Transformación Log-polar en FPGAs Utilizando CORDIC

Pardo F., Boluda J.A., Sosa J.C.

Departamento de Informática, Universitat de València
Avda. Vte. Andrés Estellés s/n, 46100 Valencia, España
Fernando.Pardo@uv.es, Jose.A.Boluda@uv.es, jucesosa@alumni.uv.es
http://tapec.uv.es/

Resumen. Este artículo describe el desarrollo de un circuito reconfigurable para la transformación de imágenes cartesianas en log-polares. El circuito que realiza la transformación log-polar se basa en un bloque que transforma las coordenadas cartesianas a log-polares. La transformación polar se implementa mediante el algoritmo CORDIC, y la logarítmica mediante un circuito adicional. El resultado es un circuito que realiza la transformación log-polar de píxeles a una velocidad de unos 60 MHz. La reconfigurabilidad de la FPGA que contiene el circuito es importante, pues permite el cambio de los parámetros de la transformación sobre la marcha.

1 Introducción

La representación log-polar presenta ventajas matemáticas como la simplificación de rotaciones y escalados. También presenta la ventaja de la reducción selectiva de la información, ya que la resolución de una imagen log-polar varía desde el centro, donde es la más alta, hasta la periferia, donde la resolución es baja. Normalmente la parte interesante de la escena se encuentra en el centro, que es la parte con mayor resolución. Precisamente ésta es la estrategia utilizada por la mayoría de sistemas visuales biológicos.

Gracias a estas propiedades, la representación log-polar ha demostrado su utilidad en tareas como el cálculo del tiempo al impacto de alta velocidad [1], donde las propiedades matemáticas simplifican el cálculo del flujo óptico, y la reducción de la información permite una alta tasa de imágenes procesadas.

Hay varias posibilidades para la obtención de imágenes log-polares. La más simple consiste en transformar la imagen cartesiana, que es la obtenida desde casi cualquier cámara, mediante un programa *software*. Esta solución tiene la ventaja de ser simple y flexible, pero no es muy adecuada en aplicaciones de tiempo real, puesto que consume tiempo del procesador. Otro problema es la falta de encaje entre los píxeles cuadrados de la imagen cartesiana y los log-polares, especialmente en el centro donde la resolución de la imagen log-polar es incluso mayor que la de la cartesiana.

Otra posibilidad consiste en utilizar una cámara especial basada en un sensor log-polar [2]. Esta es la mejor solución para aplicaciones de tiempo real, pero presenta también algunos problemas. Por ejemplo, estas cámaras log-polares no están tan disponibles como el resto y son algo caras. Por otro lado, la calidad de imagen obtenida, aunque buena, no es tan alta como la del resto de cámaras.

La propuesta presentada en este artículo es apropiada para aplicaciones de tiempo real, y ofrece una calidad de imagen normal. Esta solución está basada en el circuito LOG-CORDIC que es una implementación a medida de la transformación de coordenadas cartesianas a polares utilizando el algoritmo CORDIC. Esta solución tiene la ventaja de la reconfigurabilidad, que permite ajustar los parámetros de la transformación, y la alta velocidad de ejecución, liberando al procesador de trabajo extra. El problema de esta solución es el mismo que el de la transformación software, y es el error introducido al transformar píxeles cuadrados en log-polares. Sin embargo, para la mayoría de aplicaciones, este error es muy pequeño y no modifica los resultados.

1.1 Circuitos para la transformación entre cartesianas y log-polares

Un circuito para la transformación de imágenes cartesianas a log-polares se puede realizar siguiendo diferentes estrategias. La más simple consiste en una *Look Up Table* (LUT) para la transformación de las coordenadas [3]. En este caso, el circuito consiste en una LUT, implementada mediante ROM o RAM, y un circuito a medida para la transformación de la imagen a partir de los valores guardados en la LUT. El único problema que puede presentar está en la propia LUT, ya que las imágenes cartesianas suelen ser grandes (512×512 o más). Las coordenadas log-polares que se tienen que guardar en la LUT necesitan al menos unos 14 bits (76×128 píxeles) o incluso más. Todo esto implica una LUT de unos 512 Kbytes. Este tamaño se puede reducir teniendo en cuenta la simetría de la transformación, pero aun así harían falta unos 128 Kbytes de memoria. Por otro lado, una limitación en el tamaño de la LUT implica una limitación tanto en la resolución de la imagen cartesiana como en la log-polar.

Otra opción, que ha sido la finalmente implementada, consiste en hacer un circuito para calcular las coordenadas log-polares a partir de las cartesianas. La ventaja de esta opción es que se puede implementar todo en un único módulo, que puede formar parte de un circuito mayor dentro de la FPGA.

Esta ventaja es la que se ha aprovechado en este caso, ya que se dispone de una tarjeta para el desarrollo de aplicaciones (la APEX PCI *development board* de Altera) donde se ha implementado este circuito como parte de una aplicación mayor en las que son necesarias imágenes log-polares. De esta manera se puede implementar el sistema en una sola FPGA (APEX 20K1000CF) sin la necesidad de circuitos externos.

Después de estudiar varias posibilidades para el cálculo aritmético de la transformación, se decidió dividir el cálculo en dos fases: en la primera se calculan las coordenadas polares (radio y ángulo) correspondiente a las coordenadas cartesianas (x, y). En la segunda fase, se calcula el logaritmo del radio, aparte de otras compensaciones y factores, obteniéndose las coordenadas log-polares, que luego se utilizan para pasar de imágenes cartesianas a log-polares.

2 La teoría de CORDIC

CORDIC (COordinate Rotation DIgital Computer) es un algoritmo iterativo para el cálculo de la rotación de un vector de dos dimensiones en coordenadas lineales, circulares o hiperbólicas. La idea de este algoritmo es la realización de rotaciones mediante operaciones simples de suma y desplazamiento. A partir de las ecuaciones

de la rotación, es posible obtener operaciones trigonométricas como el seno y coseno, y otras más complejas como la transformación de coordenadas cartesianas a polares [4].

Sean (x, y) las coordenadas cartesianas de un punto, y (x_n, y_n) el mismo punto pero rotado un ángulo θ. La ecuación de esta rotación es la siguiente:

$$\begin{aligned} x_n &= x_0 \cos\theta - y_0 \sin\theta \\ y_n &= y_0 \cos\theta + x_0 \sin\theta \,. \end{aligned} \tag{1}$$

que se puede rescribir de esta forma más conveniente:

$$\begin{aligned} x_n &= \cos\theta(x_0 - y_0 \tan\theta) \\ y_n &= \cos\theta(y_0 + x_0 \tan\theta) \,. \end{aligned} \tag{2}$$

Si se restringen los ángulos de rotación de manera que $\tan\theta = 1/2^i$ $(i = 0, 1, \ldots, n)$, la multiplicación por la tangente se reduce a un desplazamiento. Por otro lado, es posible descomponer cualquier rotación en una serie de rotaciones cada vez más pequeñas, cuyos ángulos siguen la restricción anterior. De esta manera, cualquier rotación se puede realizar mediante una serie de operaciones de suma y desplazamiento y una multiplicación final (debida al factor $\cos\theta$). En la etapa i de la serie se cumpliría:

$$\begin{aligned} x_{i+1} &= K_i(x_i - y_i d_i 2^{-i}) \\ y_{i+1} &= K_i(y_i + x_i d_i 2^{-i}) \,. \end{aligned} \tag{3}$$

donde:

$$\begin{aligned} K_i &= \cos\theta_i = \cos\left(\frac{1}{\arctan^{-1} 2^{-i}}\right) = \frac{1}{\sqrt{1+2^{-2i}}} \\ A_n &= \prod_{i=0}^{n} \frac{1}{K_i} = \prod_{i=0}^{n} \sqrt{1+2^{-2i}} \\ d_i &= \pm 1 \,. \end{aligned} \tag{4}$$

Se ha definido una constante A_n como el producto de las inversas de los K_i. Este A_n es un factor de ganancia que hay que aplicar al final. Si esta operación CORDIC es parte de un circuito con otras etapas de cálculo, es posible posponer la aplicación de esta constante y calcularla junto con otros factores. Esto es lo que se ha hecho en el circuito de transformación LOG-CORDIC, donde el cálculo de este factor no supone ninguna sobrecarga, pues se ha incluido en la última etapa junto con otros efectos.

El factor d_i fija la dirección de la rotación elemental i-ésima. El signo de d_i se elige con el objetivo de alcanzar el ángulo a rotar. Una manera de que el resultado converja, consiste en definir una variable de diferencias que informe sobre la proximidad del resultado al objetivo en cada etapa. Esta variable es la que se utiliza para calcular d_i:

$$\begin{aligned} z_{i+1} &= z_i - d_i \arctan 2^{-i} \\ d_i &= \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{if } z_i \geq 0 \,. \end{cases} \end{aligned} \tag{5}$$

Las ecuaciones (5) y (3), sin las constantes K_i, se denominan ecuaciones CORDIC en *modo rotación*. Las únicas operaciones necesarias para el cálculo son sumas y desplazamientos, ya que los valores de $\arctan 2^{-i}$ son previamente calculados y guardados

en una pequeña tabla que forma parte del circuito. Estas ecuaciones resultan útiles para el cálculo de funciones trigonométricas. Para ello, dados unos valores iniciales (x_0, y_0, z_0), el circuito CORDIC realiza los siguientes cálculos:

$$\begin{aligned} x_n &= A_n(x_0 \cos z_0 - y_0 \sin z_0) \\ y_n &= A_n(y_0 \cos z_0 + x_0 \sin z_0) \\ z_n &= 0 \end{aligned} \tag{6}$$

Este algoritmo resuelve las operaciones de seno y coseno, sin más que poner el ángulo a calcular en z_0 y dándole a x_0 e y_0 los valores adecuados según la operación. Por ejemplo, el seno se calcula con $x_0 = 1$ y $y_0 = 0$, obteniéndose el resultado en y_n.

Hay otro modo de funcionamiento del CORDIC llamado *modo vector*. En este modo se minimiza el término y_n en lugar del ángulo z_n. Con este simple cambio, en z_n se obtiene θ y en x_n se devuelve r que son precisamente las coordenadas polares:

$$\begin{aligned} x_n &= A_n\sqrt{x_0^2 + y_0^2} \qquad = r \\ y_n &= 0 \\ z_n &= z_0 + \arctan\left(\frac{y_0}{x_0}\right) = z_0 + \theta \end{aligned} \tag{7}$$

La ecuación para el signo de los d_i en el modo vector debe ser tal que se minimice el término y_i. Por lo tanto, la definición de d_i en modo vector queda:

$$d_i = \begin{cases} -1 & \text{if } y_i < 0 \\ +1 & \text{if } y_i \geq 0\,. \end{cases} \tag{8}$$

Este modo vector ha sido el utilizado para transformar las coordenadas cartesianas a log-polares. Las coordenadas (x, y) de cada píxel de la imagen entran en el circuito CORDIC como (x_0, y_0) mientras que $z_0 = 0$. De esta manera, el resultado es directamente el par (r, θ) es decir, el punto (x, y) pero en coordenadas polares. El algoritmo mostrado es el más simple, pues es suficiente para el propósito que se persigue, pero hay otras implementaciones de CORDIC que pueden tener un mayor rendimiento [5].

3 Representación log-polar

El sistema de coordenadas log-polar (ξ, γ) es muy similar al sistema de coordenadas polar (r, θ), ya que $\xi = \log r$ (aproximadamente) y $\gamma = \theta$ (exactamente). La figura 1 muestra la transformación entre el plano focal de la cámara (izquierda) y el plano cortical o representación log-polar de la imagen (derecha). El plano cortical se divide en dos áreas: la fóvea y la retina. La fóvea es la parte central de la imagen y la retina es la periferia. Es necesario hacer esta distinción puesto que las transformaciones en la retina y en la fóvea son diferentes. En la retina se sigue la transformación log-polar típica, pero esta transformación tiene un límite conforme se acerca al centro, puesto que existe una singularidad logarítmica cuando el radio tiende a cero. Por esta razón, hasta un cierto radio, la transformación es polar en lugar de log-polar, lo que permite un mismo plano de procesamiento.

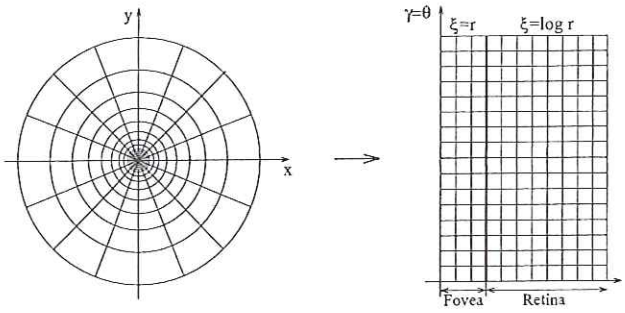

Fig. 1. Representación log-polar; del plano focal al plano cortical

Las siguientes ecuaciones definen las coordenadas (ξ, γ) de la transformación.

$$\xi = \begin{cases} \xi_f \frac{r}{A} & \text{si } r < A \text{ (fóvea)} \\ \xi_f + \frac{1}{B} \log \frac{r}{A} & \text{si } r \geq A \text{ (retina)} \end{cases} \qquad (9)$$
$$\gamma = \theta \, .$$

donde ξ_f es la frontera entre la retina y la fóvea, es decir, el número de circunferencias que tiene la fóvea. Por otro lado, (r, θ) son las coordenadas polares calculadas a partir de las coordenadas cartesianas (x, y) como $r = \sqrt{x^2 + y^2}$ y $\theta = \arctan \frac{y}{x}$.

Las constantes A y B de la ecuación (9), sirven para ajustar la transformación entre la imagen cartesiana y la log-polar: la constante A es el radio de la fóvea medido en píxeles cartesianos, mientras que B es el factor de crecimiento exponencial, cuyo valor se calcula a partir de A y del tamaño total de la imagen cartesiana, de manera que el borde de la imagen log-polar coincida con el borde de la cartesiana.

4 Transformación de las imágenes

El circuito para la transformación de imágenes cartesianas a log-polares está basado en el circuito para pasar de coordenadas cartesianas a log-polares (LOG-CORDIC). La arquitectura del sistema completo se muestra en la figura 2.

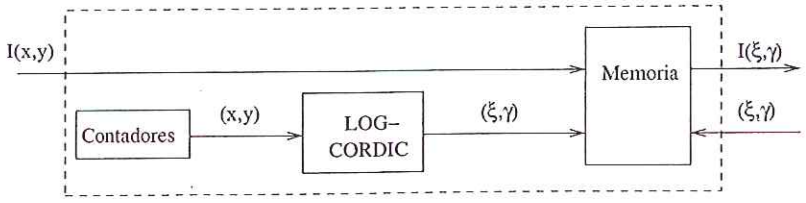

Fig. 2. Diagrama de bloques del circuito para la transformación de imágenes cartesianas a log-polares

El circuito consta de tres bloques: los contadores, el circuito LOG-CORDIC y la memoria. El bloque de contadores genera las coordenadas cartesianas de cada píxel en cada ciclo de reloj, al tiempo que el nivel de gris de un píxel entra en el circuito. Hay un retraso entre las coordenadas y el nivel de gris correspondiente para que coincidan después del cauce segmentado. Las coordenadas (x, y) se transforman en coordenadas log-polares (ξ,γ) en el bloque LOG-CORDIC. Por último, el nivel de gris se escribe en una memoria interna en la dirección (ξ,γ). Esta memoria almacena la imagen log-polar resultante y puede ser leída posteriormente para su procesado.

En realidad, el circuito para la memoria es más complicado que el mostrado en la figura 2, ya que el tamaño de la memoria es menor que $\xi \times \gamma$. El objetivo no es sólo ahorrar bits de memoria en la FPGA, sino también resolver un problema con el mapeado log-polar. Este problema tiene que ver con el hecho de que a un píxel cartesiano le pueden corresponder varios en el plano log-polar. Por ejemplo, si suponemos una resolución de 128 píxeles por circunferencia, un único píxel en el centro de la imagen cartesiana se corresponde con toda la primera circunferencia en la fóvea, es decir, a un píxel cartesiano le corresponderían 128 log-polares. Para resolver este problema, se han colocado pre-codificadores en el bus de direcciones de la memoria, de manera que a un determinado conjunto de varias direcciones le corresponde un mismo byte. En el caso anterior, existe un único byte que es accedido por cualquiera de los 128 píxeles del primer anillo. Lo mismo ocurre con cualquier otro píxel cartesiano, especialmente cercano al centro, donde un mismo píxel cartesiano se corresponde con varios log-polares. La correspondencia entre estos grupos de píxeles ha sido previamente calculada y forma parte de la descripción del circuito.

La implementación del circuito completo se ha realizado en una FPGA de Altera, concretamente la EP20K1000CF672C7ES que es capaz de albergar un gran número de elementos de proceso, de hecho, este circuito ocupa sólo una pequeña parte de la FPGA, siendo en realidad una etapa de una aplicación más grande que incluye una interfaz PCI y varios elementos de proceso. Esta FPGA forma parte de la tarjeta de desarrollo PCI de Altera utilizada para los diferentes experimentos.

5 El circuito LOG-CORDIC

El circuito LOG-CORDIC tiene como misión la transformación de coordenadas cartesianas a log-polares siguiendo la transformación de la ecuación (9). Las constantes A y B de esta ecuación están incluidas en la componente radial ξ. Aun es más, cualquier factor de amplificación que pudiera tener el algoritmo CORDIC, como la constante A_n, también está incluida en el cálculo de ξ. Por otro lado, el único factor que interviene en la componente angular γ, es el número de píxeles por circunferencia. Todos estos factores deben ser incluidos en el circuito.

La componente angular γ se calcula mediante el algoritmo CORDIC. El número de píxeles por circunferencia se codifica en el circuito a través del cálculo de la tabla de arco-tangentes ($\arctan 2^{-i}$). Si los arco-tangentes se calculan en el rango $[0..\gamma_{max})$ (donde γ_{max} es el número de píxeles por circunferencia) el ángulo resultante será directamente el píxel de la circunferencia sin necesidad de realizar ningún otro cálculo posterior. Si en la transformación se cambia el número de píxeles por circunferencia, es necesario reconfigurar la FPGA con una nueva tabla que recoja el cambio. Estos

cambios en la componente angular no afectan al cálculo de la componente radial r, que es posteriormente usada para el cálculo de la componente log-radial ξ.

CORDIC suministra directamente el valor de la componente angular tanto en coordenadas polares como en log-polares, ya que $\theta = \gamma$, sin embargo, sólo entrega la componente radial en coordenadas polares y con algún factor de amplificación que debe ser contrarestado. El resultado radial del circuito CORDIC debe sufrir varias transformaciones hasta obtener la componente ξ: se deben tener en cuenta los factores A_n, A y B, además de las diferencias entre la retina y la fóvea tal como se mostraba en la ecuación (9). En esta transformación, aparte de multiplicaciones, divisiones, sumas y restas, hay que aplicar un logaritmo. En principio es complejo implementar un circuito que realice todas estas operaciones, especialmente el logaritmo.

Como el número de componentes radiales (r) y log-radiales (ξ) es pequeño, es posible calcular una tabla de correspondencias entre ambas componentes. No importa lo compleja que sea la relación entre ambas componentes radiales, pues ésta se calcula fuera de la FPGA. Para unos valores de r y ξ típicos, esta tabla ocupa el 10% de un circuito equivalente que calculase únicamente un logaritmo simple. Por otro lado, la tabla tarda un ciclo de reloj en calcular la correspondencia, mientras que cualquier otro circuito tendría una latencia de varios ciclos, si bien se podría encauzar para tener un resultado por ciclo.

La figura 3 muestra el diagrama de bloques del circuito LOG-CORDIC. El primer bloque calcula el cuadrante donde se encuentra el punto cartesiano, y traslada las coordenadas (x, y) al primer cuadrante, ya que es donde funciona CORDIC. El segundo bloque es el circuito CORDIC en su modo vector, y está formado por varias etapas segmentadas. Cada etapa segmentada calcula una de las iteraciones del algoritmo. Se han implementado 10 etapas. El tercer bloque devuelve el resultado a su cuadrante correspondiente y lo redondea al entero de 10 bits más cercano (internamente CORDIC utiliza 12 bits). El último bloque es la tabla que resuelve todas las contribuciones de diferentes parámetros, además de la transformación lineo-logarítmica (fóvea-retina).

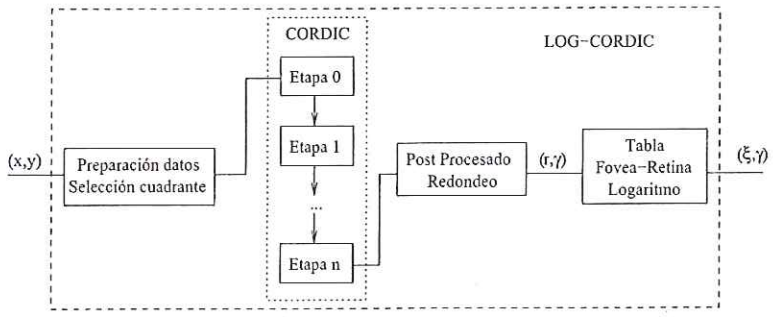

Fig. 3. Diagrama de bloques del circuito LOG-CORDIC

Este circuito LOG-CORDIC ha sido implementado en el dispositivo de Altera mostrado en la sección anterior. Se ha descrito en VHDL y todas las tablas se han calculado con un programa en C++. El diseño de CORDIC tiene como base un *Core* existente descrito en VHDL [6]. A partir de este módulo original se han realizado

modificaciones para adaptarlo a las necesidades particulares del diseño. Se ha definido un bus de datos interno de 12 bits, que es suficiente para imágenes cartesianas de 512×512 píxeles y log-polares de 76×128, aunque se podrían utilizar mayores tamaños sin problemas. Con estas dimensiones el circuito LOG-CORDIC completo utiliza unos 1.200 bloques lógicos de un dispositivo que tiene más de 30.000. El módulo CORDIC ocupa la mayor parte del total con 950 bloques lógicos. Sin un gran esfuerzo en la optimización temporal, el circuito alcanza ya los 60 MHz que es unas diez veces más rápido que lo que necesita la aplicación. Es posible obtener un mayor rendimiento utilizando técnicas para obtener un emplazado eficiente de CORDIC en FPGAs [7,8].

6 Conclusiones

Se ha presentado un circuito para transformar imágenes cartesianas a log-polares. El corazón de este circuito es el circuito LOG-CORDIC que transforma coordenadas cartesianas a log-polares. Se ha modificado una implementación de CORDIC existente para este propósito específico. También se ha añadido circuitería extra para poder compensar los factores de amplificación de CORDIC y para poder realizar la transformación líneo-logarítmica. La reconfigurabilidad de la FPGA es importante, ya que permite el cambio de los parámetros de la transformación durante la ejecución. Este circuito es parte de un proyecto mayor que utiliza el procesamiento de imágenes para la navegación de un robot. La reconfigurabilidad permite cambiar el programa de la FPGA dependiendo de la tarea que se esté llevando a cabo.

Agradecimientos

Este trabajo ha sido financiado por el proyecto TIC2001-3546 del Ministerio de Ciencia y Tecnología Español y los fondos FEDER.

Referencias

1. Pardo, F., Boluda, J.A., Coma, I., Mico, F.: High-speed log-polar time to crash calculation for mobile vehicles. Image Processing & Communications **8–2** (2002) 23–32
2. Pardo, F., Dierickx, B., Scheffer, D.: Space-variant non-orthogonal structure CMOS image sensor design. IEEE Journal of Solid State Circuits **33–6** (1998) 842–849
3. Ruiz del Solar, J., Nowack, C., Schneider, B.: Vipol: A virtual polar-logarithmic sensor. In: Scandinavian Conference on Image Analysis, SCIA'97, Finland (1997) 739–744
4. Andraka, R.: A survey of CORDIC algorithms for FPGA based computers. In: International Symposium on FPGAs, Monterey, California, USA, ACM/SIGDA (1998)
5. Antelo, E., Villalba, J., Bruguera, J., Zapata, E.: High Performance Rotation Architectures Based on Radix-4 CORDIC Algorithm. IEEE Transactions on Computers **46–8** (1997) 855–870
6. Herveille, R.: CORDIC Core Specification. http://www.opencores.org/projects/cordic (2001)
7. Valls, J., Kuhlmann, M., Parhi, K.K.: A unified algorithm for elementary functions. In: IEEE Workshop on Signal Processing Systems (SIPS2000), Louisiana, USA (2000) 336–345
8. Vladimirova, T., Tiggeler, H.: FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm. In: MAPLD'99. (1999)