INICIALIZACIÓN AUTOMÁTICA DE POSE: NUEVOS RESULTADOS UTILIZANDO FPGA'S PARA EL REGISTRO Y APAREAMIENTO DEL MODELO

***José R. Atoche Enseñat*[a], *Luis A. Muñoz Ubando*[b], *Ángel Sebastiá Cortés*[c], *Johan J. Estrada López*[d]**

[a d] Instituto Tecnológico de Mérida, Departamento de Ing. Eléctrica y Electrónica, Yucatán, México.
[b] Universidad Autónoma de Yucatán. Facultad de Matemáticas, Yucatán, México.
[c] Universidad Politécnica de Valencia. Departamento de Ing. Electrónica, Valencia, España.

[a] jatoche@itmerida.mx, [b] luisalbertomunozubando@yahoo.com.mx, [c] asebasti@eln.upv.es

RESUMEN

Con miras a obtener resultados en tiempo real que permitan inicializar la pose de objetos en movimiento, este trabajo presenta los primeros resultados de la implementación hardware en FPGA's de las etapas de: registro del modelo a la nueva posición, cálculo y discriminación de puntos auto-ocluidos y apareamiento de puntos en la imagen y el modelo mediante una adaptación del algoritmo ICP. Se presenta el algoritmo utilizado y se detallan las adecuaciones que se necesitaron para poder implementar en Hardware las etapas mencionadas, finalmente se muestra la diferencia obtenida entre la ejecución del algoritmo en una PC y en un FPGA.

1. INTRODUCCIÓN

Aunque existen algunos métodos que permiten, de manera automática, inicializar la pose de un objeto a partir de su modelo CAD y una imagen, todos ellos además de ser iterativos requieren de una gran cantidad de cálculos complejos, con lo cual es imposible con un procesador convencional poder procesar las imágenes y al mismo tiempo ejecutar el algoritmo de inicialización de pose a la velocidad a la que una cámara estándar captura las imágenes (30 cuadros por segundo).

El método de inicialización de pose elegido es el presentado por Wunsch e Hirzinger en [1]. Como las imágenes de la cámara no producen una información completa de las 3D, la idea clave propuesta en este articulo es explotar la perspectiva inversa, formada por los rayos de proyección determinados por el punto focal de la cámara y los puntos característicos en la imagen, para buscar la correspondencia con el modelo del objeto en el espacio 3D. Wunsch no presenta en su artículo ninguna propuesta para la etapa detección de bordes y obtención de los puntos característicos de la imagen. En [2] se presenta una solución implementable en hardware para la detección de bordes, utilizando el algoritmo SUSAN [3]. En [4] se analiza la unión del método SUSAN con el método de Wunsch y se desarrolla de manera más amplia la función de los Estimadores M como estimadores robustos dentro del método planteado por Wunsch. De [5] el dato más importante que utilizamos, es la corrección de la ecuación para el cálculo de la matriz de rotación presentada por Wunsch, aunque en [5] se plantea la utilización de imágenes en color y la modificación necesaria del algoritmo SUSAN, en el presente trabajo se utilizarán imágenes en tonos de gris. En [6] se presentan los detalles de la implementación hardware tanto del método SUSAN para detección de contornos como de la etapa de extracción de puntos característicos de la imagen. En esta etapa se realiza un muestreo de la imagen de contornos y tomando en cuenta los parámetros de calibración de la cámara, se entrega una lista de puntos 3D sobre un plano paralelo al de la imagen, que representan los rayos de proyección generados por los puntos muestreados. A partir de este momento el algoritmo ya no trabaja con imágenes sino con listas de datos, reduciéndose enormemente, con esto, la cantidad de información a procesar.

El siguiente paso del algoritmo es el registro del modelo a su posición actual, después de lo cual deben eliminarse del modelo los puntos que se encuentran auto-ocluidos. La lista de puntos resultante es introducida al algoritmo de apareamiento (una variante del ICP, Iterative Closest Point, introducido por Besl y Mckay [7]) junto con la lista de puntos extraída de la imagen. El algoritmo de Apareamiento entrega una lista de pares de puntos 3D (Imagen, Modelo), las etapas siguientes del algoritmo buscan minimizar el error entre estos pares de puntos. Este problema es resuelto por el método de mínimos cuadrados, generando una matriz de correlación cruzada (Qxy) a la cual se le aplica una descomposición en valores singulares (Qxy = UDV) y se calcula la matriz de rotación que

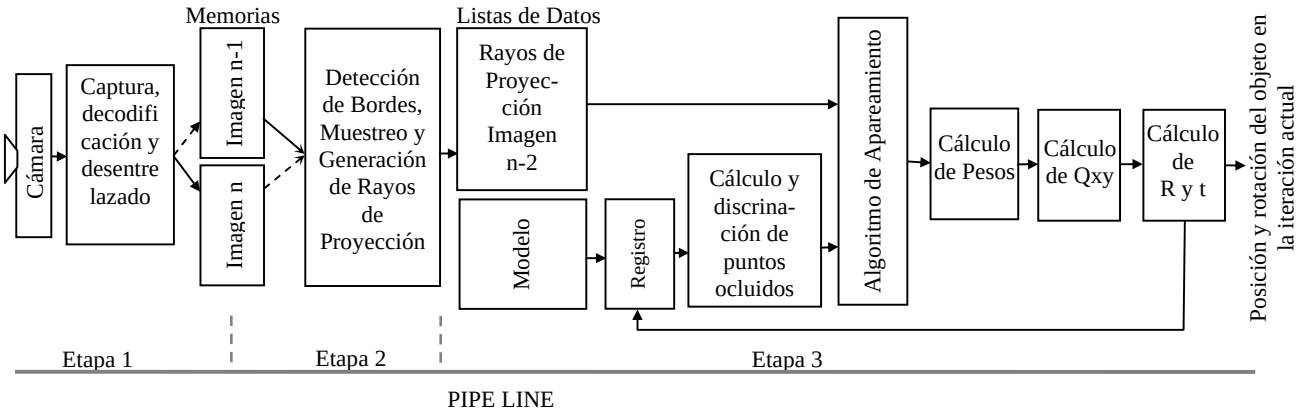

Figura 1. Esquema general de los módulos para la implementación Hardware.

minimiza los errores, mediante la descomposición polar de Qxy ($R = UV^T$) [5] [8], finalmente se calcula el vector de traslación.

Con esto hemos obtenido los datos para realizar una nueva iteración, registramos el modelo a la nueva posición, calculamos el apareamiento, la matriz de correlación cruzada, etc. La idea planteada es poder realizar la mayor cantidad de iteraciones posibles durante el tiempo en que una nueva imagen es capturada por la cámara. Como se muestra en [5] y [6] la mayor parte de los cambios se dan en las primeras 20 a 50 iteraciones, aunque el modelo no llegaba a converger, sin embargo se ha diseñado un controlador sencillo que permite al modelo converger en las primeras 30 a 40 iteraciones aún cuando se encuentre relativamente alejado de la posición final. Con esto el objetivo se fija en alcanzar unas 10 o 20 iteraciones por cuadro, ya que aún cuando el modelo se encuentre alejado de la posición real del objeto bastaría con algunos cuadros para lograr su registro. Cuando el modelo se encuentra en posiciones muy cercanas a la posición real del objeto la convergencia se logra en las primeras iteraciones.

En el artículo se presentan los detalles de la implementación hardware de las etapas de: registro de los puntos del modelo, cálculo y discriminación de puntos auto-ocluidos por las caras del modelo y apareamiento entre los puntos en la imagen y los puntos visibles en el modelo mediante una adaptación del algoritmo ICP. Finalmente se presenta una comparación de los tiempos empleados para ejecutar dichos algoritmos en la PC y en el FPGA.

2. DESCRIPCIÓN DEL ALGORITMO

El algoritmo de inicialización de pose puede representarse en los siguientes pasos:

- Extracción de puntos característicos de la imagen
  - Filtrado de Bordes.
  - Muestreo y generación de lista de puntos 3D (rayos de proyección de la imagen).
- Algoritmo de inicialización de pose.
  - Registro del modelo a la posición actual.
  - Cálculo y discriminación de puntos auto-ocluidos.
  - Apareamiento de puntos en el modelo y en los rayos de proyección de la imagen.
  - Calculo de la Matriz de correlación cruzada (Qxy).
  - Descomposición de Qxy en valores singulares.
  - Cálculo de la nueva posición del modelo (R y t).

Para la implementación Hardware del algoritmo se han utilizado técnicas de paralelización y de pipe line, tanto para cada una de las etapas como para el algoritmo en su conjunto. El siguiente dibujo muestra la forma en que se realiza el pipe line en el algoritmo general. Las 3 etapas de pipe line nos permiten conocer la posición del objeto solo 2 cuadros después de que la cámara terminó de tomar la imagen. La primera etapa de pipe line es necesaria debido a que la cámara que se esta utilizando es de formato entrelazado, si se tuviera una cámara no entrelazada podría evitarse esta etapa reduciéndose la latencia a sólo 1 cuadro.

3. REGISTRO DEL MODELO

Para reducir el número de operaciones complejas necesarias en el cálculo de los puntos auto-ocluidos, se ha incluido información adicional en el modelo, transformando el modelo de 2 listas presentado en [5] a un modelo de 4 listas, de las cuales 3 deben registrarse.
Todas las listas a registrar están formadas por coordenadas 3D, de manera que el algoritmo de registro se reduce a aplicar la siguiente ecuación a cada punto a registrar:

$$y_i = \mathbf{R}\, x_i + \mathbf{t} \qquad \text{Ec. 1}$$

donde x_i es el punto a registrar, $\mathbf{R}$ es la matriz de rotación de 3 x 3, $\mathbf{t}$ es el vector de traslación y y_i es el punto rotado y trasladado. Efectuando la suma tenemos que:

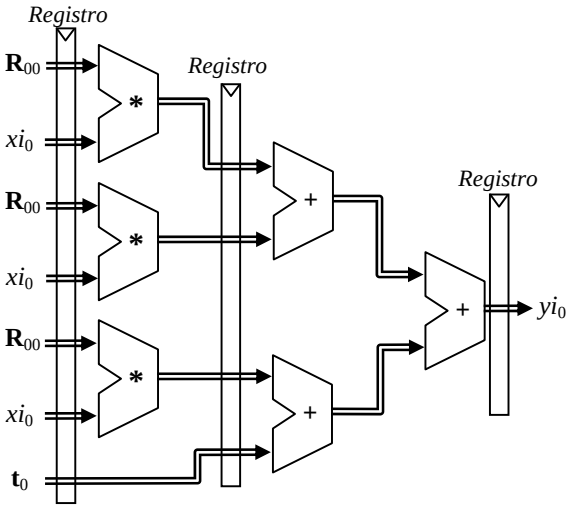

Figura 2. Conexión del componente 0 de *yi*

$$yi = \begin{bmatrix} yi_o \\ yi_1 \\ yi_2 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{00}xi_0 + \mathbf{R}_{01}xi_1 + \mathbf{R}_{02}xi_2 + \mathbf{t}_0 \\ \mathbf{R}_{10}xi_0 + \mathbf{R}_{11}xi_1 + \mathbf{R}_{12}xi_2 + \mathbf{t}_1 \\ \mathbf{R}_{20}xi_0 + \mathbf{R}_{21}xi_1 + \mathbf{R}_{22}xi_2 + \mathbf{t}_2 \end{bmatrix} \quad \text{Ec.2}$$

por lo cual para implementar cada uno de los componentes de y_i se necesitan tres multiplicadores y tres sumadores como se muestra en la figura 2. Con el fin de utilizar al máximo los recursos disponibles se ajustaron todos los algoritmos a operaciones de punto fijo de 18 bits, ya que los multiplicadores que traen integrados los FPGA's son de esa capacidad.

Las entradas se establecieron en formato 11.7 con signo (11 bits enteros 7 fraccionarios) en cm., lo cual permite obtener coordenadas de hasta 10 metros con precisión de +- 0.1 mm. Las salidas de los multiplicadores y los sumadores subsecuentes se ajustaron a este formato. Puede verse con esto que implementar el registro en paralelo utiliza 9 multiplicadores y 9 sumadores de 18 bits. Registrando las entradas y salidas de los multiplicadores se logra una frecuencia de funcionamiento arriba de los 100 MHz, logrando con esto, registrar un punto 3D por pulso de reloj.

4. DETECCIÓN DE PUNTOS AUTO OCLUIDOS

La idea base es muy simple, se trata de probar cada punto del modelo para ver si se encuentra ocluido por el segmento de plano formado por cada cara del mismo. Para probar si un punto 3D se encuentra ocluido por un segmento de plano calculamos la proyección del punto en el plano y valoramos una de las siguientes 3 posibilidades excluyentes:

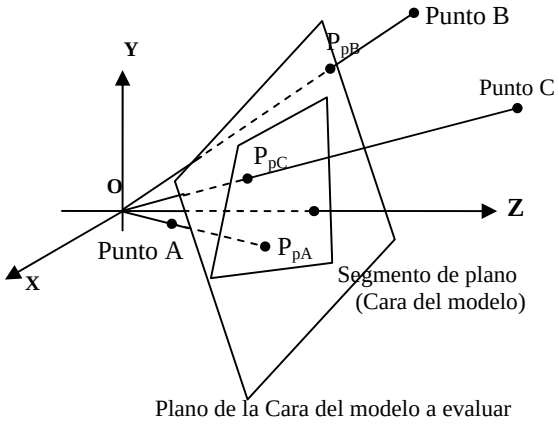

Figura 3. Oclusión de un punto por un segmento de plano.

- La norma de la proyección al plano es mayor que su propia norma, el punto no puede ser ocluido (se encuentra adelante del plano). Punto A Figura 3.
- La proyección se encuentra fuera de la cara, el punto no puede ser ocluido. Punto B Figura 3.
- La proyección se encuentra dentro de la cara, el punto se encuentra ocluido. Punto C Figura 3.

La proyección del Punto P(a, b, c) sobre el plano se encuentra con la siguiente ecuación:

$$\mathbf{P_p} = (a \cdot kte, b \cdot kte, c \cdot kte) \quad \text{Ec. 3}$$

$$kte = \frac{\mathbf{n_p} \cdot \mathbf{V_1}}{\mathbf{n_p} \cdot \mathbf{P}} \quad \text{Ec. 4}$$

donde $\mathbf{n_p}$ es un vector normal al plano que se incluye como información adicional del modelo y $\mathbf{V_1}$ es un punto en el plano (se utiliza uno de los vértices de la cara). Al registrarse el modelo $\mathbf{n_p}$ debe rotarse con traslación 0 ($\mathbf{t}$ = (0, 0, 0)). El siguiente paso es encontrar la norma del punto y de su proyección, lo cual implica 3 potencias y una raíz cuadrada, sin embargo, como ambos puntos se encuentran sobre la misma línea y lo único que se necesita es una medida proporcional a su distancia al origen se ha utilizado en la implementación hardware lo siguiente:

$$dP = || Px || + || Py || + || Pz || \quad \text{Ec. 5}$$

$$dP_p = || P_px || + || P_py || + || P_pz || \quad \text{Ec. 6}$$

donde al comparar dP con dP_p podemos saber cual se encuentra más cercano al origen. Finalmente debemos averiguar si la proyección se encuentra dentro del polígono formado por la cara, para esto, el procedimiento consiste en averiguar si el punto se encuentra "adentro" o "afuera" de cada una de las líneas definidas por las aristas del

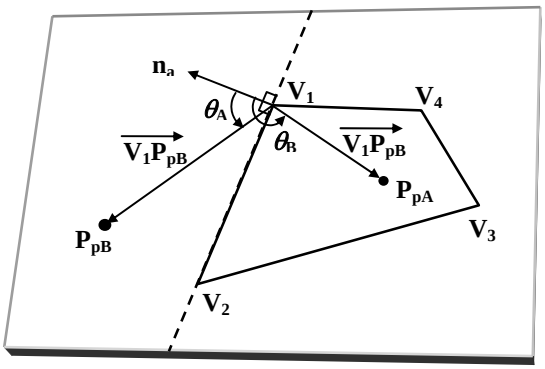

Figura 4. Si θ es obtuso $\mathbf{P_p}$ se encuentra hacia "adentro" del polígono.

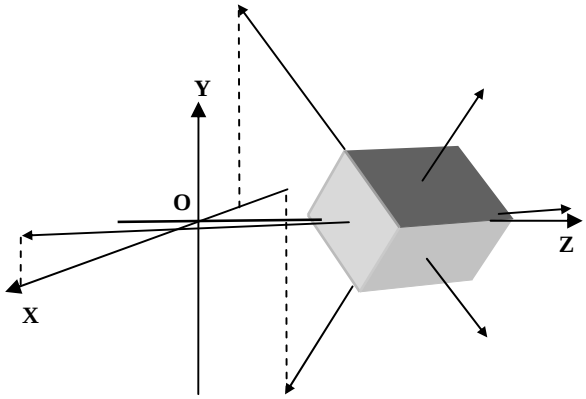

Figura 5. Los planos cuya normal no corta el plano XY no son visibles desde el origen

polígono. "Adentro" significa que se encuentra del mismo lado de la arista que el resto del polígono, "afuera" es el lado opuesto. Si el punto $\mathbf{P_p}$ se encuentra "adentro" de ***todas*** las aristas, se encuentra dentro del polígono, y por tanto, **P** está ocluido por el mismo. Para averiguar de que lado de la arista se encuentra $\mathbf{P_p}$ utilizamos la siguiente ecuación:

$$\mathbf{n_a} \cdot \mathbf{V_1P_p} = \| \mathbf{n_a} \| \, \| \mathbf{V_1P_p} \| \, \text{Cos}\, \theta \qquad \text{Ec. 7}$$

donde $\mathbf{V_1P_p}$ es el vector $\mathbf{P_p} - \mathbf{V_1}$, $\mathbf{V_1}$ es uno de los vértices de la arista y $\mathbf{n_a}$ es un vector perpendicular a la arista en cuestión dirigido hacia "afuera" que se agrega al modelo como información adicional y que debe rotarse con traslación 0 al registrar el modelo. De la figura 4 podemos ver que solo si $\mathbf{n_a} \cdot \mathbf{V_1P_p} < 0$ podemos decir que $\mathbf{P_p}$ se encuentra hacia adentro del polígono.

($\theta < 90^o$)	Cos $\theta > 0$	**y**	$\mathbf{n_a} \cdot \mathbf{V_1P_p} > 0$
($\theta = 90^o$)	Cos $\theta = 0$	**y**	$\mathbf{n_a} \cdot \mathbf{V_1P_p} = 0$
($\theta > 90^o$)	Cos $\theta < 0$	**y**	$\mathbf{n_a} \cdot \mathbf{V_1P_p} < 0$

Tabla 1. Si $\theta > 90°$ $\mathbf{n_a} \cdot \mathbf{V_1P_p}$ es negativo.

Hasta aquí se han mostrado los procedimientos a seguir para resolver los problemas planteados, ahora se mostrarán algunas mejoras que aceleran la ejecución del proceso.

Dado que se ha incluido como información adicional en el modelo la normal a cada cara del modelo, mas aún, la normal a la cara, dirigida hacia fuera del modelo, podemos rápidamente encontrar que planos no son visibles y eliminarlos en la búsqueda de puntos ocluidos. El algoritmo de decisión es: si la normal a la cara cruza el plano XY, es decir, su componente Z < 0, la cara puede ser visible, en caso contrario no es visible y por tanto no puede ocluir ningún punto (Figura 5).

Con lo anterior podemos excluir de la búsqueda todas aquellas caras que no son visibles, reduciendo importantemente el número de búsquedas a realizar.

Para la implementación hardware, de la cual se muestra en la Figura 6 un esquema de el data path, en este diseñó se utiliza una arquitectura en pipe line, que acepta datos cada dos pulsos de reloj. Para esta arquitectura alimentamos el punto **P** a probar y vamos cambiando las aristas de las caras contra las cuales probamos el punto.

El primer resultado que debemos calcular es la proyección del punto al plano, esto se realiza mediante las ecuaciones 3 y 4. Después de esto por un lado se encuentran las normas de **P**p y de **P** de acuerdo a las ecuaciones 5 y 6 y se comparan para ver si el punto se encuentra detrás del plano; y por otro calculamos el producto punto $\mathbf{n_a} \cdot \mathbf{V_1P_p}$ para ver si el punto esta "dentro" de la arista. Finalmente solo si ambos $\|\mathbf{P}p\| - \|\mathbf{P}\|$ y $\mathbf{n_a} \cdot \mathbf{V_1P_p}$ son negativos para cada una de las 4 aristas de la cara el punto estará ocluido, poniendose a '0' la señal Pvisible al mismo tiempo que el No. De arista de salida se encuentra en 3, con lo cual se debe marcar el punto indicado por la señal IndicePunto como ocluido.

El control de esta arquitectura lo ejecutan dos máquinas de estados, una se encarga de cargar los datos en las entradas y la otra de manejar los datos de salida.

5. APAREAMIENTO

Con el fin de establecer la correspondencia entre los rayos de proyección (lista PuntosI) y los puntos del modelo (ListaVisibles), utilizamos una adaptación del algoritmo de puntos cercanos (ICP: Iterative Closest Point), introducido por Besl y Mckay [7]. Cada rayo de proyección esta representado por una línea definida por el origen y el punto en el plano de la imagen cuyas coordenadas son (PuntoI(i)_x, PuntoI(i)_y, f). El objetivo es buscar, para cada rayo de proyección, el punto más cercano en el modelo y almacenar en la lista de salida las coordenadas de los

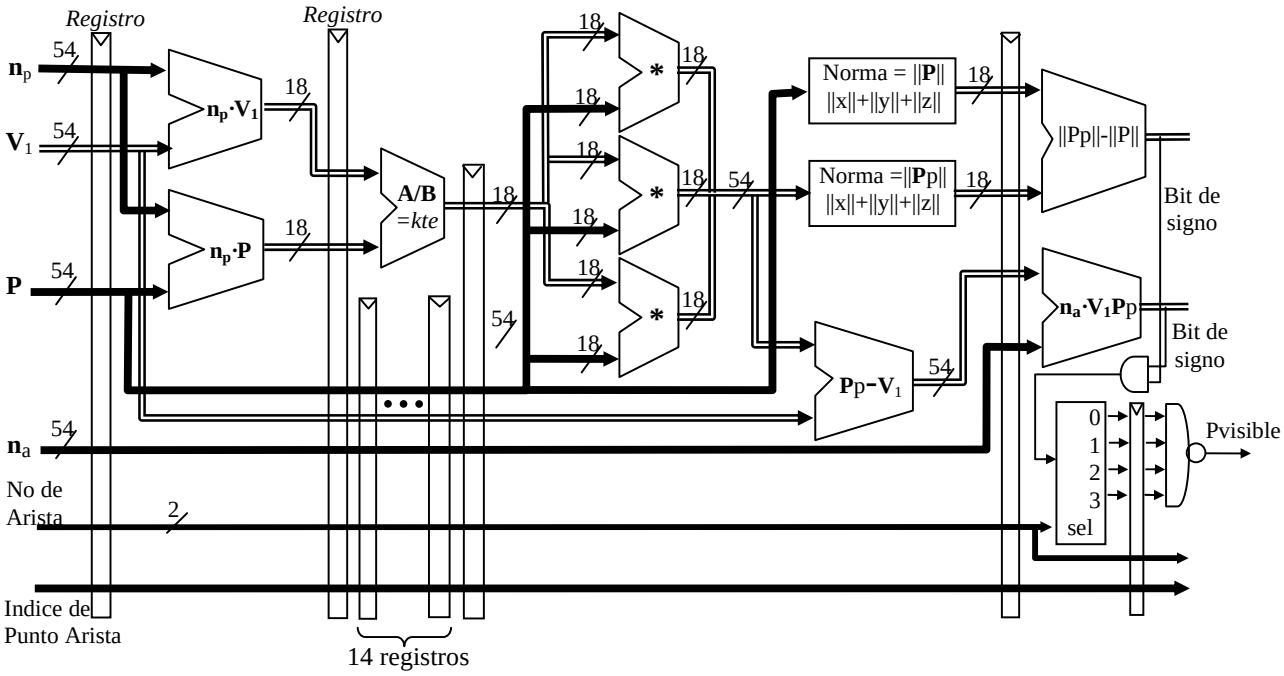

Figura 6. Data Path para el cálculo de Puntos Ocluidos

puntos apareados en el rayo de proyección y en el modelo. Para resolver esto es necesario hallar la distancia mas corta entre un punto (P) y una recta en el espacio (Figura 7). Como dicha recta pasa por el origen y por el punto PuntoI(i), esto se logra haciendo una descomposición del vector P en sus componentes paralelo (Pr) y perpendicular (Pc) al vector PuntoI(i). La norma de la componente perpendicular es la distancia mas corta entre el punto P y la línea, y la componente paralela es el punto sobre el rayo de proyección que genera esta distancia. Esto se logra con las ecuaciones que se muestran a continuación:

$$\mathrm{Pr} = \frac{\mathrm{P} \cdot \mathrm{Punto(i)}}{\mathrm{PuntoI(i)} \cdot \mathrm{PuntoI(i)}} \mathrm{PuntoI(i)} \qquad \text{Ec. 8.}$$

$$\|\mathrm{Pc}\| = \|\mathrm{P} - \mathrm{Pr}\| \qquad \text{Ec. 9.}$$

Obteniendo la distancia mas corta entre un rayo de proyección y un punto en el modelo, se repite esta operación y se guarda el punto en el modelo y en el rayo de proyección que generaron la distancia más corta. Se realiza esta operación con todos los rayos de proyección y se genera la Lista de Pares. En la ecuación 9 para evitar la raíz cuadrada inherente al cálculo de la norma únicamente se toma la suma del cuadrado de cada componente quedando como:

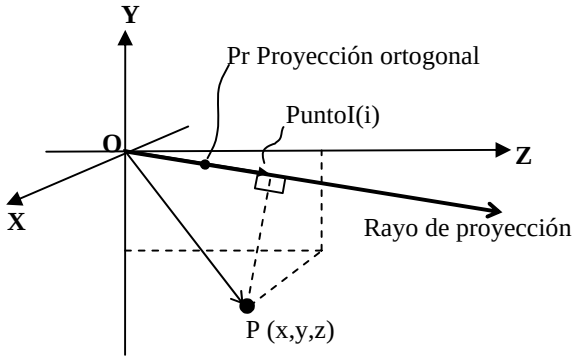

Figura 7. Distancia de un punto a una recta en el espacio

$$\|\mathrm{Pc}\| = \left(\mathrm{P_x} - \mathrm{Pr_x}\right)^2 + \left(\mathrm{P_y} - \mathrm{Pr_y}\right)^2 + \left(\mathrm{P_z} - \mathrm{Pr_z}\right)^2 \qquad \text{Ec. 10}$$

En la implementación hardware se ha utilizado una arquitectura en pipeline que acepta datos cada pulso de reloj, con lo cual ahora incluso los productos punto deben incluir un registro intermedio después de los multiplicadores.

El primer paso es calcular los productos punto: P·PuntoI(i) y PuntoI(i)·PuntoI(i), se dividen y multiplican de acuerdo a la ecuación 8 para obtener Pr, después se calcula la distancia de acuerdo a la Ec. 10, finalmente se almacena el par de puntos para el primer dato de la lista evaluado, luego solo se guarda si la distancia calculada es

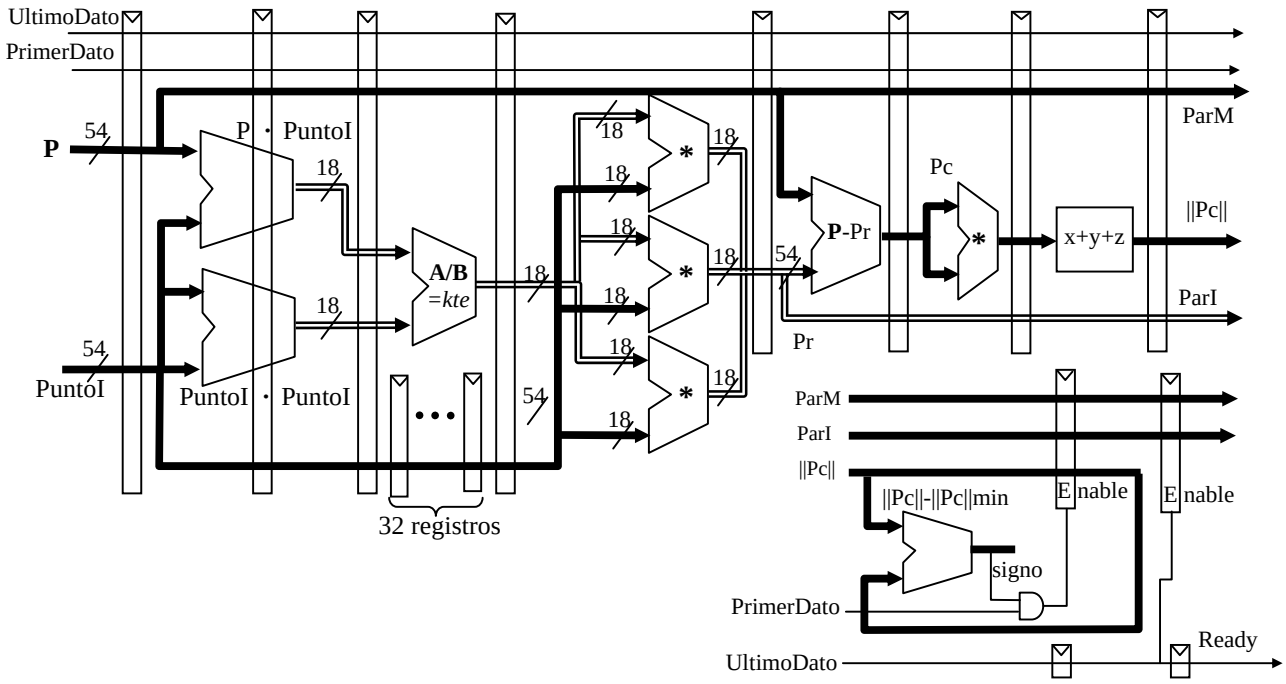

Figura 8. Data Path para el apareamiento

menor a la guardada. Cuando se llega al último dato se carga el par de puntos en el registro de salida y se tiene un pulso en la señal ready.

El control lo ejecutan dos máquinas de estados, una se encarga de cargar los datos en las entradas y la otra de manejar los datos de salida.

6. RESULTADOS Y TRABAJOS FUTUROS

A continuación en la Tabla 2 se presentan los resultados obtenidos al implementar los esquemas planteados y sus controles en una Virtex 2 XC2V200 de Xilins:

	Registro	Puntos Ocluidos	Aparea-miento	Total
Multiplicadores	9	12	12	33
Slices	7%	18%	31%	48%

Tabla 2. Implementación en una Virtex2 XC2V2000

En la tabla 3 mostramos la aceleración obtenida al compara el sistema implementado en el FPGA, funcionando con un reloj interno de 100 MHz, contra el mismo sistema implementado en software en una PC con un Procesador AMD a 1.33 GHz con sistema operativo Linux. Se está trabajando en la implementación del algoritmo planteado en un FPGA Virtex 4 en el cual se espera obtener el doble de velocidad en frecuencia de reloj y al tener más área disponible se podra aumentar el paralelismo en el cálculo del apareamiento con lo cual se obtendrán a su vez mejoras en el tiempo total del proceso.

Como trabajos futuros, se está trabajando en la implementación total del algoritmo para lograr generar una herramienta que deje libre al procesador principal para que se dedique a labores de más alto nivel, de manera que puedan ser probadas en tiempo real aplicaciones de robótica, manipulación, control, etc.

Proceso	Tiempo en ms		
	Software	Hardware	aceleración
Registro	0.0882	0.00836	10.6
Puntos Visibles	1.4230	0.27663	5.1
CLP	25.9500	0.83502	31.1
Total	27.4612	1.12001	24.5

Tabla 3. Comparación de tiempos de ejecución.

7. REFERENCIAS

[1] P. Wunsch, G. Hirzinger, “Registration of CAD-Models to Images by Iterative Inverse Perspective Matching”, 13th International Conference on Pattern Recognition. Viena, Austria, pp. 77-83, Agosto 1996.

[2] Migel Arias-Estrada, César Torres-Huitzil, “A Real-time FPGA Architecture for Computer Vision”, Journal of Electronic Imaging (SPIE-IS&T), Vol 10, No 1, pp. 289-296, January 2001.

[3] S. M. Smith, J. M. Brady, “SUSAN – A New Approach to Low Level Image Processing”, Technical Report, FMRIB, 1995.

[4] J. R. Atoche Enseñat, M. Rojas Solís, L. A. Muñoz Ubando, “Introducción a los Métodos de Registro 2d a 3d en Visión Artificial.”, Congreso de Instrumentación SOMI XVII, Mérida, Yucatán, México, Octubre de 2002.

[5] J. R. Atoche Enseñat, M. Rojas Solís, L. A. Muñoz, A. Espinosa, “Registro de modelos 3D en imágenes utilizando Perspectiva inversa”, VI Congreso Mexicano de Robótica, COMRob 2004, Torreón, Coah., México, Octubre de 2004.

[6] J. R. Atoche Enseñat, L. A. Muñoz, A. Espinosa, Á. Sebastia C. “Towards The Automatic Pose Computation Via Registration: New Results On Computing Edges Using Fpga’s”, 8th International Symposium on Laser Metrology, Mérida, Yucatán, México, Febrero de 2005.

[7] P. J. Besl and N. D. McKay, “A method for Registration of 3-D Shapes”, IEEE Trans. Pattern Analysis and Machine Intelligence, 14 (2), Febrero 1992, pp 239-256.

[8] G. Nakos y D. Joyner. Álgebra Lineal con Aplicaciones. International Thomson Editores. México 1999.