

Laboratorio de Métodos Numéricos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico 1

Photometric Stereo

Integrante	LU	Correo electrónico
Hosen, Federico	825/12	federico.hosen@gmail.com
Palladino, Julián Alberto	336/13	julianpalladino@hotmail.com
Rajngewerc, Guido	379/12	guido.raj@gmail.com
Silvani, Damián Emiliano	658/06	dsilvani@gmail.com

Utilización de métodos de factorización de matrices para la estimación de profundidades a partir de imágenes estáticas con distintas iluminaciones. Análisis temporal y de resultados variando distintos parámetros, propuestas y soluciones sobre algunos de los problemas encontrados y posibles trabajos futuros.

Fotometría Estéreo Factorización Cholesky Factorización LU

Índice

1. Introducción	3
2. Desarrollo	3
2.1. Flujo del trabajo	3
2.2. Calibración	3
2.2.1. Color a escala de grises	3
2.2.2. Estimación de dirección de luz y detección de reflejo especular	4
2.2.3. Cálculo de la <i>bounding box</i> , centro y radio	4
2.3. Leer y procesar las imágenes	5
2.4. Obtener I_0 y ρ	5
2.4.1. Obtenerlo en cada iteración	5
2.4.2. Método alternativo: O de un sólo píxel	6
2.4.3. Conclusión	6
2.5. Reconstrucción 3D	6
2.5.1. Pseudocódigo	6
2.5.2. Matriz de ecuaciones M	6
2.5.3. Resolución del sistema	7
2.6. Sobre la factorización Cholesky de la matriz de la ecuación 15	8
3. Experimentación	9
3.1. Comparación entre los distintos métodos de obtención de dirección de luz en la calibración	9
3.1.1. Obtención de la dirección	9
3.1.2. Impacto de los métodos de combinación de rojo, verde y azul en la calibración	10
3.2. Diferencia temporal los diferentes métodos numéricos aplicados a la obtención de normales	12
3.2.1. Análisis de tiempos	12
3.3. Impacto de la elección de las 3 direcciones de iluminación en el cálculo de las normales	13
3.3.1. Introducción	14
3.3.2. ¿Cómo incide S sobre el cálculo de n ?	15
3.3.3. Evaluando resultados coherentes según S	15
3.3.4. Confirmación empírica de nuestra hipótesis	15
3.3.5. Comprobación empírica usando la esfera	16
3.3.6. Comprobación empírica usando el buda	17
3.4. Impacto de la elección de las 3 direcciones de iluminación en el armado del modelo 3D	18
3.4.1. Analizando el modelo 3D del caballo	18
3.4.2. Analizando el modelo 3D del buda	19
3.4.3. Conclusión	21
3.5. Comparación de tiempos: Eliminación Gaussiana vs. Cholesky	21
3.6. Especificaciones técnicas relativas a la experimentación	22
3.6.1. Medición de tiempos	22
3.6.2. Especificación técnica de la PC utilizada	22
4. Conclusión	23
4.1. Trabajo futuro	23
4.1.1. Implementación con paralelismo	23
4.1.2. Conclusiones finales	24
5. Apéndices	26
5.1. Planos normales variando las direcciones de luces	26

1. Introducción

La fotometría estéreo es una técnica de reproducción tridimensional de objetos a partir de un conjunto de fotografías. El método permite calcular los vectores normales a la superficie de un objeto en cada píxel de la imagen, observando a éste bajo distintas condiciones de luz[3].

En este trabajo práctico, se implementó el procedimiento de calibración y reconstrucción del mapa de normales y de profundidad para un conjunto de imágenes de un objeto. Los métodos numéricos empleados para resolver los sistemas de ecuaciones que la técnica plantea fueron *eliminación Gaussiana*, *factorización LU* y *factorización de Cholesky*.

2. Desarrollo

2.1. Flujo del trabajo

A medida que avanzamos pensando los distintos problemas a resolver terminamos encontrando el siguiente flujo sobre el cual iteramos hasta obtener el código aquí descripto.

1. Calibración
2. Leer y procesar las imágenes
3. Obtener I_0 y ρ
4. Obtener vector normal v a cada pixel
5. Construir matriz M tal que $M \cdot M^t = A$
6. Definir el sistema de ecuaciones $M^t \cdot M \cdot z = M^t \cdot v$
7. Resolver el sistema $A \cdot z = b$

2.2. Calibración

Para el proceso de calibración se implementó un programa `calibrate` que toma como parámetros una ruta a un conjunto de imágenes RGB de formato `ppm`. Se espera que las imágenes representen una esfera de material no especular, con una fuente de luz con distintas direcciones de luz. Además, una de las imágenes debe ser una máscara de la esfera.

En la calibración se estima la dirección de la fuente de luz para cada imagen. Para lograr esto, primero se busca el reflejo especular de la esfera. Dado que se asume que la esfera es de un material con superficie Lambertiana, el brillo sólo depende de la orientación de ésta con respecto a la fuente de la luz.

2.2.1. Color a escala de grises

La primera dificultad que se nos presentó fue la de trabajar con imágenes RGB. Lo que necesitamos tanto para la calibración como para el cálculo de las normales es interpretar la intensidad de luz de cada píxel. Primero se decidió tomar el promedio del valor de cada canal RGB. Como alternativa, se calculó el *luma*, un promedio ponderado entre los 3 valores, donde se tiene en cuenta la manera en la que el ojo humano percibe los colores[2]. Sean R, G, B los valores rojo, verde y azul respectivamente, L es el luma y se define como:

$$L = 0,2126 \times R + 0,7152 \times G + 0,0722 \times B$$

2.2.2. Estimación de dirección de luz y detección de reflejo especular

Siendo la superficie Lambertiana, la cantidad de luz reflejada en una localidad de la superficie es proporcional al coseno del ángulo formado por la normal en ese punto y la fuente de luz incidente. Si el brillo en un punto de la imagen es máximo, podemos asumir que el ángulo allí es muy pequeño (porque $\cos(x) \rightarrow 1$ si $x \rightarrow 0$), con lo cual, la normal es una buena estimación del vector de dirección de luz.

Por otro lado, como el objeto usado en la calibración es una esfera, la normal de cualquier punto de la superficie es conocida, y está dada por la ecuación de la esfera:

$$\begin{aligned}x^2 + y^2 + z^2 &= r^2 \\ \Rightarrow |z| &= \sqrt{r^2 - x^2 - y^2}\end{aligned}$$

Entonces, la dirección de luz incidente se puede estimar tomando el opuesto a la normal en el punto de mayor brillo.

Nuestro primer intento para encontrar el punto de mayor brillo en la esfera fue buscar el máximo valor en la matriz recorriéndola linealmente. Luego notamos que podría suceder que haya algún píxel con un valor máximo, pero que no represente con exactitud el verdadero área de incidencia máxima de la fuente de luz (ver figura 1). Teniendo en cuenta este problema derivamos una solución: en lugar de obtener el píxel individual de máximo brillo buscamos una ventana donde el brillo sea mayor al resto de la imagen.

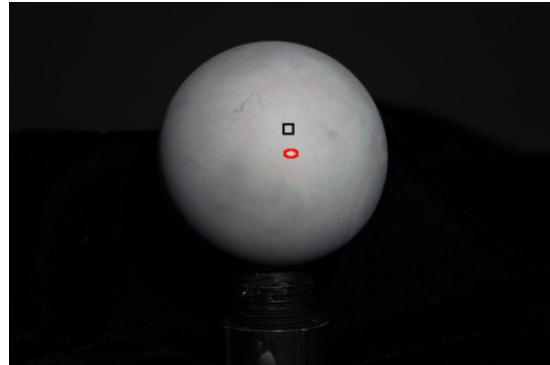


Figura 1: En rojo se detalla el píxel más brillante de la imagen. En negro la ventana de 10x10 con más brillo

2.2.3. Cálculo de la *bounding box*, centro y radio

Para calcular la bounding box de la esfera se utilizó la máscara, porque la imagen tiene sólo dos colores y resulta fácil buscar las siguientes variables:

- **T** = Primer fila que con un píxel blanco desde arriba
- **B** = Primer fila que con un píxel blanco desde abajo
- **L** = Primer columna que con un píxel blanco desde la izquierda
- **R** = Primer columna que con un píxel blanco desde la derecha

Y a partir de allí encontrar:

- **Radio** = $\frac{B-T}{2}$
- **Fila del centro** = $\frac{B-T}{2} + T$
- **Columna del centro** = $\frac{R-L}{2} + L$

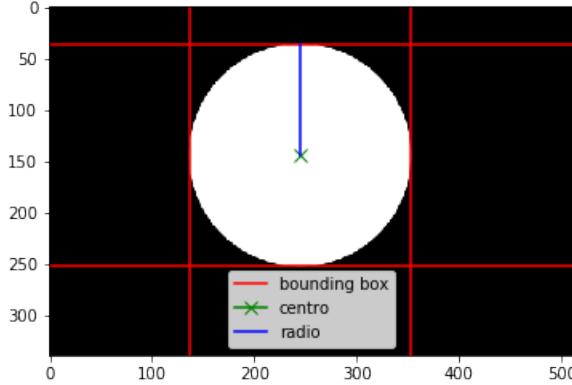


Figura 2: Obtención del centro y radio a partir del cuadrado que contiene al círculo

En la figura 2 puede verse una representación gráfica de este proceso.

Luego, la normal $n = (n_x, n_y, n_z)$ está dada por:

$$\begin{aligned} n_x &= p_x - c_x \\ n_y &= p_y - c_y \\ n_z &= \sqrt{r^2 - p_x^2 - p_y^2} \end{aligned}$$

y por lo tanto, $l = -n$ es el vector de dirección de luz.

2.3. Leer y procesar las imágenes

Habiendo obtenido la calibración, el paso siguiente constaba en leer y hacer el preprocesamiento necesario de las imágenes para trabajar con ellas. Desarrollando esta parte del trabajo es que surge la necesidad de definir una estructura propia que modele correctamente lo que queríamos lograr. Entre las opciones viables consideramos como la mejor el definir nuestra propia clase de *Matrix*. La implementación consta de un template que nos permite versatilidad a la hora de definir los tipos de los elementos y por dentro se implementa como `vector<vector<T>>`. Al leer las imágenes con los archivos brindados por la cátedra en `ppmloader.cpp` utilizamos la misma idea que la definida en 2.2.1 para pasar de imágenes RGB a escalas de grises. Con esta información pudimos cargar las imágenes a estructuras `Matrix<uchar>`

2.4. Obtener I_0 y ρ

Una parte clave del cálculo de las normales es lograr calcular correctamente la constante $I_0\rho$. Para lograrlo, surgieron dos métodos posibles:

2.4.1. Obtenerlo en cada iteración

La primer idea consiste en calcular la constante antes mencionada cada vez que resolvemos el sistema de ecuaciones $S.m = I$. Una vez resuelto el sistema y obtenido el vector columna m , es posible calcular su norma 2 y de esa manera conseguir $I_0\rho$.

El mayor beneficio de este método es la sencillez. Implica simplemente un pequeño cálculo constante cada vez que se resuelve el sistema de ecuaciones $S.m = I$.

Por otro lado, la desventaja es que el valor $I_0\rho$ no será estrictamente constante en toda la imagen, sino que será sensible a errores de cálculos de punto flotante. Por lo tanto, en algunos píxeles de “buena calidad” conseguiremos un $I_0\rho$ más correcto, mientras que en otros píxeles, de “mala calidad”, tendremos más margen de error.

La definición de buena o mala calidad de un píxel nos parece sensato darla según qué tan oscuro sea su color, o sea, qué tan cercanos al cero van a dar su vector de intensidades I .

Por estos posibles problemas, emerge otro método posible.

2.4.2. Método alternativo: O de un sólo píxel

Como $I_0\rho$ es constante para toda la imagen, otra idea consiste en calcularlo **una sola vez**, y luego usar ese valor para realizar los cálculos en el resto de la imagen.

El beneficio de este método es que podemos elegir un '*buen*' píxel para calcular $I_0\rho$, y asegurarnos que el valor obtenido sea lo más cercano posible a lo que buscamos.

La desventaja de esta manera es la complejidad que conlleva: ¿Cómo elegimos un buen píxel? Esto incluye realizar análisis teóricos, complementado con una experimentación para poder corroborar la teoría.

Además, con este método la imagen entera depende de un buen algoritmo de elección de píxel, ya que el $I_0\rho$ es utilizado para calcular **todas** las normales. Mientras que en el anterior método si surgía algún error al calcular el $I_0\rho$, solamente influía en un píxel.

2.4.3. Conclusión

Lamentablemente por falta de tiempos (y como esta parte del trabajo no es prioridad), no se llegó a realizar las comparaciones y experimentaciones previamente planteadas. En lugar de ello, el método que finalmente fue implementado fue el primero explicado, y, como se verá en los tests más adelante, los resultados obtenidos fueron suficientemente similares a los de la cátedra como para que no se vuelva un problema a resolver.

2.5. Reconstrucción 3D

En esta sección tratamos de desarrollar los fundamentos teóricos y las técnicas prácticas utilizadas para obtención de los valores de profundidad para los píxeles de la imagen.

2.5.1. Pseudocódigo

Resultado: Matriz N de normales para cada píxel
 construir matriz S a partir de s_1, s_2, s_3 ;
 hacer factorización LU de S ;
para cada píxel (x, y) **hacer**
 | armar vector I de los valores en (x, y) ;
 | resolver sistema $Lx = I$ con forward substitution;
 | resolver sistema $Um = x$ con backward substitution;
 | $N_{x,y} = \frac{m}{\|m\|}$;
fin

Algoritmo 1: Cálculo de normales

Resultado: vector z de profundidades para cada píxel
 construir matriz M y vector v a partir de las normales N ;
 $A = M^t M$;
 $b = M^t v$;
 resolver sistema $Az = b$ con Cholesky (o EG);

Algoritmo 2: Estimación de la profundidad

2.5.2. Matriz de ecuaciones M

Para cada píxel (x, y) , se puede inferir su profundidad $z^{x,y}$ a partir de las ecuaciones 11 y 12, que relaciona la normal en el punto con los valores de profundidad de los píxeles ubicados debajo y a la derecha. Como fue mencionado anteriormente por cada píxel en la posición (x, y) hay dos ecuaciones que relacionan las normales con las profundidades:

$$\begin{pmatrix} n_z^{0,0} & -n_z^{0,0} & & & & & & \\ n_z^{1,0} & & -n_z^{1,0} & & & & & \\ & n_z^{2,0} & & -n_z^{2,0} & & & & \\ & & n_z^{3,0} & & -n_z^{3,0} & & & \\ & & & n_z^{4,0} & & -n_z^{4,0} & \dots & \\ \vdots & \vdots \\ n_z^{0,0} & & & & & \dots & -n_z^{0,1} & \\ & n_z^{1,0} & & & & \dots & & -n_z^{1,1} \\ \vdots & \vdots \end{pmatrix} = \begin{pmatrix} z^{0,0} \\ z^{1,0} \\ z^{2,0} \\ z^{3,0} \\ z^{4,0} \\ \vdots \\ z^{0,1} \\ z^{1,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} n_x^{0,0} \\ n_x^{1,0} \\ n_x^{2,0} \\ n_x^{3,0} \\ n_x^{4,0} \\ \vdots \\ n_y^{0,1} \\ n_y^{1,1} \\ \vdots \end{pmatrix}$$

Figura 3: Matriz de ecuaciones M

- $n_x = n_z z^{x,y} - n_z z^{x+1,y}$ (11)
- $n_y = n_z z^{x,y} - n_z z^{x,y+1}$ (12)

En primer lugar, no vamos a trabajar con todos los píxeles, si no sólo con aquellos que estén dentro de la máscara y además cuya componente z de su normal sea distinta de 0. Descartamos los píxeles fuera de la máscara porque no tiene sentido estimar la profundidad de algo que no es el objeto que interesa, y además si se incluyen píxeles cuya normal en z se anule, se corre el riesgo de que la matriz tenga filas nulas, lo cual ocasionaría que $M^t M$ no sea s.d.p. impidiendo resolver el sistema usando la factorización de Cholesky. Llamaremos N entonces a los píxeles que reúnen las dos condiciones mencionadas, es decir, las que están en la máscara y además la componente z de su normal no se anula.

La idea es entonces armar una matriz M para representar de forma matricial las ecuaciones relacionan la profundidad de cada píxel con las normales de los píxeles aledaños. En las primeras N filas de M estarán las ecuaciones que relacionan cada píxel con el de la derecha, y en las siguientes N filas estarán las ecuaciones que relacionan cada píxel con el que está debajo.

En la figura 3 se puede observar la forma que tiene la matriz M en el sistema $Mx = v$, con las $2N$ ecuaciones y N incógnitas, o sea, $M \in R^{2N \times N}$. Sobre la forma de M , es importante recordar que no se trabaja con todos los píxeles, si no con aquellos pertenecientes a la máscara y que además tienen la componente z normal no nula. Dicho esto, las primeras N filas de M forman dos bandas, y en las últimas N filas los valores pueden estar más *dispersos*, pues la cantidad de píxeles válidos que hay entre un píxel y el píxel debajo correspondiente no siempre es la misma, depende principalmente de como sea la máscara de la imagen, entonces en la parte inferior de la matriz pueden haber varias bandas.

Con respecto a los píxeles cuyos vecinos (derecho o inferior) no eran válidos, sus coeficientes en la matriz aparecen con 0. Es decir, supongamos que el píxel (x, y) no tiene vecino derecho válido, entonces $-n_z^{x,y}$ aparecerá con valor 0 en la matriz M .

Dado que la matriz es de gran tamaño, al momento de implementar las funciones que resolvían el sistema, nos dimos cuenta que la estructura de representación que veníamos usando no nos iba a servir para instanciar la matriz en memoria. De todas formas, pudimos observar que la matriz no sólo es esparsa, si no que por la manera en la que *ordenamos* las ecuaciones en la matriz, los coeficientes forman siempre diagonales. Teniendo en cuenta esto último, se implementó una nueva clase **BandMatrix**, que sólo guarda los valores de las diagonales. De esta forma guardamos sólo los valores no nulos, y tenemos acceso en tiempo constante a una posición cualquiera de la matriz.

2.5.3. Resolución del sistema

Para resolver el sistema de ecuaciones $Mz = v$, se recurrió al método de ecuaciones normales, multiplicando M^t por izquierda en ambas partes de la ecuación y resolviendo el sistema $Az = b$, con $A = M^t M$ y $b = M^t v$ mediante factorización Cholesky y eliminación gaussiana.

2.6. Sobre la factorización Cholesky de la matriz de la ecuación 15

En la ecuación 15 tenemos

$$Az = b$$

y queremos resolver este sistema primero descomponiendo la matriz A con la factorización Cholesky de manera que

$$A = LL^t$$

donde L es una matriz triangular inferior, con los elementos de la diagonal mayores a cero. Recordamos la propiedad que dice:

Propiedad 2.1. *Una matriz tiene factorización de Cholesky si y sólo si es simétrica definida positiva.*

Por lo tanto, para probar que se puede factorizar con Cholesky en A , **basta probar que la matriz A es simétrica definida positiva**. Para lograrlo, demostraremos primero ciertos lemas, que usaremos en nuestra demostración final.

Lema 2.1. *Sea $v \in \mathbb{R}^n$, $v^t v = (\|v\|_2)^2$*

Demostración. Por definición de norma 2 vale que:

$$\|v\|_2 = \sqrt{v_1^2 + v_2^2}$$

$$\text{Entonces, } \|v\|_2 = \sqrt{v_1^2 + v_2^2} \iff (\|v\|_2)^2 = v_1^2 + v_2^2 = v^t v$$

□

Lema 2.2. *Sea $A \in \mathbb{R}^{m \times n}$ es rango máximo, $Mt = A^t A$ es definida positiva.*

Demostración. Demostraremos por absurdo. Supongamos que no es definida positiva. Entonces $\exists x \neq 0$, tal que $x^t Mx \leq 0$

Veamos también que, por lema 2.1:

$$x^t A^t Ax = (Ax)^t (Ax) = (\|Ax\|_2)^2$$

Por lo tanto, podemos decir:

$$\exists x \neq 0, \text{ tal que } (\|Ax\|_2)^2 \leq 0$$

Como la norma es positiva:

$$\exists x \neq 0, \text{ tal que } (\|Ax\|_2)^2 = 0$$

Y, al ser positiva, podemos pasar la raíz y queda:

$$\exists x \neq 0, \text{ tal que } \|Ax\|_2 = 0$$

Por propiedades de normas vectoriales, sabemos que:

$$\|v\|_2 = 0 \iff v = \bar{0}$$

Por lo tanto, concluimos:

$$\exists x \neq 0, \text{ tal que } (\|Ax\|_2) \leq 0 \iff \exists x \neq 0, \text{ tal que } Ax = \bar{0} \iff \exists x \neq 0, \text{ tal que } x \in Nu(A)$$

Pero habíamos dicho que A es de rango máximo, por lo tanto $Nu(A) = 0$. Entonces $x = \bar{0}$. Absurdo! Habíamos dicho que $x \neq 0$.

El absurdo vino de suponer que M no es definida positiva. Por lo tanto, queda demostrado que sí lo es.

□

Lema 2.3. *Sea $A \in \mathbb{R}^{m \times n}$, $M = A^t A$ es simétrica*

Demostración. Queremos ver que $M^t = M$.

$$M^t = (A^t A)^t = A^t \cdot (A^t)^t = A^t \cdot A = M$$

□

Como último paso, si sabemos que la M que armamos en el algoritmo es de rango máximo, podemos decir que, gracias a los lemas que hemos demostrado, vale que $A = M^t M$ es simétrica definida positiva. De esa forma, gracias a la propiedad antes mencionada, al ser A s.d.p. se puede realizar la factorización de Cholesky sobre ella.

Recordemos cómo habíamos armado la M (explicada previamente), y pensemos cómo tendría que ser M para que no quede de rango máximo.

Tenemos dos ecuaciones por incógnita, una relacionando $Z_{x,y}$ y $Z_{x,y+1}$, otra relacionando $Z_{x,y}$ y $Z_{x+1,y}$. Por lo tanto, cada columna tiene como mucho cuatro elementos distintos de cero, excepto por la primera, que tiene dos.

De esta manera, para que no quede de rango máximo, hay dos opciones: Que haya una columna de ceros y/o que una columna sea combinación lineal de las otras.

Para que haya una columna de ceros, es necesario que hayan cuatro n_z (elementos de la tercera coordenada de alguna normal), pertenecientes a cuatro píxeles distintos, que den cero. Pero, como se habló previamente, no incluimos en nuestro sistema de ecuaciones ninguna tercera coordenada de una normal. Esto se hizo aprovechando la máscara y revisando que ninguna normal de, por casualidad, con tercer coordenada igual a cero.

Por lo tanto, para que M no sea de rango máximo, tiene que ocurrir que una columna sea combinación lineal de las otras.

Luego de consultarla en el laboratorio (más específicamente, Néstor), concluimos que la probabilidad de caer en este caso es despreciable. Además, empíricamente nunca nos ocurrió que la matriz $M^t M$ no quede s.d.p., por lo tanto confirmamos que es altamente improbable que M quede de rango máximo. De esta manera concluimos que A puede ser factorizada utilizando Cholesky.

3. Experimentación

3.1. Comparación entre los distintos métodos de obtención de dirección de luz en la calibración

3.1.1. Obtención de la dirección

Como se explicó anteriormente, para obtener las direcciones de luz en las imágenes es necesario encontrar el punto de la esfera que sea más brillante, entonces el método de detección de dicho sector determinará cuan buena sea la estimación de la dirección. El método de detección utilizado se basa en detectar la ventana más brillante de la esfera, es decir, tomando un cuadrado de tamaño $N \times N$, éste se mueve píxel por píxel calculando la suma de todos los píxeles que quedan dentro de la ventana.

Considerando el método empleado, se considera de interés probar distintos tamaños de ventana y ver cómo esto impacta en las direcciones obtenidas, comparándolas contra las direcciones provistas por la cátedra, que serán tomadas de referencia (sabiendo que son una estimación también, y no las direcciones *verdaderas*, pero aún así las consideramos como referencia dado que no conocemos las direcciones reales).

Se realizó el siguiente experimento:

- Se calculan las direcciones de luz de cada imagen, utilizando la esfera cromada y mate, con tamaños de ventana **1, 5, 10, 15, 20, 30, 50 y 100**.
- Luego se calculan las distancias vectoriales (norma dos) entre las direcciones obtenidas contra su correspondiente dirección de referencia.
- Dado un tamaño, se grafican las distancias de las 12 direcciones.
- Además, para condensar más la información para cada tamaño de ventana se gráfica la distancia media.

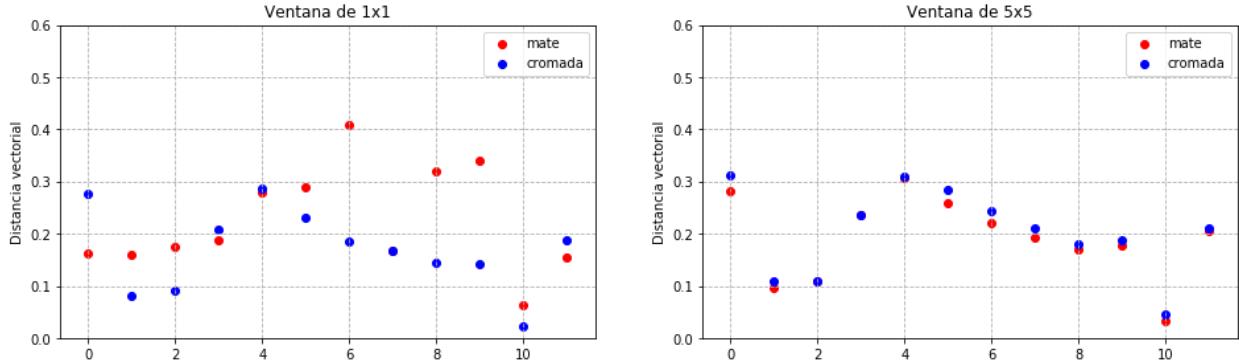


Figura 4: El eje X representa las distintas dirección de luz. Tamaños de ventana 1×1 y 5×5

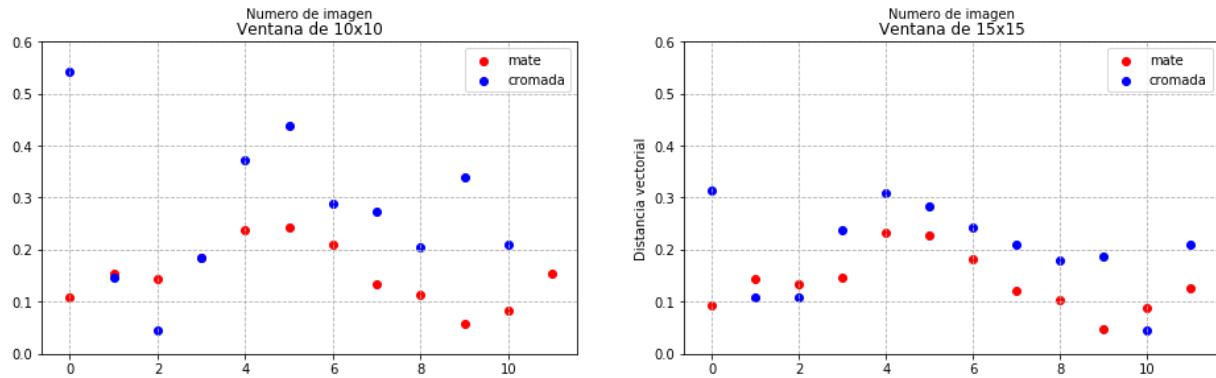


Figura 5: El eje X representa las distintas dirección de luz. Tamaños de ventana 10×10 y 15×15

Los resultados arrojan que tomar ventanas muy pequeñas o muy grandes resultan en malos resultados, lo cual tiene sentido, pues si se busca una ventana de 1×1 en realidad se está buscando el píxel más brillante, y si se toma una ventana muy grande 100×100 , seguramente habrá varias áreas “muy brillantes” en la esfera cromada y no queda claro cuál elegirá el algoritmo.

Se observa lo siguiente:

- En general los resultados son mejores utilizando la esfera mate, esto puede apreciarse en todos los gráficos, pero se observa de forma clara en la media de los errores.
- En el tamaño 50×50 (figura 9) se encuentra un mínimo de error promedio (aproximadamente 11 %).
- Con respecto a la utilización de la esfera cromada, se ve que se estabiliza en los valores de 15×15 a 30×30 (figura 9).

Esto indica que la esfera mate es más estable que la cromada, pues nunca hay grandes saltos de errores entre los tamaños de las ventanas, y encuentra valores muy buenos en 50×50 . Se puede adjudicar esta diferencia al tipo de superficies de cada esfera, la esfera mate al tener una superficie lambertiana refleja la luz de igual forma en todas direcciones, por lo cual debería haber una zona de mayor brillo que se va difuminando por la esfera, en cambio la cromada tiene sólo un foco muy iluminado, casi saturado en el cual no es claro cual es el punto más brillante.

3.1.2. Impacto de los métodos de combinación de rojo, verde y azul en la calibración

Se considera de interés hacer un análisis sobre el impacto de la proyección de RGB a escala de grises en la calibración, pues éste es un punto clave en la obtención de normales, con lo que afectará de lleno el

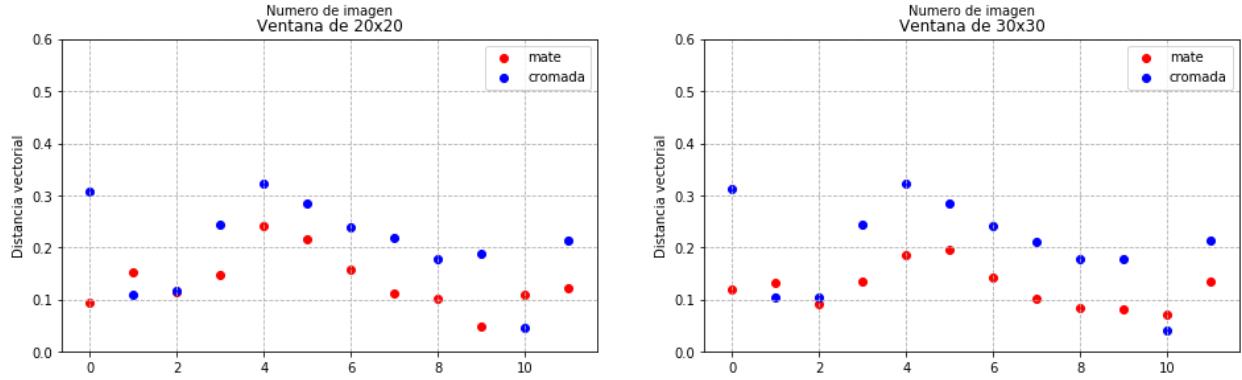


Figura 6: El eje X representa las distintas dirección de luz. Tamaños de ventana 20×20 y 30×30

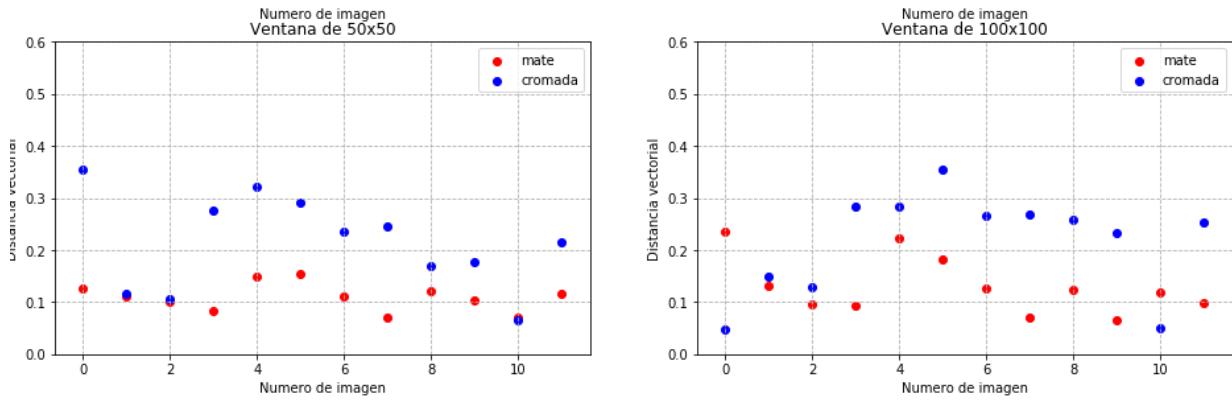


Figura 7: Diferencia entre direcciones obtenidas contra las provistas, variando el tamaño de ventana.

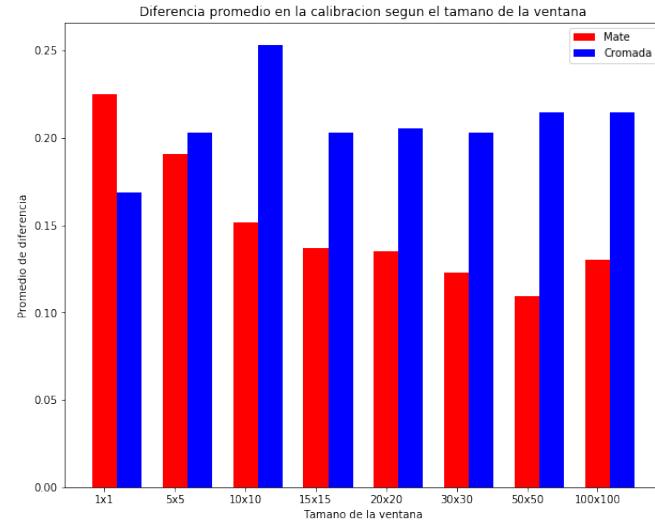


Figura 8: Media de las distancias según tamaño de ventana.

resultado final.

Siguiendo el procedimiento realizado anteriormente, comparamos el error de las direcciones estimadas usando el promedio clásico, y un promedio ponderado (2.2.1).

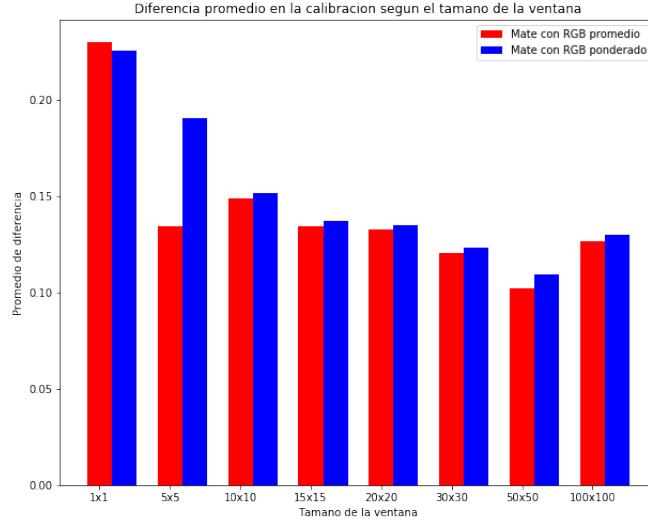


Figura 9: Diferencia entre direcciones obtenidas contra las provistas, usando distintos métodos para la conversión de grises

Los resultados indican que no hay mucha diferencia entre ambas transformaciones a escala de grises. Quizás esto se deba a que las imágenes de las esferas no son particularmente *coloridas*, podría ser que experimentar con imágenes más coloridas arrojen resultados más interesantes.

3.2. Diferencia temporal los diferentes métodos numéricos aplicados a la obtención de normales

3.2.1. Análisis de tiempos

Para la obtención de las normales, se debe resolver varias veces un sistema de ecuación en el cual sólo varía el término independiente. Para resolver un sistema de ecuaciones se puede utilizar eliminación gaussiana (desde ahora **EG**) directa, o realizar una factorización LU del sistema para luego resolver dos sistemas triangulares por separado. Este último acercamiento tiene ventajas cuando se desea resolver un mismo sistema múltiples veces y sólo varía el término independiente.

Dado que la obtención de normales entra en el último caso, se considera pertinente hacer un análisis para determinar si tiene una ventaja significativa en este caso particular. Para poder hacer esto se realizan mediciones de tiempos en la obtención de normales, utilizando los dos métodos mencionados, variando el tamaño de las matrices.

Para esto se toman matrices (cuadradas) variando su tamaño, yendo desde 100×100 hasta 1000×1000 , aumentando el tamaño en 50 en cada paso. Para cada tamaño se corrieron 100 veces ambos métodos, y se tomó el promedio. Se presentan en la figura 10 y en la figura 11 los tiempos obtenidos, graficados *crudos* y divididos por una función lineal, esperando así poder analizar su comportamiento.

Ahora bien, el algoritmo de EG tiene una complejidad de $O(n^3)$, en cambio realizar la factorización LU $O(n^3)$ y resolver un sistema triangular $O(n^2)$. A priori, parecería lógico pensar que la obtención de normales es un caso donde hacer factorización LU para luego resolver múltiples veces sistemas triangulares reduzca notablemente el tiempo necesario para generar las normales.

Sin embargo, como está plasmado en los resultados, los tiempos de obtención son virtualmente los mismos para ambos métodos. ¿A qué se debe esto? Si bien es cierto que resolver k sistemas de ecuaciones donde sólo varía el término independiente es $O(k * n^3)$, mientras que usando factorización LU es $O((n^3) + k * n^2)$, en este caso particular $n = 3$ con lo cual, la complejidad para ambos métodos es $O(k)$, k siendo la cantidad de píxeles de la imagen.

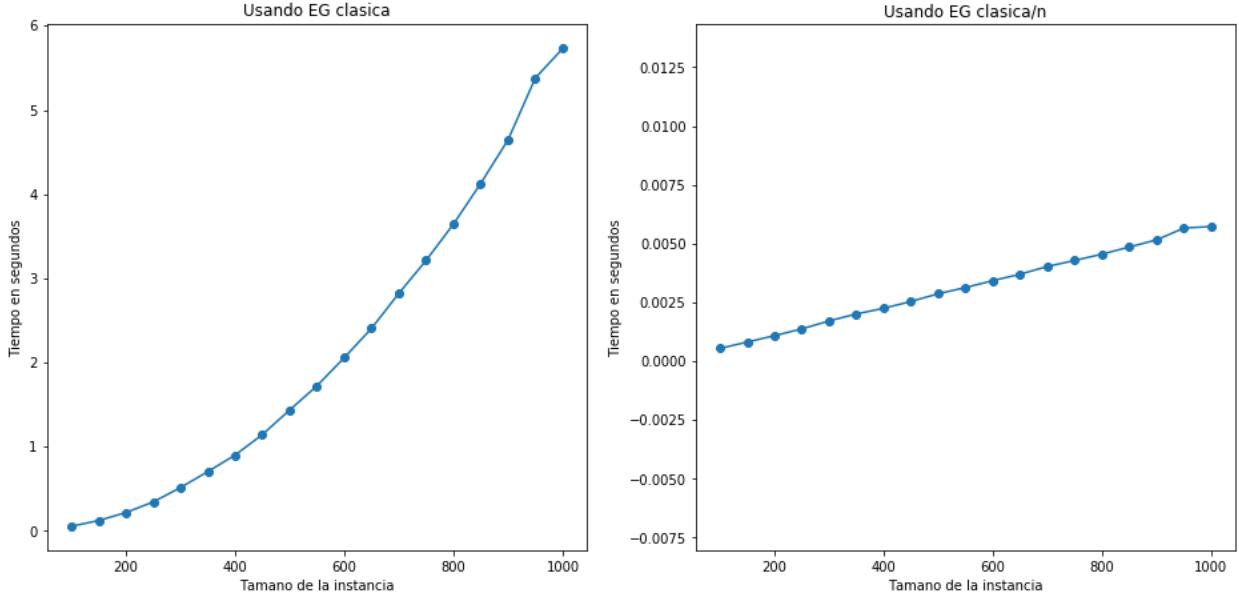


Figura 10: Comparación de tiempos usando EG

Considerando que se toman matrices cuadradas para la experimentación resulta que $k = m^2$, obteniendo así una complejidad cuadrática en m , siendo éste el ancho (o alto) de la matriz. Por dicho motivo, si se grafican los tiempos en función de m , se obtendrá una función cuadrática, y si se divide por alguna función lineal se obtendrá otra función lineal. Esto es observable en la figura 10 y en la figura 11, las funciones graficadas se comportan de forma lineal, corroborando lo dicho anteriormente.

Se puede concluir entonces que dado que el sistema es muy chico, da lo mismo cual método de resolución de sistemas se utilice, pues quien domina los tiempos es la cantidad de veces que se resuelve el sistema, no la resolución del sistema en sí. De hecho como el método de factorización LU debe resolver dos sistemas, en una matriz de 3×3 esto puede generar más overhead que resolver uno sólo usando EG.

3.3. Impacto de la elección de las 3 direcciones de iluminación en el cálculo de las normales

Para analizar el impacto de la formación de S (la matriz de luces) en el cálculo de las normales, deberemos realizar:

1. Generación de distintas matrices S usando las direcciones de luces de la cátedra. Queremos tener en cuenta la influencia del número de condición, por lo tanto generaremos distintos S con $\kappa(S)$.
2. Cálculo de las normales según las distintas S generadas
3. Análisis de la imagen generada por las normales, y comparación con la imagen generada por las normales provistas por la cátedra
4. Análisis numérico de las normales, y comparación numérica con las normales provistas por la cátedra.

La tabla se interpreta de la siguiente manera: Con un pequeño script de Python fueron chequeadas todas las combinaciones de luces posibles, y tomado 5 combinaciones distintas. Cada combinación tiene un porcentaje correspondiente a las otras combinaciones que tienen un peor número de condición. O sea, la combinación de percentil i , indica que tiene mejor número de condición que el $i\%$ de las combinaciones.

De esta manera, el del percentil 0 corresponde a la mejor combinación de luces en cuanto al número de condición de su S , y el del percentil 100 corresponde a la peor.

Los gráficos de los planos de las normales de cada combinación se adjuntan en el Apéndice.

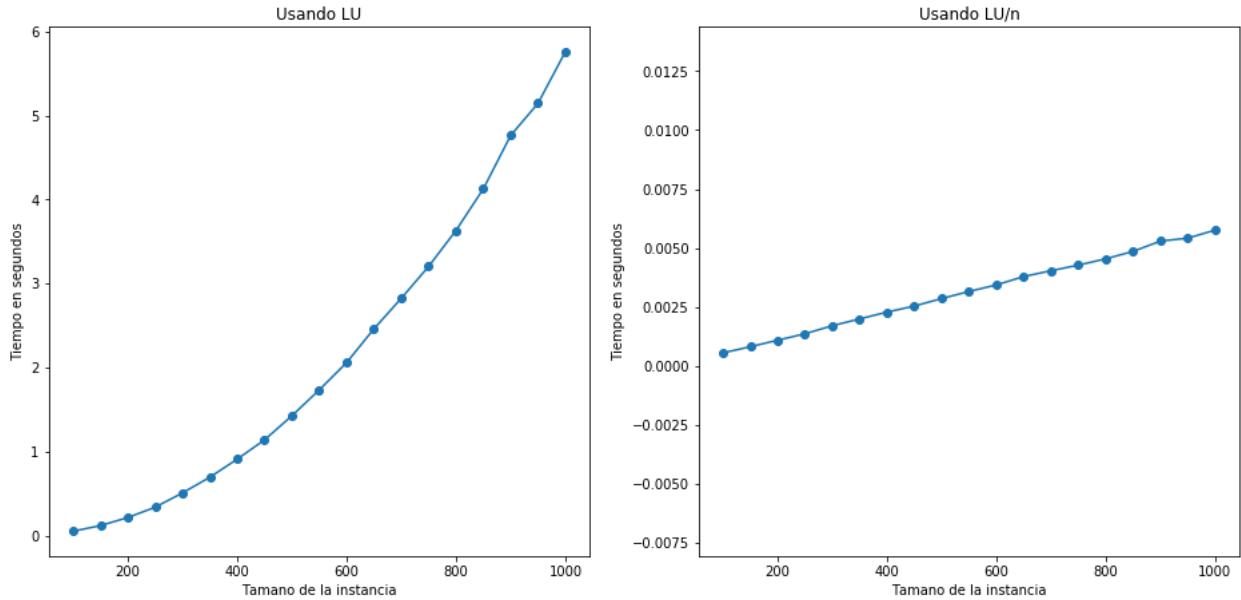


Figura 11: Comparación de tiempos usando LU

Cuadro 1: Comparación de direcciones de luz

Percentil	Dirección 1	Dirección 2	Dirección 3	Número de condición
0	0	5	10	5.48463645012
25	0	5	11	17.6842766004
50	3	8	11	26.4965999233
75	0	7	11	53.7015936876
100	3	4	7	2100.6680286

3.3.1. Introducción

La elección de las 3 imágenes con sus respectivas direcciones de iluminación es clave al momento de calcular las normales. Veamos por qué.

De las 3 imágenes que elegimos, se deduce la matriz de direcciones de luces que llamamos S (igual que como se llaman en el enunciado).

Recordamos que S se define de la siguiente manera:

$$S = \begin{pmatrix} s_x^1 & s_y^1 & s_z^1 \\ s_x^2 & s_y^2 & s_z^2 \\ s_x^3 & s_y^3 & s_z^3 \end{pmatrix}$$

Donde (s_x^i, s_y^i, s_z^i) es el vector de luces de la imagen i .

Y el cálculo de las normales se resuelve calculando:

$$\begin{pmatrix} s_x^1 & s_y^1 & s_z^1 \\ s_x^2 & s_y^2 & s_z^2 \\ s_x^3 & s_y^3 & s_z^3 \end{pmatrix} \cdot m = I$$

Donde $m, I \in R^{3x1}$, con I las intensidades del mismo pixel en las 3 imágenes; y donde el vector de la normal n se deduce de $n = m/(I_0\rho)$ (con $(I_0\rho)$ una constante cuyo cálculo no viene al caso).

3.3.2. ¿Cómo incide S sobre el cálculo de n?

Preguntarnos cómo incide S sobre el cálculo de n es equivalente que preguntarnos cómo incide S sobre el cálculo de m, ya que tienen una relación lineal definida por la constante ($I_0\rho$).

Entonces, ¿Qué problema podría ocurrir al tener una matriz S “de mala calidad”? Lo esperado, en términos generales, es que los vectores m de cada píxel nos queden *coherentes*, pero ¿Cómo definiríamos coherencia en este caso?

Un enfoque básico al respecto es que los píxeles con vectores de intensidades similares deberían dar valores de normales similares.

En otras palabras: Sean I_1, I_2 dos vectores de intensidades de dos píxeles diferentes cualesquiera. Definimos *coherencia* para sus m_1, m_2 correspondientes si vale que, cuanto más cercanos sean I_1 e I_2 , más cercanos son m_1 y m_2 .

3.3.3. Evaluando resultados coherentes según S

Formalicemos aún más este concepto de valores coherentes. Recordamos la definición de **Número de condición**:

Definición 3.1. Sea $A \in R^{m \times n}$, se define número de condición de A como $\kappa(A) = \|A\| \cdot \|A^{-1}\|$

Como hasta ahora no hemos ahondado en la materia en cuál es la mejor norma a tomar a la hora de calcular número de condición. Un test interesante habría sido tomar distintas normas y revisar empíricamente cuál es más estricta y cuáles son los valores de respectivos números de condición que ofrecen mejores resultados al calcular las normales. Lamentablemente por falta de tiempo no hemos podido realizar dicho experimento.

Dicho esto, para elegir las normales tomamos como referencia la bibliografía de la materia[1], cuando dice que, en general, es más rápido computacionalmente tomar norma infinito, y más exacto (aunque más complejo) tomar norma 2. Como los tiempos de ejecución no nos preocupan tanto en este caso, tomaremos norma 2. A su vez, afirma que tomar una matriz de $\kappa(A) = 10^k$, nos hace perder alrededor de k dígitos de precisión, por las operaciones de punto flotante realizadas.

3.3.4. Confirmación empírica de nuestra hipótesis

Tenemos nuestra hipótesis: **El número de condición de la matriz S determinará la correctitud de los cálculos de las normales.** Es decir, si elegimos 3 direcciones de luces tal que la Matriz S resulte con $K(S)$ muy elevado, es más probable ocurran errores numéricos al calcular las normales.

Entonces, queremos revisar:

1. Si nuestra hipótesis teórica realmente vale en la práctica
2. En caso de valer, quisiéramos saber cuál es efectivamente la relación entre número de condición de la matriz de luces y exactitud al calcular las normales. ¿Será lineal, cuadrática, cúbica?

Idea del esquema del test:

1. Armar distintas matrices de luces S, con distintos números de condición.
2. Calcular las normales de un objeto 3D, usando dicha matriz S.
3. Revisar qué tan bien se han calculado las normales en el paso anterior.
4. Concluir una relación (o no) del número de condición de las luces, y la exactitud del cálculo de las normales.

Pero ahora se presenta una nueva dificultad: Se tiene que definir una forma de revisar qué tan bien se calcularon las normales. Con este propósito, las ideas que surgen son:

- Comparar visualmente el gráfico de las normales obtenidas con el gráfico de las normales provistas por la cátedra.
- Comparar numéricamente las normales obtenidas con las normales provistas por la cátedra.

Como consideramos que ambas son totalmente válidas y que aportan tipos de información complementaria, decidimos no descartar ninguna y concluimos el esquema del test de la siguiente manera.

Esquema definitivo del test:

1. Armar distintas matrices de luces S, con distintos números de condición.
2. Calcular las normales de un objeto 3D, usando dicha matriz S.
 - a) Comparar visualmente el gráfico de las normales obtenidas con el gráfico de las normales provistas por la cátedra.
 - b) Comparar numéricamente las normales obtenidas con las normales provistas por la cátedra.
3. Concluir una relación (o no) entre número de condición de las luces y la exactitud del cálculo de las normales.

3.3.5. Comprobación empírica usando la esfera

Para corroborar empíricamente nuestra hipótesis teórica debemos comparar los gráficos de las normales de cada combinación de luces, chequeando al mismo tiempo su respectivo número de condición. (Ver tabla 1, y el gráfico de las normales en el apéndice).

Repetimos la tabla de las distintas combinaciones de luces, esta vez notando cuál figura corresponde a cada corrida. Las imágenes son colocadas en el apéndice, para poder darles suficiente tamaño como para que sean apreciables. La tabla queda:

Cuadro 2: Comparación de direcciones de luz, con sus respectivas figuras

Percentil	Dirección 1	Dirección 2	Dirección 3	Numero de condición	Figura
0	0	5	10	5.48463645012	19
25	0	5	11	17.6842766004	20
50	3	8	11	26.4965999233	21
75	0	7	11	53.7015936876	22
100	3	4	7	2100.6680286	23

Analicemos los 5 casos desde lejos: Comenzando por la mejor combinación (percentil 0), observamos que se distingue claramente la figura de la esfera, y también el fondo. Se distinguen las líneas que forma el "telón" que se divisa en la oscuridad detrás de la esfera, y la pequeña estructura sosteniendo la esfera. Se notan ciertas normales que aparecen quedando en cero, siempre fuera de la esfera, a causa de estar ubicadas en píxeles tan oscuros.

En la segunda mejor combinación analizada (percentil 25), se sigue distinguiendo la esfera con claridad, aunque el fondo comienza a perder detalle: Las líneas que definen el telón de atrás se pierden, y tenemos más normales en cero.

En la tercera mejor combinación analizada (percentil 50), se nota una pequeña diferencia en la esfera, aunque se sigue distinguiendo con claridad. Las líneas de la parte inferior del fondo ya quedaron completamente perdidas, y arriba a la derecha notamos una imperfección: Antes teníamos a las normales provocaban una textura sólida en el fondo, pero ahora en esa zona la textura está más ruidosa.

En la cuarta imagen (percentil 75), este ruido antes mencionado se mantiene, y notamos que se deja de distinguir con tanta claridad la esfera en la zona inferior izquierda de la misma. Se comienzan a mezclar la esfera con el fondo.

En la quinta y última (percentil 100, la peor combinación), notamos que bastantes más normales dan cero en el fondo. Además, el ruido que antes mencionábamos ahora se encuentra en la parte superior derecha de

la esfera.

Analicemos las normales más de cerca, comparando la mejor y la peor:

Ya teniendo un panorama general de como se forman las normales de la imagen, buscamos profundizar nuestra visión en la zona de la esfera, para observar con más detalle ese ruido que mencionamos que se genera en la peor combinación, y contrastarlo con cómo queda la mejor combinación.

En la figura 24 tenemos el plano de las normales, con zoom en la zona de la esfera, utilizando la mejor combinación de luces posibles. Notamos la uniformidad con la que están definidas las flechas, o sea, en una misma zona, todas las flechas apuntan al mismo lugar. Asimismo, los cambios entre las distintas zonas de la esfera se reflejan de forma gradual en las flechas: No hay modificaciones bruscas en sus direcciones. Estos comportamientos se corresponden con la figura de la esfera, y denotan resultados razonables.

Por otro lado, en la figura 25 tenemos el plano de las normales, con zoom en la zona de la esfera, utilizando la peor combinación de luces posibles. Apreciamos más de cerca el ruido que encontramos previamente, y notamos que, efectivamente, esto que habíamos identificado como ruido, son normales que dan resultados más azarosos. No se corresponden todos a una misma zona, ni están espaciados en algún patrón razonable. Esto, remitiendo a nuestro análisis del número de condición de la matriz de luces, refleja que el cálculo de las normales utilizando una matriz de luces S con $\kappa(S)$ muy alto, es extremadamente sensible a arrojar resultados incoherentes. Recordemos conceptualmente qué nos indica un $\kappa(S)$ alto. Teniendo $S \cdot x = b$, un $\kappa(S)$ alto nos indica que, si se realiza un pequeño cambio en b , puede causar un cambio brusco en el x deseado. O sea que al más mínimo error de precisión, el sistema de ecuaciones es muy sensible a arrojar una normal muy lejana a lo que tiene que dar, y esto es lo que está ocurriendo en el plano de normales de la peor combinación de luces.

Observación: Lamentablemente por falta de tiempo no pudimos hacer una comparación numérica. De todas maneras, nos parece suficientemente satisfactoria la comparación visual, y el análisis exhaustivo que hemos hecho sobre ella.

3.3.6. Comprobación empírica usando el buda

A pesar de estar satisfechos con el análisis de la esfera para justificar nuestras hipótesis, un contraargumento de su validez podría ser que es que estamos analizando sobre la misma imagen que utilizamos para calibrar (la esfera). Por lo tanto, buscamos aportar nuevos resultados, utilizando una imagen distinta. Analizaremos el buda.

Analizamos directamente con zoom para ver de cerca a las normales del objeto. Damos 3 imágenes:

- El buda con la mejor combinación de luces: figura 27
- El buda con la combinación media de luces: figura 28
- El buda con la peor combinación de luces: figura 29

Al comparar estas tres imágenes se hace más difícil que con la esfera, ya que el objeto no es tan uniforme, y no queda tan claro cuales deberían ser los valores de las normales en cada punto.

De todas maneras, se puede apreciar, análogamente que para la esfera, qué tan similar es una normal con respecto a sus direcciones vecinas y qué tan graduales son los cambios de las direcciones entre distintas áreas del objeto (notar que analizar numéricamente esto habría sido un excelente punto de análisis, aunque hacerlo bien sería muy complejo).

Veamos un ejemplo. En la figura de mejor combinación de luces se notan ciertas uniformidades en distintas áreas del rostro del buda. En los costados se aprecian normales horizontales, mientras que en el centro de la cara las normales se tornan verticales, con un poco de graduación entre ambas. En cambio, al empeorar las direcciones de luces, se pierde esa uniformidad.

Asimismo, en el centro de la imagen de mejor combinación, desde el pecho del buda hacia sus costados, se pueden apreciar nuevamente tanto uniformidades como cambios graduales y colectivos de las normales. En cambio, en la imagen de peor caso las normales parecieran tener más incidencia de un componente aleatorio y ruido.

Observación: ¿Por qué decimos que es malo este ruido en los planos de las normales?

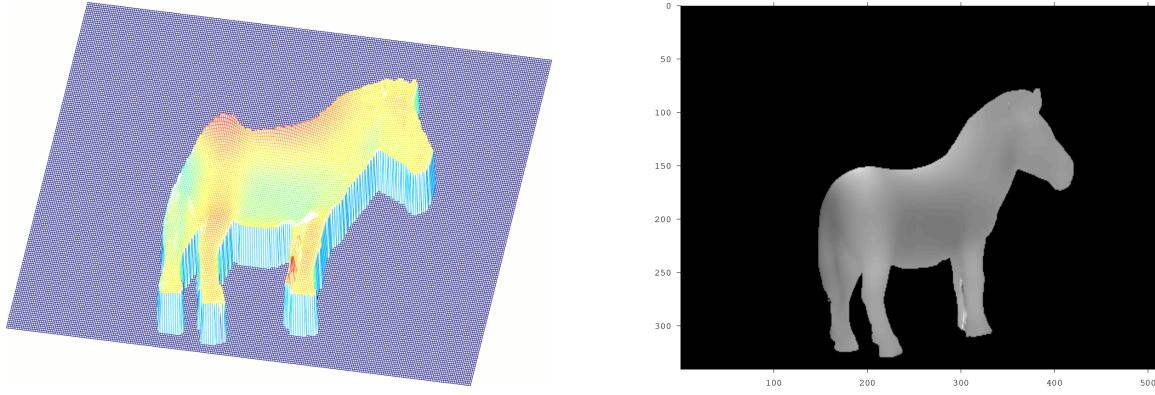


Figura 12: Imagen del caballo calculada con las luces 0, 5 y 10. La mejor combinación.

Porque esta componente de ruido en las normales luego afectará de mala manera el cálculo de las profundidades, ya que, al tener áreas con uniformidad, las profundidades van a resultar más uniformes, sin picos bruscos. De igual forma, un cambio gradual entre distintas áreas va a resultar en profundidades suaves.

En cambio, si el ruido de las normales se torna demasiado grande, los picos serán más frecuentes, y no habrá una similitud entre los z de píxeles contiguos, resultando en una predicción errónea. Esto se debe a que estas imágenes que analizamos no tienen bordes muy marcados, sino que tienen un contorno suave.

3.4. Impacto de la elección de las 3 direcciones de iluminación en el armado del modelo 3D

Buscamos comparar cómo se arma el modelo 3D en base a distintas combinaciones de direcciones de luces. Para ello, utilizamos los resultados del experimento en el que analizamos los números de condición de las combinaciones de luces. Recordamos que, entre todas las combinaciones, tenemos:

- La combinación 0, 5 y 10 resultó en una matriz de iluminación S de $\kappa(S) = 5,48363$. Es la mejor combinación, en cuanto al $\kappa(s)$.
- La combinación 3, 8 y 11 resultó en una matriz de iluminación S de $\kappa(S) = 26,49660$. La combinación “del medio” en cuanto al $\kappa(S)$, tiene la misma cantidad de mejores combinaciones que de peores.
- La combinación 3, 4 y 7 resultó en una matriz de iluminación S de $\kappa(S) = 2100,66802$. La peor combinación, en cuanto a $\kappa(S)$.

3.4.1. Analizando el modelo 3D del caballo

Corremos tres instancias,

Análisis del mejor caso, figura 12: Nos vemos limitados porque tenemos que analizar un modelo en 3 dimensiones con una foto en dos. De todas maneras, observando los colores (que se van transformando de azul a rojo, cuanto más alto de el z), es apreciable visualmente que dan resultados razonables en los valores de profundidad. Podemos ver como se forma el contorno de la figura con claridad, observándose mayores alturas en “la panza” del caballo, y en la parte posterior del mismo, detalles correspondientes con el modelo 3D utilizado de input. También es apreciable que no se forman agujeros en el medio (o sea, valores de z significativamente menores de lo que deberían valer), que significaría que hay cierta parte del modelo que no tuvo suficiente información como para calcular buenos valores de profundidad, o sea, que ninguna de las 3 imágenes usadas para el análisis tiene cierta parte del modelo bien iluminada.

Análisis del caso medio, figura 13: En estos resultados vemos que el modelo comienza a perder forma. Por un lado, distinguimos más picos (como en el lomo del caballo), y por otro, distinguimos estos *agujeros* mencionados previamente, correspondientes a partes del caballo no iluminadas correctamente por ninguna de las tres direcciones de luz. Ambas deformaciones del modelo se deben a que las combinaciones de luces aportan mucha información sobre partes similares del caballo, y poca sobre otras partes, donde no

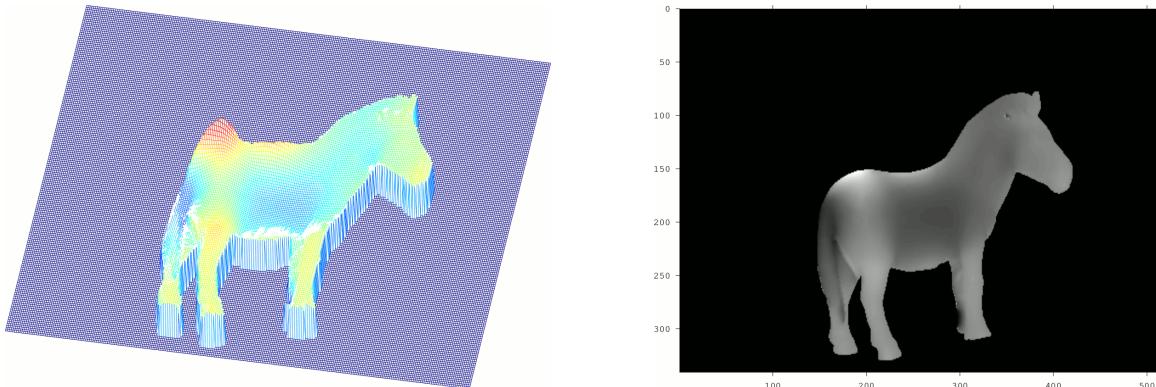


Figura 13: Gráfico y heatmap del caballo con las luces 3, 8 y 11 (la combinación del medio)

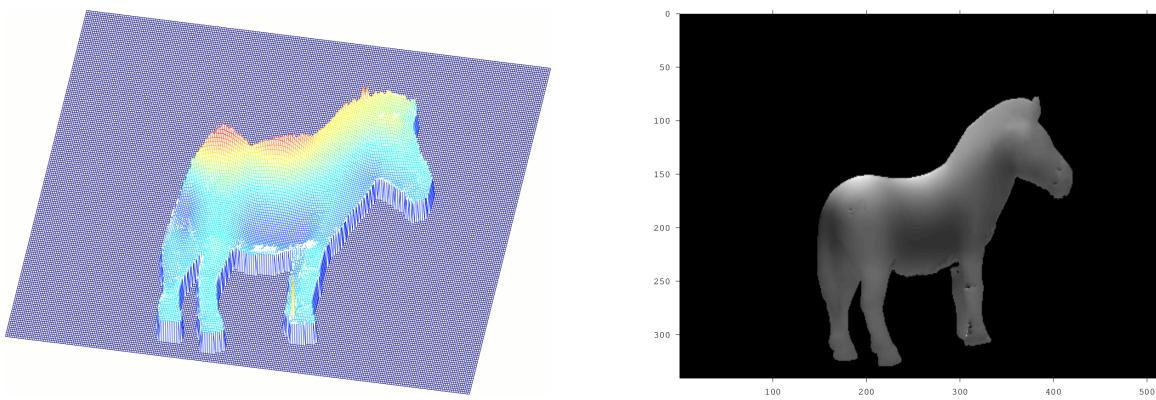


Figura 14: Gráfico y heatmap del caballo con las luces 3, 4 y 7 (la peor combinación)

llega la luz. En este caso, parecería que las luces dejan la parte inferior del caballo mal iluminada, por lo que indicaría que las 3 luces apuntan desde arriba. O sea, en lugar de que las luces se complementen, aportan información redundante en la parte superior, y no llegan a juntar suficiente información para la parte inferior de la figura.

Análisis del peor caso, figura 14: Aquí vemos las imperfecciones de la combinación media magnificadas. Notamos un agujero de mayor tamaño en la parte inferior de la figura, y también un agujero nuevo en la parte inferior de la pierna posterior del caballo. Al mismo tiempo que crecen los agujeros de abajo, crecen los picos de arriba: Lo que antes era un pequeño pico aislado se convirtió en un pico que cubre toda la frontera superior del caballo.

3.4.2. Analizando el modelo 3D del buda

Para complementar estos resultados, realizamos en mismo análisis sobre la imagen del buda. Calculamos igual que antes: Para las combinaciones mejor, media y peor calculamos el modelo 3D y un heatmap (cuyos píxeles blancos indican más profundidad, y negros menos).

En la figura de mejor combinación (figura ??), se aprecia claramente la figura del buda, con todos sus detalles. El contorno parece suave, no se aprecian muchos picos ni *agujeros* de baja profundidad como ocurría con el caballo.

En la figura de combinación media (figura ??), ya podemos ver ciertos picos (por los pies del buda, y en su rodilla izquierda), y se comienzan a perder ciertos detalles, que podrían seguir allí, solo que muy sutiles para nuestro análisis visual.

Ya en la peor combinación nos queda un resultado muy diferente al de la mejor. Con el caballo no parecía haber tanta diferencia, pero con buda aparentemente el impacto es mucho más grave: Se formó una laguna gradiente hacia abajo, en la que las profundidades van tiendiendo hacia el cero. Al mismo tiempo, igual que

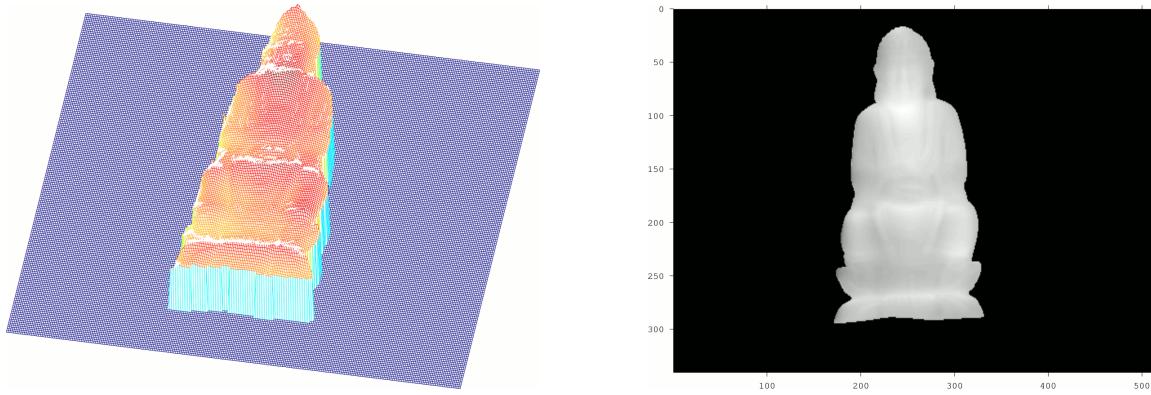


Figura 15: Gráfico y heatmap del buda con las luces 0, 5 y 10 (la mejor combinación)

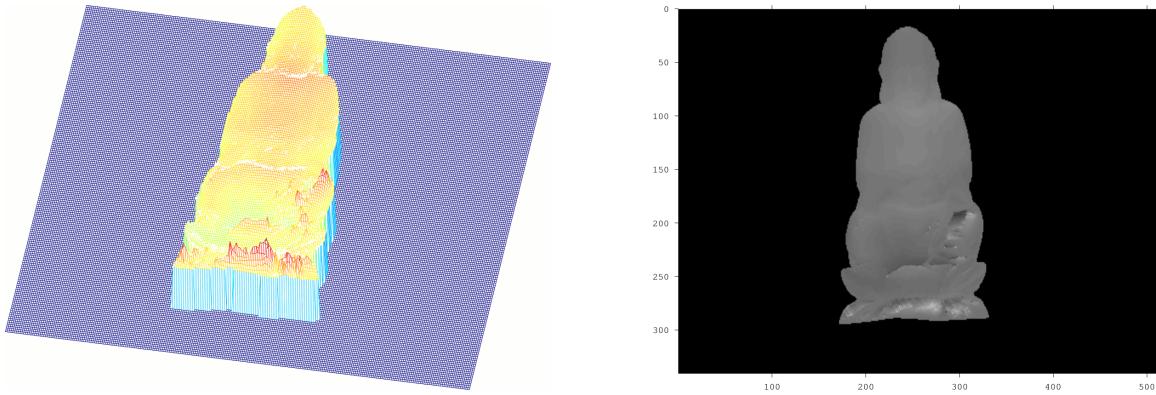


Figura 16: Gráfico y heatmap del buda con las luces 3, 8 y 11 (la combinación media)

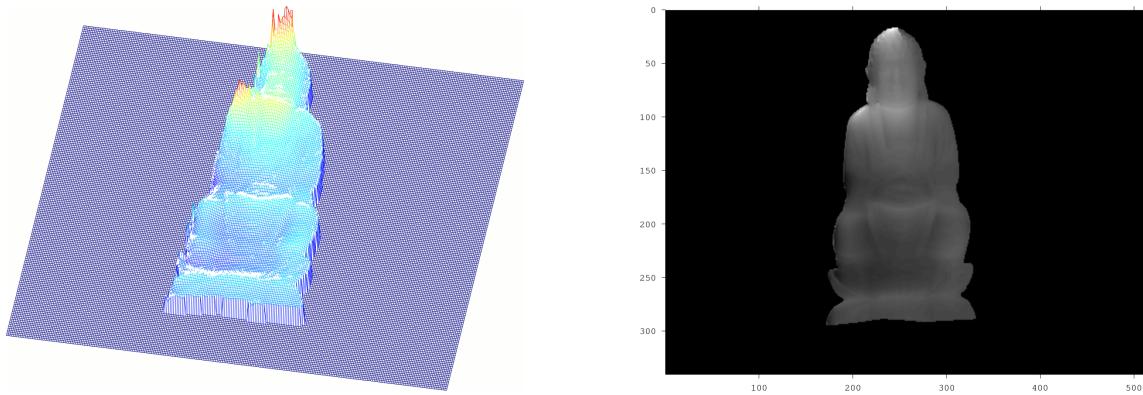


Figura 17: Gráfico y heatmap del buda con las luces 3, 4 y 7 (la peor combinación)

habíamos visto en el caballo, mientras se forman lagunas por algunos lados, se forman picos por otros. De igual manera que antes, las lagunas se forman en la parte inferior (donde la luz no llega tanto), y los picos en la parte superior (de donde viene la luz).

De esta manera, utilizamos dos imágenes distintas para corroborar que nuestro análisis tenga validez, y buscamos llegar a una conclusión.

3.4.3. Conclusión

Como era esperado, cuando las tres luces no aportan información sobre una parte de la figura, ésta se ve debilitada en el modelo 3D. Esto se puede apreciar por los *agujeros* que se forman, o sea, por las lagunas de píxeles que surgen, en las que las profundidades dan menor a lo que debería dar.

Por otro lado, nos llevamos una sorpresa: No sólo las partes del modelo con poca información se ven afectadas, parecería ser que las partes con información redundante resultan con más picos. Además, comparando las figuras de la combinación media y la peor, resultó que parecería ser que cuanto más redundante sea la información obtenida para ciertos píxeles, más picos surgen en ellos.

¿Por qué ocurre esto? Recordemos el experimento que analiza las normales con distintas combinaciones de luces. Habíamos visto que, al elegir buenas combinaciones de luces, las normales forman áreas uniformes y cambian muy gradualmente. En cambio, con malas combinaciones, se agrega ruido, y ciertas normales se ven afectadas por el alto número de condición de la matriz de luces, resultando en valores muy distintos a lo que deberían dar. Este ruido, justamente, es lo que causa los picos en las profundidades.

3.5. Comparación de tiempos: Eliminación Gaussiana vs. Cholesky

A la hora de resolver los sistemas de ecuaciones se implementaron dos algoritmos, Cholesky y Eliminación gaussiana. Para poder analizar cual podría considerarse ventajoso sobre el otro decidimos realizar una comparativa de tiempos modificando los tamaños de las imágenes utilizadas. Es importante destacar que cuando implementamos los algoritmos conseguimos optimizar de gran manera la implementación de Cholesky aprovechando la estructura de matriz banda. Por otro lado no logramos obtener un enfoque particular al esquema de la Eliminación Gaussiana que evite los accesos innecesarios a la matriz o tome ventaja sobre la estructura planteada.

En la tabla 3 pueden apreciarse los resultados obtenidos. Además creemos importante resaltar que no realizamos un análisis con imágenes más grandes porque la Eliminación Gaussiana requería de tiempos de ejecución poco prácticos para los tiempos disponibles en el marco de esta experimentación.

Cuadro 3: Comparación de tiempos de ejecución para distintos tamaños de imágenes variando el método de resolución

	20x20	30x30	40x40	50x50	60x60
Cholesky	0.004694	0.013285	0.039614	0.090348	0.163457
Eliminación Gaussiana	0.172111	1.65393	9.13952	24.6951	68.0621

Con el fin de poder comparar gráficamente los resultados obtenidos en el marco de esta experimentación es que realizamos la figura 18. En la primer imagen se puede ver que los tiempos de ejecución de la EG superan ampliamente aquellos logrados por el algoritmo de Cholesky, tanto es así que la escala no permite apreciar ambas mediciones a la vez por lo que adicionalmente mostramos el comportamiento de Cholesky, pero multiplicando sus tiempos por 100 para que se pueda apreciar la abismal mejora que resulta este algoritmo sobre EG. Con facilidad se observa que ni empeorando 100 veces los tiempos los resultados temporales de ambos algoritmos se acercarían.

Además si bien las imágenes pequeñas (menores a 20x20) son despreciables a fines prácticos, pues no es realista querer realizar una técnica como la fotometría estéreo en imágenes de tan poca resolución, quisimos mostrar que la diferencia temporal en la única zona de la primer imagen donde parecen comportarse similarmente sigue siendo porcentualmente grande.

En vistas de los resultados obtenidos y sabiendo que en la teoría coinciden en una complejidad temporal de $O(n^3)$ (aunque Cholesky tiene una menor constante) llegamos a la conclusión de que no existe fundamento real que nos lleve a elegir el algoritmo de Eliminación Gaussiana por sobre la Factorización de Cholesky seguida de forward substitution. Para confirmar nuestra conclusión restaría repetir la comparativa con alguna optimización de EG para nuestra estructura, si es que existe.

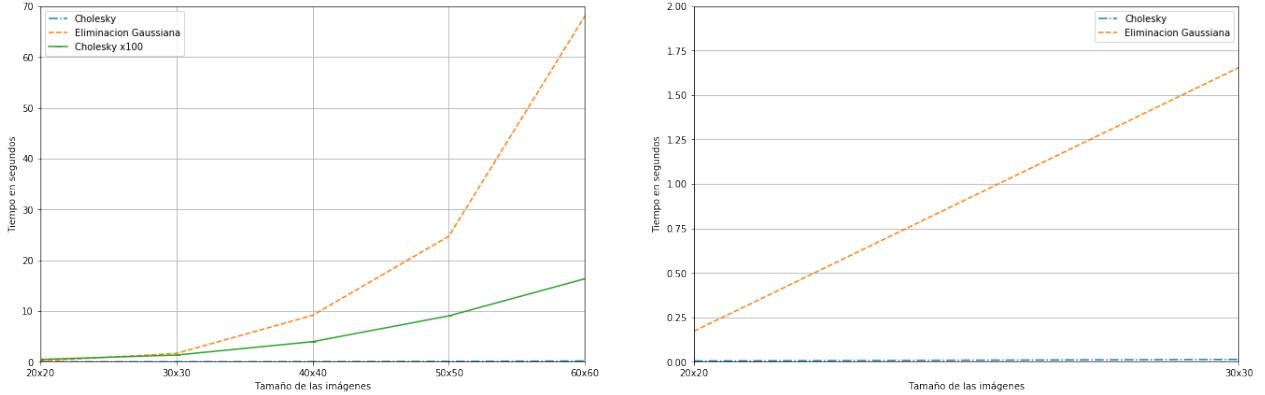


Figura 18: Comparativa de tiempos de ejecución para distintos tamaños de imágenes variando el método de resolución

3.6. Especificaciones técnicas relativas a la experimentación

3.6.1. Medición de tiempos

Para las mediciones de tiempos se utilizó de base el código presentado por la cátedra en la clase del 01/09/2017. A continuación puede verse un modelo del código utilizado para dichas mediciones.

```
clock_t start = clock();
llamado_a_funcion_a_medir();
clock_t end = clock();
double segs = (double)(end-start) / CLOCKS_PER_SEC;
cout << segs << endl;
```

3.6.2. Especificación técnica de la PC utilizada

Para las mediciones y comparaciones presentadas se utilizó siempre la misma PC, contando con las siguientes especificaciones respecto a su hardware y software:

```
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  61
Model name:             Intel(R) Core(TM) M-5Y10c CPU @ 0.80GHz
Stepping:                4
CPU MHz:                800.039
CPU max MHz:            2000,0000
CPU min MHz:            500,0000
BogoMIPS:                1995.35
Virtualization:         VT-x
```

4. Conclusión

4.1. Trabajo futuro

Se nos ocurrieron muchas alternativas a funciones o decisiones que fuimos tomando, para testear por tiempos y también por calidad, calculando tradeoffs, entre otras:

- Optimizar Eliminación Gaussiana para la representación de matriz banda.
- Utilizar pivoteo parcial y/o total al realizar Eliminación Gaussiana y PLU. Esto se nos ocurrió apenas lo vimos en la teórica, y no llegamos a implementarlo.
- Calcular la normal para los bordes aproximando por otros planos tangentes. O sea, en los casos borde, en lugar de tomar el plano tangente dado por los vectores $(x, x+1)e(y, y+1)$, tomar $(x, x-1)e(y, y-1)$

También se nos vinieron a la mente otros experimentos posibles como:

- Elección de luces: como incide en el calculo de I0 y Ro. El píxel elegido para obtener I0 y Ro, ¿incide en los valores obtenidos? ¿Cuánto incide? ¿Conviene tomar el más brillante? ¿Cómo elegirlo?
- Obtención de normales.
 - Analizar la pérdida de precisión si se usa la suma de Kahan o no (precisión vs. complejidad)
- Estimación de profundidades
 - ¿Cómo afectan la estimación de las profundidades el cálculo de las normales.
 - ¿Qué métodos de solución de los sistemas lineales arrojan mejores cálculos de profundidades?

4.1.1. Implementación con paralelismo

A medida que avanzamos con el TP tuvimos que ir realizando pruebas con imágenes de diversos tamaños. A pesar de que es evidente, nos llamó la atención lo rápido que funcionaban los distintos algoritmos con imágenes más pequeñas (por supuesto que estos tiempos se vinculan directamente a las complejidades de los distintos algoritmos).

Pensando en este aspecto, se nos ocurrió pensar que sucedería si una misma imagen es partida en varias más pequeñas y se realiza el procesamiento en paralelo aprovechando que hoy en día la mayoría de las PCs cuentan con procesadores multithread.

En cuanto a cómo funcionaría esta implementación con los algoritmos utilizados, creemos que el único inconveniente podría venir de los bordes. Por como armamos los v al partir las imágenes tendríamos más píxeles (x, y) tales que obtener su vector normal utilizando $(x+1, y)$ o $(x, y+1)$ sería imposible.

Para suplir esto consideramos que podría ser útil redefinir los vectores normales en dichos píxeles obteniendo en cambio vectores formados por $(x, y-1)$ o $(x-1, y)$. Este cambio es definitivamente posible, pero es necesario notar que podría llevar a que los sistemas lineales planteados pierdan algunas de las propiedades útiles para resolverlos. Por ejemplo, podríamos obtener una matriz M que no sea rango columna completa y no sea posible resolver el sistema mediante la factorización Cholesky.

Es aquí donde sería necesario ahondar un poco más en cuáles serían los *trade-offs* que se estaría dispuesto a hacer. Si se cuenta con suficientes procesadores el hecho de perder propiedades útiles podría no ser tan costoso pues se deberían resolver sistemas más pequeños, pero análogamente con sistemas menos capaces quizás se gana más con las optimizaciones posibles sobre matrices con propiedades conocidas. Finalmente concluimos que una implementación aprovechando el poder de *multithreading* de los procesadores actuales podría resultar en un proyecto interesante y con el potencial de mejorar significativamente los tiempos de corrida del programa presentado.

4.1.2. Conclusiones finales

A lo largo del desarrollo del trabajo presentado nos hemos cruzado con varias dificultades, que nos han dejado varios aprendizajes.

En primer lugar, nos vimos obligados a buscar representaciones alternativas de matrices, ya que al construir M , no alcanza la memoria RAM de una máquina promedio para poder almacenarla. Aprovechamos el hecho de saber cómo estaba compuesta la matriz, es decir, sabíamos que la matriz era esparsa, lo cual nos permitió representar sólo los valores no nulos. De esta forma no sólo pudimos tener en memoria a M , si no que pudimos mejorar en gran medida los tiempos de ejecución de los algoritmos necesarios para resolver los distintos problemas presentados (multiplicación, sistemas de ecuaciones, etc.).

Por otro lado, a mitad de desarrollo del trabajo nos vimos varias veces trabados frente a la imposibilidad de saber con algún grado de confianza si las profundidades estimadas tenían sentido, o si no lo tenían, cómo saber en qué etapa del pipeline había un problema. Pudimos paliar esta situación implementando varias de las funcionalidades en *Matlab*, para tener un punto de referencia contra el que contrastar, y creemos que ésta fue una de las mejores decisiones que tomamos. Nos proveyó de una base importante, que nos ayudó a apuntalar nuestro desarrollo en *C++*. Como reflexión, hubiera sido mucho más sensato primero resolver el trabajo en *Matlab*, y luego migrarlo a *C++*.

Si bien esto mencionado no es central al trabajo, sí presentó grandes dificultades implementativas que consumieron gran parte del tiempo que podríamos haber dedicado a experimentación. Si desde un primer momento hubiéramos sido conscientes de la utilidad de ésto, quizás hubiésemos manejado los tiempos de otra manera.

Yendo a la problemática abarcada en el trabajo desarrollado, pudimos concluir que la elección de los vectores direcciones de las luces es fundamental para generar un campo normal fidedigno al objeto que se quiere modelar. Esto lo fundamentamos teóricamente y luego lo corroboramos empíricamente con experimentos, ambos ítems en la sección 3.3. A su vez, en la sección 3.4, entendimos que un campo normal sin ruido es clave para poder realizar una certera estimación de las profundidades, pues éstas se derivan del sistema de ecuaciones que relacionan las normales con las profundidades. En ambos experimentos (3.3 y 3.4), de forma paralela, empeoramos gradualmente las direcciones de luces observando de qué forma nuestros resultados se veían afectados. Por un lado, el plano normal va incrementando en ruido, formando movimientos bruscos y rompiendo uniformidades. Luego entendimos que estos ruidos y movimientos bruscos son los que formarían picos y lagunas en el modelo 3D: Por un lado, pequeños lugares donde el z toma valores bruscos y erróneos, y por otro grandes zonas de píxeles donde el z se acerca al cero. Gracias a estos dos análisis separados, pudimos concluir cómo es que la correcta elección de los vectores dirección de luz es de vital importancia para poder estimar las profundidades.

Como concluimos que la elección de las direcciones de luces son de semejante importancia, resulta importante tener un método eficiente para estimarlas, y por lo tanto es ese el primer proceso delicado del pipeline, que en definitiva es encontrar la zona más brillante de la esfera, como lo explicamos en la sección 3.1. El método que concluimos como mejor es tomar una ventana de 50×50 píxeles y usar la esfera mate que mostró reflejar la luz de manera más *estable*, por ser una superficie *lambertiana*.

Recapitulando, entendemos que el proceso general que llevamos a cabo es viable como técnica práctica para obtener un modelo 3D de objetos a partir de imágenes estáticas. Aún más, podemos concluir que la correcta utilización de los distintos métodos, la optimización de las estructuras y los algoritmos involucrados y la correcta experimentación son estrictamente necesarios para llegar a obtener un sistema integral utilizable. Como detallamos a lo largo del trabajo hay límites estructurales (poder de procesamiento, cantidad de memoria RAM disponible) que no sólo dificultan sino que hasta pueden imposibilitar el uso de esta técnica en su totalidad. Podemos detallar los tiempos de ejecución de Cholesky y EG (optimizado y no optimizado respectivamente, ver sección 3.5) o la necesidad de crear estructuras específicas que aprovechen los conocimientos que tenemos de la técnica (ver sección 2.5.2) como algunos de los ejemplos con los que nos topamos.

Es sobre estos ejemplos particulares sobre los que más tuvimos que trabajar y por ello consideramos sumamente satisfactorio el hecho de haber encontrado ciertas técnicas y análisis que hayan convertido algoritmos que en principio parecían lentos, como Cholesky, en herramientas muy interesantes y que funcionan

sorprendentemente rápido. Comparativamente como concluimos en la sección citada anteriormente logramos tiempos de ejecución muy superiores a la técnica de EG y que sobrepasan lo que esperábamos lograr. De la mano de este análisis se desprende también el destaque a la adaptación que realizamos para las diversas estructuras de matrices, esparza, en bandas o común, y que nos llevó a entender cómo conocer y analizar una instancia de un problema para poder transformar algo que inicialmente parece irrealizable en un programa realmente utilizable en la práctica.

En resumen, consideramos que este trabajo abarca el desarrollo necesario, su correcta optimización para el problema específico a resolver y una sólida base de experimentación con fundamentos teóricos, y nos ayudó a comprender los fines prácticos que pueden tener los diversos métodos numéricos vistos en la materia.

5. Apéndices

5.1. Planos normales variando las direcciones de luces

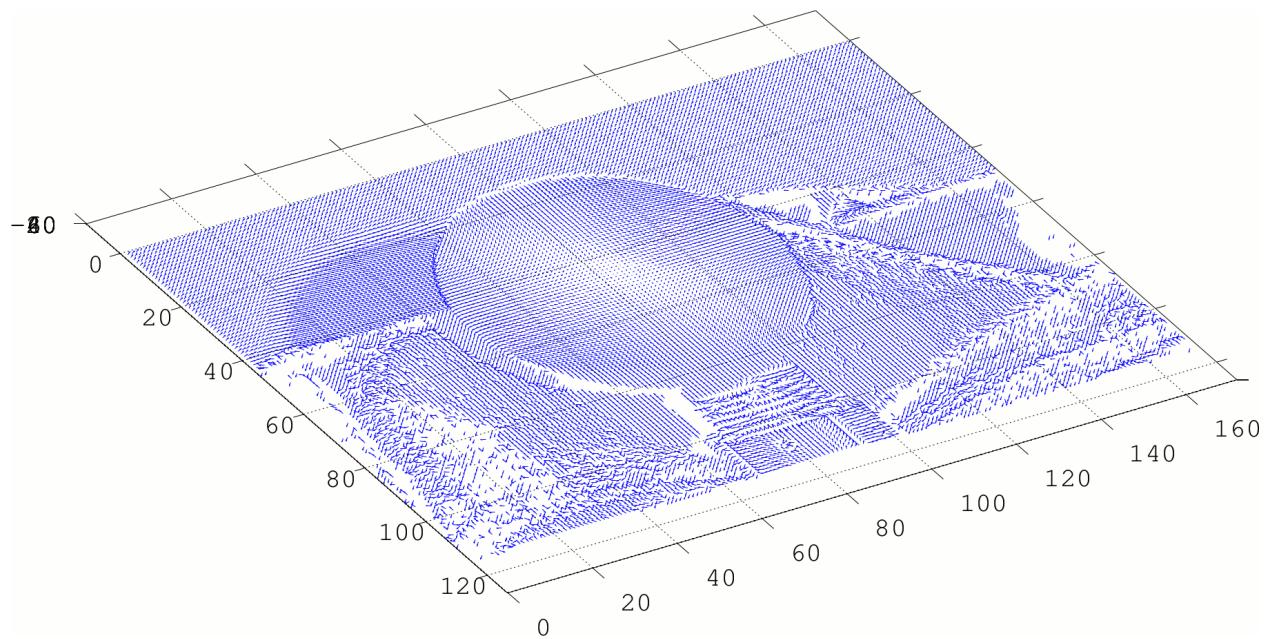


Figura 19: Cálculo de las normales para la imagen de la esfera (La mejor)

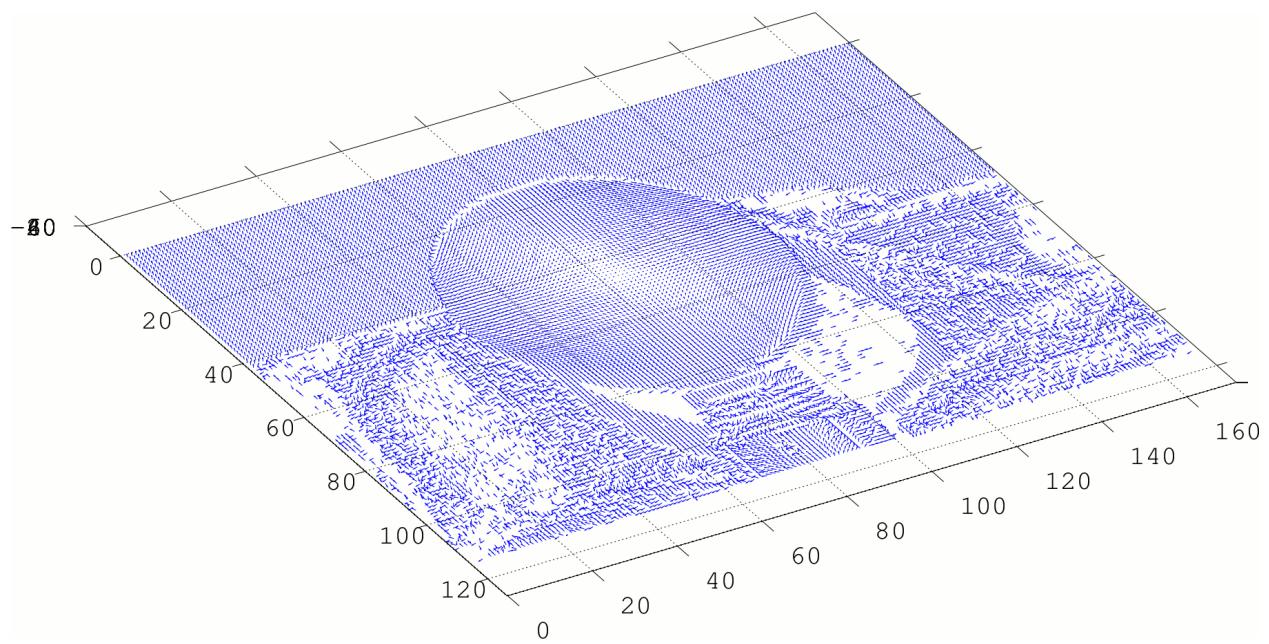


Figura 20: Cálculo de las normales para la imagen de la esfera, (La que es peor al 25 % de las combinaciones)

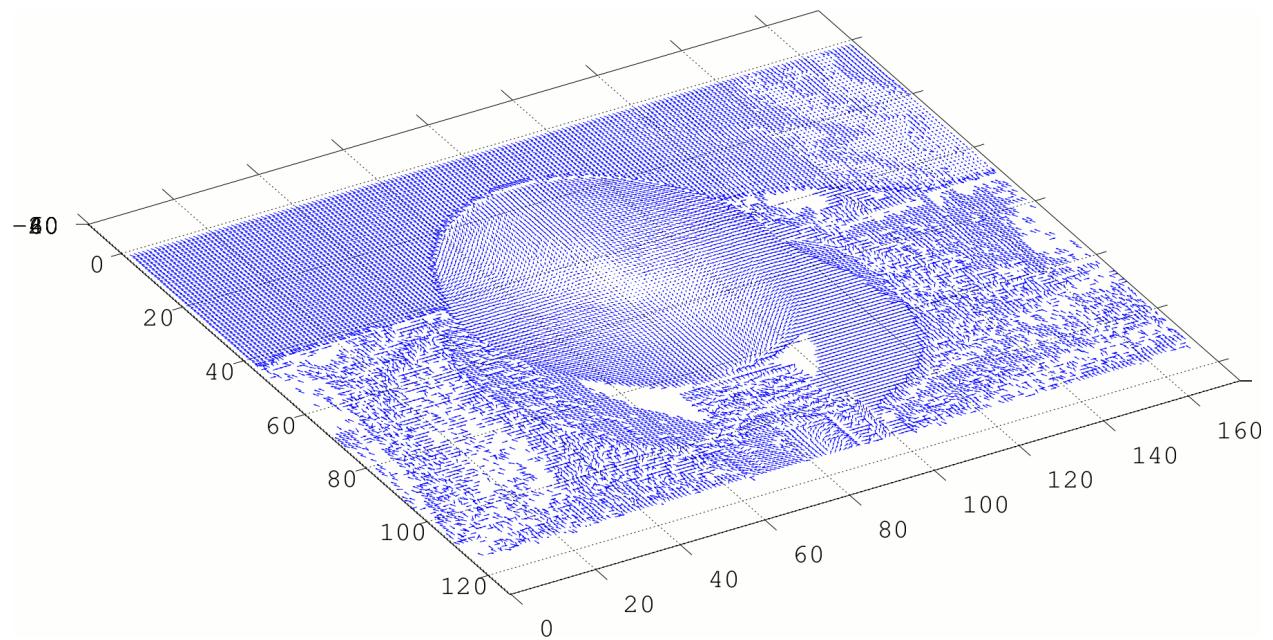


Figura 21: Cálculo de las normales para la imagen de la esfera, (La que es peor al 50 % de las combinaciones)

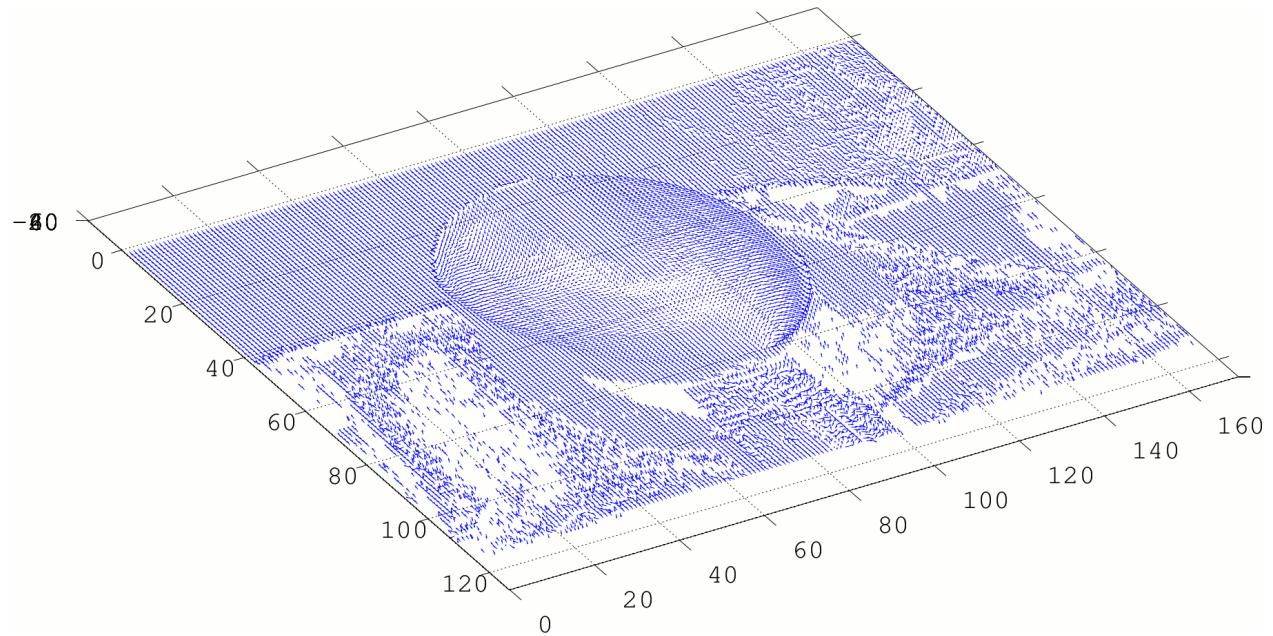


Figura 22: Cálculo de las normales para la imagen de la esfera, (La que es peor al 75 % de las combinaciones)

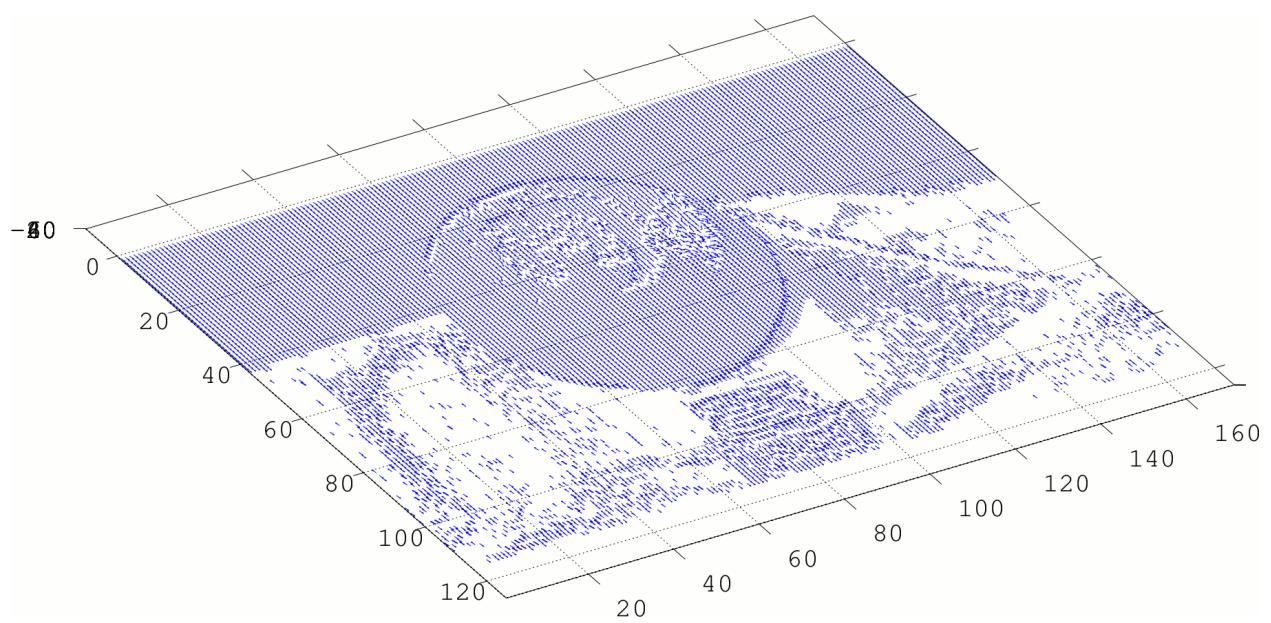


Figura 23: Cálculo de las normales para la imagen de la esfera, (La peor)

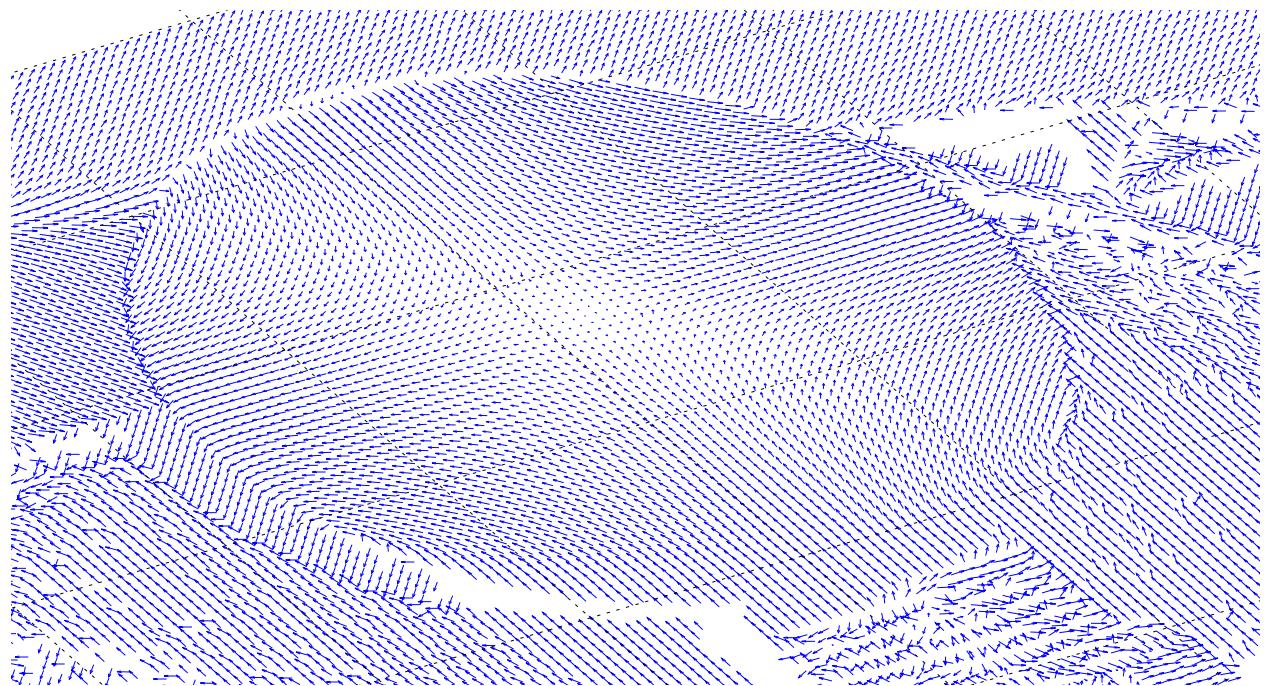


Figura 24: Cálculo de las normales, haciendo zoom en la esfera, usando la mejor combinación de luces

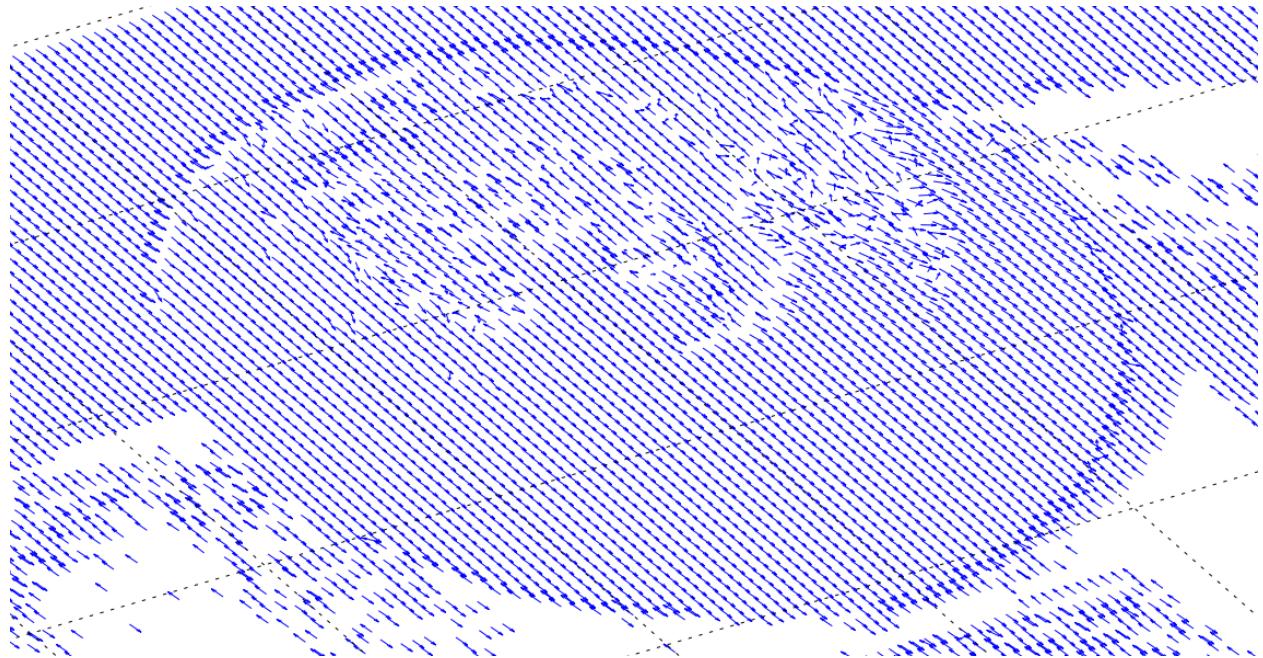


Figura 25: Cálculo de las normales, haciendo zoom en la esfera, usando la peor combinación de luces

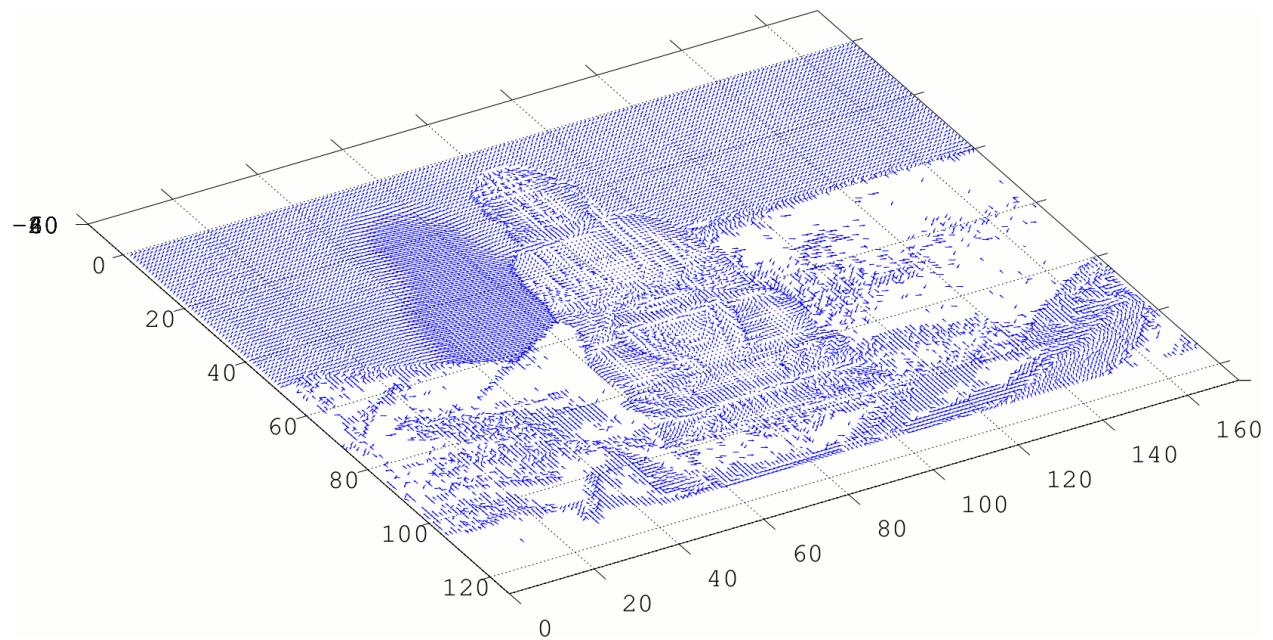


Figura 26: Cálculo de las normales, usando la mejor combinación de luces

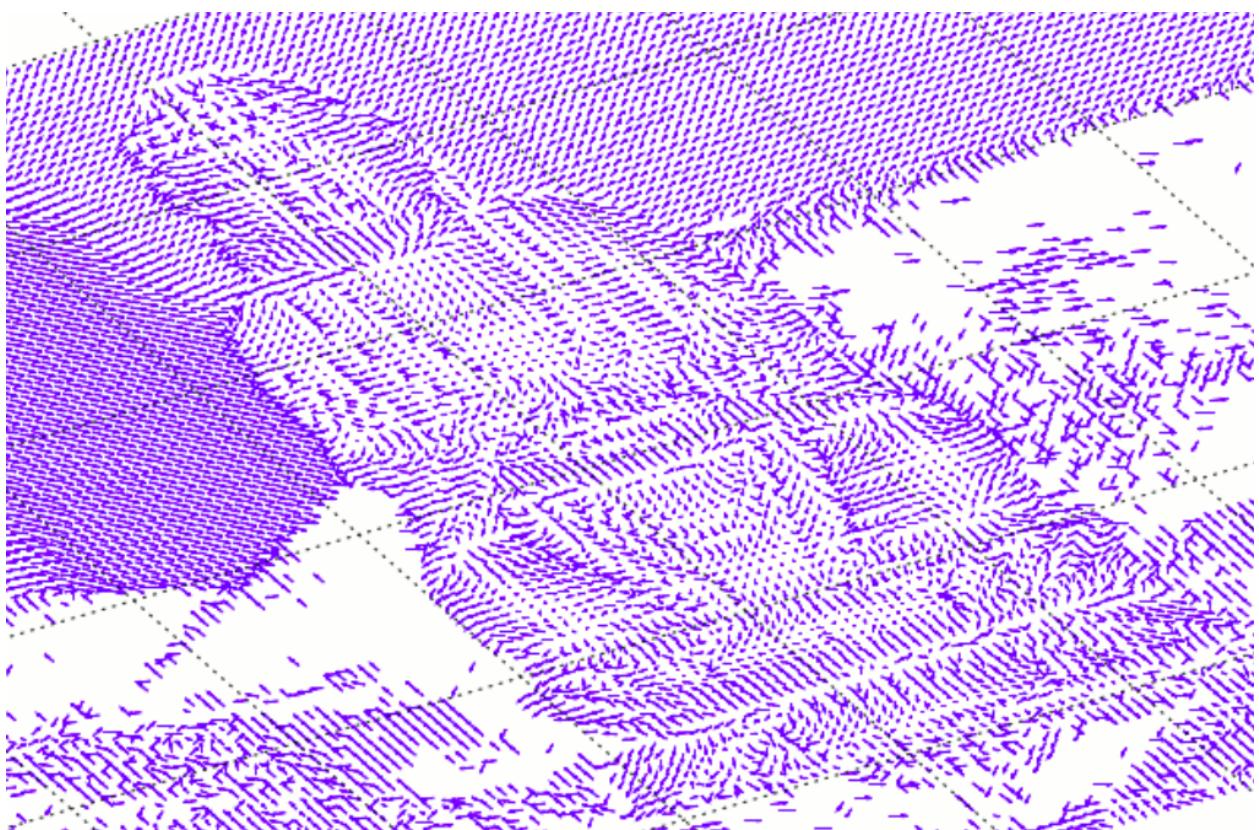


Figura 27: Cálculo de las normales, haciendo zoom en el buda, usando la mejor combinación de luces

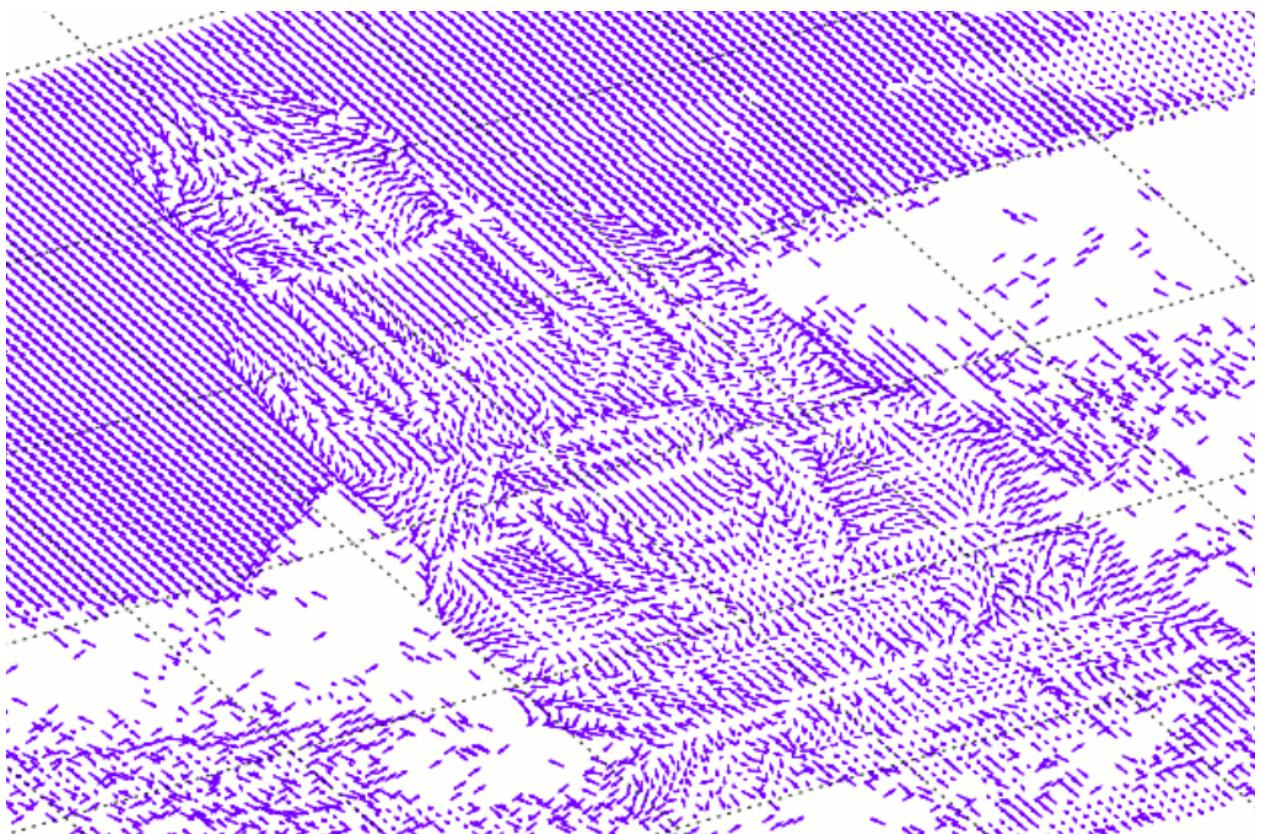


Figura 28: Cálculo de las normales, haciendo zoom en el buda, usando la combinación media de luces

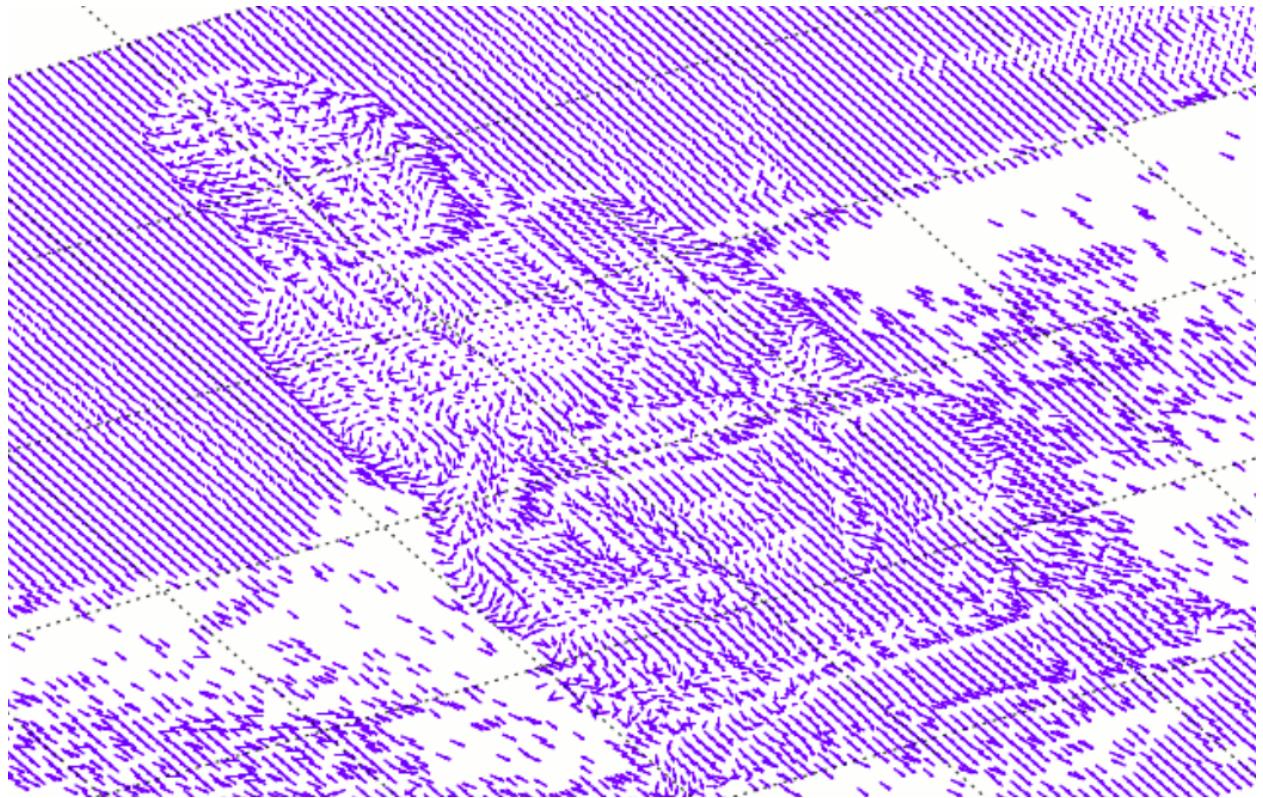


Figura 29: Cálculo de las normales, haciendo zoom en el buda, usando la peor combinación de luces

Referencias

- [1] David Kincaid E. Cheney. *Numerical Mathematics and Computing*. 1980.
- [2] UIT-R. *Recomendación BT.709-6*. URL: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-S.pdf.
- [3] R. J. Woodham. «Photometric method for determining surface orientation from multiple images». En: *Optical Engineerings* 19.1 (1980), págs. 139-144. DOI: <http://dx.doi.org/10.1117/12.7972479>.