UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Metodos Numericos

Segundo cuatrimestre de 2017

7 de septiembre de 2017

Fotometria Estereo

1. Introducción Teórica

1.1. Fundamentos de los métodos involucrados en el trabajo

El foco de este trabajo practico es la resolucion de sistemas de ecuaciones lineales. La idea principal de la resolucion de dichos sistemas es despejar todas sus variables. En este trabajo utilizamos multiples metodos numericos que simplifican la tarea de encontrar la solucion de estos sistemas. El metodo mas comun es el de eliminacion gaussiana para transormar al sistema de ecuaciones en otro equivalente(traingula el sistema) donde se pueden despejar las incognitas utilizando "backwards substitution". Estas tres operaciones basicas son multiplicar una fila por un escalar, sumar o restar un multiplo de una fila con otra y intercambiar filas. Siempre teniendo en cuenta que una fila no denota solo los coeficientes de las variables de una ecuacion sino tambien su termino independiente.

Utilizaremos la tecnica de fotometria estereo que nos permite medir la posicion y distancia de los objetos con respecto al sensor(camara), es decir, obtener las coordenadas x,y,z basandonos unicamente en las propiedades reflectivas de la luz, para objetos no especualares y en un ambiente con iluminacion controlada.

1.1.1. Intoduccion Teorica algoritmos utilizados

En el desarrollo se usan diferentes metodos para resolver sistemas lineales, algunos mejores que otros, siempre tratando de no perder presicion por errores de redondeo u otros incombenientes. Los algoritmos que usamos son el algoritmo de eliminacion gaussiana, factorizacion LU, factorizacion de cholesky u otras variantes de ellos. Un problema que no debe pasar desapercibido cuando trabajamos con operaciones aritmeticas en la computadora es el de perdida de digitos significativos. Esto ocurre porque la computadora trabaja con aritmetica de digitos finitos lo cual quiere decir que la representacion de los numeros en la computadora tienen una cantidad de digitos finitos para ser representados. La mejor forma que nosotros encontramos de acotar este tipo de errores es implementado el metodo de pivoteo parcial en nuestro algoritmo de eliminacion gaussiana.

1.1.2. Eliminacion Gaussiana

El metodo de eliminacion gaussiana tiene complejidad temporal $O(n^3)$ y cuando tenemos que resolver multiples sistema de ecuaciones lineales esta complejida no es tan buena. En la division de los pivotes, se selecciona el mas grande de entre las filas mirando en esa columna, asi al dividir por el numero mas grande el error del numero al que estoy dividiendo, no se ve tan reflejado en las cuentas ya que estoy achicando el error o no lo estoy agrandando demasiado.

1.1.3. factorizacion LU

Encontrar la factorizacion LU de una matriz tiene costo $O(n^3)$, la resolucion del sistema $Ax = LUx = b$ tiene costo $O(n^2)$, lo cual hace que en las situaciones donde hay que resolver muchas veces el sistema $Ax = b$ para diferentes b sea mucho mas eficiente que utilizar el algoritmo de eliminacion de gauss. Tomamos la decision de usar la factorizacion PLU, ya que nos permite pivotear y es muy similar. La existencia de la factorizacion LU no esta garantizada para cualquier matriz. Las condiciones necesarias y suficientes para que exista la factorizacion LU son que no aparezca un zero como pivote a lo largo de la factorizacion LU o que todas las submatrices principales sean inversibles. Aunque la factorizacion LU no siempre existe, la factorizacion PLU si.

1.1.4. Factorizacion Cholesky

Esta factorizacion se usa al obtener la matriz de pixeles para buscar las profundidades, y reescribir las ecuaciones para que la nueva matriz sea una matriz definida positiva, asegurando que existe la factorizacion cholesky para la matriz con la cual podemos trabajar ya que la solucion original al sistema no cambia.

1.1.5. Matrices Banda

En la construccion de la matrix de pixeles(para armar el espacio normal de ecuaciones), se puede armar la matriz de tal forma que quede una matriz banda, la cual hace que la matriz tenga muchos ceros y poder optimizar algunos algoritmos en funcion de eso.

1.1.6. Matrices Esparzas

Para la reconstruccion 3D, como las matrices con las que trabajamos son banda y esparzas podemos utilizar una estructura en la cual la matriz considere a los elementos que no son cero y asi disminuir la complejidad tanto temporal y espacial de los algoritmos

1.1.7. Definiciones

Una matriz es simetrica y definida positiva ¡=¿tiene factorizacion cholesky

2. Desarrollo

Durante el desarrollo se fue observando que al trabajar con aritmetica finita se presenta la posibilidad de errores de redondeo en varias oportunidades por este motivo se decidio utilizar double's para la una mejor representacion de los numeros. Con lo cual se decidio utilizar una cota para este error (llamada epsilon) la cual es un numero pequenio que se utiliza para la comparacion con el cero. en otros de los lugares donde tuvimos que considerar este error fue al momento de triangular las matrices en donde para cada columna se buscaba en que fila se encontraba mayor valor, asi al restarle a la siguientes filas un multiplo de la mayor se minimiza el error de redondeo

El programa desarrollado se puede dividir en dos partes: calibracion del sistema y reconstruccion del modelo 3D. La primera parte consiste en conseguir las direcciones de fuentes de luz y elegir de estas las que nos permitan hacer una reconstruccion optima de los modelos. En la segunda parte utilizando las direcciones obtenidas durante la calibracion, se estiman las dirreciones normales de la superficie del modelo y con estas se obtienen las profundidades

2.0.1. Calibracion

En el desarrollo de este trabajo practico lo primero que hicimos fue la calibracion del sistema. Lo cual consistio en encontrar las coordenadas del punto mas brillante de 12 imagenes de una esfera. Luego utilizamos la coordenadas del punto mas brillante junto con la ecuacion de la esfera para poder encontrar la direccion de la luz incidente a ella.

Para encontrar el punto mas brillante, nuestra primer idea fue analizar uno por uno todos los pixeles de cada imagen hasta encontrar el pixel con la máxima intensidad en el rango de 1 a 255, pero nos dimos cuenta que esta no era la mejor estrategia. El problema que surgió al tratar de seleccionar el pixel mas brillante de esta manera es que había mas de un pixel con el máximo valor de brillo en una imagen y en estos casos decidimos elegir uno de todos los pixeles con mayor intensidad al azar. Esto llevo a que en muchos casos eligiésemos mal el puntos mas brillante. Lo que no nos habíamos dado cuenta al principio fue que las imágenes discretisan el espacio y esto es lo que lleva que aya mad de un pixel que tengan la máxima intensidad de la imagen, pero esto no necesariamente quiere decir que la normal de todos estos puntos sean la dirección de la luz. Ademas no es muy lógico elegir el punto mas brillante al azar ya que este representa la dirección de la iluminación la cual es unica. Luego pensamos en que el brillo forma una aureola y que la esfera tiene algunas irregularidades sobre su superficie, entonces no alcanza con buscar el pixel mas brillante, por lo tanto se utilizamos las vecindades para lograr agrandar el espectro. Una vez seleccionadas las coordenadas del punto mas brillante de una imagen utilizamos la ecuación de la esfera, $r^2=(x-x_0)^2 + (y-y_0)^2+ (z-z_0)^2$, para obtener las direcciones de la iluminación. De esta ecuacion los parametros que conocemos son las coordenadas (x_0,y_0) del centro de la esfera, el radio r y las coordenadas del pixel que consideramos mas brillante (x, y). Si C es el centro de la esfera y P el punto donde se encuentra el pixel mas brillante, entonces el vector que representa la dirección de la iluminación es C-P = (x_0 - x, y_0 - y, z_0 - ($\sqrt{r^2-(x-x_0)^2-(y-y_0)^2+z_0}$)) = ($x_0$ - x, y_0 - y, $\sqrt{r^2-(x-x_0)^2-(y-y_0)^2}$).

2.0.2. Obtencion de direcciones de fuentes de luz

Dado que la superficie de los modelos es Lambertiana tiene la prodiedad de que absorbe la luz uniformemente en cada punto, con lo cual la normal en el punto con mayor intensidad de luz de la superficie corresponde con la direccion de la iluminacion. Usamos para la calibracion el modelo de la esfera porque al conocer su geometria y las intensidades en cualquier parte de su superficie podemos calcular las normales en todo punto

Para determinar la intensidad de un pixel utilizamos un promedio de los tres colores(azul, verde, rojo) dandole mas peso al verde dado que este es el color que el ojo humano percibe en mayor proporcion.

//Por tener la imagen con intensidades Discretas, no necesariamente habria una con el mayor valor.

Nuestra primera estrategia para encontrar el punto con mayor intensidad de luz fue recorrer los pixeles de las imagenes uno por uno y guardar la posicion del de mayor intensidad. El problema que tuvimos es que como la esfera no es de color uniforme, pueden haber puntos que sean mas brillantes por tener un tono de color mas claro y no por ser el punto con mayor intensidad luz.

La segunda estrategia fue considerar vecindades fijas que es un poco mas simple que tomar vecindades que se pudieran contraer o expandir(ya que en los bordes esto podria fallar), obteniendo una mejor aproximacion al pixel buscado.

2.0.3. Eleccion de las direcciones de fuente de luz

Para eleccion de las direcciones de fuente de luz: Lo primero fue hacer las diferentes permutaciones de tres direcciones de luz sin tomar elementos repetidos y aplicar el algoritmo de la eliminacion de gauss para ver si la matriz es inversible o no. Una vez corroborado que la matriz fuera inversible, mediante el numero de condicion asociado a la matriz podemos discriminar entre las matrices para una mejor estimacion del valor, mirando el numero de condicion que esta relacionado con la estabilidad numerica de un sistema lineal ya que una matriz mal condicionada es propensa a tener graves errores y no darnos soluciones del sistema en las cuales podemos confiar. Nosotros supones que el error que vamos a obtener al calcular las normales va a estar directamente relacionado al condicionamiento de la matriz que utilizamos. Dicho de otra manera, el mejor gráfico de las normales va a estar dado por la matriz mejor condicionada. Para corroborar esto buscamos la matrices con el máximo y mínimo grado de condicionamiento y comparamos visualmente las diferencias entre los campos normales que obtuvimos utilizando cada una.Con eso calculamos el numero de condicion de cada matriz, para luego tomar los maximos y minimos para poder comparar los resultados. Una vez obtenidos las direcciones de iluminacion, el siguiente paso del desarrollo fue elegir las tres direcciones de luz para después obtener las normales. Nuestra idea fue elegir aquella combinacion de direcciones que tengan el menor número de condición. Para hacer esto programamos un algoritmo que calcula todas las posibles permutaciones de tres vectores, teniendo 12 vectores para elegir, junto con el numero de condición asociado a cada una de estas matrices. Hay que tener cuenta que al calcular las posibles permutaciones de las direcciones consideramos a una elección de tres vectores como un conjunto, ya que el numero de condicion no cambia si permutamos las columnas o filas de una matriz. El condicionamiento de una matriz, dado por su numero de condicion, nos pareció la mejor forma de escoger las direcciones ya que, por lo que aprendimos en clase una matriz mal condicionada es propensa a tener graves errores y no darnos soluciones del sistema en las cuales podemos confiar. Pensando en el numero de condiciones en términos geométricos, también nos dimos cuenta de que el condicionamiento de una matriz va a mejorar a medida que las ecuaciones lineales de la matriz esten lo mas lejano posible de ser paralelas, lo cual tiene sentido ya que tener iluminación dispersa ayuda a ver un objeto mejor en el mundo real. Nosotros supones que el error que vamos a obtener al calcular las normales va a estar directamente relacionado al condicionamiento de la matriz que utilizamos. Dicho de otra manera, el mejor gráfico de las normales va a estar dado por la matriz mejor condicionada. Para corroborar esto buscamos la matrices con el máximo y mínimo grado de condicionamiento y comparamos visualmente las diferencias entre los campos normales que obtuvimos utilizando cada una.

El procedimiento lo hicimos en C++. Para conseguir todas la conbinaciones posibles de las direcciones de luz pusimos en una lista todas las filas (direcciones de luz) sin ninguna repetición. A esta lista se la recorre con 3 indices, los cuales nunca comparten el mismo valor y nunca se cruzan por lo tanto no se crean nuevas matrices con filas conmutadas, ni matrices con filas idénticas. Al mismo tiempo se le calcula el numero de condición correspondiente a cada matriz y se guarda solamente la que posee el numero de condición menor.

Con las tres direcciones de iluminación ya elegidas el siguiente paso del desarrollo es el calculo de las normales en cada pixel. Esta es la etapa del trabajo practico en la cual implementamos y utilizamos los algoritmos de eliminación de gauss y la factorizacion LU.

Nuestro algoritmo de eliminación gaussiana, hace n iteraciones donde n es la cantidad de filas de la matriz a triangular. En la i-esima iteración toma el elemento a_{ii} como pivote. Si el pivote en cualquiera de las iteraciones es zero entonces no existe factorizacion LU pura, aunque si existe factorizacion PLU. Si el pivote no es cero el algoritmo hace otras k iteraciones donde k es la cantidad de filas que tiene por debajo. El algoritmo procede a calcular el múltiplo exacto por el cual tiene que multiplicar al pivote para poder eliminar al elemento, que este en la misma columna que el, y en la fila debajo si. Esto lo repite k veces, una ves para cada fila que se encuentra por debajo del pivote actual. Una vez terminadas las k iteraciones para el i-esimo pivote, el algoritmo incrementa en 1 a i y pasa al siguiente pivote. Resulta evidente que los pivotes va a ser todos los elementos de la diagonal.

Obtener la factorizacion LU no es mas que guardar los múltiplos que utilizamos para diagonalizar una matriz en el algoritmo de eliminacion gaussiana. Es importante notar que nosotros implementamos un algoritmo de factorizacion PLU donde la factorizacion LU es el caso especifico donde la matriz P es la matriz identidad. Por lo cual, la complejidad temporal de obtener la factorizacion LU es igual a la complejidad del algoritmo de eliminacion de gauss, la cual es $O((n)^3)$.

Aunque sus complejidades son iguales, la diferencia se encuentra al momento de resolver varios sistemas. La eliminación gaussiana va a tener la misma complejidad cada vez que la utilizamos. En contraste, la factorización LU solo se obtiene una vez y luego solo se debe resolver dos sistemas triangulares, esto lleva a que su complejidad es $O((n)^2)$. Como dice el enunciado del TP, los valores sx,sy,sz, de la ecuación 5 no cambian pixel a pixel sino que solamente se modifica el valor de la intensidad en (x,y) para la imagen correspondiente Ii, y por lo tanto poseen la misma matriz pero con distinto término independiente. Por lo cual, si se encuentra una convinación de s_1, s_2 y s_3 que posea factorización LU se podria bajar sustancialmente la complejidad de obtener las normales de una imagen, ya que estas poseen una gran cantidad de pixeles.

Nuestra primer idea para tratar de comprobar nuestra hipótesis de que la factorizacion LU es mas rápida que la eliminación de gauss fue tomar distintos tamaños de matrices y trangularlos una vez con cada algoritmo para ver que método era mas rápido. Este experimento no pone a prueba nuestra hipótesis porque lo que intentábamos comprobar es que la factorizacion LU es mas rápida que la eliminación de gauss para resolver un sistema lineal despues de la primera vez que se resuelve el sistema. Si resolvíamos una sola vez cada tamaño de matriz, ambos algoritmos tardarían lo mismo. Por lo tanto, decidimos hacer un experimento donde mantuvimos constante el tamaño del sistema lineal a resolver y lo que variamos en cada instancia de teste fueron las cantidades de términos independientes. Por ejemplo, la primera instancia resolvimos el sistema Ax=b con 100 b diferentes y luego incrementamos la cantidad de términos independientes de 100 en 100, hasta llegar a 1000. Es

importante remarcar que estos términos independientes fueron generados aleatoriamente con una función encontrada en la siguiente pagina:

2.1. Construccion de la matriz M

Dadas las ecuaciones 11 y 12, para generar el sistema de ecuaciones para todos los pixeles para poder generar la matriz banda.

2.1.1. Armando la matriz M

La segunda forma (naiv), para la cual tomamos la misma relacion anterior, es ver la grilla de pixeles en forma de F1++F2++...++Fn concatenacion de los pixeles (donde F1 contiene a todos los pixeles correspondiente a todas las columnas de la fila 1 de la grilla, analogamente Fn). con los cual, si separamos las ecuaciones y formamos 2 matrices una para las ecuaciones correspondientes al eje X y otra matriz correspondiente al eje Y. Tenemos una Matriz Banda 1x2 Para el eje X y 1x(cantidad De Columnas de las grilla +1) para el eje Y. Combinandolos de manera adecuada, se obtiene la matriz banda(anhelada).(X1/Y1/.../Xn/Yn)

MxMt es simetrica, es valido para toda matriz.

2.1.2. Descripcion de la matriz M

La matriz M tiene dimension 2NxN y es banda PxQ, por lo tanto Mt es de dimension Nx2N y es banda QxP, luego siendo A = MxMt tiene dimension NxN y es banda Qx(Q+1). Ya que como esta armada la matriz M, se forma una escalera donde al moverse por las columnas,mirando desde la q+1 columna, la primer coordenada tiene el primer elemento igual a cero y todas los demas no necesariamente distinto de cero.Al mirar la siguiente columna la cantidad de elementos que no son cero incrementan en dos y asi cada vez que se mira la siguiente columna. Luego al mirar las filas, se logra ver un patron en el cual se agrega un cero a las primeras coordenadas de la fila cuando se dan pasos de a dos filas. Lo cual forma un patron interesante al ser multiplicado por su transpuesta, esto relaciona la cantidad de pixeles con el ancho de banda de la matriz el cual queda dependiendo de la cantidad de pixeles y del ancho de la imagen.

2.1.3. Mirando ceros de la parte superior

Para la matriz A resultante al juntar lo visto, y siguiendo la misma idea con la matriz transpuesta, se observan los mismos patrones.Por lo tanto al hacer la multiplicacion, se puede observar un patron que siguen cada una de las filas de la matriz resultante , en particular la fila i(i ¡=Q) tiene sus coordenadas igual a cero desde la posicion Q+i+1, eso forma una diagonal de elementos no necesariamente igual a cero a partir de la fila 1 columna q.

2.1.4. Mirando los ceros de la parte inferior

Para la fila i(i ¿Q), tiene el primer cero a partir de la fila q+1 en la coordenada 1 y va incrementando un cero a las primeras coordenada del vector fila cada ves que avanza, con lo cual eso forma una diagonal de elementos no necesariamente igual a cero a partir de la fila q columna 1.

2.1.5. Forma de la matriz resultante

Para obtener la matriz A que da la multiplicacion de la matriz MxMt, dado que son matrices banda y que M es banda PxQ donde p=cantidad de elementos(pixeles)-1 y Q = cantidad de columnas de la imagen. Si la imagen es una tira de pixeles entonces P = Q, sino P ¿Q. La matriz resultante tiene dimension NxN y es banda Qx(Q+1).

5:Dudas. ¿Porque no importa la cantidad de pixeles a la hora de ver la anchura de la matriz?.

2.1.6. Justificacion de utilizacion de la factorizacion de cholesky Para la matriz A

Para poder justificar cholesky, nos alcanza con demostrar que las columnas de la matriz M son Li ya que por la definicion de matriz definida positiva, vale que para $w^tAw > 0 \ \forall$ w $\in \mathbb{R}^{nxn}$, la demostracion es por el contraresiproco con lo cual suponemos lo contrario existe un $(\exists w)Aw = 0 \wedge w \neq 0$, como A = $M^tM \rightarrow w^tM^t = (Mw)^t(Mw) = ||Mw||^2$ y como esto es la norma del vector, la norma de ese vector es 0, como demostramos que las columas eran Li, con lo cual llegamos a un absurdo que provino de suponer que existe tal w, por lo tanto la matriz A es definida positiva. Y usando la Propiedad*, tenemos que la matriz A tine factorizacion cholesky unica.

Propiedad*: Una matriz es definida positiva ¡=¿tiene factorizacion cholesky unica.

2.1.7. Implementacion

Para la estructura de representacion de la clase matriz, se utilizo la estructura arreglo de arreglo. Para la parte de implementacion, se creo una clase Matriz y una clase Matriz-Esparza y todas las operaciones necesarias para el desarrollo del los experimentos. En los algoritmos se trato de aprobechar tanto como fuera posible los patrones o irregularidades que se observaban tanto en la matriz M , como la matriz A(re sultante de M^tM) y asi poder optimizar el tiempo y la estructura

que se usa. La estructura que se usa, permite que la transposicion de una matriz se realize en O(1). la variable cols guarda el orden de las columnas, de tal manera que permutar una fila sea en O(1).

3. Resultados

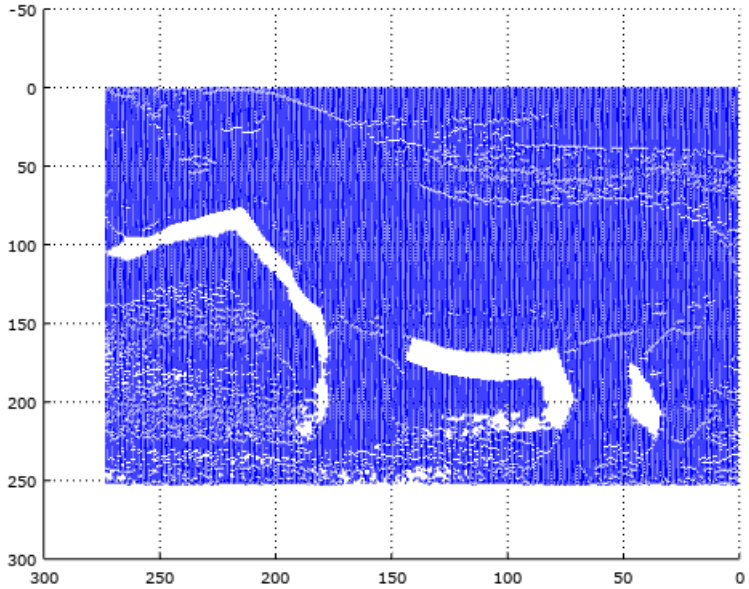
Hicimos la elección de direcciones de iluminación con el procedimiento explicado en la parte de desarrollo y obtuvimos la siguiente matriz, que tiene el numero de condicion 15.0503:

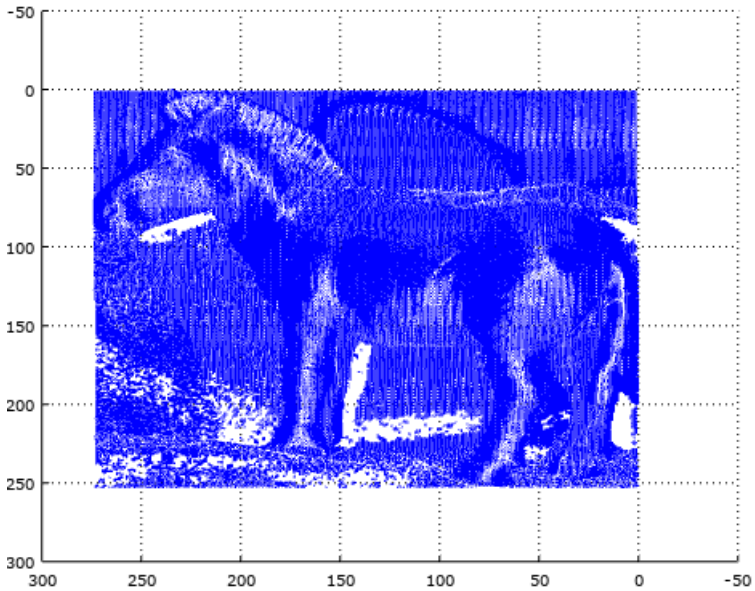
$$\begin{pmatrix} 0{,}15625000 & -0{,}59765625 & 0{,}78637964 \\ -0{,}11718750 & -0{,}57421875 & 0{,}81027151 \\ 0{,}03515625 & -0{,}44140625 & 0{,}89661840 \end{pmatrix}$$

y la siguiente matriz que tiene el máximo numero de condición, 407,581:

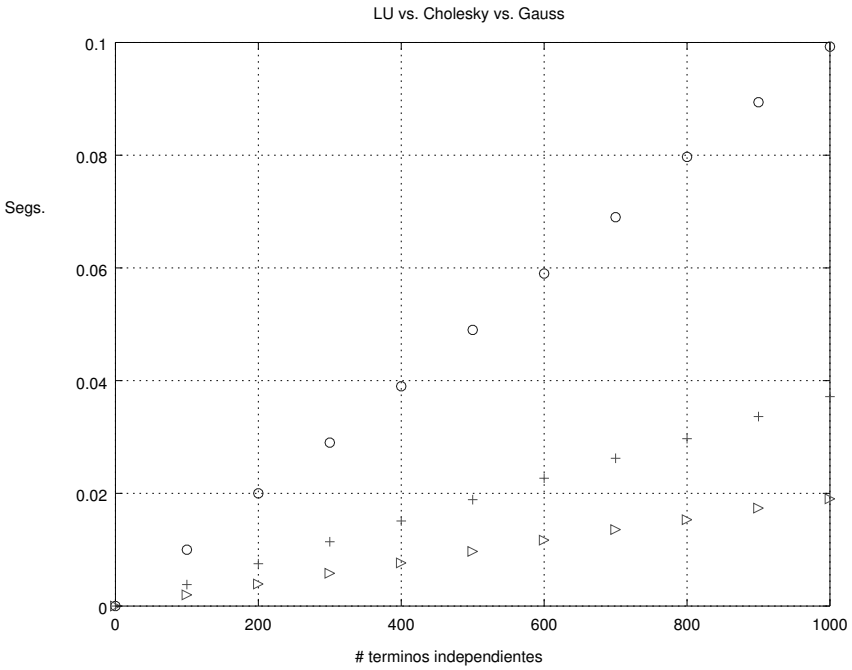
$$\begin{pmatrix} -0{,}03906250 & -0{,}56250000 & 0{,}82587400 \\ -0{,}04296875 & -0{,}56250000 & 0{,}82567998 \\ -0{,}04687500 & -0{,}56250000 & 0{,}82546743 \end{pmatrix}$$

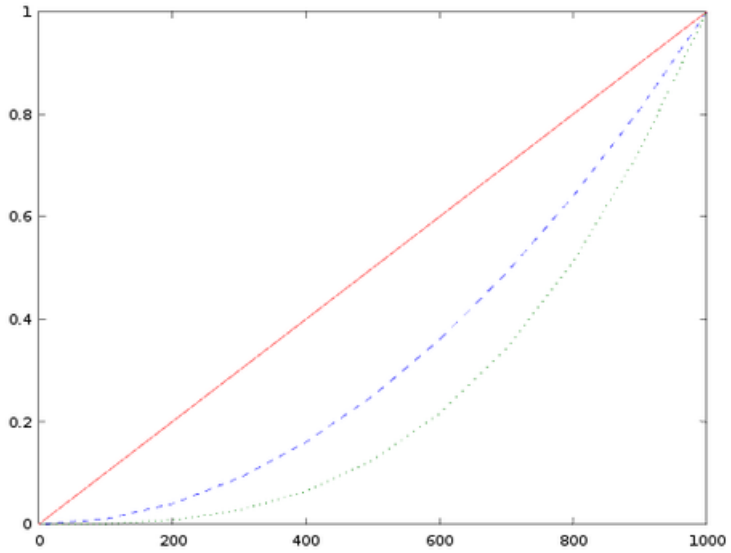
Luego generamos los campos de las normales y obtuvimos los siguientes resultados:



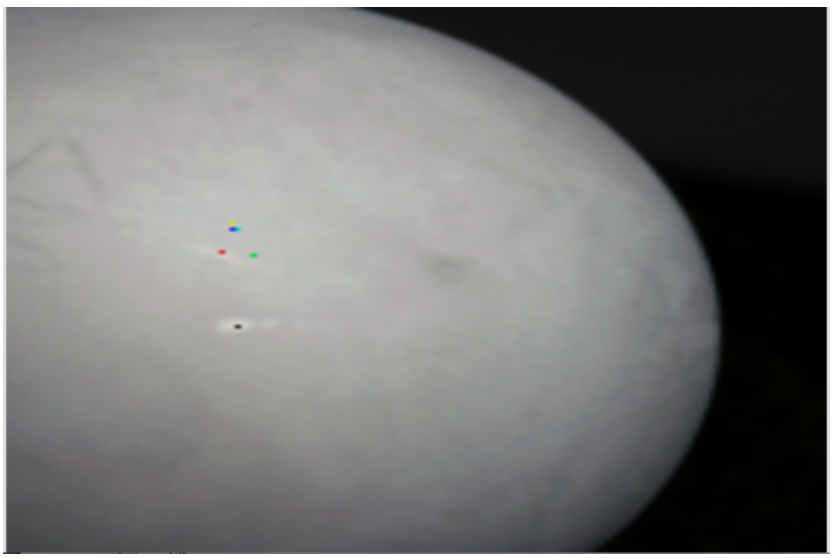


El siguiente grafico muestra los resultados que obtuvimos en e experimento donde comparamos las complejidades temporales del algoritmo de eliminacion de guass, la facotrizacion LU y Chelosky.

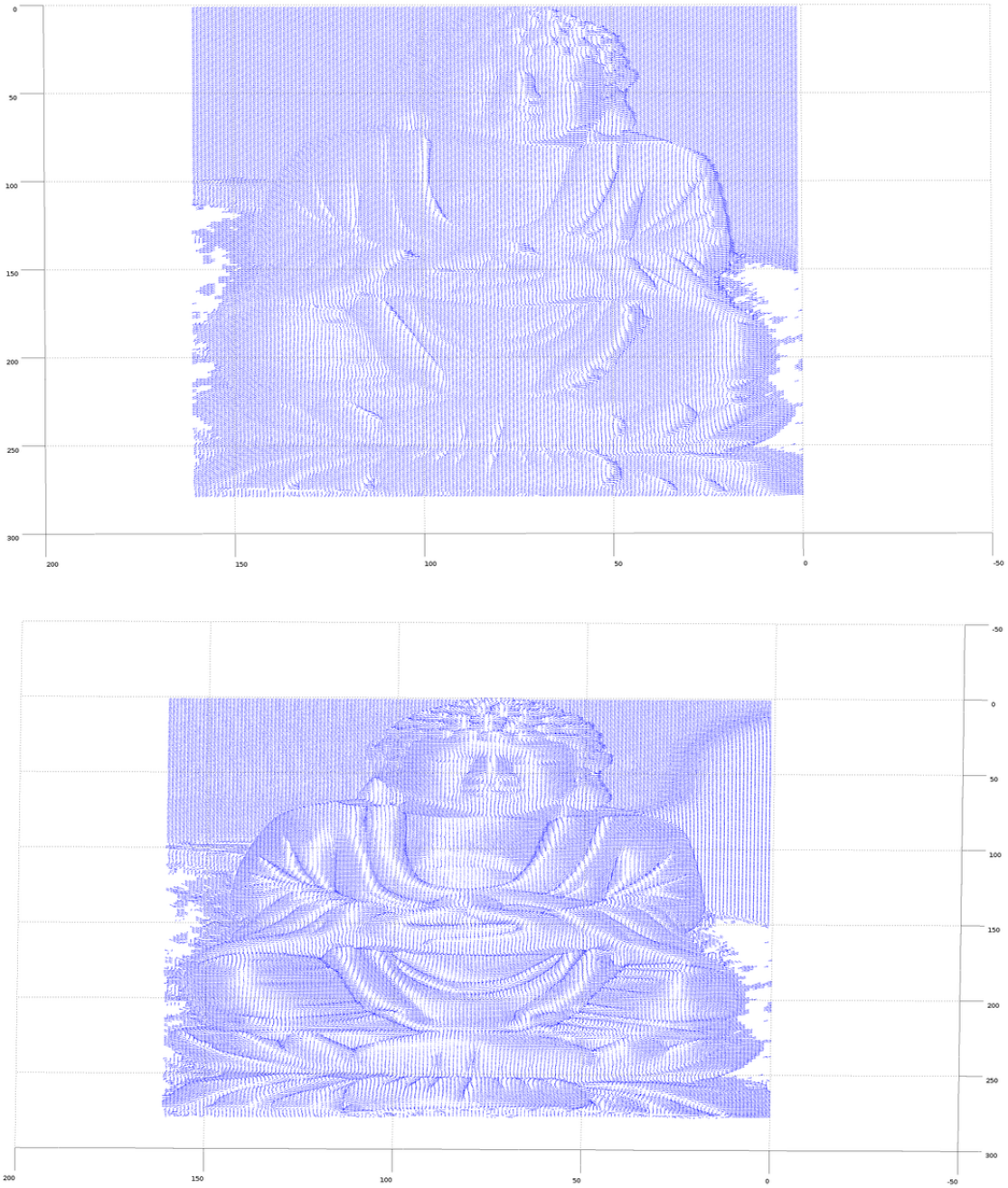




La imagen mate.6 tiene marcada con puntos de diferentes colores las siguientes vecindades: 0(negro), 2(rojo), 4(verde), 6(azul), 8(amarillo) y 10(cian).



Los siguientes campos pertenecen al resultado de calibrar con vecindad cero y vecindad 6.



Nuestra hipótesis de que el numero de condición afecta directamente la estimación de las normales se puede ver confirmada claramente en las imágen del caballo. El campo de normales del caballo en las figuras 1 esta estimado con la elección de direcciones de iluminación que causan que la matriz de la ecuación 5 tanga el máximo numero de condición en comparación a todas las otras posibles combinaciones de direcciones. En contraste la figura 2 utiliza la conbinación de direcciones de illuminacion que generan la matriz con el mejor numero de condición. Esto se da porque un numero de condición alto implica que las direcciones de iluminación están apuntando desde una posición muy similar lo cual es equivalente a tener una sola direccion para generar las normales y también las profundidades.

4. Discusión

4.1. Calibración

La imÃ¡gen en la cual vimos la mayor diferencia es en Mate.6, dónde se observan esos factores de los que hablamos en el desarrollo, entre aquellos, las manchas de distinto color y la aureola de luz. Pór ejemplo se ve el de vecindad 0 (púnto negro) bién lejos de los demas, por lo claro de la mancha blanca. Las vecindades de 2 y 4 (rojo y verde) también fueron corrompidos por el ruido, aunque llegaron un poco mas cerca. Luego a partir de los 6, 8 y 10 (azúl, amaríllo y cian) se podria creer que llega a un lugar en donde por más que agrandes el número de cantidad de vecindades no modificaria la posición, también viéndolo a simple vista se puede corroborar que es de dónde uno diria que proviene la luz. Luego en el próximo experimento

se observa que al modificar la cantidad de vecindades repercute fuertemente en la realizacion del resto del modelo, ya que con una mala dirección de la luz se perdera el efecto 3D.

4.2. Complejidades

Nuestra hipotesís, y a partir de lo que nos comentaron en las clases era que los algoritmos de eliminación gaussiana, factorización LU y Cholesky iban a tener un orden de complejidad cúbico y cuadrado, respectivamente, lo que al ver los resultados de las experimentaciones nos dice lo contrario. Las complejidades nos resultaron lineales, lo cual al pensarlo un momento se nos ocurrio que lo que puede estar pasando es debido a que la pendiente crece poco, a lo que deberíamos utilizar una cantidad mayór de términos independientes. Esto último no lo pudimos comprobar por falta de tiempo.

4.3. Número de condición

Nuestra hipótesis de que el número de condición afecta directamente la estimación de las normales se puede ver confirmada claramente en las imágen del caballo. El campo de normales del caballo en las figuras 1 esta estimado con la elección de direcciones de iluminación que causan que la matriz de la ecuación 5 tenga el máximo número de condición en comparación a todas las otras posibles combinaciones de direcciones. En contraste la figura 2 utiliza la conbinación de direcciones de iluminación que generan la matriz con el mejor número de condición. Esto se da porque un número de condición alto implica que las direcciones de iluminación están apuntando desde una posición muy similar lo cual es equivalente a tener una sola dirección para generar las normales y también las profundidades.

4.4. RGB

La visión humana tiende a diferenciar más escalas de verde lo cual hace que al hacer un promedio ponderado para el color verde se obtengan mejores resultados. Esto lo tomamos en cuenta al momento de promediar los colores de los pixeles para buscar la dirección proveniente de la luz. Al no poder encontrar una gran diferencia, decidimos utilizarlo.

5. Conclusiones

Las conclusiones finales de este trábajo desde principio a fin son las siguientes:

Al momento de la calibración de la imágen es fundamental tomar bien las medidas, ya que si se arrastra este error, a medida que se le aplican los metodos se potencia.

Se debe seleccionar de buena manera las direcciones de luz para eliminar las mayores posibilidades de tener algún error de redondeo al medir las normales.

Al momento de calcular las profundidades se debe ser inteligente y poder rebuscarse para no consumir tanta memoria, ya que se trabajan con matrices demasiado grandes para tenerlas en memoria.

6. Referencias

1-Numerical Analysis - 9th Edition - (Burden and Faires)