# Proyección de nube de puntos 3D a través de imágenes estereoscópicas con OpenCV y CUDA

## 3D point cloud projection through stereoscopic images with OpenCV and CUDA

Autor: Luis David Arias Manjarrez
*Facultad de Ingenierías, Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, Pereira, Colombia*
e-mail: david-0296@utp.edu.co

***Resumen*— En la actualidad, las maquinas requieren más información visual de lo que se requería anteriormente. Mientras que hace algunos siglos el mayor reto era fijar permanentemente una imagen por medio de la luz, hoy en día se busca la mejor forma de capturar la información de todo un objeto y proyectarla correctamente a altas velocidades. Algunas técnicas como la estereoscopia, fotogrametría y nube de puntos han permitido solucionar parcialmente este problema, pero al trabajar con tanta información, no son lo suficientemente óptimas para procesarse en una CPU convencional. La presente investigación, tiene como objetivo realizar una demostración del mapeo de objetos en un espacio 3d a través de la estereoscopia, y posteriormente realizar una comparación de la velocidad del procesamiento en CPU vs GPU.**

***Palabras clave*— Fotogrametría, estereoscopía, nube de puntos, visión por computador, CUDA, OpenCV, procesamiento masivamente paralelo, computación de alto rendimiento.**

***Abstract*— Currently, machines require more visual information than was previously required. While a few centuries ago the biggest challenge was to permanently fix an image by means of light, nowadays we are looking for the best way to capture the information of an entire object and project it correctly at high speeds. Some techniques such as stereoscopy, photogrammetry and point cloud have partially solved this problem, but when working with so much information, they are not optimal enough to be processed in a conventional CPU. The present investigation, has like objective realize a demonstration of the mapping of objects in a space 3d through the stereoscopy, and later realize a comparison of the speed of the processing in CPU vs GPU.**

***Key Word* —Photogrammetry, stereoscopy, point cloud, computer vision, CUDA, OpenCV, massively parallel processing, high performance computing.**

## I. INTRODUCCIÓN

Los usuarios perciben un mundo tridimensional, hacen sus movimientos sobre 3 ejes y diferencian lo que está lejos y lo que está cerca de ellos, mientras que las computadoras solo se limitan a recibir instrucciones de un puntero que se mueve en X e Y. ¿Y si se lograra hacer que una maquina perciba el espacio como lo percibe un usuario normal?

Las computadoras reciben cientos de imágenes cada segundo, sean de la web o de sus propias cámaras, imágenes que, si se pudieran enlazar de una forma más precisa, se lograría una proyección más acertada de los problemas y se resolverían de una forma más fácil.

El problema es que para procesar tantas imágenes y encontrar una relación entre ellas una CPU no ofrece el rendimiento necesario para hacerlo, por lo que la solución mas viable en la actualidad es el uso de la computación masivamente paralela.

A pesar de que las GPUs se fabricaron para procesar videojuegos, estos dispositivos compuestos por cientos de núcleos de procesamiento han aportado grandes avances científicos a la solución de problemas de computer visión, la misma rama de la computación que se trabajará en el presente artículo.

## II. COMPRENDIENDO LA VISION HUMANA

En algún momento el hombre se llegó a preguntar el por qué poseía dos ojos en lugar de uno solo, y aunque la respuesta la consiguió con los tropiezos, llegó a la conclusión de que un solo ojo no es suficiente para sobrevivir en un mundo lleno de obstáculos.

Fecha de Recepción: 5 de Diciembre de 2017
Fecha de Aceptación:

Cada ojo ve el mismo objeto desde un ángulo distinto, revelando las partes que el otro ojo no puede captar.

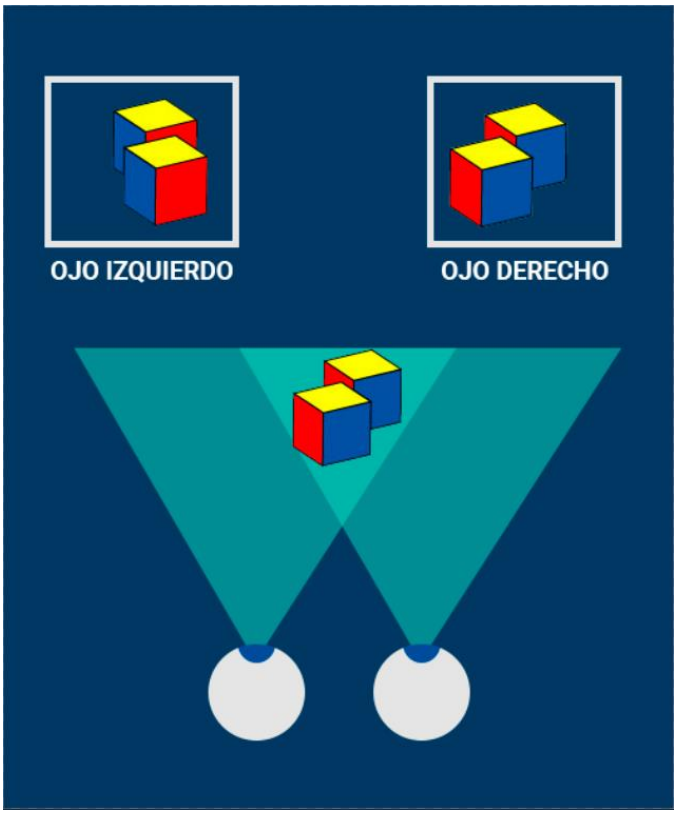


*Figura 1 Representación aproximada de la visión estereoscópica*

La visión estereoscópica es la capacidad que tiene el ser humano de integrar en una imagen tridimensional (con longitud, altura y profundidad), las dos imágenes que proveen sus ojos.

Muchos inventos se basaron en la visión humana y sus fenómenos; la estereoscopia se utiliza tanto para la ciencia, como para el entretenimiento, de aquí nacen las películas en 3D y todas sus variaciones: anaglifo, polarizado, activo, parallax, etc.

Un invento esencial para esta investigación, inspirado totalmente en la visión humana, es la cámara fotográfica, la cual ha tenido una evolución sorprendente desde su invención en el siglo XVII. Esta caja oscura con material fotosensible permite capturar la luz que recibe y fijarla de forma permanente en un lienzo físico o virtual. El problema se encuentra en que se esta proyectando un espacio tridimensional en un plano de dos dimensiones por lo que toca recurrir a la estereoscopia para reconstruir parcialmente el objeto fotografiado.

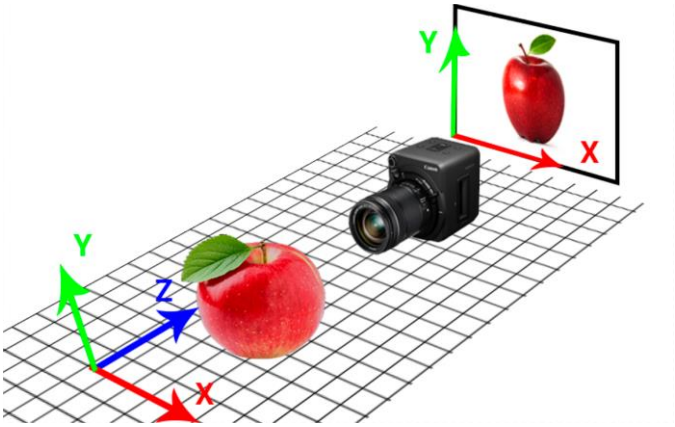


*Figura 2 Diferencia de dimensiones espaciales en fotografía simple*

## III. MODELANDO EL PROBLEMA

Teniendo en cuenta que hay un cambio de dimensiones al tomar una única fotografía, el presente artículo tiene como objetivo evidenciar la efectividad de utilizar múltiples fotografías para reconstruir una imagen en 3 dimensiones a una velocidad considerable.

Todo el procesamiento se compone de las siguientes fases:

- Lectura y calibración de 2 imágenes estereoscópicas
- Análisis de correspondencia de imágenes
- Cálculo de disparidades
- Proyección en nube de puntos de la imagen parcialmente reconstruida.

### A. CALIBRACIÓN DE CÁMARAS

Trabajar con cámaras en computer visión requiere de un preprocesamiento de las imágenes antes de abordar el problema objetivo. Este preprocesamiento comprende lo que es la reducción de distorsiones y rectificación de imágenes (si es necesario).
Una imagen puede ser afectada por dos tipos de distorsiones, distorsión radial y distorsión tangencial.

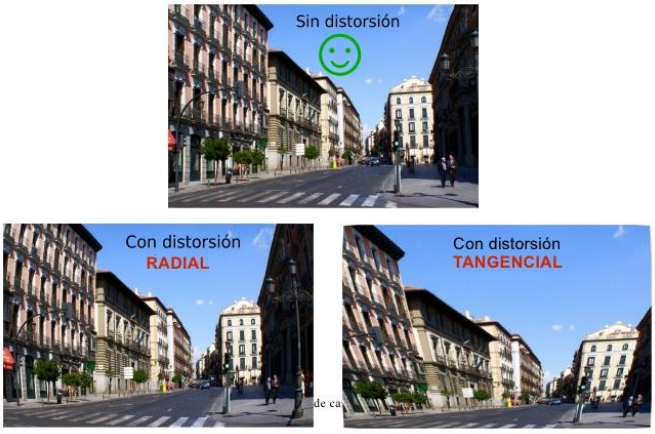


*Figura 3 Tipos de distorsiones (Radial y Tangencial), Jorge Tarlea, Sistema de posicionamiento de objetos mediante visión estéreo embarcable en vehículos inteligentes*

La distorsión radial es producida cuando se usan distancias focales cortas (lentes de mayor campo de visión) mientras que la distorsión tangencial se produce por errores de construcción (falta de alineación entre los componentes ópticos).

Otro de los problemas que se pueden presentar al tomar una fotografía se debe a la perspectiva cónica, en computer vision a veces es necesario tener una imagen con perspectiva ortogonal.

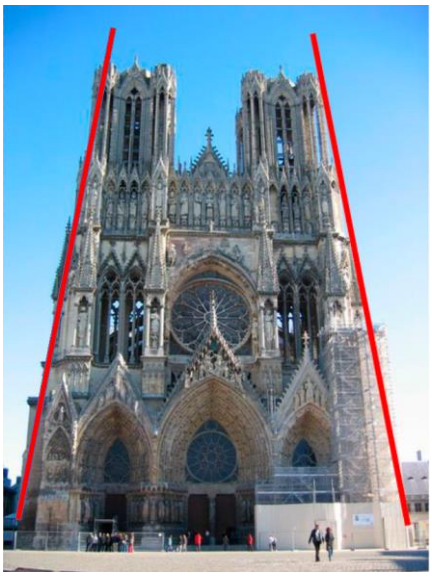

*Figura 4 Problemas de perspectiva, Catedral de Reims, Wikiwand.*

Todos los fenómenos anteriormente mencionados se corrigen mediante un cálculo utilizado por OpenCV para la calibración de cámaras.

*Ecuación 1 Cálculo de calibración de cámaras*

$$s\, m' = A[R|t]M'$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- **(X,Y,Z)** son las coordenadas del punto 3d en el espacio
- **(u,v)** Son las coordenadas del punto de proyección en píxeles
- **A** una matriz de cámara, o una matriz de parámetros intrínsecos
- **(cx,cy)** es un punto principal que generalmente se encuentra en el centro de la imagen
- **fx,fy** son las distancias focales expresadas en unidades de píxeles.
- **[R | t]** es la matriz de rotación-traducción conjunta se denomina matriz de parámetros extrínsecos.

Para la correcta calibración de la cámara es necesario tener las matrices de parámetros intrínsecos y extrínsecos definidas. La matriz de parámetros extrínsecos permite obtener la proyección 2D sobre el plano de la cámara de las coordenadas 3D de un punto de la escena. Con la matriz de parámetros intrínsecos se consiguen las coordenadas 2D en pixeles de la imagen. Cabe aclarar que en el problema trabajado en esta investigación no se requiere realizar la rectificación de las imágenes.

## B. CORRESPONDENCIA DE IMÁGENES

Este proceso se encarga de encontrar los puntos similares entre las 2 imágenes estereoscópicas. Existen algunas líneas a seguir para resolver el problema:

- Usar un enfoque discreto, encontrando correspondencias en puntos determinados.
- Usar correspondencias para cada pixel de las imágenes, guiándose por los colores.

OpenCV trabaja con el concepto de Geometria Epipolar para el proceso de correspondencia de imágenes.

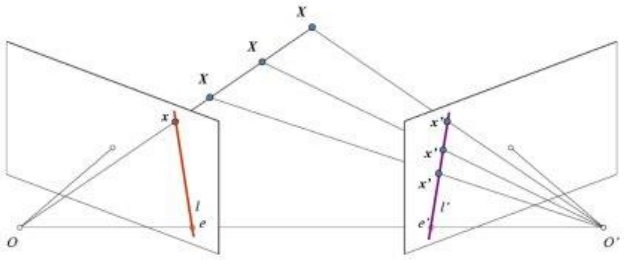


*Figura 5 Geometría de vistas múltiples. Fuente: Documentación OpenCV*

Analizando la *Figura 5*, si se usara solo la cámara izquierda, no se podría encontrar el punto 3D correspondiente al punto $x$ en la imagen ya que la línea $OX$ puede proyectar múltiples puntos $x$ en profundidad. Pero teniendo 2 cámaras, es posible hallar una intersección entre la línea $OX$ y $O'X$ dando lugar a una triangulación del punto 3D. Estas líneas que se trazan se conocen como epilineas o lineas epipolares.

Para encontrar un punto $x$ de la imagen izquierda en la imagen derecha, solo basta con buscar a lo largo de su línea epipolar; esta regla se conoce como restricción epipolar. Y el plano generado por la triangulación $XOO'$ se le conoce como plano epipolar. $O$ y $O'$ son los centros de cámara, donde se puede apreciar que una proyección de la cámara izquierda se encuentra directamente con una proyección de la cámara derecha, a este punto $e$ se le conoce como epipolo.

Se requieren dos últimos factores, la matriz esencial E que contiene información sobre traslación y rotación describiendo la segunda cámara con relación a la primera por medio de coordenadas globales, y la matriz fundamental F que contiene la misma información que la matriz E, más la información sobre los parámetros intrínsecos de ambas cámaras con el objetivo de poder relacionarlas en coordenadas de pixeles.

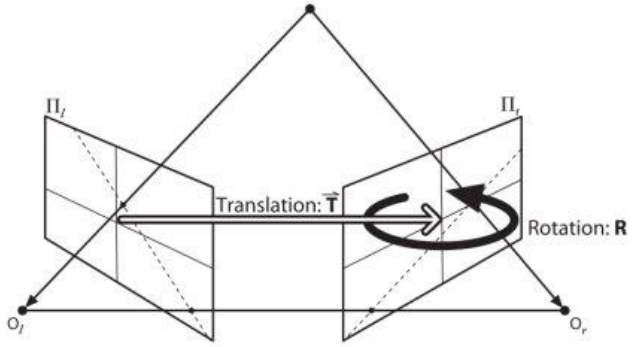


*Figura 6 Información sobre traslación y rotación de las imágenes a analizar. Fuente: Documentación OpenCV*

C. CÁLCULO DE DISPARIDAD

Luego de localizar los puntos similares en las dos imágenes estéreo, se necesita hacer un cálculo de distancias entre las dos imágenes sobre un mismo punto para determinar su profundidad.

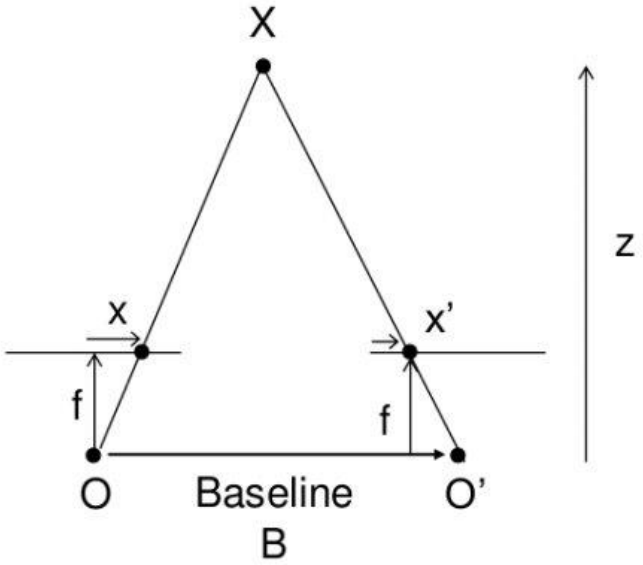


*Figura 7 Representacion de triangulos equivalentes. Fuente: Documentación OpenCV*

El cálculo esta dado por la siguiente ecuación:

*Ecuación 2 Cálculo de disparidad entre dos puntos de correlación*

$$disparity = x - x' = \frac{Bf}{Z}$$

$x$ y $x'$ es la distancia entre los puntos en el plano de la imagen correspondiente al punto de la escena 3D y su centro de cámara. $B$ es la distancia entre las dos cámaras (Que se conoce) y $f$ es la distancia focal de la cámara (Ya conocida). Conociendo estos parámetros se puede concluir que la profundidad de un punto en una escena es inversamente proporcional a la diferencia en la distancia de los puntos de imagen correspondientes y sus centros de cámara.

D. PROYECCION DE NUBE DE PUNTOS

Una nube de puntos es un conjunto de vértices en un sistema de coordenadas tridimensional. Estos vértices se identifican habitualmente como coordenadas X, Y, y Z y son representaciones de la superficie externa de un objeto. Las nubes de puntos se crean habitualmente con un láser escáner tridimensional o por fotogrametría.

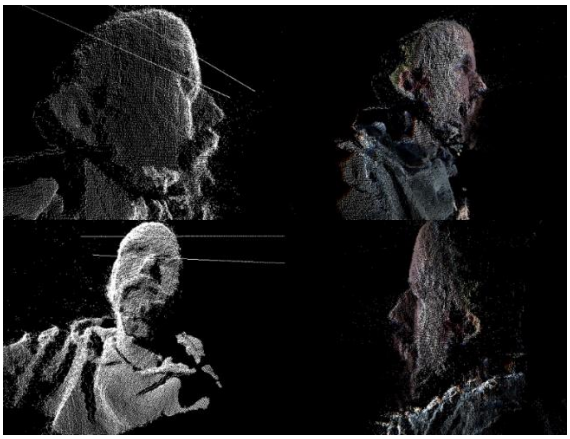

*Figura 8 Proyección de nube de puntos con Kinect. Fuente: Kyle McDonald, Flickr*

Con los procedimientos anteriormente explicados, se poseen todos los parámetros necesarios para reconstruir la imagen en una nube de puntos:

- Posición global (X,Y,Z) de cada punto de correlación
- Posición de pixel (u,v) en la imagen original

Solo es necesario almacenar los datos con un formato adecuado e importarlos a un software de visualización en 3D compatible.

## IV. DESARROLLO CON OPENCV

OpenCV es la librería de computer visión indicada para resolver el problema planteado en este artículo, ya que proporciona al desarrollador un arsenal de herramientas y la posibilidad de trabajar con imágenes estereoscópicas a través del módulo Stereo, además tiene implementación en diferentes lenguajes de programación (Python, C/C++, Matlab, Java)

El proceso se hace mucho mas simple al trabajar con los módulos ya implementados de OpenCV, los cuales cuentan con la compatibilidad de múltiples formatos de imagen y de color.

Antes de explicar brevemente las características utilizadas de la librería, se presenta el siguiente diagrama, el cual especifica cuales son las entradas y salidas esperadas.

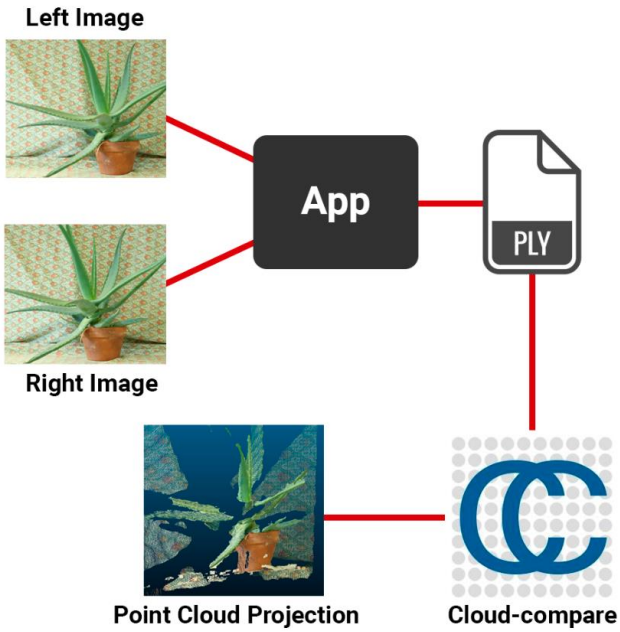


*Figura 9 Diagrama que representa el flujo de procesamiento de las imágenes*

## A. LECTURA DE IMÁGENES

La primera parte del proceso consiste en importar las 2 imágenes estereoscópicas a la aplicación, se realiza una reducción de calidad de la imagen para ser procesada en menor tiempo.

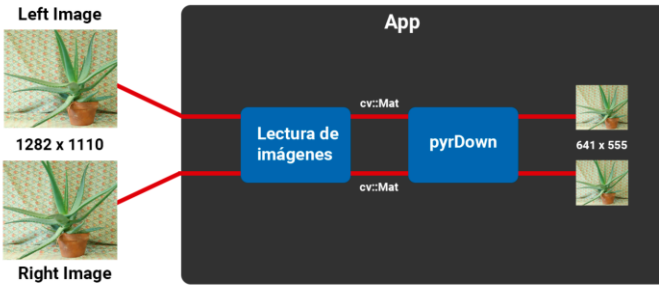


*Figura 10 Lectura de imagenes y reducción de calidad mediante la funcion pyrDown*

*Figura 11 Imágenes estereoscópicas de prueba. Fuente: Opencv Samples*

## B. DEFINICION DE PARAMETROS Y MÉTODO DE PROCESAMIENTO

Se continua con la definición de los parámetros esenciales dependiendo del método de procesamiento seleccionado, en este caso se utilizó el algoritmo SGBM (Semi Global Block Matching) una modificación del algoritmo H. Hirschmuller, es single-pass, de correspondencias entre bloques en lugar de pixeles y con uso de distintas métricas de costos.

Los parámetros definidos para crear el objeto StereoSGBM son:

- Disparidad mínima: El valor mínimo que se debe poseer al calcular disparidades.
- Numero de disparidades: Se calcula restando la disparidad mínima a la máxima. Debe ser divisible entre 16.
- Tamaño de bloque coincidente: El tamaño del bloque de búsqueda de correspondencias, debe ser impar.
- P1 Y P2: controlan la suavidad de la disparidad, entre mas grandes los valores, mas suave es la disparidad, cada uno representa distintos costes de cambio entre los pixeles vecinos. El algoritmo SGBM requiere que P2 > P1.
- Máxima diferencia permitida: es el límite de disparidad de un pixel entre las 2 imágenes.
- Relación de exclusividad: Es un margen de porcentaje que define el mejor costo de una coincidencia encontrada respecto a las otras para determinar que es correcta. Un valor entre 5 y 15 es suficientemente bueno.
- Tamaño de ventana de moteado: Es el tamaño máximo de una región con disparidad uniforme, normalmente se asocia con la reducción de ruido.
- Rango de moteado: es la variación máxima de disparidad dentro de cada componente conectado.

También se establece una matriz Q de transformación de perspectiva, la cual se puede obtener mediante la rectificación de las imágenes, o se pueden aproximar los datos que la componen.

*Ecuación 3 Matriz Q para proyección en nube de puntos. Los parámetros se encuentran explicados en la Ecuación 1*

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix}$$

Todos estos parámetros a excepción de la matriz Q, se cargan en un objeto StereoSGBM que se encargará de realizar los respectivos cálculos sobre las imágenes de entrada.

## C. CÁLCULO DE DISPARIDAD

Todo el procesamiento de correspondencia de imágenes y cálculo de disparidad se lleva a cabo por el objeto StereoSGBM anteriormente declarado. Con solo un llamado a la función de calcular, y la especificación de imágenes de entrada y de salida, se procesa y almacena una imagen de disparidad en escala de grises.

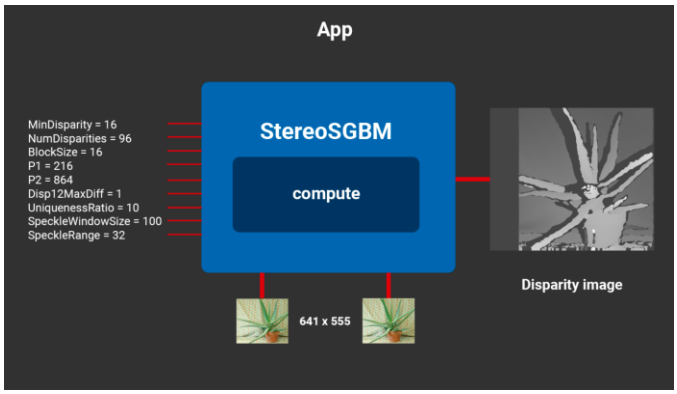


*Figura 12 Cálculo de disparidad mediante SGBM Compute*

Cada disparidad se traduce a un valor de pixel entre 0 y 255, para convertirse en una imagen en formato CV_8U[2].

*Figura 13 Imagen de disparidad obtenida*

## D. PROYECCION DE NUBE DE PUNTOS

Aunque el mapa de disparidad posee la información necesaria para identificar los puntos más cercanos a la cámara, se necesita ubicar cada pixel en un sistema de coordenadas globales para poder proyectarse correctamente en una nube de puntos. OpenCV cuenta con un método para tomar el mapa de disparidad y la matriz de transformación Q y generar los puntos x,y,z de cada pixel.

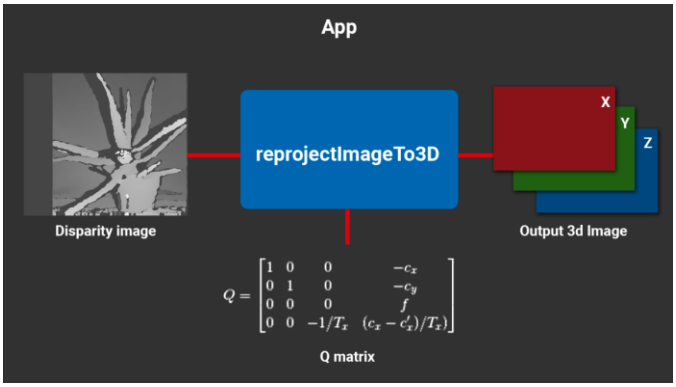


*Figura 14 Cálculo de coordenados globales de pixel*

Los canales de color (R,G,B) pasan a ser canales de coordenadas (X,Y,Z) y cada punto se almacena en su respectiva posición u,v en sus respectivos canales. Finalmente se construyó una función que se encarga de leer la imagen 3D de salida y almacenarla en un archivo ply (Polygon File Format) siguiendo la estructura requerida.

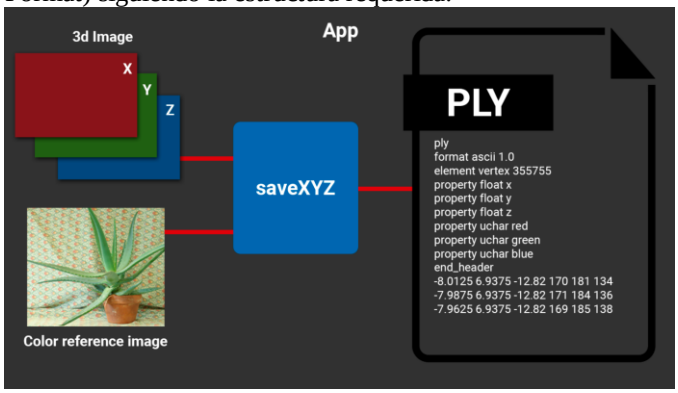


*Figura 15 Archivo final de salida ply, requiere matriz de posiciones y matriz de color para generarse.*

El archivo generado puede leerse en una variedad de programas de visualización 3D, en esta investigación se utilizó el software cloud-compare para proyectar los puntos obtenidos sobre un visor tridimensional.

*Figura 16 Modelo 3D generado por el archivo ply en Cloud-compare.*

Como se puede apreciar en la *Figura 16*, el modelo generado pasa de ser una imagen plana a una colección de puntos en un espacio tridimensional; puede parecer una proyección incompleta, pero esto se debe a que la información del objeto proviene de solo dos imágenes. Si se utilizaran mas cámaras que cubrieran todos los ángulos del objeto, se obtendría una representación completa con todos los vértices.

## V. ACELERANDO EL CODIGO

Hasta ahora, todo lo que se ha implementado se procesa por la CPU, y aunque parece ser eficiente, el procesamiento puede ser aún más rápido si se decide llevar a la computación masivamente paralela. Este enfoque de la computación se encarga de llevar problemas grandes y complejos, a problemas mas pequeños y que se puedan solucionar en paralelo.

La tecnología utilizada en esta investigación para acelerar los métodos ofrecidos por OpenCV, es CUDA, una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema. Su modularidad permite integrarse perfectamente con otras librerias, en este caso OpenCV, ya que esta posee soporte para CUDA y versiones aceleradas de sus métodos de procesamiento originales.

Se seleccionó la versión acelerada del cálculo de correspondencia y disparidad (cuda::compute), la cual trabaja con otro tipo de matrices (cuda::Gpumat). El flujo de procesamiento de la imagen cambia un poco adaptándose a la arquitectura host-device.

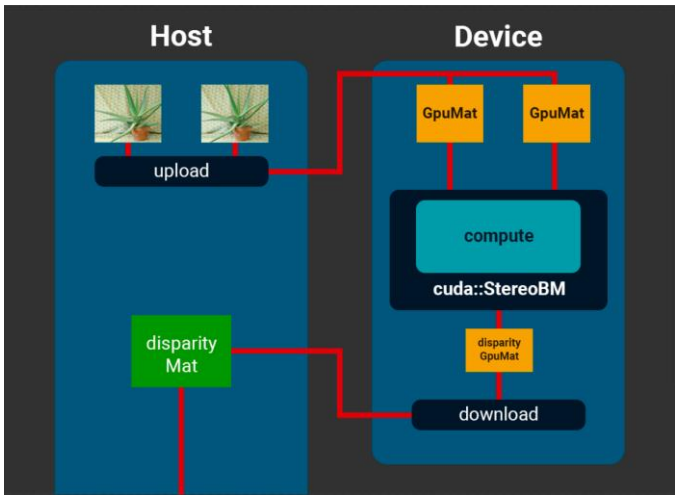


*Figura 17 Flujo de procesamiento en computación masivamente paralela. Las imagenes se cargan desde el host al device y se procesan mediante un kernel alojado en la GPU, al finalizar se descarga el mapa de disparidad calculado del device al host.*

Analizando la *Figura 17* los cambios son pocos, pero significativos, ya que se debe preparar todos los parámetros necesarios desde el host y enviarlos al device para su procesamiento. OpenCV con su módulo cudastereo permite hacer este trabajo de una forma sencilla y con pocas líneas de código, los métodos disponibles actualmente para la correspondencia de imágenes estéreo son limitados, y trabajan con imágenes en escala de grises, por lo que aún no son lo suficientemente precisos, generando un mapa de disparidad como el siguiente.

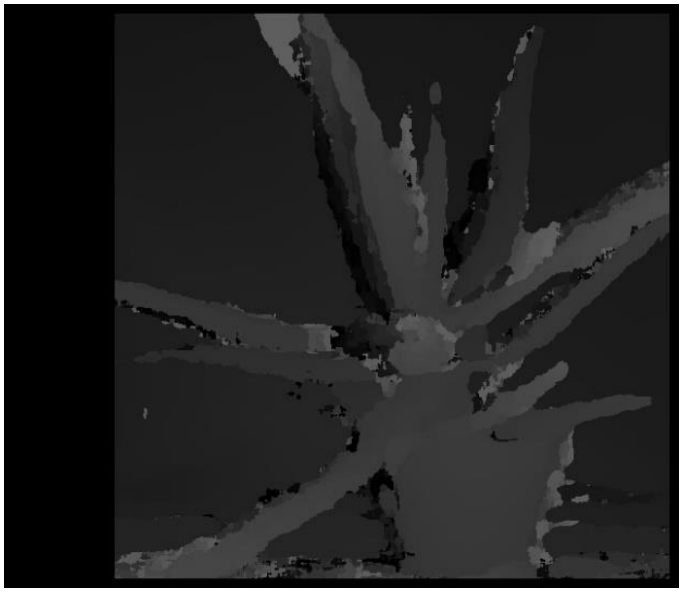

*Figura 18 Mapa de disparidad calculado por la gpu, se pueden apreciar bordes menos definidos y mayor cantidad de ruido.*

Al tener un mapa de disparidad con mayor ruido y menor definición, la proyección de puntos se va a encontrar más dispersa.

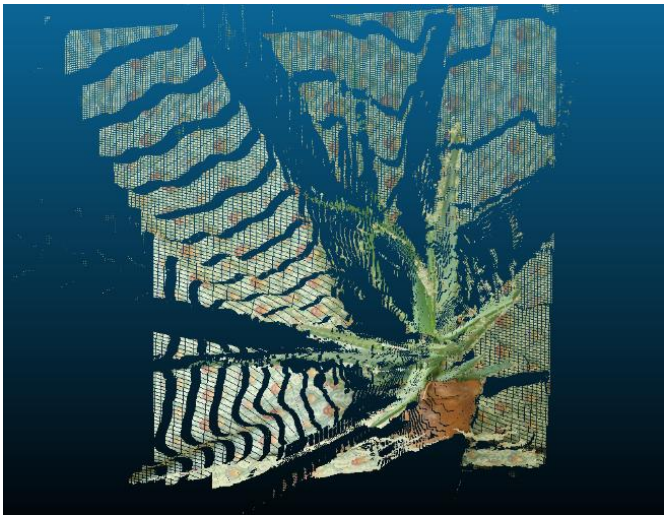

*Figura 19 Proyeccion de nube de puntos basada en un mapa de disparidad calculado por GPU.*

## VI. RESULTADOS

El algoritmo se ejecutó 20 veces por cada método en un computador con las siguientes características:

- Procesador Intel Core i7 6700k 4.0Ghz
- RAM 16GB DDR4 2100 Mhz
- Tarjeta de video Nvidia GTX 970 4GB GDDR5
- Almacenamiento masivo principal SSD 240 GB

Se midió el tiempo de procesamiento de los cálculos de correspondencia y disparidades. Los resultados fueron en promedio:

*Tabla 1 Comparación de tiempos de procesamiento CPU vs GPU*

| Promedio CPU | Promedio GPU |
|---|---|
| 96.398575 ms | 4.0661415 ms |

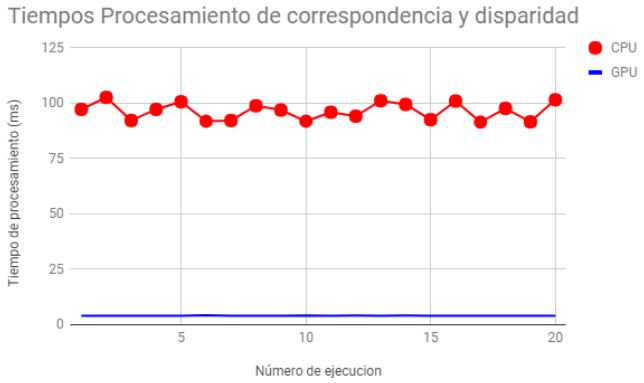


*Gráfico 1 Comparación de tiempos de procesamiento*

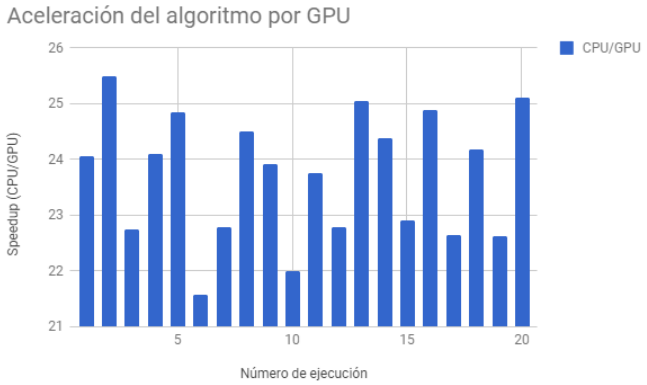


*Gráfico 2 Gráfico de Speedup*

Analizando los gráficos 1 y 2, la versión del código por GPU supera por muchísima ventaja a la versión por CPU. A pesar de que es menos precisa, el tiempo de procesamiento tan bajo permite que se pueda aumentar el número de cámaras logrando procesar un modelo completo antes que el modelo parcialmente construido por la CPU y las dos cámaras.

## VII. CONCLUSIONES

El impacto de la tecnología y la visión por computador en el cine, la arquitectura y los videojuegos ha logrado que las computadoras hagan todo el trabajo mecanizado que anteriormente hacían los seres humanos, facilitando herramientas que permiten que las maquinas vean lo que los usuarios no pueden apreciar con la misma abstracción.

La computación masivamente paralela y la visión por computador son enfoques tecnológicos que van de la mano, siendo un arma muy poderosa para el tratamiento de datos masivos y solución de problemas complejos.

En la presente investigación se logró demostrar la eficiencia de procesamiento de una GPU respecto a los cálculos secuenciales de una CPU, superándolos por mucha diferencia, pero con una leve reducción de calidad. El reto es encontrar el balance perfecto entre velocidad y calidad que asegure un rendimiento óptimo y con los resultados correctos.

## REFERENCIAS

Código desarrollado:

[1]. Luis David Arias Manjarrez. Generación de nube de puntos 3D a través de imágenes estereoscópicas. Disponible: https://github.com/DavidAriasDeveloper/point-cloud

Referencias bibliográficas:

[2]. Clínica Baviera. ¿Qué es la visión estereoscópica? [Online]. Disponible: https://www.clinicabaviera.com/blog/bye-bye-gafasconoce-tus-ojosque-es-la-vision-estereoscopica/

[3]. jderobot. Geometria Epipolar [Online]. Disponible: http://jderoboturjc.blogspot.es/1403282746/geometria-epipolar/

[4]. Arturo Martínez Cruz. ¿Por qué usar lentes para ver en 3D? [Online]. Disponible: http://www.parentesis.com/noticias/ciencias/Por_que_usar_lentes_para_ver_en_3D

[5]. Documentación OpenCV. Depth Map from Stereo Images [Online]. Disponible: https://docs.opencv.org/3.1.0/dd/d53/tutorial_py_depthmap.html

[6]. Documentación OpenCV. Camera Calibration and 3d Reconstruction [Online]. Disponible: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[7]. Wikipedia. Estereoscopía [Online]. Disponible: https://es.wikipedia.org/wiki/Estereoscop%C3%ADa

[8]. Joel Corrado Juárez. Calibrar una cámara con OpenCV[Online]. Disponible: http://tecnicasdevision.blogspot.com.co/2014/05/calibrar-una-camara-con-opencv.html