# UNA ARQUITECTURA SEGMENTADA PARA EL CÁLCULO DEL TIEMPO AL IMPACTO CON VISIÓN LOG-POLAR

Jose A. Boluda[1], Fernando Pardo[1], Francisco Blasco[1], Joan Pelechano[2]
Jose.A.Boluda@uv.es

[1]Departament d'Informàtica i Electrònica. Universitat de València.
C/ Doctor Moliner, 50. 46100 Burjassot, (València)

[2]Institut de Robòtica. Universitat de València.
C/ Hugo de Moncada, 4. 46010 València

***Resumen:*** *Esta comunicación presenta una arquitectura segmentada basada en lógica reconfigurable para procesamiento de imágenes a alta velocidad. La arquitectura está orientada a la optimización de algoritmos diferenciales de visión, en los que se puede incluir una gran cantidad de algoritmos interesantes para navegación de vehículos autónomos. La utilización del sistema de coordenadas log-polar es fundamental para reducir la cantidad de datos a procesar y simplificar los algoritmos de visión. La utilización de FPGAs basadas en tecnología SRAM combina reconfigurabilidad con altas prestaciones. Como aplicación particular se presenta el diseño, bajo esta arquitectura, del algoritmo de cálculo de tiempo al impacto con prestaciones estimadas de cerca de 90 imágenes por segundo.*

## 1. Introducción

La navegación de plataformas móviles autónomas en entornos no-estructurados es un importante campo de investigación en robótica. Dentro de los múltiples métodos de sensorización de una plataforma móvil destaca la sensorización visual por combinar una alta precisión con un gran alcance.

Un problema que aparece con la sensorización visual es la gran cantidad de información que debe ser almacenada y procesada. Cuanto más precisa se quiera la información visual, más resolución tendrán las imágenes, y por tanto más datos deberán procesarse.

A esta primera restricción de cantidad de información a procesar, hay que sumarle la complejidad de los algoritmos de visión artificial a utilizar, (se puede ver un ejemplo en [BB95]), y las restricciones de tiempo real que la plataforma móvil tendrá. Esta potencia de cálculo, necesaria para la plataforma móvil, es un problema que no se puede resolver incorporando una gran cantidad de recursos hardware. El hecho de ser autónoma impone limitaciones de peso y consumo de potencia.

Otra consideración a tener en cuenta es la flexibilidad requerida a un módulo de sensorización visual. No es recomendable conseguir la velocidad *hardware* a costa de la rigidez de los circuitos hechos a medida. Por otra parte, la deseable flexibilidad *software* implica una mayor lentitud en los algoritmos de visión artificial que se deseen implementar. Es deseable, por tanto, combinar la flexibilidad del software con las prestaciones del hardware para, de esta manera, poder implementar varios algoritmos de visión artificial en el módulo de sensorización visual.

En la presente comunicación, se presenta la arquitectura de un módulo de sensorización visual para plataformas móviles que afronta la resolución de estos problemas. Como ejemplo concreto, que muestra la validez de la aproximación, se plantea el diseño del algoritmo de cálculo del tiempo al impacto bajo esta arquitectura.

## 2. Visión log-polar

En los seres vivos más evolucionados, la retina muestra una distribución no uniforme de celdas visuales. Con la distribución espacio-variante de fotocélulas en los sensores visuales de los seres vivos se consigue una resolución que también es espacio-variante. De esta manera, se tiene una mayor resolución en el centro de la imagen, que es donde se supone se tendrá el área de interés, manteniendo un campo de visión suficientemente ancho para apercibirse de cambios en la periferia. Un ejemplo de visión espacio variante es la distribución log-polar, que se puede observar en la figura 1.

La utilidad de la visión log-polar para visión artificial ha sido ampliamente estudiada de modo *teórico* desde hace dos décadas [WC79] [TS92], encontrándose interesantes propiedades de simplificación de ciertos algoritmos, además de la

inherente reducción de información debida a la distribución espacio variante.

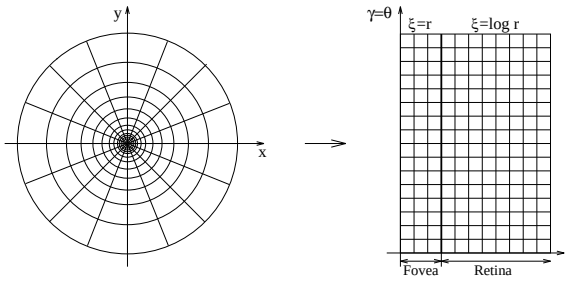


**Figura 1**: Transformación log-polar

Como consecuencia del interés teórico de la visión log-polar, ha habido varios intentos de desarrollar sensores que incorporasen la transformación log-polar a modo de *ojos artificiales* [VKC+89] [WRL95]. Estos desarrollos planteaban problemas de continuidad en la imagen y complejidad del hardware que hacían muy difícil o imposible su utilización en aplicaciones prácticas *reales* de visión artificial.

Recientemente ha sido diseñado y construido un sensor log-polar basado en tecnología CMOS que resuelve los problemas de continuidad de la imagen y hace mucho más sencillo su interfase con una CPU [PDS97] [PDS98]. A partir del sensor log-polar CMOS se ha desarrollado una cámara [BPK+96] y una tarjeta de adquisición de imágenes de alta velocidad para bus PCI [Bla98] que está permitiendo su utilización en una plataforma autónoma.

## 3. Cálculo del tiempo al impacto

Un algoritmo de visión artificial que es muy interesante para una plataforma autónoma es el del cálculo del tiempo al impacto con los diferentes objetos de la escena. La plataforma móvil, dotada de una sistema de adquisición de imágenes, será capaz de estimar un mapa del tiempo al impacto de los diferentes objetos de la escena. De esta manera se podrán evitar choques y planificar de manera adecuada la navegación en entornos no-estructurados.

El cálculo del tiempo al impacto se simplifica de una manera importante utilizando el formalismo log-polar [TS91]. Cuando el centro del sensor coincide con el FOE (foco de expansión de la imagen), condición fácilmente asumible con una plataforma móvil, el tiempo al impacto $\tau$ viene expresado por la ecuación 1.

$$\tau = K \frac{-\frac{\partial E}{\partial \xi}}{\frac{dE}{dt}} \qquad (1)$$

Donde $E(\xi, \theta, t)$ sería la imagen de 255 niveles de gris en función de las coordenadas log-polares y el tiempo, y K una constante de proporcionalidad que depende del sensor log-polar.

El cómputo se reduce al cálculo del cociente del gradiente de la coordenada radial entre la primera derivada temporal. Realizar el mismo cálculo en coordenadas cartesianas implicaría realizar el cálculo del flujo óptico en ambas coordenadas x e y, además de realizar el cálculo de la distancia al FOE. El cálculo del flujo óptico es especialmente costoso [BB95], si se quiere realizar con una precisión aceptable, y el cálculo de distancias al FOE implica la aparición del cálculo de raíces cuadradas.

Con la aproximación log-polar todo queda reducido a simples cálculos diferenciales, tanto espaciales como temporales, lo que unido a la reducción de pixels a procesar, permitirá un alto ratio de imágenes por segundo.

## 4. El cauce reconfigurable

**Requisitos del módulo reconfigurable**

Para diseñar la arquitectura del módulo de sensorización que se desea implementar hay que, en primer lugar, definir los objetivos a cumplir. De esta manera, se pretende realizar una módulo visual que sea útil para navegación robótica.

Las técnicas de detección de movimiento, o desde un punto de vista más amplio, algoritmos con una variación temporal de las imágenes, se pueden agrupar en 3 grandes grupos:

1. **Cálculo de flujo óptico**: Técnica tan costosa como precisa se quiera hacer [BB95].
2. **Extracción y encaje de puntos relevantes**: Estas técnica sufre también de un alto coste temporal así como de dependencia de la escena para extraer los puntos que se van a considerar relevantes [DAD97].
3. **Técnicas diferenciales**: Son el método de estimación de movimiento y cálculo de parámetros dinámicos con menor coste computacional y más orientados a una implementación hardware.

El cálculo del tiempo al impacto log-polar es un ejemplo de un algoritmo diferencial que se puede implementar con hardware específico. Un ejemplo de otro algoritmo log-polar y diferencial distinto (en este caso de detección de movimiento) puede encontrarse en [BDPP97].

Un requisito para la arquitectura reconfigurable será que pueda implementar de manera eficiente algoritmos diferenciales, tanto espaciales como temporales.

Otro requisito para el módulo de visión será la flexibilidad. Se desea tener un módulo que pueda implementar diversos algoritmos de visión

artificial, en función de los intereses de la plataforma móvil. De esta manera la plataforma puede estar parada y puede ser interesante que simplemente detecte movimiento en una zona de la imagen, o puede estar en movimiento y deba detectar los objetos que en la escena tengan movimiento propio.

A esta condición de flexibilidad *software* hay que añadir la condición de que se desea la rapidez *hardware*. Esta última condición se deberá conseguir sin convertir el módulo de visión en una colección de recursos hardware que implementarán una colección predeterminada de algoritmos, seleccionando en cada caso el robot los recursos que interesen.

Una plataforma autónoma tiene limitaciones de peso y consumo de potencia, y el módulo de visión será uno más de los que sensorizarán el robot. No se puede cargar la plataforma con excesivo *peso muerto*.

Otro requisito del módulo reconfigurable seria el poco peso, dimensión y consumo combinado la *escalabilidad*. El módulo debe ser capaz de implementar algoritmos nuevos, en los que no se ha pensado a la hora de diseñarlo, con ligeras modificaciones.

**La arquitectura global del sistema.**

Para conseguir la reconfigurabilidad del software con las prestaciones del hardware, se ha recurrido a la utilización de lógica reconfigurable. Más concretamente a las FPGAs de arquitectura híbrida basadas en tecnología SRAM FLEX8000 de Altera. Esta familia tiene una arquitectura que combina las altas prestaciones de las CPLDs (Complex Programmable Logic Devices), con la complejidad de las FPGAs (Field Programmable Gate Arrays).

Para explotar el alto grado de paralelismo de los algoritmos de visión artificial [Pit93], (especialmente los diferenciales), se va a dividir un algoritmo en diversas etapas. Cada etapa la ejecutará en paralelo un Elemento de Proceso (EP), implementado de esta manera una cauce segmentado.

En un cauce segmentado el tiempo de proceso de un dato (o byte de la imagen) será igual al de la etapa más lenta, que actuará como cuello de botella del cauce. Cuanto más *profunda* sea la segmentación (mayor número de etapas) mayor será el flujo de datos. Además del paralelismo temporal implícito en el cauce segmentado, cada EP podrá a su vez utilizar paralelismo espacial para aumentar la velocidad del sistema.

En cualquier caso la escalabilidad estará directamente relacionada con la velocidad del cauce. A mayor número de etapas, mayor número de EPs y mayor rapidez.

En la figura 2 se puede observar un esquema general de la arquitectura propuesta. El sistema de control de la plataforma móvil decidirá que algoritmo de visión artificial es interesante utilizar y configurará cada EP.

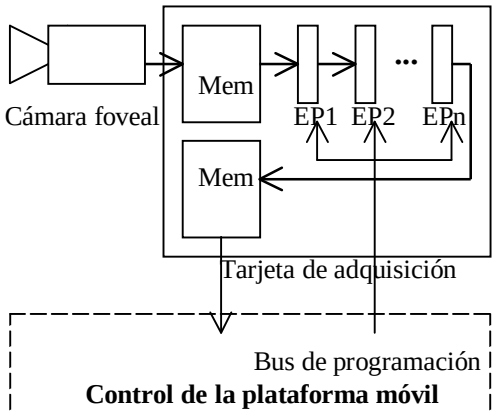


**Figura 2**: Arquitectura global del sistema

El flujo de imágenes es gobernado por una tarjeta especial de adquisición de imágenes log-polares, que las capturará de la cámara log-polar y se las transmitirá al primer EP o etapa del cauce. La imagen fluirá a través del cauce conforme va siendo procesada, hasta llegar al último EP. Una vez los datos salen del último EP vuelven a la tarjeta de adquisición, siendo visibles por el sistema de control de la plataforma móvil.

**El elemento de proceso**

El EP, para acelerar el cálculo de las derivadas temporales, dispondrá de dos bloques de memoria de doble puerto. La filosofía de este cauce de datos es que el $EP_i$ pueda acceder a la imagen *presente* que le esta suministrando el $EP_{i-1}$, además de acceder simultáneamente a la imagen almacenada en la memoria del $EP_{i-1}$. Esta memoria de doble puerto estará en dos bloques, de manera que el $EP_i$ está generando una imagen resultado mientras la almacena en un bloque, y el $EP_{i+1}$ accede al otro bloque de memoria donde estará la imagen anterior.

Con este cauce de datos se acelera el cálculo de derivadas temporales, evitando la colisión entre la lectura/escritura al estar el puerto de la izquierda de la memoria siempre en modo escritura, y el puerto de la derecha siempre en modo lectura.

Además el tener dos bloques tampoco hay colisiones de lectura/escritura simultanea en la

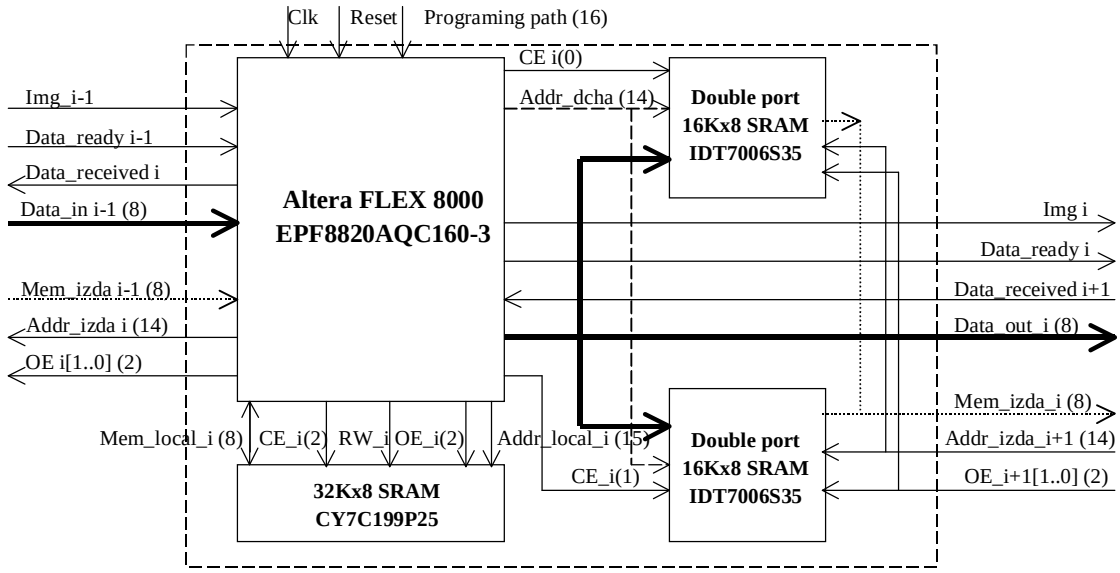


**Figura 3:** El Elemento de proceso i-ésimo

misma celda. Si se lee de un bloque se escribe en el otro. Una imagen log-polar completa ocupa 9.728 bytes, por tanto en una memoria de 16 KBytes se puede almacenar de forma completa. La figura 3 muestra un esquema del elemento de proceso i-ésimo **[BPP98]**.

La FPGA reconfigurable que ejecutará las tareas de esa etapa del cauce, incorporará además una memoria local para de 32 Kbytes almacenar resultados intermedios o imágenes intermedias.

Es deseable que el EP tenga una sola interfaz externa suficientemente general para que sea posible implementar todas las etapas del cauce en él. De esta manera sólo diseñando un PCB se tiene el diseño físico de cualquier etapa del cauce. La funcionalidad del EP se definirá cuando la plataforma móvil programe la FPGA del EP.

La transmisión de datos a través del cauce se realizará con un simple protocolo de dato_listo / dato_recibido. La escalabilidad esta garantizada gracias a la generalidad de la arquitectura. A más etapas en el cauce (número de EPs), se tendrá una mayor profundidad en la segmentación con menor número de ciclos por etapa, siendo el tiempo de proceso por dato igual al de la etapa más lenta. Por el contrario a mayor profundidad en la segmentación aumentará el tiempo de latencia.

La escalabilidad (ilimitada en teoría) vendrá limitada por los requisitos de espacio y consumo de la plataforma autónoma. Cabe hacer notar que físicamente el EP es una pequeña tarjeta con sólo 4 componentes y que se han implementado varios algoritmos de visión artificial con sólo 3 EPs.

## 5. Los EPs del tiempo al impacto

El cálculo del tiempo al impacto con el formalismo log-polar se expresa en la ecuación 1. el tiempo al impacto es proporcional al cociente del gradiente radial entre la derivada temporal. Para que los resultados sean correctos deben evitarse los cambios bruscos de tono de gris en la imagen realizando previamente un suavizado de la imagen.

El cálculo del tiempo al impacto, utilizando la arquitectura segmentada propuesta, se realizara en las etapas:

**A) Suavizado:** Este EP realiza un suavizado de la imagen sumando los valores de gris de un punto sus 6 vecinos más próximos. Este EP utiliza la memoria local, para almacenar los bytes vecinos que se han suministrado previamente, e intervienen en el cálculo del suavizado de un byte. El coste es de 4 ciclos por byte procesado.

**B) Gradiente y Derivada:** Este módulo realiza el cálculo de la derivada espacial (radial) utilizando su memoria local para almacenar la fila anterior. La derivada temporal se calcula accediendo simultáneamente a la imagen presente, y a la imagen anterior almacenada en el EP anterior del cauce, en este caso el de suavizado. El coste es de 4 ciclos por byte procesado.

**C) División:** Este módulo realiza la división entera del gradiente y la deriva temporal calculados en la etapa anterior. Para ello se utiliza el algoritmo de división entera sin restauración, siendo el coste de 16 ciclos por dato procesado. Si la división no se puede

calcular, (divisor nulo o resultado negativo), el coste es de 6 ciclos.

El diseño de las FPGAs de los EPs ha sido realizado utilizando VHDL, estando el PCB de los EPs en fase de construcción, habiéndose realizado previamente exhaustivas simulaciones.

En el peor caso, (cuando se tenga que realizar la división por todos los pixels de la imagen), el cauce suministra un dato válido cada 21 ciclos de reloj. Como cada imagen tiene un total de 9.728 pixels, esto nos da un resultado por imagen de 21 ciclos x 9.728 pixels = 204.288 ciclos. El reloj de los EPs lo suministra la tarjeta de adquisición para bus PCI que funciona a una velocidad de 33MHz (30 ns de periodo). Los elementos de proceso tendrán un periodo de reloj de 60ns, resultado de dividir la frecuencia de la tarjeta PCI. Con este periodo de reloj se pueden procesar 81 imágenes log-polares por segundo.

Otras simulaciones de otros algoritmos diferenciales de detección de movimiento han mostrado resultados de más de 100 imágenes por segundo.

## 6. Conclusiones

Una arquitectura segmentada para visión log-polar ha sido diseñada y esta en fase de construcción. La utilización de lógica programable permite combinar la reconfigurabilidad del software con la velocidad del hardware. El formalismo log-polar reduce la cantidad de información a procesar y el paralelismo temporal del cauce segmentado aumenta la velocidad de proceso.

La modularidad del sistema, además de su pequeño tamaño, permite plantear su utilización como módulo de sensorización visual de una plataforma móvil autónoma que está en fase de construcción.

## 7. Agradecimientos

La Generalitat Valenciana esta financiando este trabajo con el proyecto GV97-TI-05-27.

## 8. Bibliografía

**[BB95]** S.S. Beauchemin y J.L. Barron. *The computation of optical flow.* ACM Computing Surveys, 27(3): 443-467, 1995.

**[Bla98]** Francisco Blasco. *Entorno de adquisición de imágenes log-polares a alta velocidad*. Proyecto fín de carrera. Ingeniería Informàtica. Departament d'Informàtica y Electrònica. Universitat de València.

**[BDPP97]** J.A. Boluda, J.J. Domingo, F. Pardo y J. Pelechano. *Detecting motion independent of the camera movement through a log-polar differential approach.* Springer Lectures Notes on Computer Science. Vol. 1296. Pag: 702-710

**[BPK+96]** J.A. Boluda, F. Pardo, T. Kayser, J.J. Pérez y J. Pelechano. *A new foveated space-variant camera for robotic applications.* IEEE International Conference on Electronics, Circuits and Systems, Volumen 2, páginas 680-683.

**[BPP98]** J.A. Boluda, F. Pardo y J. Pelechano. *Reconfigurable architectures for machine perception. An approach for autonomous vehicle navigation.* Workshop on European Scientific and Industrial Collaboration on promoting advanced technologies in manufacturing, WESIC'98. Girona, Spain, June 1998. Pag: 359-363.

**[DAD97]** Juan Domingo, Guillermo Ayala, Elena Díaz. *A method for multiple rigid-object motion segmentation based on detection and consistent matching of relevant points in image sequences*. IEEE international Conference on Acoustics, Speech and Signal Processing. Vol. 4: 3021-3025, April 1997.

**[Pit93]** Ioannis Pitas, editor. *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks.* Series in Parallel Computing. WILEY, 1993.

**[PDS97]** F. Pardo, B. Dierickx and D. Scheffer *CMOS Foveated Image Sensor: Signal Scaling and Small Geometry Effects*. IEEE Transactions on Electron Devices. Pag. 1731-1737, Vol. 44, No. 10, October 1997.

**[PDS98]** F. Pardo, B. Dierickx and D. Scheffer *Space-Variant Non-Orthogonal Structure CMOS* Image Sensor Design IEEE Journal of Solid-State Circuits. Pag. 842-849, Vol. 33, No. 6, June 1998.

**[TS91]** M. Tistarelli y G. Sandini. *On the advantages of polar mapping for direct estimation of time-to-impact from optical flow.* IEEE Trans. on PAMI, PAMI-15, No. 4:401-410, 1991.

**[TS92]** M. Tistarelli y G. Sandini. *Dynamic aspects in active vision*. CVGIP: Image Understanding, 56 No.1: 108-129, 1992.

**[VKC+89]** J. Van der Spiegel, H. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Belluti y G. Socini. *A Foveated Retina-Like Sensor using CCD Technology*. Kluwer Academic, Boston 1989.

**[WC79]** C. Weiman y G. Chaikin. *Logarithmic spiral grids for image processing and display*. Computer Graphics and Image Processing, 11:197-226, 1979.

**[WRL95]** R. Wodnicki, G. Roberts, M. Levine. *A foveated image sensor in standard CMOS technology*. Custom Integrated Circuits Conference, Santa Clara, California, May 1995.