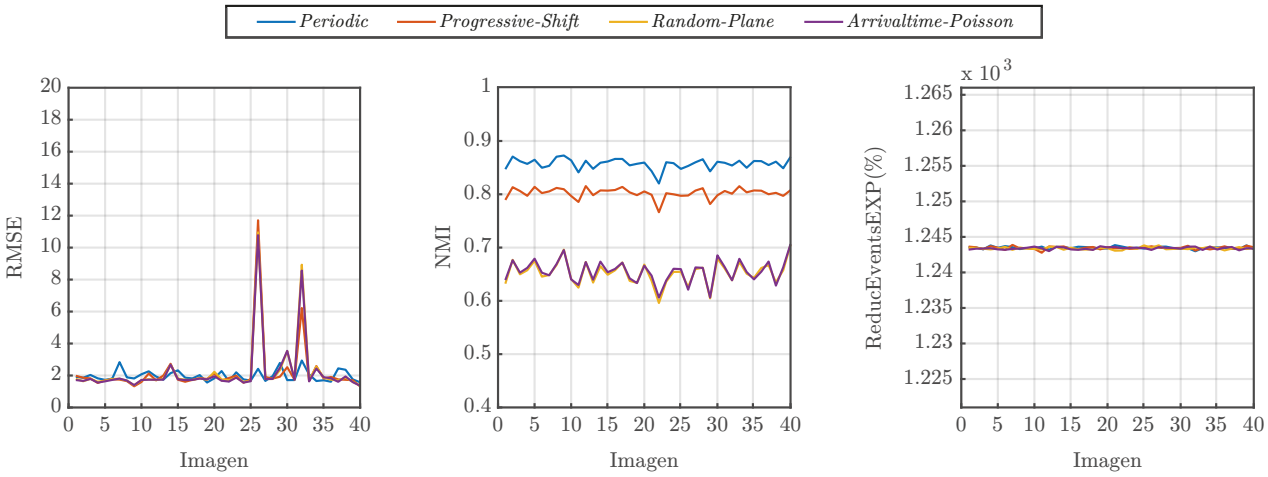


FIGURA 4.1: *RMSE*, *NMI* y *ReducEventsEXP*(%) utilizando batería de imágenes para $n = 12$.

Por otro lado, los resultados numéricos cuando se considera el modelo de saturación del sistema de arbitraje se pueden observar en la figura 4.2. En este caso, no se ha representado la reducción de eventos puesto que dependería de la frecuencia de operación del sistema de arbitraje, tomada como $2MHz$ para este estudio de acuerdo a lo reportado en [10]. Además, se han añadido los resultados que tendría un sensor de luminancia asíncrono saturado que no implementara *Quanta Imaging*. A simple vista se observa que tanto el *RMSE* como el *NMI* empeoran considerablemente con respecto al caso ideal, pero sus valores son notablemente mejores que los obtenidos por el sensor sin *Quanta imaging*. Esto se debe, tal y como pondrá de manifiesto la comparativa visual, a que el uso de ventanas provoca que la probabilidad de lectura de un determinado píxel por parte del sistema de arbitraje sea independiente de su frecuencia. De esta forma, se registran eventos de toda la matriz en lugar de solamente de aquellos píxeles que tienen mayores niveles de iluminación.

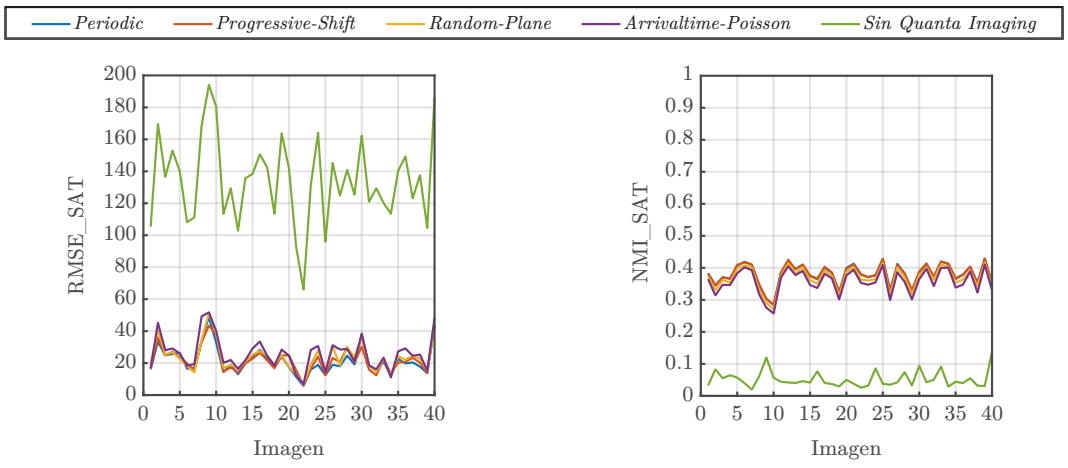


FIGURA 4.2: *RMSE* y *NMI* incluyendo modelo de saturación para $n = 12$.

4.1.2. Comparativa Visual.

En las figuras 4.3, 4.4, 4.5, 4.6 se muestran algunos ejemplos de las imágenes generadas a partir de la batería. En concreto, las figuras 4.3, 4.4 se corresponden con las imágenes que presentaban picos en el *RMSE* (Fig. 4.1), mientras que en la figura 4.5 se observa una imagen de exterior con bastante contraste y en la figura 4.6 la lectura de un *display* digital. Para cada una de las imágenes de muestra se ha representado en la parte superior la imagen obtenida por el sensor de luminancia asíncrono sin (izquierda) y con saturación (derecha) del sistema de arbitraje, mientras que en la parte inferior aparecen las imágenes correspondientes a cada tipo de ventana sin (arriba) y con saturación (abajo) junto con los valores de *RMSE*, *NMI* y número de eventos en cada caso.

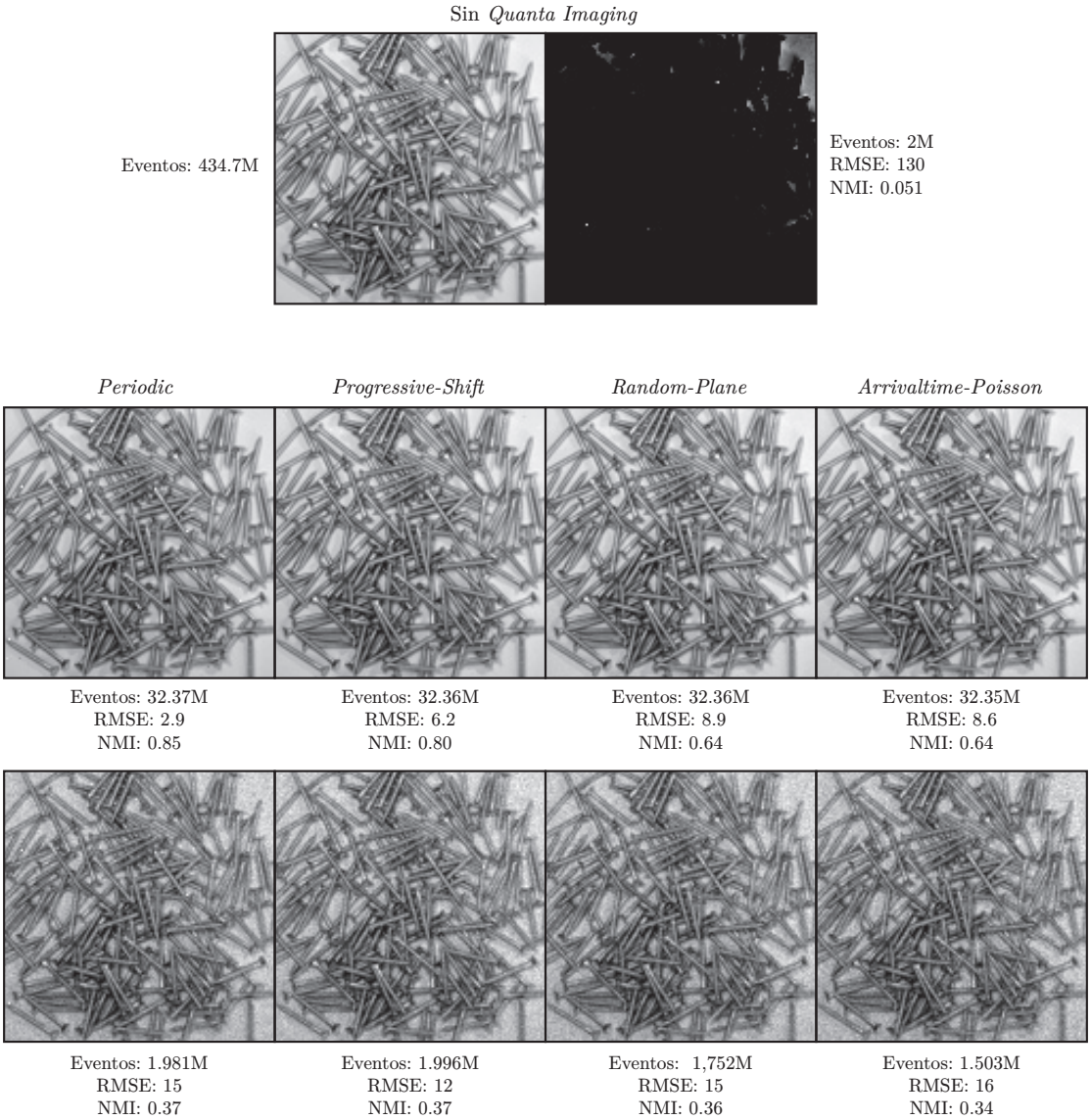


FIGURA 4.3: Resultados generados con la imagen 32. Tiempo de captura: 1 segundo.

En cuanto a las imágenes sin saturación se puede observar que, en todos los ejemplos mostrados, es posible con una cantidad de eventos diez veces menor reconstruir la imagen con un aspecto visual muy parecido al que conseguiría un sensor que no implementara *Quanta Imaging*. Incluso las figuras 4.3 y 4.4 que presentaban valores de *RMSE* más altos de lo esperado, mantienen un resultado visual muy similar.

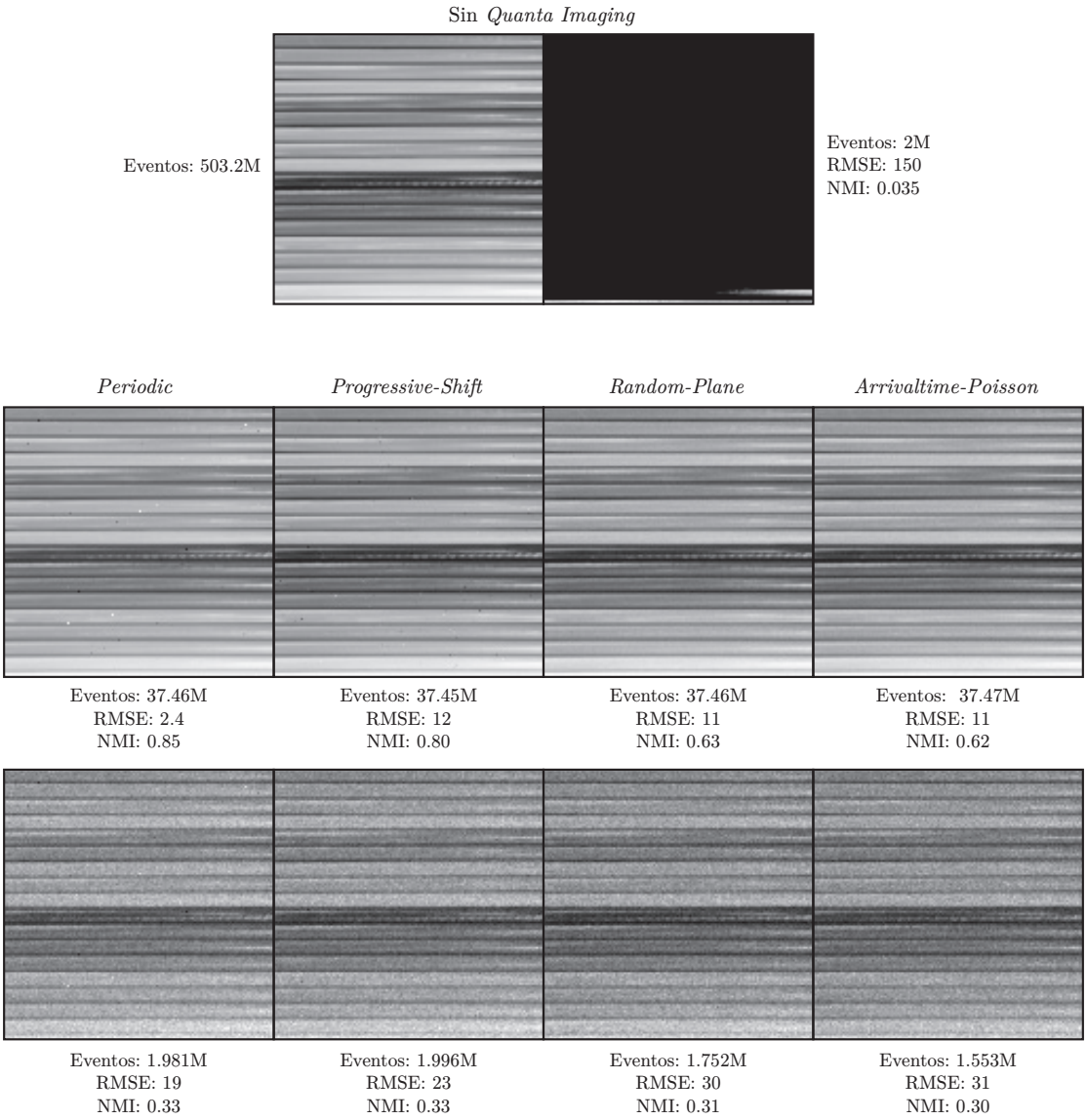


FIGURA 4.4: Resultados generados con la imagen 26. Tiempo de captura: 1 segundo.

Se puede observar que los resultados obtenidos por los diferentes tipos de ventanas apenas presentan diferencias entre ellos. Sin embargo, tal como era de esperar, en el caso de las imágenes obtenidas por las ventanas *Periodic* y *Progressive-Shift* se aprecia un ruido de tipo 'sal y pimienta', aunque menor en esta última, que en el caso de la *Random-plane* y la *Arrivaltime-Poisson* no aparece. Como se ha explicado a lo largo del trabajo este ruido se debe a la sincronización que se produce entre las frecuencias de los píxeles y las ventanas.

Con respecto a las imágenes que se obtienen incluyendo el modelo de saturación del sistema de arbitraje, se puede observar claramente otra de las ventajas que ofrece el uso de ventanas, aparte de la reducción del número de eventos, como es una lectura más equitativa de toda la matriz que evita que solamente unos cuantos píxeles, lo más iluminados, acaparen toda la atención del sistema de arbitraje.

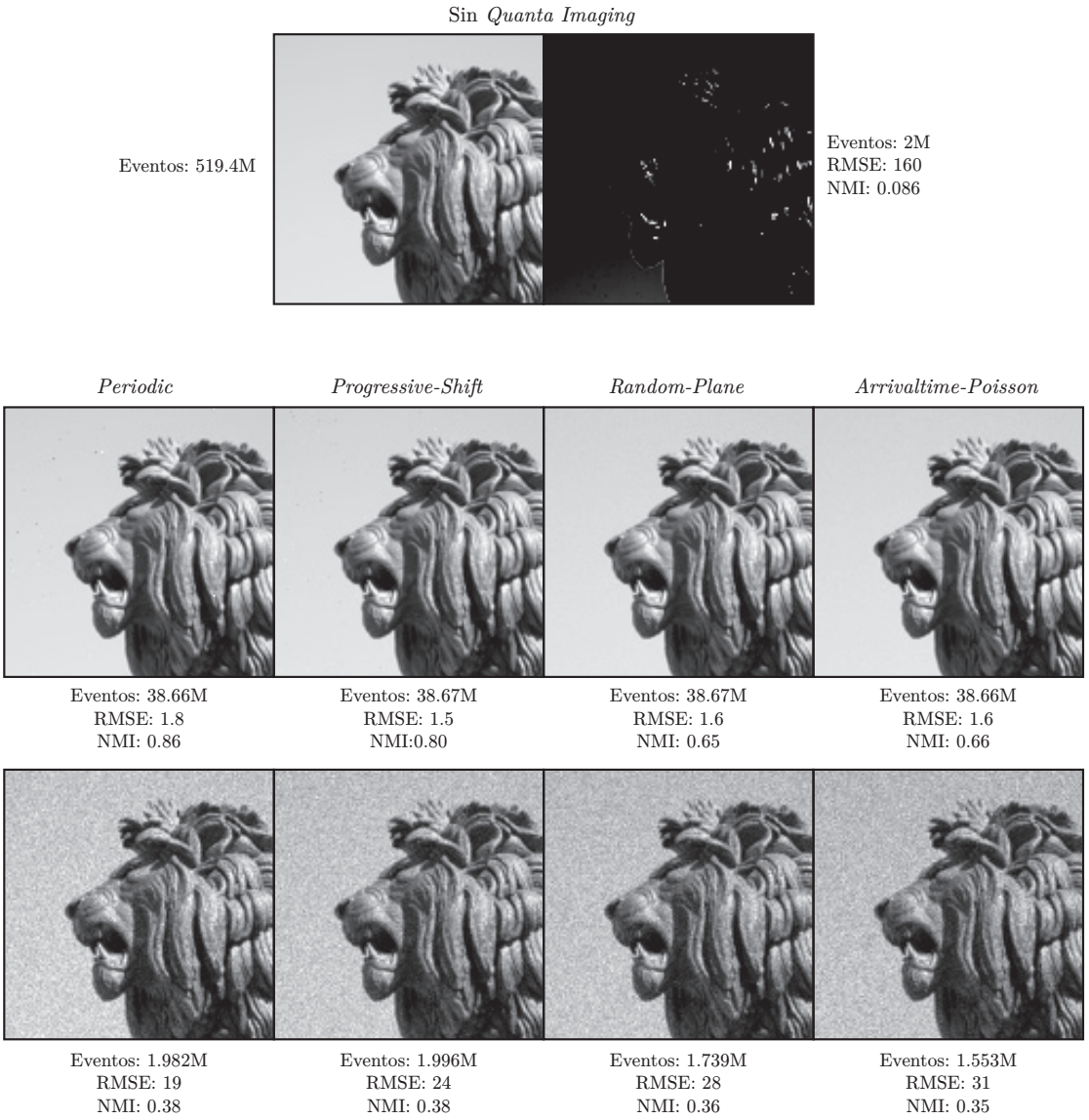


Figura 4.5: Resultados generados con la imagen 24. Tiempo de captura: 1 segundo.

Se observa claramente como el sensor de luminancia asíncrono sin *Quanta Imaging* solo es capaz de atender a los píxeles que pulsan con mayor frecuencia, lo que provoca en la imagen que solamente aparezcan unos cuantos píxeles con un nivel de iluminación máximo mientras que el resto se encuentran totalmente oscuros. En el caso de utilizar ventanas se puede apreciar que, pese a existir un ruido de fondo y una degradación en la calidad de la imagen como apuntaban las gráficas de las métricas, es posible identificar en las imágenes claramente qué objetos aparecen en ellas.

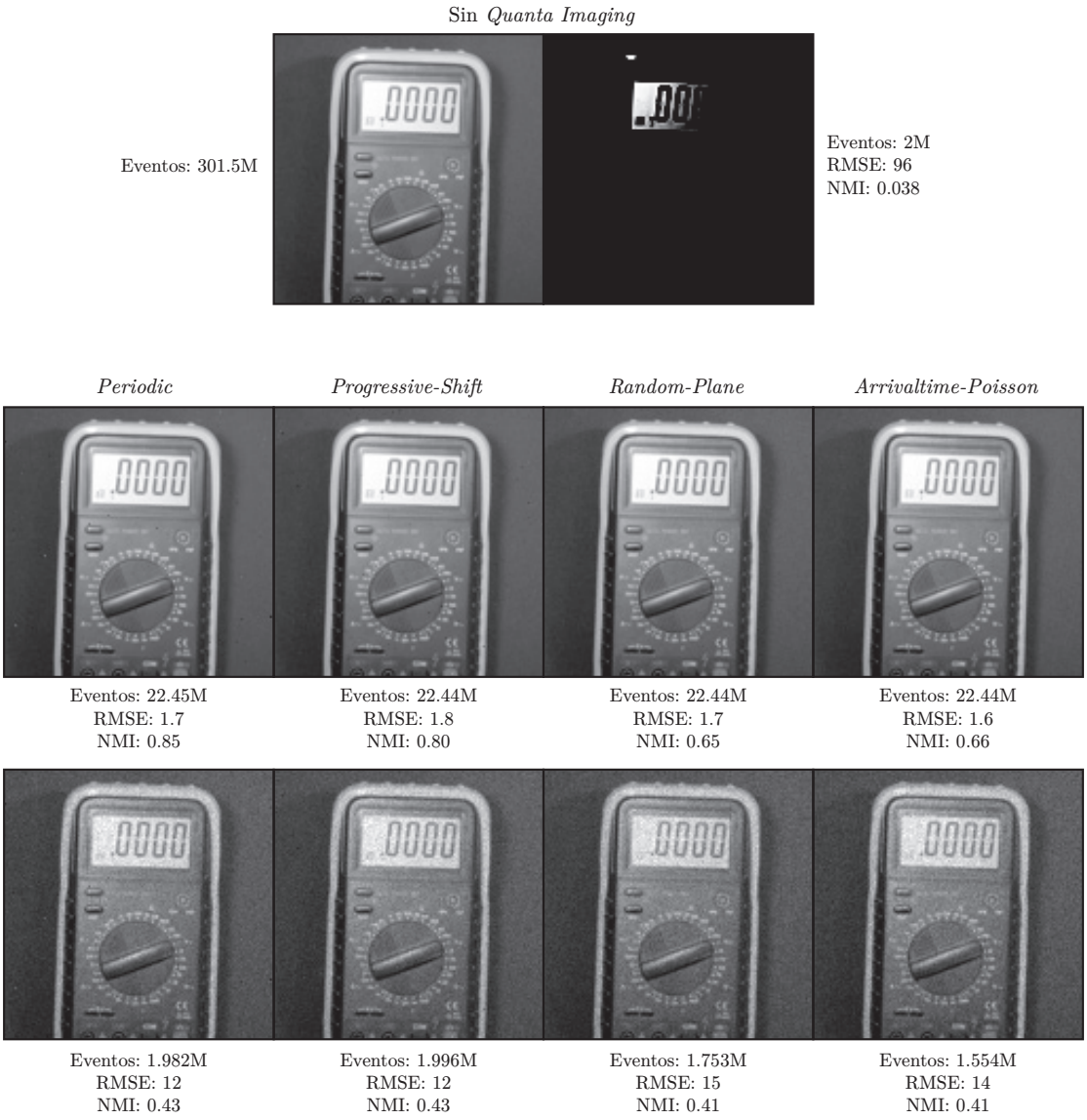


FIGURA 4.6: Resultados generados con la imagen 25. Tiempo de captura: 1 segundo.

4.2. Caso 2: Simulación 30 FPS

En este caso la generación de imágenes a partir de la batería se realiza a una tasa de 30 FPS utilizando 255 planos ($n = 8$) y considerando nuevamente que los píxeles pulsan a una frecuencia máxima $f_{max} = 55\,kHz$. Las imágenes utilizadas son las mismas que para el caso anterior y se encuentran disponibles en el Anexo B. En cuanto a las especificaciones para cada tipo de ventana aparecen recogidas en la tabla 4.3.

TABLA 4.3: Especificaciones de cada ventana para 30 FPS considerando $n = 8$ y una $f_{max} = 55\,kHz$.

Tipo de ventana	D	τ_{on} [µs]	T_binary_plane [µs]
Periodic	0.140	18.182	130.719
Progressive-Shift	0.162	18.182	332.537
Random-Plane	0.140	18.182	130.719
Arrivaltime-Poisson	0.213	18.182	85.398

Los resultados que se obtienen en las condiciones anteriores utilizando como imagen un degradado de 16 bits se resumen en la tabla 4.4. Estos valores servirán nuevamente para contrastar los resultados obtenidos por cada tipo de ventana utilizando la batería de imágenes.

TABLA 4.4: Valores de *RMSE*, *NMI* y *ReducEventsEXP*(%) utilizando un degradado de 16 bits con $n = 8$.

Tipo de ventana	*RMSE*	*NMI*	*ReducEventsEXP*(%)
Periodic	9.092	0.625	616
Progressive-Shift	6.500	0.526	618
Random-Plane	6.528	0.448	623
Arrivaltime-Poisson	6.456	0.451	619

4.2.1. Comparativa Numérica.

En la figura 4.7 se han representado los resultados de manera similar al caso de 1 FPS sin saturación. Como se observa, los resultados también son similares a los obtenidos en la caracterización (Tabla 4.4). Cabe destacar que los picos del *RMSE* se siguen produciendo en las mismas imágenes que anteriormente. Viendo estos resultados, es razonable pensar que el aumento del *RMSE* con respecto al valor esperado se deba a una estimación errónea de algún nivel de iluminación muy frecuente en la imagen. Sin embargo, como siempre se le asigna el mismo valor, no se aprecia visualmente una pérdida de información notable en las imágenes. En general, y como era de esperar, las métricas empeoran al reducir el número de planos, ya que la estimación de los niveles de brillo de cada uno de los píxeles se realiza con un número menor de muestras de la escena.

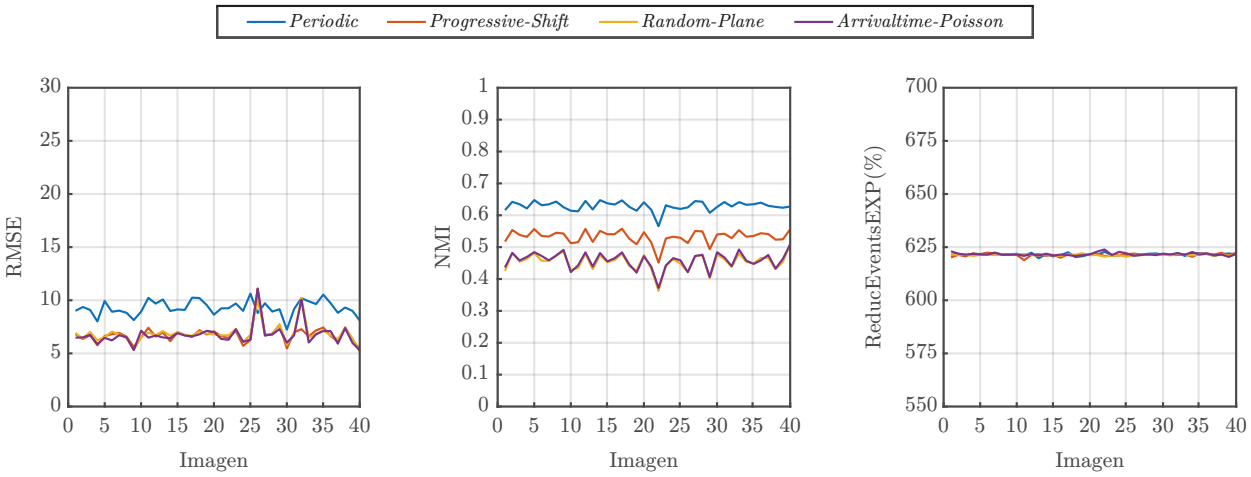


FIGURA 4.7: *RMSE*, *NMI* y *ReducEventsEXP*(%) utilizando batería de imágenes para $n = 8$.

En cuanto a los resultados considerando el modelo de saturación, representados en la figura 4.8, empeoran con respecto al caso ideal y con respecto al caso de 1 FPS debido a la reducción en el número de planos utilizados. El *RMSE* es prácticamente la mitad cuando se aplican las ventanas, pero la *NMI* no presenta prácticamente diferencias apreciables.

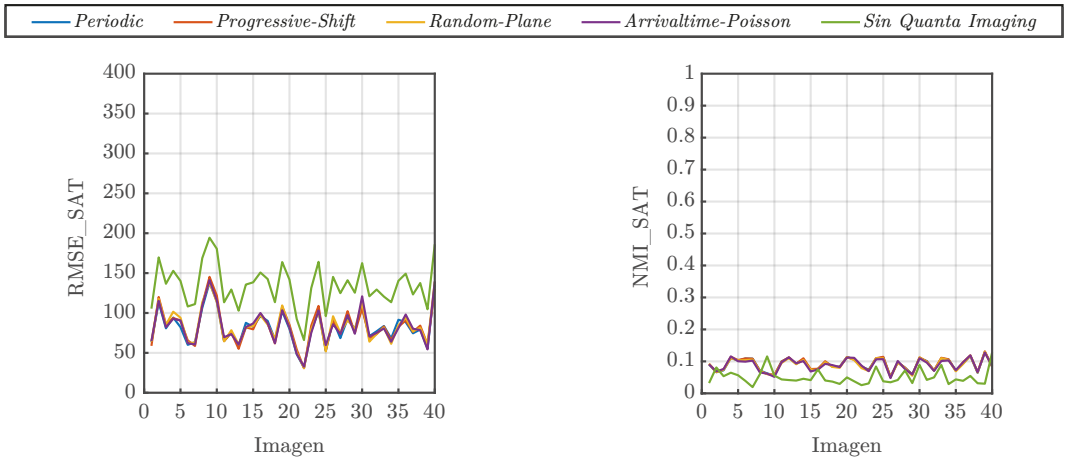


FIGURA 4.8: *RMSE* y *NMI* incluyendo modelo de saturación para $n = 8$.

4.2.2. Comparativa Visual.

En las figuras 4.9, 4.10, 4.11, 4.12 se muestran los mismos ejemplos que se utilizaron en el caso de estudio de 1 FPS. En cuanto a los resultados ideales, los diferentes tipos de ventanas presentan un comportamiento similar. Sigue apareciendo el ruido tipo 'sal y pimienta' en las ventanas *Periodic* y *Progressive-Shift*, mientras que la *Random-Plane* y la *Arrivaltime-Poisson* lo consiguen eliminar a costa de aumentar el ruido de fondo.

Con respecto a los resultados considerando el modelo de saturación, en el caso de las imágenes generadas por un sensor asíncrono que no implemente *Quanta Imaging* son similares al caso de 1 FPS. En el caso de las diferentes ventanas, como cabría esperar de los resultados numéricos, las imágenes presentan un aspecto visual notablemente peor. Aparece bastante más ruido que dificulta la visualización de texturas y objetos pequeños, como ocurre en las figuras 4.9, 4.10. No obstante, pese a la gran reducción que se produce en el número de eventos las formas y contornos siguen siendo visibles, como ocurre con el león de la figura 4.11 o el multímetro de la figura 4.12.

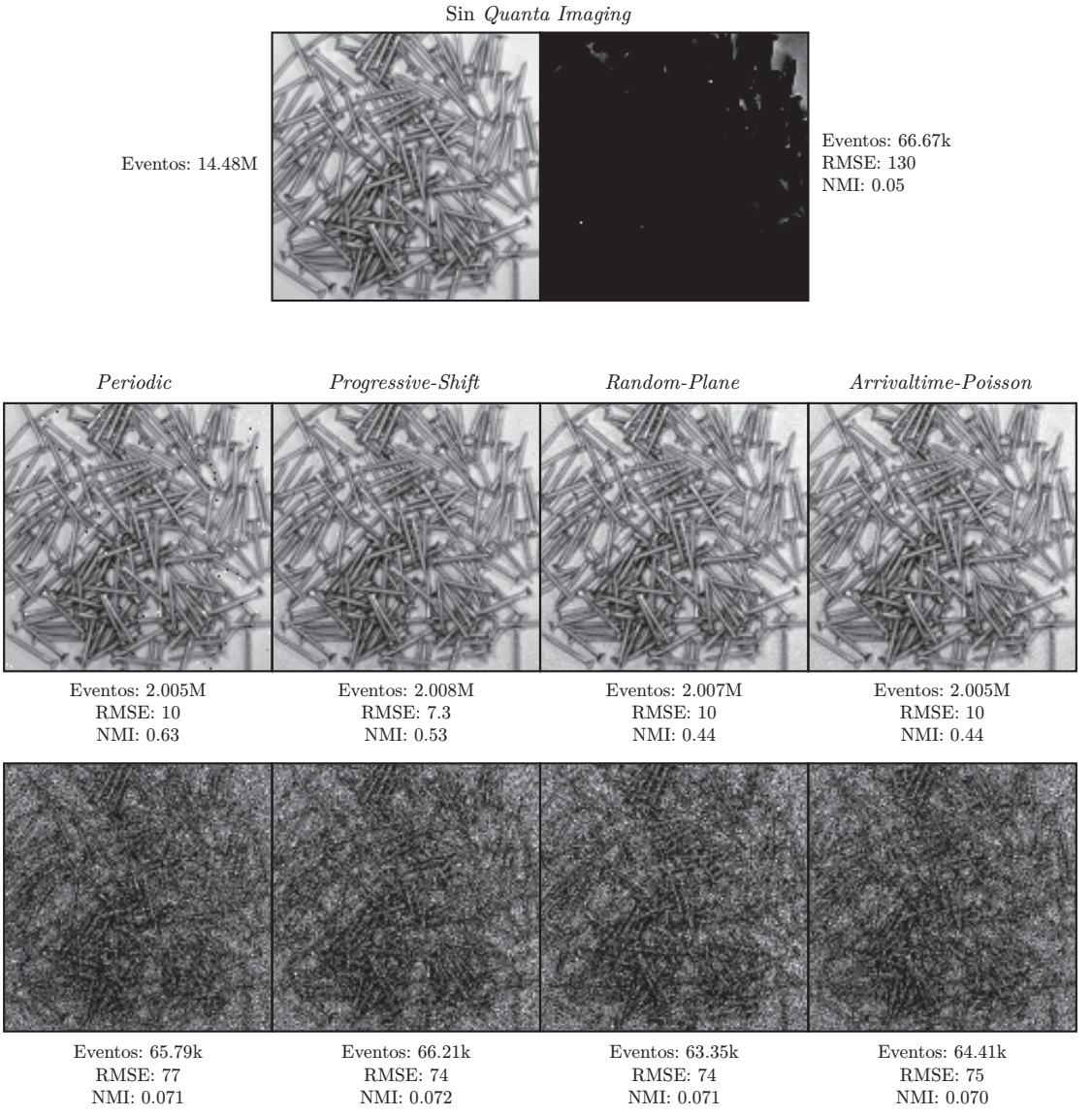


FIGURA 4.9: Resultados generados con la imagen 32. Tiempo de captura: 1/30 segundos.

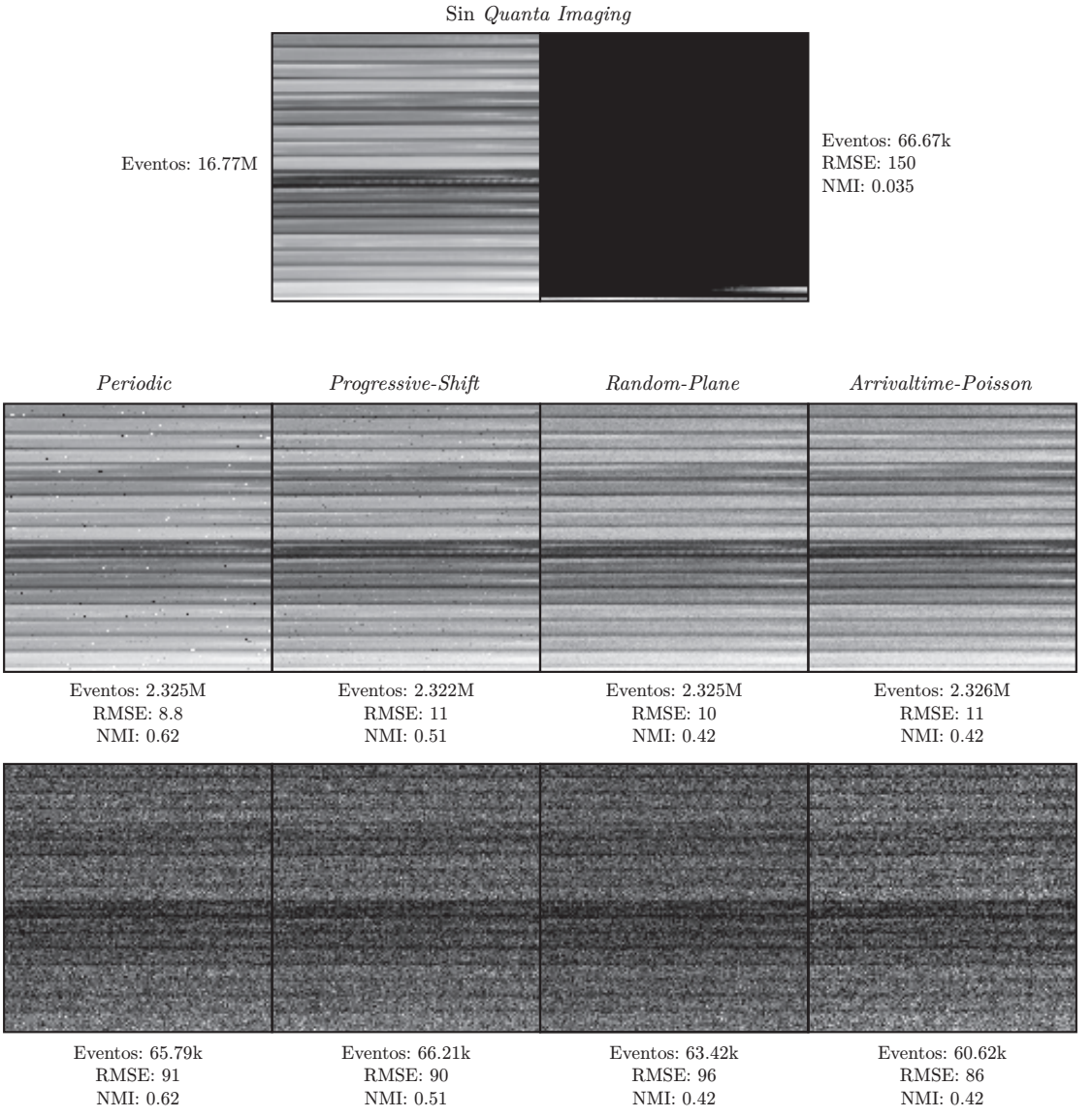


FIGURA 4.10: Resultados generados con la imagen 26. Tiempo de captura: 1/30 segundos.

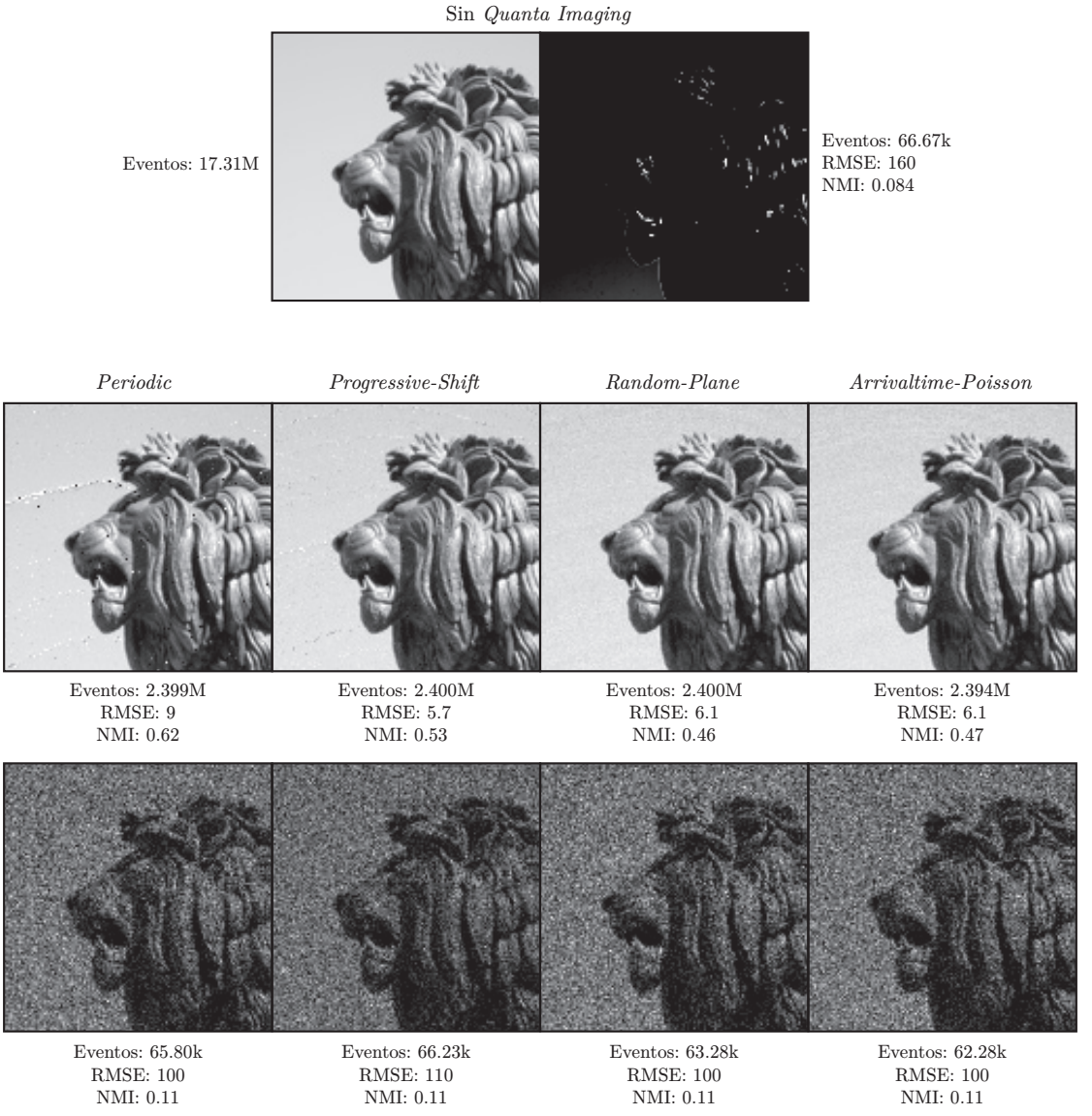


FIGURA 4.11: Resultados generados con la imagen 24. Tiempo de captura: 1/30 segundos.

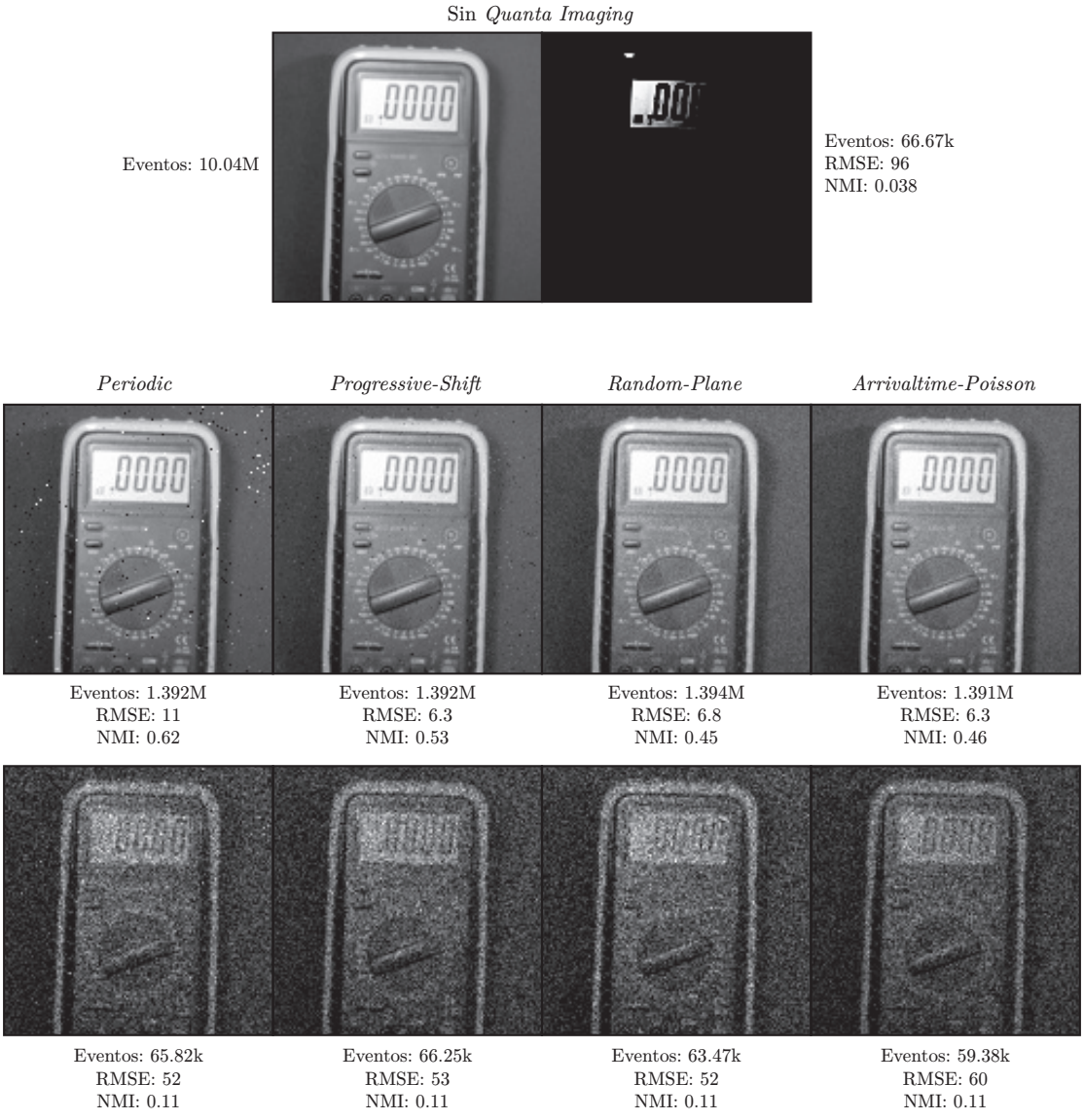


FIGURA 4.12: Resultados generados con la imagen 25. Tiempo de captura: 1/30 segundos.

4.3. Caso 3: Aplicación en datos experimentales

Para concluir este capítulo se probará la técnica con un conjunto de datos experimentales obtenidos de un sensor real de luminancia asíncrono en un entorno de laboratorio. Las condiciones de iluminación en que se realizaron las medidas garantizaban que el sistema de arbitraje no estuviese saturado. Los datos fueron almacenados en forma de tabla, de manera que cada fila se corresponde con un nuevo evento mientras que las columnas contienen la dirección del píxel correspondiente y el instante de lectura de dicho evento. La reconstrucción de una imagen a partir de los datos consistiría en asignar el nivel de iluminación en función del número de veces que han pulsado los distintos píxeles durante el tiempo que duró la captura de los datos.

En el caso de realizar la reconstrucción mediante *Quanta Imaging* se debe, en primer lugar, generar las ventanas de observación y determinar qué píxeles pulsan dentro de las mismas. Puesto que la tabla contiene únicamente los instantes de lectura de los píxeles, la mejor aproximación para emular la aplicación de las ventanas consiste en discriminar directamente los eventos que tienen una marca temporal comprendida entre los instantes de inicio y fin de cada una de ellas. Tras esto, se procedería igual que en el caso anterior, se realizaría un conteo del número de veces que ha pulsado cada píxel tras el filtrado para asignarles un nivel de iluminación.

Los resultados obtenidos a partir de los datos se representan en la figura 4.13. A la izquierda se muestran dos imágenes distintas sin aplicar *Quanta Imaging*, mientras que a la derecha se muestran tras aplicar las ventanas de observación.

FIGURA 4.13: Ejemplos de imágenes reconstruidas a partir de datos experimentales.

Se puede apreciar que dichas imágenes son bastantes oscuras. Para obtener una representación visualmente más atractiva, se ha optado por aplicar una curva de *Tone Mapping* que aumente el rango de grises de los píxeles más oscuros y limite el de los más iluminados. En [10] se pueden encontrar ejemplos concretos de histogramas

y las funciones lineales a tramos definidas para mejorar la visualización de las imágenes.

Para este caso en concreto, la elección de la curva se ha realizado siguiendo los siguientes pasos:

1. Cálculo del histograma de los niveles de iluminación de la imagen.
2. Determinación de los valores de iluminación más frecuentes atendiendo a los picos del histograma.
3. Definición de una función lineal a tramos que asigne más niveles de gris a los valores de iluminación en torno a los cuales se sitúan los picos del histograma.

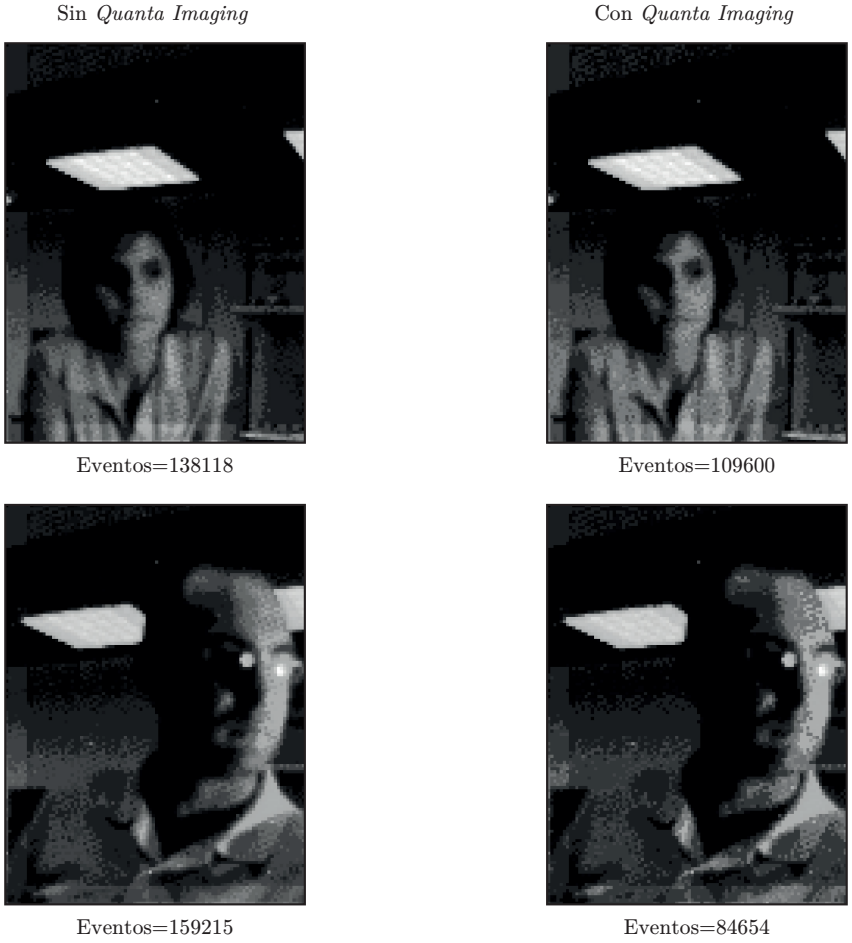


FIGURA 4.14: Imágenes reconstruidas aplicando *Tone Mapping*.

Como se puede apreciar, la imagen sigue siendo considerablemente oscura. La razón es que el sensor con el que se realizaron las capturas estaba diseñado para generar tasas de eventos bajas en las zonas poco iluminadas. Debido a esto, y a las condiciones del experimento, un número importante de píxeles no llegaron a pulsar nunca, de modo que la información asociada a estos no puede ser mejorada mediante *Tone Mapping*. Sin embargo, en el caso de los píxeles que sí pulsaron, la información es mejorada y se consiguen diferenciar los retratos que aparecen en primer plano así como algunas formas que se encuentran al fondo sin perder la información relativa al foco de luz del techo.

Con este experimento se comprueba que al aplicar las ventanas en un caso en el que el número de eventos es ya de por sí reducido, se puede reconstruir la imagen obteniendo un resultado equivalente.

Por último, en la figura 4.15 se muestran los histogramas y las curvas de *Tone Mapping* utilizadas para cada una de las imágenes resultantes.

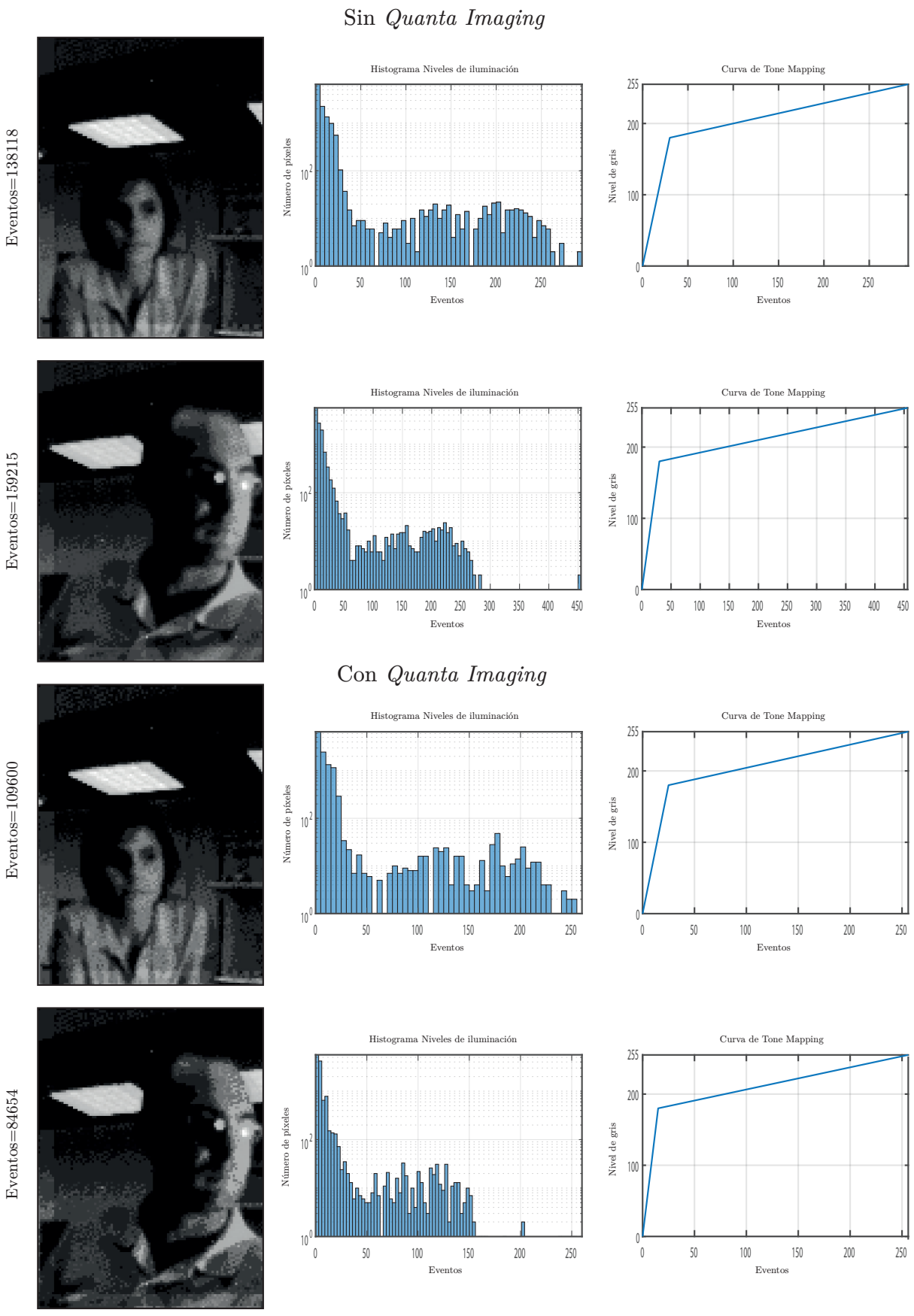


FIGURA 4.15: Histogramas y curvas de *Tone Mapping*.

Capítulo 5

Conclusiones

El estudio realizado en este trabajo muestra que la técnica de adquisición de datos *Quanta Imaging* se presenta como una opción viable para reconstruir imágenes en sensores de luminancia asíncronos o de tipo 'octopus' con un número de eventos reducido. Esta reducción es muy interesante ya que podría permitir reducir el consumo del sistema de arbitraje, dificultar su saturación o incluso ampliar el tamaño de la matriz.

Se ha conseguido realizar una caracterización de cuatro tipos de ventana que permite ilustrar de manera clara e intuitiva las diferentes consideraciones de diseño a tener en cuenta a la hora de seleccionar el ciclo de trabajo de las mismas. Esto permite, dadas unas determinadas especificaciones de sistema, optimizar la configuración de éstas tanto para aplicaciones que requieran de una mayor calidad de imagen, como para aplicaciones que requieran una mayor tasa de FPS.

La metodología desarrollada para configurar las ventanas permite también el escalado de la técnica a otras frecuencias de operación, ya que el tamaño de las ventanas se ajusta de acuerdo a la frecuencia máxima a la que se espera que pulsen los píxeles. En este sentido, es importante destacar que los valores de las métricas de *RMSE* y *NMI*, así como la reducción de eventos que consigue la técnica desarrollada, son independientes de la frecuencia de operación. Por lo que los resultados son escalables a otras frecuencias. No obstante, las tasas de FPS que se podrían alcanzar sí que dependen de la frecuencia, las cuales aumentan conforme aumenta esta última a costa, eso sí, de incrementar la tasa de eventos por segundo.

De lo anterior se deduce que el uso de la técnica no estaría limitado exclusivamente a los sensores de luminancia asíncronos, podría implementarse en cualquier arquitectura de sensor cuyos píxeles presenten una salida en forma de tren de pulsos. Posibles candidatos que podrían beneficiarse de su uso serían los sensores basados en dispositivos sensibles a un único fotón, como es el caso de los SPADs, cuyas tasas de eventos son difíciles de manejar debido a que los píxeles pulsan con frecuencias en el orden de los MHz.

En cuanto a los resultados de simulación muestran que los diferentes tipos de ventanas desarrollados ofrecen resultados muy parecidos. A nivel numérico, tanto los valores de las métricas (*RMSE* y *NMI*) como el número de eventos dadas unas determinadas condiciones de operación, resultan similares. A nivel visual, para un número de planos reducidos las diferencias son más notorias. En el caso de las ventanas periódicas (*Periodic* y *Progressive-Shift*) destaca el ruido de tipo 'sal y pimienta', mientras que en el caso de las aleatorias (*Random-Plane* y *Arrivaltime-Poisson*) descata el ruido aleatorio. Conforme aumenta el número de planos, las diferencias entre los resultados de un tipo de ventana y otro se reducen cada vez más. Por tanto, un criterio para elegir entre el uso de una ventana u otro podría venir dado por la implementación real de estas. En el caso de las periódicas, parece más adecuado seleccionar la *Periodic* por presentar una dificultad técnica que se puede intuir menor. De igual forma, en el caso de las aleatorias, la elección de la *Random-Plane* podría resultar más adecuada desde el punto de vista práctico. Otro criterio podría venir dado si las imágenes obtenidas van a ser *post*-procesadas. El ruido sal y pimienta al provocar picos tan notorios sobre la característica de la imagen es más fácil de filtrar que el ruido de tipo aleatorio que presenta una respuesta mucho más plana sobre la característica.

De manera general, se puede concluir que el uso de ventanas que impliquen patrones sistemáticos presentarán ruido de tipo 'sal y pimienta' debido a la sincronización de las frecuencias de los píxeles con las ventanas, mientras que las que se caractericen por ser aleatorias tendrán un ruido de fondo aleatorio debido a la propia aleatoriedad del proceso de generación. En ambos casos, tal y como se expuso en el capítulo anterior, el ruido será menor conforme aumente el número de planos binarios.

En el caso de los resultados de las simulaciones incluyendo los modelos de saturación desarrollados, sugieren que la técnica también ofrece la ventaja de permitir una lectura más equitativa de los píxeles, lo que a su vez permite evitar o reducir el problema de que solo unos cuantos píxeles acaparen toda la atención del sistema de arbitraje.

Un punto fuerte a destacar de la técnica es que tanto la mejora en el caso de que exista saturación, como la reducción en el número de eventos, solo requiere añadir dos llaves adicionales a nivel de píxel.

Por otra parte, la reconstrucción de imágenes realizada a partir los datos experimentales obtenidos en un laboratorio mediante un sensor real, resulta ser una prueba valiosa para comprobar y demostrar la validez de la técnica en una situación más próxima a la realidad.

Para finalizar, con el objetivo de dar continuidad al trabajo aquí desarrollado y explotar las ventajas que ofrece la técnica, se plantean las siguientes líneas de trabajo futuras:

- Implementación en Verilog de los diferentes tipos de ventanas. La generación de ventanas temporales con una resolución en el orden de los microsegundos con las tecnologías actuales es asequible. Sin embargo, la generación de ventanas con resoluciones en el orden de los nanosegundos no es trivial, pudiendo presentar una serie de limitaciones prácticas que deben ser estudiadas.

- Aprovechar la información temporal de las ventanas y combinarla con la información del número total de pulsos que han sido observados de cada píxel para realizar una mejor estimación del nivel de iluminación de los píxeles.

- Estudiar el efecto de que un píxel pueda pulsar más de 1 vez dentro de la misma ventana debido a la resolución discreta que tendrá el generador de ventanas real frente a la resolución infinita del espectro de frecuencias que presenta la luz.

- Desarrollar y estudiar nuevos tipos de ventanas.

Referencias

[1] M. A. Mahowald and C. Mead, *The Silicon Retina*, vol. 264. 1991.

[2] J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Applications of event-based image sensors—Review and analysis," *International Journal of Circuit Theory and Applications*, vol. 46, no. 9, pp. 1620–1630, 2018.

[3] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE J Solid State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[4] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120dB 30mW Asynchronous Vision Sensor that Responds to Relative Intensity Change," *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, pp. 844–847, 2006.

[5] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," *Experiments in Fluids*, vol. 51, no. 5, pp. 1465–1469, 2011.

[6] M. Litzenberger, B. Kohn, G. Gritsch, N. Donath, C. Posch, N. A. Belbachir, and H. Garn, "Vehicle Counting with an Embedded Traffic Data System using an Optical Transient Sensor," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 36–40, 2007.

[7] Y. Hu, H. Liu, M. Pfeiffer, and T. Delbruck, "DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition," *Frontiers in Neuroscience*, vol. 10, no. AUG, pp. 1–5, 2016.

[8] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "CIFAR10-DVS: An Event-Stream Dataset for Object Classification," *Frontiers in Neuroscience*, vol. 11, no. MAY, 2017.

[9] 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.

[10] J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodríguez-Vázquez, "A Wide Linear Dynamic Range Image Sensor Based on Asynchronous Self-Reset and Tagging of Saturation Events," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 6, pp. 1605–1617, 2017.

[11] J. A. Leñero-Bardallo, L. Farian, J. M. Guerrero-Rodriguez, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Sun Sensor Based on a Luminance Spiking Pixel Array," *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6578–6588, 2017.

[12] J. A. Leñero-Bardallo, P. Häfliger, R. Carmona-Galán, and A. Rodríguez-Vázquez, "A Bio-Inspired Vision Sensor With Dual Operation and Readout Modes," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 317–330, 2016.

[13] Ł. Farian, P. Häfliger, and J. A. Leñero-Bardallo, "A Miniaturized Two-Axis Ultra Low Latency and Low-Power Sun Sensor for Attitude Determination of Micro Space Probes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1543–1554, 2018.

[14] J. A. Leñero-Bardallo, M. Delgado-restituto, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Sensitivity to Illumination," vol. 65, no. 11, pp. 3854–3863, 2018.

[15] J. A. Leñero-Bardallo, D. H. Bryn, and P. Häfliger, "Bio-inspired Asynchronous Pixel Event Tricolor Vision Sensor," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 3, pp. 345–357, 2014.

[16] J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodríguez-Vázquez, "A high dynamic range image sensor with linear response based on asynchronous event detection," *2015 European Conference on Circuit Theory and Design, ECCTD 2015*, 2015.

[17] P. Häfliger, *A spike based learning rule and its implementation in analog hardware.* PhD thesis, 2000.

[18] K. A. Boahen, "Point-to-Point Connectivity Between Neuromorphic Chips Using Address Events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[19] J. A. Leñero-Bardallo, F. Pérez-Peña, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Pipeline AER arbitration with event aging," *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017.

[20] E. R. Fossum and D. B. Hondongwa, "A Review of the Pinned Photodiode for CCD and CMOS Image Sensors," *IEEE Journal of the Electron Devices Society*, vol. 2, no. 3, pp. 33–43, 2014.

[21] E. R. Fossum, "What To Do With Sub-Diffraction-Limit (SDL) Pixels? - A Proposal for a Gigapixel Digital Film Sensor (DFS).," in *Proceedings of the 2005 IEEE Workshop on Charge-Coupled Devices and Advanced Image Sensors, Karuizawa, Japan*, pp. 9–11, 2005.

[22] E. R. Fossum, "Some Thoughts on Future Digital Still Cameras," in *Image Sensors and Signal Processing for Digital Still Cameras* (J. Nakamura, ed.), pp. 305–314, CRC Press, 2005.

[23] E. R. Fossum, "Gigapixel Digital Film Sensor (DFS) Proposal," in *Nanospace Manipulation of Photons and Electrons for Nanovision Systems, Proceedings of the 7th Takayanagi Kenjiro Memorial Symposium and the 2nd International Symposium on Nanovision Science, University of Shizuoka, Hamamatsu, Japan, 25–26 October*, 2005.

[24] E. R. Fossum, J. Ma, and D. Hondongwa, "Jot Devices and the Quanta Image Sensor," pp. 247–250, 2014.

[25] E. R. Fossum, D. Hondongwa, J. Ma, S. Masoodian, Y. Song, and K. Odame, "Quanta Image Sensor (QIS): Early Research Progress," *Optics InfoBase Conference Papers*, vol. 1, pp. 27–29, 2013.

[26] N. Teranishi, "Required Conditions for Photon-Counting Image Sensors," *IEEE Transactions on Electron Devices*, vol. 59, no. 8, pp. 2199–2205, 2012.

[27] E. R. Fossum, "Modeling the Performance of Single-Bit and Multi-Bit Quanta Image Sensors," *IEEE Journal of the Electron Devices Society*, vol. 1, no. 9, pp. 166–174, 2013.

[28] E. R. Fossum and J. Ma, "A Pump-Gate Jot Device With High Conversion Gain for a Quanta Image Sensor," *IEEE Journal of the Electron Devices Society*, vol. 3, no. 2, pp. 73–77, 2015.

[29] E. R. Fossum, "The Quanta Image Sensor (QIS): Concepts and Challenges," in *Proc OSA Topical Mtg Computational Optical Sensing and Imaging*, 2011.

[30] E. R. Fossum, S. Masoodian, A. Rao, J. Ma, and K. Odame, "A 2.5 pJ/b Binary Image Sensor as a Pathfinder for Quanta Image Sensors," *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 100–105, 2016.

[31] E. R. Fossum, "Multi-Bit Quanta Image Sensors," in *Proceedings of the International Image Sensor Workshop*, pp. 292–295, 2015.

[32] E. R. Fossum, J. Ma, S. Masoodian, L. Anzagira, and R. Zizza, "The Quanta Image Sensor: Every Photon Counts," *Sensors (Switzerland)*, vol. 16, no. 8, 2016.

[33] O. A. Elgendy, *Image Processing For Quanta Image Sensors.* PhD thesis, Purdue University, 2019.

[34] S. Masoodian, *Readout Circuits For Quanta Image Sensors.* PhD thesis, Dartmouth College, 2016.

[35] N. A. Dutton, I. Gyongy, L. Parmesan, S. Gnecchi, N. Calder, B. R. Rae, S. Pellegrini, L. A. Grant, and R. K. Henderson, "A SPAD-Based QVGA Image Sensor for Single-Photon Counting and Quanta Imaging," *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 189–196, 2016.

[36] B. E. A. Saleh and M. C. Teich, *Fundamentals of Photonics*. New York, NY, USA: Wiley, 1991.

[37] F. Hurter, V. Driffield, and W. Fergusen, "Photo-Chemical Investigations and a New Method of Determination of the Sensitiveness of Photographic Plates," *The Journal of the Society of Chemical Industry*, vol. 9, pp. 76– 122, 1890.

[38] S. H. Chan, J. Ma, Y.-W. Chung, A. Gnanasambandam, and S. Masoodian, "Photon-Counting Imaging with Multi-Bit Quanta Image Sensor," pp. 1–4, 2019.

[39] E. R. Fossum, S. Member, L. E. O. Anzagira, and S. Member, "A 1 µm-Pitch Quanta Image Sensor Jot Device With Shared Readout," vol. 4, no. 2, pp. 83–89, 2016.

[40] S. H. Chan and A. Gnanasambandam, "HDR Imaging With Quanta Image Sensors: Theoretical Limits and Optimal Reconstruction," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1571–1585, 2020.

[41] S. H. Chan and O. A. Elgendy, "Optimal Threshold Design for Quanta Image Sensor," *IEEE Transactions on Computational Imaging*, vol. 4, no. 1, pp. 99–111, 2017.

[42] E. R. Fossum, "Photon Counting Error Rates in Single-Bit and Multi-Bit Quanta Image Sensors," *IEEE Journal of the Electron Devices Society*, vol. 4, no. 3, pp. 136–143, 2016.

[43] R. Van Rullen and S. J. Thorpe, "Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex," *Neural Computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[44] F. Rieke, D. Warland, R. d. R. van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code. Computational Neuroscience.* 1997.

[45] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating Mutual Information," *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 69, no. 6, 2004.

[46] Á. E. Fernández, "Optimisation of Algorithms to Compute Information Theoretic Indexes," 2013.

[47] I. Arroyo, L. C. B. M, R. Nat, H. Llinás, and F. L. Muñoz, "Poisson and Gamma Distributions : A Discrete and Continuous Relationship," vol. 12, no. 1, pp. 99–107, 2014.

[48] N. Asuni and A. Giachetti, "TESTIMAGES: A Large Data Archive For Display and Algorithm Testing," *Journal of Graphics Tools*, vol. 17, no. 4, pp. 113–125, 2013.

[49] N. Asuni and A. Giachetti, "Testimages: A large-scale archive for testing visual devices and basic image processing algorithms," *Italian Chapter Conference 2014 - Smart Tools and Apps in Computer Graphics, STAG 2014*, pp. 63–70, 2014.

Anexos

Anexo A

Scripts

A.1. Reconstrucción de imágenes mediante *Quanta Imaging*. Matrices

MATLAB®

```
%% Simulation parameters
phi_max = 1e6;                          % Maximum Photon Flux [phe/s]
phi_min = 1e2;                          % Minimum Photon Flux [phe/s]
w_matrix = 256;                         % Matrix Width (No. Columns)
h_matrix = 256;                         % Matrix Height (No. Rows)
n_QI = 8;                               % Bits Number QI image
n_binary_planes = 8;
Binary_planes = 2^n_binary_planes-1;    % Planes to generate QI image
tau = 1e-6;                             % Observation time
type_tau = 1;                           % Selects the method observation
k_Th = 1;                               % Sets pulses within window
T_binary_planes = 1.1*tau;              % Sets the time for a binary plane.

%% Sample Image
im_g = imread('Image.png');
im_rs = imresize(im_g,[h_matrix w_matrix]);
n_im_rs = 16;

%% Calculate Phi Matrix [phe/s]
PHI = im2phi(im_rs,phi_max,phi_min,n_im_rs);

%% Calculate Frequency Matrix [pulses/s]
F_PULSE = phi2fpulse(PHI);

%% Generate vector of observation times (tau) [s]
vector_TAU = gentauvector(tau,type_tau,Binary_planes);

%% Calculate H Matrix [Expected pulses/pixel]
H = F_PULSE.*vector_TAU;

%% Calculate Poisson Matrix [Observed pulses/pixel]
P = poissrnd(H);


```

```
%% Calculate Binary Matrix
B = poiss2binary(P,k_Th);

%% Calculate Count Matrix
COUNT = single(sum(B,3));

%% QI Image
im_QI = count2im(COUNT,n_QI);

function [Phi] = im2phi(im,phi_max,phi_min,nbits)
% im2phi - Transforms the image matrix into a photon flux matrix
% [(phe/s)/pixel] by linear interpolation
PV_max = 2^nbits-1;
PV_min = 0;
Phi = (phi_max-phi_min)/(PV_max-PV_min).*single(im) + phi_min;
end

function [Fpulse] = phi2fpulse(Phi)
% phi2fpulse - Transforms the photon flux matrix [(phe/s)/pixel] into a
% frequency matrix [(pulses/s)/pixel]
Fpulse = Phi;
end

function [vtau] = gentauvector(tau,tipo,binary_planes)
% gentauvector - Generates a three-dimensional vector with the observation
% time (tau) for each frame.
switch tipo
case 1
tau_tri = ones(1,1,binary_planes);
tau_tri(1:end) = tau;
vtau = single(tau_tri);
case 2
tau_tri = ones(1,1,binary_planes);
tau_max = 14e-8;
tau_min = 1e-10;
tau_tri(1:end) = tau_min:(tau_max-tau_min)/(binary_planes-1):tau_max;
vtau = single(tau_tri);
end
end

function [Binary_matrix] = poiss2binary(Poisson_matrix,Threshold)
% poiss2binary - Sets all pixels in each frame whose value is greater
% than the specified threshold to 1
Binary_matrix = Poisson_matrix >= Threshold;
end
```

Código A.1: Reconstrucción de imágenes mediante *Quanta Imaging*. Matrices

A.2. Reconstrucción de imágenes mediante *Quanta Imaging*. Bucles

MATLAB®

```
%% Simulation parameters
phi_max = 1e6;                          % Maximum Photon Flux [phe/s]
phi_min = 1e2;                          % Minimum Photon Flux [phe/s]
w_matrix = 256;                         % Matrix Width (No. Columns)
h_matrix = 256;                         % Matrix Height (No. Rows)
n_QI = 8;                               % Bits Number QI image
n_binary_planes = 8;
Binary_planes = 2^n_binary_planes-1;    % Planes to generate QI image
tau = 1e-6;                             % Observation time
type_tau = 1;                           % Selects the method observation
k_Th = 1;                               % Sets pulses within window
T_binary_planes = 1.1*tau;              % Sets the time for a binary plane.

%% Sample Image

im_g = imread('Image.png');
im_rs = single(imresize(im_g,[h_matrix w_matrix]));
n_im_rs = 16;
tam = size(im_rs);

%% Calculate Phi Matrix [phe/s]

PHI = single(zeros(tam(1),tam(2)));
PV_max = 2^n_im_rs-1;
PV_min = 0;

for jj=1:tam(1)
for ii=1:tam(2)
PHI(jj,ii) = im_rs(jj,ii)*((phi_max-phi_min)/(PV_max-PV_min)) + phi_min;
end
end

%% Calculate Frequency Matrix [pulses/s]

F_PULSE = PHI;

%% Generate vector of observation times (tau) [s]

vector_TAU = gentauvector(tau,type_tau,Binary_planes);

%% Calculate H Matrix [Expected pulses/pixel]

H = zeros(tam(1),tam(2),Binary_planes);

for kk=1:Binary_planes
for jj=1:tam(1)
for ii=1:tam(2)
H(jj,ii,kk) = F_PULSE(jj,ii)*vector_TAU(kk);
end
end
end
```

```
52 %% Calculate Poisson Matrix [Observed pulses/pixel]
53 P = zeros(tam(1),tam(2),Binary_planes);
54
55 for kk=1:Binary_planes
56 for jj=1:tam(1)
57 for ii=1:tam(2)
58 P(jj,ii,kk) = poissrnd(H(jj,ii,kk));
59 end
60 end
61 end
62
63 %% Calculate Binary Matrix
64 B = zeros(tam(1),tam(2),Binary_planes);
65
66 for kk=1:Binary_planes
67 for jj=1:tam(1)
68 for ii=1:tam(2)
69 B(jj,ii,kk) = P(jj,ii,kk)>0;
70 end
71 end
72 end
73
74 %% Calculate Count Matrix
75 COUNT = single(zeros(tam(1),tam(2)));
76
77 for jj=1:tam(1)
78 for ii=1:tam(2)
79 for kk=1:Binary_planes
80 COUNT(jj,ii) = COUNT(jj,ii) + B(jj,ii,kk);
81 end
82 end
83 end
84
85 %% QI Image
86 im_QI = single(zeros(tam(1),tam(2)));
87 Count_min = min(min(COUNT));
88 Count_max = max(max(COUNT));
89
90 for jj=1:tam(1)
91 for ii=1:tam(2)
92 im_QI(jj,ii) = floor(((COUNT(jj,ii)-Count_min)/(Count_max-Count_min))...
93 .*(2^n_QI-1));
94 end
95 end
```

Código A.2: Reconstrucción de imágenes mediante *Quanta Imaging*. Bucles

A.3. Generación de imagen en *Spiking Luminance Sensor*.

MATLAB®

```
%% Simulation parameters
f_max = 55e3;                   % Maximum Pulse Frequency [pulses/s]
f_min = 1e2 ;                   % Minimum Pulse Frequency [pulses/s]
w_matrix = 256;                 % Matrix Width
h_matrix = 256;                 % Matrix Height
n_SS = 8;                       % Bits number SS image
F_ARB = 2e6;
T_image_SS = 1000e-3;           % Is the time available to generate an image
vc_mne = 2;

%% Sample image
im = imread('Image.png');
im_g = rgb2gray(im);
im_rs = imresize(im_g,[h_matrix w_matrix]);
n_im_rs = 16;

%% Calculate Pixel Frequency Matrix [(pulses/s)/pixel]
F_rs = im2frequency(im_rs,f_max,f_min,n_im_rs);

%% Calculate Spiking Matrix [(Pulses in window/pixel)/Image]
t_1 = 0;
t_2 = T_image_SS;

S_SS = pulses(vc_mne, F_rs, t_1, t_2);

%% SS SATURATION
MER_SS = F_ARB*T_image_SS;

if sum(S_SS(:)) < floor(MER_SS)
  S_SS_SAT = S_SS;
else
  S_SS_SAT = S_SS;
  toDelete = abs(floor(MER_SS) - sum(S_SS_SAT(:)));
  weight = 1./F_rs(:);

  while toDelete > 0
    possibles = find(S_SS_SAT(:) > 0);
    deletion = floor(toDelete*weight(possibles)/sum(weight(possibles)));
    a = toDelete - sum(deletion);
    if a > 0
       [v,index] = sort(weight(possibles),'descend');
       while a > length(index)
        deletion = deletion + 1;
        a = toDelete - sum(deletion);
       end
       if a > 0
        deletion(index(1:a)) = deletion(index(1:a)) + 1;
       end
    end
    S_SS_SAT(possibles) = S_SS_SAT(possibles) - deletion;
    negatives = find(S_SS_SAT(:) < 0);
```

```
52      if ~isempty(negatives)
53        S_SS_SAT(negatives) = 0;
54        toDelete = abs(floor(MER_SS) - sum(S_SS_SAT(:)));
55      else
56        toDelete = 0;
57      end
58    end
59  end
60
61  %% Measured Pixel Frequency Matrix [(pulses/s)/pixel]
62  F_MEA_SAT = S_SS_SAT./T_image_SS;
63
64  %% SS Image
65  im_SS_SAT = frequency2im(F_MEA_SAT, n_SS);
66
67  function [F_rs] = im2frequency(im,frequency_max,frequency_min,nbits)
68  %im2frequency - Transforms the image matrix into a frequency matrix
69  %[(pulses/s)/pixel] by linear interpolation
70  PV_max = 2^nbits-1;
71  PV_min = 0;
72  F_rs = (frequency_max-frequency_min)/(PV_max-PV_min).*single(im)...
73         + frequency_min;
74  end
75
76  function [S] = pulses(vc_mne, f_pixel, t_1, t_2)
77  %pulses - Calculates the pulses of each pixel that fall within
78  %the window of each binary plane.
79  tol = eps('single');
80  t_2_tri = ones(1,1,length(t_2)); t_2_tri(1:end) = t_2;
81  vt_2 = double(t_2_tri);
82  t_1_tri = ones(1,1,length(t_1)); t_1_tri(1:end) = t_1;
83  vt_1 = double(t_1_tri);
84  if(vc_mne == 0)
85    S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1)...
86    -(abs(floor(f_pixel.*vt_2)./f_pixel - vt_2) < tol);
87  elseif (vc_mne == 1)
88    S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1)...
89    +((vt_1~=0).*(abs(floor(f_pixel.*vt_1)./f_pixel - vt_1) < tol));
90  elseif (vc_mne == 2)
91    S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1);
92  end
93  end
94
95  function [im_SS] = frequency2im(F,nbits)
96  %frequency2im - Transforms the frequency matrix [(pulses/s)/pixel] into a
97  %image matrix by linear interpolation
98  F_min = min(min(F));
99  F_max = max(max(F));
100 im_SS = floor(((F-F_min)/(F_max-F_min)).*(2^nbits-1));
101 end
```

Código A.3: Generación de imagen en *Spiking Luminance Sensor*.

A.4. Generación de imagen en *Spiking Luminance Sensor* aplicando *Quanta Imaging*.

MATLAB®

```
%% Simulation parameters
f_max = 55e3;                          % Maximum Pulse Frequency [pulses/s]
f_min = 1e2;                           % Minimum Pulse Frequency [pulses/s]
w_matrix = 256;                        % Matrix Width
h_matrix = 256;                        % Matrix Height
n_SQS = 8;                             % Bits number SQS image
F_ARB = 2e6;
n_binary_planes = 12;
binary_planes = 2^n_binary_planes-1;   % Planes to generate SQS image
k_Th = 1;                              % Sets pulses within window
T_image_SQS = 1000e-3;                 % Is the time to generate an image
vc_mne = 2;                            % It is a control variable

%% Sample image
im_g = imread('Image.png');
im_rs = imresize(im_g,[h_matrix w_matrix]);
n_im_rs = 16;

%% Calculate Frequency Pixel Matrix [(pulses/s)/pixel]
F_rs = im2frequency(im_rs,f_max,f_min,n_im_rs);

%% Window Parameters
tau_on = 1/f_max;
T_binary_plane = T_image_SQS/binary_planes;
D_tau = tau_on/T_binary_plane;
delay_tau = 0;

%% Generate time windows vector
[t_tau, vector_TAU, T_bp] = gentauvector(tau_on, T_binary_plane, delay_tau
    , n_binary_planes,'random-plane');
t_instant = t_tau(vector_TAU == 1);
t_1 = t_instant(1:2:end);
t_2 = t_instant(2:2:end);

%% Calculate SQS Spiking Matrix [(Pulses in window/pixel)/Binary Plane]
S_SQS = pulses_in_window(vc_mne, F_rs, t_1, t_2);

%% SQS SATURATION
S_SQS_SAT = zeros(size(S_SQS));
for ii = 1:binary_planes
P = T_bp(ii);
plane = S_SQS(:,:,ii)>0;

Rate = sum(plane(:))/P;

if F_ARB < Rate
events = find(plane(:) == 1);
reorder = randperm(length(events));
neglected = events(reorder(ceil(F_ARB*P):end));
```

```
49  plane(neglected) = 0;
50  end
51  S_SQS_SAT(:,:,ii) = plane;
52  end
53
54  %% Calculate Binary Matrix
55  B_SAT = spiking2binary(S_SQS_SAT,k_Th);
56
57  %% Calculate Count Matrix
58  COUNT_SAT = single(sum(B_SAT,3));
59
60  %% SQS Image
61  im_SQS_SAT = count2im(COUNT_SAT,n_SQS);
62
63  function [F] = im2frequency(im,frequency_max,frequency_min,nbits)
64  % im2frequency - Transforms the image matrix into a frequency matrix
65  % [(pulses/s)/pixel] by linear interpolation
66  PV_max = 2^nbits-1;
67  PV_min = 0;
68  F = (frequency_max-frequency_min)/(PV_max-PV_min).*single(im) +
        frequency_min;
69  end
70
71  function [t_p, v, T_bp] = gentauvector(tau_on, T_binary_planes, ...
72                                         delay_tau, n_binary_planes,mode)
73  ton = tau_on;
74  T = T_binary_planes;
75  delay = delay_tau;
76  N = n_binary_planes;
77
78  if ton > T
79  error(''Duty CYCLE HIGHER THAN PERIOD. tau_on cannot be higher than
        T_binary_plane'')
80  end
81
82  duty = ton/T;
83  if strcmp(mode,'periodic')
84
85  t1 = 0;
86  t2 = duty*T;
87  t3 = T;
88
89  t_p = [];
90  v = [];
91
92  for ii = 0:(2^N-1)-1
93  t_p = [t_p, (ii*t3)+[t1 t1 t2 t2 t3]];
94  v = [v, 0 1 1 0 0];
95
96  T_bp(ii+1) = t3;
97  end
98
99  t_p = [0 delay t_p+delay];
100 v = [0 0 v];
```

```
101  elseif strcmp(mode,'progressive-shift')
102
103  divisions = floor(T/ton);
104  t_p = [0 delay];
105  v = [0 0];
106
107  for ii = 0:(2^N-1)-1
108  t1 = 0;
109  t2 = duty*T;
110 m = mod(ii,divisions)+1;
111  t3 = t2 + m/divisions*T;
112  T_bp(ii+1) = t3;
113  t_p(end+1:end+5) = t_p(end) + [t1 t1 t2 t2 t3];
114  v = [v, 0 1 1 0 0];
115  end
116
117  elseif strcmp(mode,'random-plane')
118
119  t_p = [0 delay];
120  v = [0 0];
121
122  for ii = 0:(2^N-1)-1
123  r = rand-0.5;
124  t1 = 0;
125  t2 = t1 + duty*T;
126  t3 = max(T*(1+r),t2+0.05*T);
127  T_bp(ii+1) = t3;
128  t_p(end+1:end+5) = t_p(end) + [t1 t1 t2 t2 t3];
129  v(end+1:end+5) = [0 1 1 0 0];
130  end
131
132  elseif strcmp(mode,'arrivaltime-poisson')
133
134  t_p = [0 delay];
135  v = [0 0];
136
137  for ii = 0:(2^N-1)-1
138  t1 = 0;
139  t2 = t1 + duty*T;
140  t3 = max(-log(1-rand)/lambda,t2+0.05*T);
141  T_bp(ii+1) = t3;
142  t_p(end+1:end+5) = t_p(end) + [t1 t1 t2 t2 t3];
143  v(end+1:end+5) = [0 1 1 0 0];
144  end
145
146  end
147
148  function [S] = pulses_in_window(vc_mne, f_pixel, t_1, t_2)
149  % pulses_in_window - Calculates the pulses of each pixel that fall
150  % within the window of each binary plane.
151  % vc_mne == 0 does not count the pulses at instants t1 and t2
152  % vc_mne == 1 if it counts them.
153  % vc_mne == 2 Only counts one.
154
```

```
155 tol = eps('single');
156
157 t_2_tri = ones(1,1,length(t_2));
158 t_2_tri(1:end) = t_2;
159 vt_2 = double(t_2_tri);
160 t_1_tri = ones(1,1,length(t_1));
161 t_1_tri(1:end) = t_1;
162 vt_1 = double(t_1_tri);
163
164 if(vc_mne == 0)
165
166 S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1)-(abs(floor(f_pixel.*vt_2)./
        f_pixel - vt_2) < tol);
167 elseif (vc_mne == 1)
168 S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1)+((vt_1~=0).*(abs(floor(
        f_pixel.*vt_1)./f_pixel - vt_1) < tol));
169 elseif (vc_mne == 2)
170 S = floor(f_pixel.*vt_2)-floor(f_pixel.*vt_1);
171
172 end
173
174 end
175
176 function [Binary_matrix] = spiking2binary(Spiking_matrix,Threshold)
177 % spiking2binary - Sets to 1 all pixels in each binary plane
178 % whose value is greater than the specified threshold
179 Binary_matrix = Spiking_matrix >= Threshold;
180 end
181
182 function [im_AAIS] = count2im(COUNT_matrix,nbits)
183 % count2im - Encoding to n-bit greyscale image
184 Count_min = min(min(COUNT_matrix));
185 Count_max = max(max(COUNT_matrix));
186
187 im_AAIS = floor(((COUNT_matrix-Count_min)/(Count_max-Count_min))...
188           .*(2^nbits-1));
189 end
```

Código A.4: Generación de imagen en *Spiking Luminance Sensor* aplicando *Quanta Imaging*.

A.5. Métricas de comparación: *RMSE* y *NMI*

MATLAB®

```
function [MI_norm,RMSE] = metrics(im_rs,im_sensor,n_im_sensor)
%metrics - Calculates the RMSE and MI of two images
%% Mutual Information Metric
%% 2D histogram calculation (Reference Image with Reference Image)
[N,Xedges,Yedges] = histcounts2(im_rs,im_rs,2^n_im_sensor);

%% Mutual Information
%Convert bins counts to probability values
pxy = N/sum(N(:));      %Joint probability distribution function of X and Y
px = sum(pxy,2);        %Marginal probability distribution function of X
py = sum(pxy,1);        %Marginal probability distribution function of Y
px_py = px.*py;
nzs = pxy > 0;
mutual_max = 0;
tam = size(pxy);

for xx = 1:tam(2)
for yy = 1:tam(1)
if nzs(xx,yy) == 1
mutual_max = mutual_max + sum(sum(pxy(xx,yy)*log(pxy(xx,yy)/px_py(xx,yy))));
end
end
end

%% 2D histogram calculation (Reference Image with Sensor Image)

[N,Xedges,Yedges] = histcounts2(im_rs,im_sensor,2^n_im_sensor);

%% Mutual Information

%Convert bins counts to probability values
pxy = N/sum(N(:));      %Joint probability distribution function of X and Y
px = sum(pxy,2);        %Marginal probability distribution function of X
py = sum(pxy,1);        %Marginal probability distribution function of Y
px_py = px.*py;
nzs = pxy > 0;
mutual = 0;
tam = size(pxy);

for xx = 1:tam(2)
for yy = 1:tam(1)
if nzs(xx,yy) == 1
mutual = mutual + sum(sum(pxy(xx,yy)*log(pxy(xx,yy)/px_py(xx,yy))));
end
end
end

MI_norm = mutual*(1/mutual_max);


```

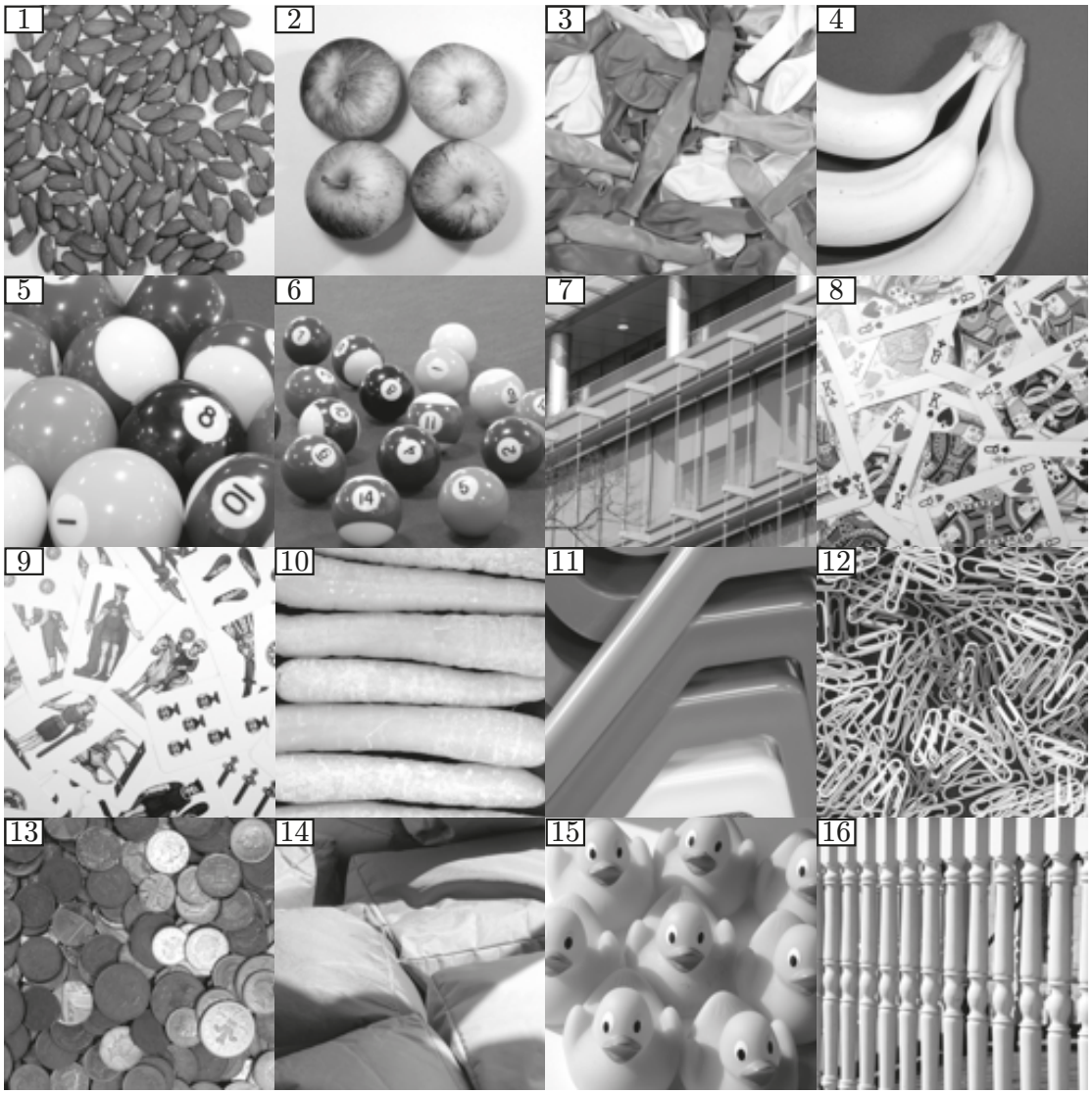
```
51 %% Root Mean Square Error Metric
52
53 tam = size(im_rs);
54 im_high= ones(tam(1,1),tam(1,2)).*(2^n_im_sensor-1);
55 im_low= zeros(tam(1,1),tam(1,2));
56 im_diff_high = single(im_high) - single(im_rs);
57 im_diff_low = abs(single(im_low) - single(im_rs));
58
59 for ii=1:1:tam(1,1)
60 for jj=1:1:tam(1,2)
61 if(im_diff_high(ii,jj) >= im_diff_low(ii,jj))
62 im_diff_max(ii,jj) = 2^n_im_sensor-1;
63 else
64 im_diff_max(ii,jj) = 0;
65 end
66 end
67 end
68
69 MSE_max = immse(single(im_rs),single(im_diff_max));
70 RMSE_max = sqrt(MSE_max);
71
72 MSE = immse(single(im_rs),single(im_sensor));
73 RMSE = sqrt(MSE);
74
75 end
```

Código A.5: Métricas de comparación: *RMSE* y *NMI*.

Anexo B

Colección de imágenes

Las imágenes que se exponen fueron tomadas de [48, 49].




17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40