

# **Reconstrucción de un objeto en 3D a partir de imágenes en 2D**

**Christian Axel Vera Cortés**

**B210737**

Matemáticas para las ciencias de la computación

Maestría en Ciencias en Ingeniería de Cómputo  
Centro de Investigación en Computación, CDMX

November 3, 2021

# Contents

	<b>Page</b>
<b>1 Introducción</b>	<b>2</b>
1.1 Justificación . . . . .	2
1.2 Objetivo General . . . . .	2
1.3 Objetivos Específicos . . . . .	2
<b>2 Estado del Arte</b>	<b>3</b>
<b>3 Fundamentos</b>	<b>4</b>
3.1 Visión estereoscópica por computadora y algoritmo BM . . . . .	4
3.2 Geometría Epipolar . . . . .	5
<b>4 Solución propuesta</b>	<b>6</b>
<b>5 Conclusión</b>	<b>8</b>
5.1 Resultados . . . . .	8
5.2 Conclusiones . . . . .	8
5.3 Trabajo a Futuro . . . . .	9

# 1 Introducción

## 1.1 Justificación

Los modelos virtuales en tres dimensiones tienen aplicaciones en distintos campos: desde la medicina, para el estudio de daño en los órganos del cuerpo a través de un modelo virtual, pasando por la industria de manufactura dónde juegan un papel importante para crear nuevas piezas usando técnicas como el control numérico o la impresión 3D.

No obstante, las herramientas para obtener información del entorno que permita hacer una reconstrucción virtual del mismo, tales con la tecnología LiDAR o las cámaras RGB-D, puede ser muy costoso.

Es por ello que en este proyecto se propuso la obtención de una técnica que permita obtener un modelo en tres dimensiones usando la cámara de un teléfono inteligente, un aparato que está al alcance de la mayoría de personas.

## 1.2 Objetivo General

Reconstruir un objeto en 3D a partir de dos fotografías tomadas desde un teléfono inteligente.

## 1.3 Objetivos Específicos

- Obtener los parámetros de la cámara del teléfono inteligente.
- Desarrollar un programa que calcule la profundidad de la imagen a partir de dos fotografías.
- Desarrollar un programa que reconstruya el objeto a partir de los datos de la imagen y la profundidad encontrada.
- Integrar los programas y mostrar el resultado de la reconstrucción.

## 2 Estado del Arte

Título	Revista	Año	Problema	Método
A multiple-point statistics algorithm for 3D pore space reconstruction from 2D images	Advances in Water Resources	2011	Reconstruir un modelo en 3D de un medio poroso a partir de imágenes en 2D de la sección del mismo	Técnica estocástica para la reconstrucción en 3D
3D model reconstruction based on multiple view image capture	Conferencia ISPACS	2012	Reconstruir un modelo en 3D a partir de imágenes tomadas por una cámara	Proceso SfM, reconstrucción con estéreo multivista y algoritmo iterativo del punto más cercano
Automated 3D Model Reconstruction from Photograph	Conferencia VSM	2014	Reconstruir automáticamente modelos 3D texturizados a partir de un número de fotografías	Proceso SfM
3D reconstruction of small sized objects from a sequence of multifocused images	Journal of Cultural Heritage	2014	Reconstruir un modelo en 3D de objetos pequeños a partir de una secuencia de macro imágenes	Técnica de estéreo multivista pasiva y algoritmo de fusión de imágenes
3D reconstruction of a scene from multiple 2D images	International Journal of Civil Engineering and Technology	2017	Reconstruir un modelo en 3D de una escena a partir de imágenes en 2D	Identificación de los puntos característicos de las imágenes en 2D

# 3 Fundamentos

## 3.1 Visión estereoscópica por computadora y algoritmo BM

Para extraer información sobre la profundidad de un objeto a partir de imágenes en dos dimensiones, se utiliza la visión estereoscópica por computadora. El principio de funcionamiento de ésta es semejante al de la visión biológica usando dos pares de ojos supliendo la función de los ojos con cámaras fotográficas.

La técnica canónica para la visión estereoscópica por computadora es el algoritmo BM (*block matching* o emparejamiento de bloques). Éste toma una pareja de imágenes que hayan tomado un mismo objeto, pero desde diferente ángulo (a la izquierda y a la derecha), y busca una correspondencia entre los píxeles: esto es, que estos píxeles tomen información de la misma zona del objeto.

Para lograr su cometido el algoritmo BM sigue los siguientes pasos:

1. Hacer un proceso de pre-filtrado para normalizar el brillo de la imagen y mejorar la textura.
2. Búsqueda de correspondencia a lo largo de las líneas epipolares horizontales usando una ventana SAD.
3. Pos-filtrado para eliminar falsas correspondencias.

En el primer paso las imágenes son normalizadas para reducir diferencias de iluminación y mejorar la textura. Para ello se hace correr una ventana sobre la imagen. El píxel central de dicha ventana, llamado  $I_c$ , es reemplazado por  $\min(\max(I_c - \bar{I}, -I_{cap}), I_{cap})$  donde  $\bar{I}$  es el valor promedio de la ventana e  $I_{cap}$  es un límite (por default se elige 30).

A continuación, se calcula la correspondencia usando la ventana SAD. Para ello el algoritmo utiliza las líneas epipolares para encontrar los puntos de encuentro de ambas imágenes. En la siguiente sección se explicará qué son las líneas epipolares.

Para llevar a cabo este proceso el algoritmo es necesario establecer de antemano las disparidades máximas y mínimas.

Finalmente, es necesario hacer pos-procesamiento para eliminar falsos positivos. Esto se hace estableciendo un umbral máximo a partir del cual serán rechazados los emparejamientos.

Una variante de BM es el algoritmo SGMB (*semi-global block matching* o emparejamiento de bloques semi-global). Éste introduce algunas ideas para mejorar los resultados de BM, pero resulta computacionalmente más costoso. Entre esas nuevas ideas destaca la introducción de restricciones

de consistencia para no depender únicamente de las líneas epipolares para hacer el emparejamiento. Dichas restricciones se hacen utilizando métricas de Birchfield–Tomasi.

## 3.2 Geometría Epipolar

La geometría epipolar es aquella que combina los modelos geométricos de dos cámaras estenopeicas. Cada cámara tiene su propio centro de proyección:  $O_l$  y  $O_r$ , y sus propios planos proyectivos:  $\Pi_l$  y  $\Pi_r$ . El punto  $P$  del mundo físico tiene una proyección en cada uno de los planos proyectivos a las cuales llamaremos  $p_r$  y  $p_l$ .

Un epipolo  $e_l$  sobre un plano  $\Pi_l$  está definido como la imagen del centro de proyección de la otra cámara  $O_r$ . El plano en el espacio formado por el punto  $P$  y los dos epipolos se llama plano epipolar y las líneas  $p_l e_l$  y  $p_r e_r$  son las líneas epipolares.

Para una mejor comprensión de lo anterior véase la figura 4.1.

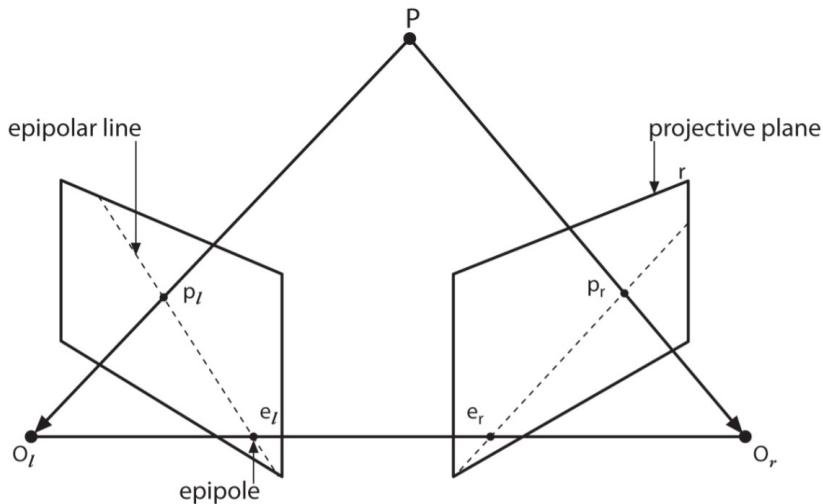


Figure 3.1: Plano epipolar.

La geometría epipolar tiene las siguientes características:

- Dada una característica en una imagen, su vista correspondiente en la otra imagen debe encontrarse a lo largo de la línea epipolar correspondiente. A esto se le conoce como restricción epipolar.
- La restricción epipolar significa que la búsqueda de características correspondientes entre dos imágenes bidimensionales se convierte en una búsqueda unidimensional a lo largo de la línea epipolar.

Estas características son aprovechadas por el algoritmo BM visto en la sección anterior para calcular la correspondencia y así obtener la profundidad de la imagen mediante una triangulación.

## 4 Solución propuesta

Para hacer la reconstrucción en 3D de un objeto a partir de imágenes en dos dimensiones se utilizó el algoritmo SGBM que se encuentra implementado en la librería OpenCV.

El primer paso para la implementación de este algoritmo es obtener los parámetros de la cámara a utilizar. Para ello es necesario tomar varias fotos, desde distintos ángulos, a un patrón de tablero de ajedrez. Como referencia véase la figura 4.1.

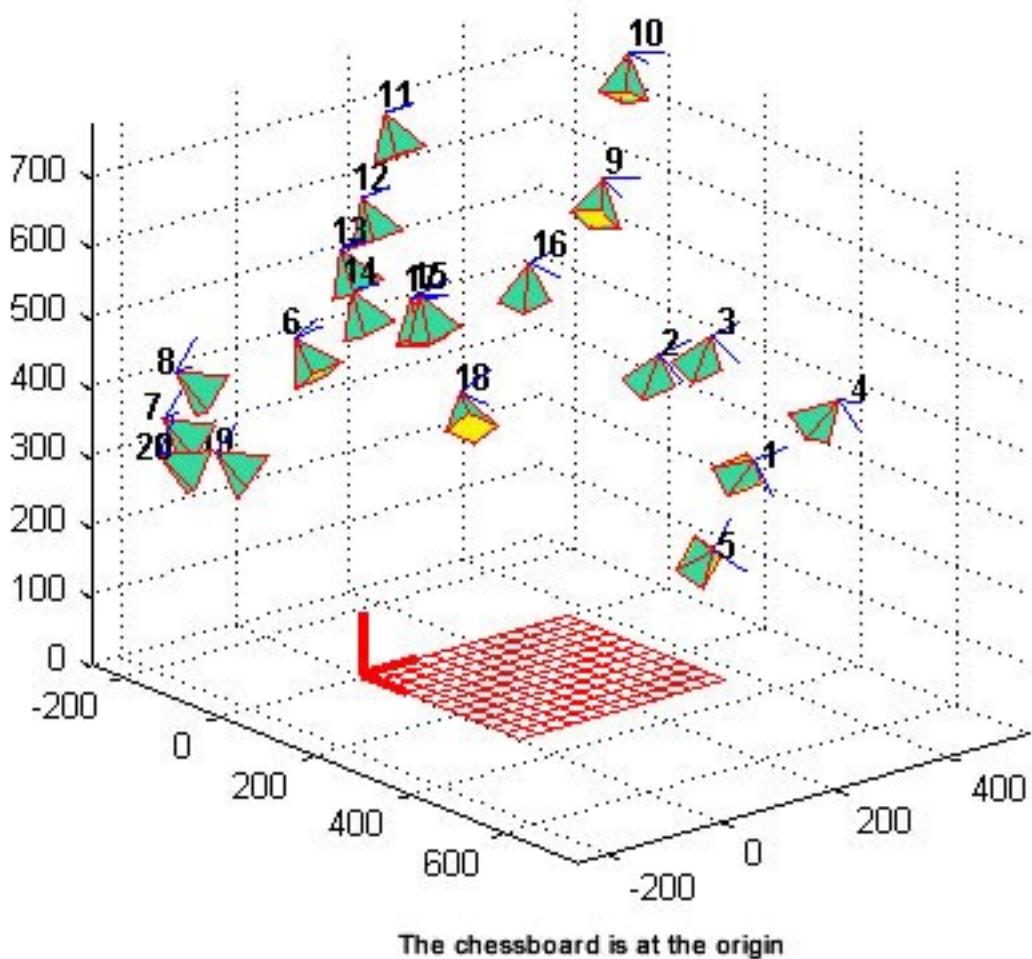


Figure 4.1: Cómo tomar fotografías del patrón.

Una vez obtenidas varias imágenes se usó la función `cv2.findChessboardCorners` para identificar los puntos del patrón de tablero de ajedrez. Éstos se guardaron en un arreglo y se le pasaron a la función

`cv2.calibrateCamera` que devolvió los parámetros necesarios para la aplicación del algoritmo. Éstos se guardaron en una carpeta especial para no tener que hacer la calibración cada vez que se corre el programa.

A continuación, se tomaron fotografías de los objetos de interés. Se procuró que estas estuvieran a la misma altura, pero tomadas desde distintos ángulos. Una más a la izquierda y otra más a la derecha del objeto.

Posteriormente se implementó el algoritmo en Python. El primer paso fue mandar a llamar las variables previamente guardadas. A continuación, se calculó una matriz de optimización basada en el parámetro de escalamiento. Con esta matriz se reduce la distorsión de la imagen utilizando la función `cv2.undistort`.

Después se implementó una función de submuestreo y se aplicó a las imágenes. Esto con el fin de mejorar la velocidad de muestreo. La desventaja fue la pérdida de información y la reducción de la precisión en el cálculo de la profundidad.

Tras este preprocesamiento se aplicó el algoritmo de SGBM. Para ello se usaron las funciones `cv2.StereoSGBM_create` la cuál crea el objeto que se usará para la implementación de SGBM. Es importante señalar que el tamaño de la ventana SAD, así como los umbrales máximos y mínimos de disparidad son establecidos manualmente. Estas variables se obtienen por ensayo y error y los valores óptimos pueden cambiar, no solamente según la cámara utilizada, sino también dependen de las propias fotografías.

En este punto del algoritmo se obtuvo el mapa de disparidad. A continuación, se obtuvo un arreglo con los colores de la imagen. Para ello es necesario obtener los valores de alto y ancho de la imagen submuestreada así como la matriz de transformación que es la responsable de proyectar la profundidad y los colores en un espacio 3D. La matriz de transformación implementada es la siguiente:

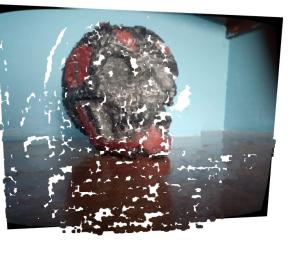
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & \frac{1}{f'} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \frac{z}{f'} \end{bmatrix}$$

Finalmente, se hace la conversión a un archivo de nube de puntos. La rutina encargada de hacer dicha conversión se puede encontrar en la documentación de OpenCV.

# 5 Conclusión

## 5.1 Resultados

En la siguiente tabla se muestra las imágenes, tanto a la izquierda como a la derecha, de tres objetos así como el resultado de la aplicación del programa:

Imagen Izquierda	Imagen Derecha	Reconstrucción en 3D
		
		
		

## 5.2 Conclusiones

Como se puede apreciar en la tabla de la sección anterior, la reconstrucción presenta deficiencias. La principal es la ausencia de algunos puntos que no fueron calculados, la cual se hace más patente al rotar

el modelo 3D (algo que no se aprecia adecuadamente en la imagen). Entre las posibles razones de los defectos de la reconstrucción se encuentran:

- Una calibración insuficiente de la cámara. Lo cual se puede solucionar con un mayor número de muestras.
- Un inadecuado ajuste de los parámetros del algoritmo.

En la siguiente sección se discutirán las posibles mejoras para un trabajo a futuro.

### 5.3 Trabajo a Futuro

Como se mencionó en el capítulo 5, la ventana SAD y los valores de umbral máximos y mínimos de disparidad son establecidos directamente por el programador y pueden variar según la cámara utilizada e incluso según la propia imagen. Para facilitar el establecimiento de dichos parámetros se sugiere añadir un sistema de ajuste en tiempo real que permita ver de forma inmediata el efecto de dicho ajuste sobre el modelo reconstruido.

Finalmente, se sugiere ampliar el trabajo para poder reconstruir modelos partiendo de más de dos imágenes para obtener un panorama más completo de la profundidad del objeto analizado.

# Bibliography

- [1] Kaehler, A. and Bradski, G. (2017) *Learning OpenCV 3*, O'Reilly Media, Inc, pp. 709-710,764-772
- [2] OpenCV-Python Tutorials (2017) *Depth Map from Stereo Images*, Consultado: 31/10/2021, Sitio web: [https://opencv24-python-tutorials.readthedocs.io/en/latest/py\\_tutorials/py\\_calib3d/py\\_depthmap/py\\_depthmap.html](https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html),