DEPARTAMENTO DE COMPUTACION

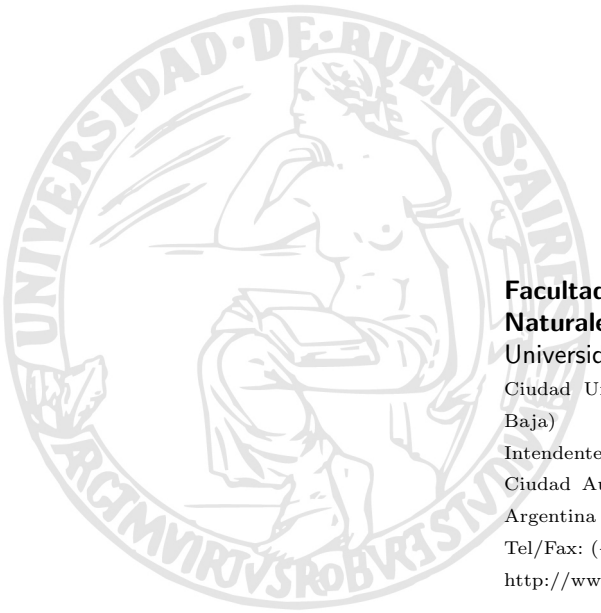
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 3 - Tomografía computada

1 de julio de 2018 — Métodos Numéricos

Grupo "Greco-Herrera-Kubrak-Salvador"

Integrante	LU	Correo electrónico
Cristian, Kubrak	456/15	Kubrakcristian@gmail.com
Marcela Alejandra, Herrera	1162/84	marcelaalejandraherrera@yahoo.com.ar
Luis Fernando, Greco	150/15	luifergreco@gmail.com
Alejo, Salvador	467/15	alejo.antonio.salvador@gmail.com



Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Abstract: El objetivo del Trabajo es realizar la simulación de una tomografía computada partiendo de imágenes tomográficas reales en formato 16 bits. La misma se realiza generando una simulación de rayos X que atraviesan la imagen, la cual es discretizada en una cuadrícula de tamaño parametrizable. Una vez generados los rayos se realiza la reconstrucción de la imagen utilizando el método de cuadrados mínimos lineales, luego se realiza una evaluación de los resultados obtenidos usando error cuadrático medio y la reconstrucción de la imagen en formato pgm de 8 bits.

Palabras clave: reconstrucción de imagen tomográfica - cuadrados mínimos lineales - error cuadrático medio

Índice

1. Introducción

Introducción

El objetivo de este trabajo práctico es evaluar un método para reconstruir imágenes tomográficas sujetas a ruido, utilizando el método de aproximación por cuadrados mínimos.

En la toma de una tomografía se emiten rayos X que atraviesan al sujeto en estudio y se realizan numerosas mediciones sobre los mismos para compensar los errores de medición que pudieran producirse. Este proceso conduce a un sistema de ecuaciones lineales sobredeterminado que por general no tiene solución, razón por la cual se utiliza el método de cuadrados mínimos para obtener un resultado aproximado. En nuestro caso para simplificar el problema vamos a suponer que las mediciones que se realizan son de los tiempos que le lleva a cada rayo atravesar el sujeto.

Estos se emiten de tal forma que atraviesan al sujeto en un plano dado tomando distintos ángulos y diferentes direcciones sobre ese plano.

La superficie del corte a estudiar se discretiza dividiéndola en celdas de acuerdo a una cuadrícula de $n*n$, siendo n un número entero. Cada celda puede ser atravesada por múltiples rayos, de hecho esto es recomendable para poder recolectar una cantidad de datos representativa que nos permita compensar errores numéricos. A continuación, con los datos recolectados se plantea un sistema de ecuaciones lineales donde los coeficientes están dados por la distancia que recorre cada rayo dentro de cada una de las celdas de la cuadrícula, el vector independiente viene dado por los tiempos que demora cada rayo en atravesar al sujeto y las incógnitas son las velocidades de los rayos dentro de cada celda. Estas velocidades son una propiedad de la materia atravesada.

Si llamamos D a la matriz de distancias por celda, t al vector de tiempos y v al vector incógnita, el sistema a resolver sería:

$$Dt = v$$

Luego como este sistema en general es incompatible, para resolver por cuadrados mínimos se utilizan las ecuaciones normales:

$$D^t Dt = D^t v$$

Centrándonos en nuestro caso, debido a que no es posible la utilización de un tomógrafo real para recolectar los datos, utilizamos imágenes tomográficas preexistentes (provistas por la cátedra). Es a partir de estas imágenes que generamos las instancias de prueba que se convierten en nuestro "sujeto" en estudio tal como se detalla más abajo.

Usamos las imágenes de partida como si fuera el sujeto al que necesitamos estudiar y simulamos matemáticamente los rayos X, agregando ruido aleatorio para simular los errores de medición. Para manejar el ruido decidimos calcular los tiempos de recorrida de cada rayo simulado a través de la imagen sin ruido (vector v), y luego perturbar los valores obtenidos aplicando ruido gaussiano multiplicativo, de modo que para cada elemento de v obtenemos un valor $v1 = v + v * \alpha * x$ con x un valor aleatorio con distribución $\mathcal{N}(0, 1)$.

Para evaluar la calidad de la reconstrucción de la imagen tomográfica utilizamos como métrica el cálculo del Error Cuadrático Medio, el cual compara

las velocidades de cada celda obtenidas a partir de las imágenes sin ruido contra las velocidades calculadas por medio de cuadrados mínimos. Como velocidad "verdadera" de cada celda se toma un promedio de la suma de los valores de intensidad de los pixeles pertenecientes a esa celda. De esta forma el error cuadrático medio se calcula según la siguiente fórmula:

$$ECM = (\sum_{k=1}^{n*n}(v^1[i] - v[i])^2)/n * n$$

La granularidad de la discretización (o sea la cantidad de pixeles por celda), la cantidad, puntos de partida y dirección de los rayos así como el nivel de ruido utilizado son parámetros de experimentación. En su determinación tenemos en cuenta un compromiso entre la calidad de la imagen reconstruída y los tiempos de ejecución, principalmente dominados por la resolución del sistema por cuadrados mínimos. Respecto de este último punto, a menor tamaño de celda se corresponde un mayor tamaño de la matriz. Por ejemplo una discretización de $n * n$ celdas, genera un sistema de ecuaciones de dimensión $n^2 * n^2$. Resolver el sistema, en nuestro caso utilizando eliminación gaussiana, nos conduce a una complejidad del orden de $\mathcal{O}(n^2 * n^2)^3 = \mathcal{O}(n^6)$, con lo cual el tiempo de procesamiento se vuelve crítico para las pruebas.

2. Desarrollo

Desarrollo

Para realizar la experimentación utilizamos la imagen tomo.png provista por la cátedra, con una resolución de 100 x 100 pixeles.

La imagen la trabajamos en formato .csv para la entrada con una profundidad de 16 bits y luego de la reconstrucción la guardamos en formato pgm de 8 bits.

Para la conversión de la imagen de 16 a 8 bits implementamos una función (to8bits) que identifica los valores de intensidad máximo y mínimo de la imagen a guardar. Al valor mínimo se le asocia el valor de salida 0 y al máximo el valor de salida 255. Los valores de intensidades intermedias se distribuyen de forma proporcional. Este procedimiento permite salvar el inconveniente de que como resultado de la eliminación gaussiana se pueden obtener valores negativos, los cuales no tienen sentido en este contexto ya que representan intensidades que siempre son valores mayores o iguales a cero, o valores mayores al máximo valor permitido para una imagen de 16 bits. Esto es posible porque estamos encontrando una solución aproximada y además estamos trabajando con ruido.

En un principio comenzamos utilizando los archivos .csv provistos por la cátedra de 512 x 512 pixeles, con estas imágenes realizamos varias pruebas fallidas que se explican más detalladamente en el apartado Generación de rayos. Debido a diversas dificultades que se nos presentaron con la elección de la estrategia para trazar los rayos, se demoró el inicio de las pruebas y optamos por utilizar imágenes más chicas para poder realizarlas en el tiempo de que disponíamos. Aunque hubiera sido interesante poder trabajar con un conjunto mayor de imágenes.

Para discretizar la imagen utilizamos valores divisores de 100. En los tests de granularidad, variamos el tamaño de las celdas tomando los valores 4x4, 5x5, 10x10, 20x20, 25x25 y 50x50 pixeles. Como ya se analizó a medida que se achica el tamaño de las celdas crece el tiempo de procesamiento, lo cual condicionó la elección de los parámetros.

Implementamos una función "pasa" que se encarga de identificar por qué pixeles pasa un rayo (los rayos son considerados como rectas en el plano, caracterizados por un punto de origen que llamamos (x_0, y_0) y un ángulo α).

Para poder describir esta función adecuadamente imaginamos la imagen ubicada en un eje de coordenadas de modo tal que su vértice inferior izquierdo se encuentre en la coordenada (0,0). Pensamos a la imagen como un subconjunto de puntos del plano.

Utilizando la tangente de α (que es la pendiente de la recta asociada al rayo), se busca la intersección con la recta $y = 0$ la cual es el punto $(x - y/tangent(\alpha), 0)$. De ahi en mas se utiliza la ecuación de la recta $y = tan(\alpha) * x + y_0$, variando el valor de x para calcular los puntos del plano (siempre hablando del plano en un sentido general) por los que pasa la recta. Si el punto hallado pertenece a la imagen se guardan las coordenadas del pixel correspondiente en un vector, para ser usadas posteriormente para el cálculo de distancias y de tiempos.

Esta función se usa para calcular el tiempo que le toma a un rayo atravesar al sujerto. Para esp se hace la suma de las intensidades de los pixeles por las que pasa el rayo ya que la intensidad de los mismos la asociamos al tiempo que

le toma a un rayo atravesar el mismo.

Finalmente se la reutilizará por ultima vez para calcular la cantidad de pixeles en cada celda por las que pasa el rayo. Para hacer esto se calcula para cada pixel por los que pasa el rayo a qué celda pertenece. Para esto es necesario realizar una conversión en las coordenadas, ya que las celdas de la imagen se identifican desde la esquina superior izquierda.

Debido a que los rayos pasan solamente por algunas celdas del total (Se puede ver gráficamente que si la grilla es de n celdas, a lo sumo serán $2 * n - 1$), para la matriz de distancias reutilizamos la estructura de matriz esparza del TP1.

Una de las decisiones que tuvimos que tomar fue de qué forma trazar los rayos simulados que íbamos a aplicar a la imagen.

En cuanto a la cantidad de rayos, la misma debe ser mayor que n^2, siendo n la cantidad de celdas de la discretización, ya que con una cantidad menor habría celdas que no son atravesadas por ningún rayo. En ese caso, hay columnas completas de ceros en la matriz de distancias, lo cual se traduce en filas de ceros en la matriz del sistema a resolver que como ya explicamos en la introducción se obtiene como $D^t Dt = D^t v$. Esta situación se corresponde con un sistema con infinitas soluciones.

En cuanto a la ubicación de los emisores, inicialmente pensamos en dos opciones que resultaron fallidas:

1. Trazar rayos horizontales y verticales, formando una cuadrícula.

2. Trazar rayos saliendo de los cuatro vértices de la imagen en distintos direcciones para barrer los 90° de cada ángulo.

En ambos casos, al comenzar la experimentación nos encontramos con dificultades.

En el primer caso al realizar la eliminación gaussiana nos encontrábamos con que la matriz tenía filas completas en cero (matriz no inversible).

En el segundo caso, si bien se ejecutaban todos los pasos del programa las imágenes reconstruídas no eran reconocibles.

Al realizar un análisis de estos resultados nos dimos cuenta que los rayos trazados de estas formas eran muy similares entre sí y en consecuencia no suministraban información sufuciente para realizar una buena aproximación a la verdadera solución del sistema. Igualmente adjuntamos la implementación en el código a título informativo.

Finalmente decidimos trazar rayos desde los cuatro laterales de la imagen. Los emisores se ubican en igual cantidad sobre los lados de la imagen, la ubicación de los emisores se selecciona aleatoriamente en base a una distribución uniforme entre 0 y k, siendo k la cantidad de pixeles por lado de la imagen (es decir suponiendo que la imagen tiene $k * k$ pixeles).

De cada uno de estos emisores salen igual cantidad de rayos en ángulos que van entre 0° a 180°. Los valores de los ángulos también se seleccionan aleatoriamente con distribución uniforme.

Si bien este método es aleatorio -con lo que los resultados difícilmente se repitan en corridas sucesivas- utilizando una cantidad suficientemente grande de rayos (como en este caso) por consecuencia de la Ley de los Grandes Números, en general los rayos quedan bien distribuídos a lo largo (y ancho) de la imagen, pudiendo obtener así resultados consistentes en varias corridas.

De los tres métodos este fue el que nos dio mejores resultados y el que terminamos eligiendo para realizar la experimentación.

Durante la experimentación, para una discretización de imagen de 10 celdas por lado (100 celdas en total), realizamos dos tipos de pruebas:

1. Sobre la cantidad de emisores, variando entre 20 y 140, aumentando de a 20 ($2 * n$ hasta $7 * 2 * n$). Sin aplicar ruido y manteniendo constante la cantidad de rayos por emisor en 100.

2. Sobre la cantidad de rayos por emisor, variando desde 50 hasta 150, aumentando de a 10. Sin aplicar ruido y manteniendo constante la cantidad de emisores en 100.

En la fig. se puede ver el trazado de rayos resultante tomando como base la imagen tomo.png en un tamaño de 25 x 25 pixeles, con una discretización en celdas de 5 x 5 pixeles, usando 25 emisores y 25 rayos por emisor.

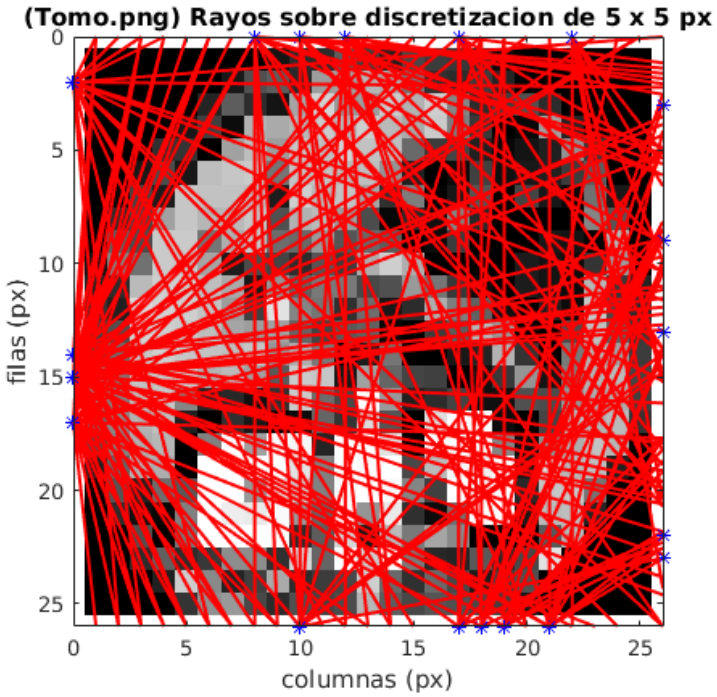


Figura 1: Trazado aleatorio de rayos

Para resolver el sistema de ecuaciones de cuadrados mínimos utilizamos las ecuaciones normales. Para esto reutilizamos la estructura de matriz esparza y el algoritmo de eliminación gaussiana del TP1, al cual modificamos para utilizar pivoteo de filas.

Durante el testeo implementamos la eliminación gaussiana utilizando una matriz completa pero no reportó ningún beneficio en el tiempo de procesamiento por lo cual descartamos la idea.

Como explicamos en la introducción utilizamos ruido gaussiano aleatorio multiplicativo sobre el vector de tiempos de recorrida de los rayos de manera que para cada elemento, el resultado luego de aplicar el ruido es $v1 = v + v * \alpha * x$ con x un valor aleatorio entre 0 y 1 con distribución normal. Durante la experimentación usamos como parámetro los valores $x * \alpha$ para $\alpha = 0.001$ y x tomando valores del conjunto en $\{1, 2, 3, 4, 5, 6, 10, 20, 50\}$

El α actúa como un regulador de ruido. Elegimos que sea multiplicativo (es decir, multiplicando el número aleatorio por el del vector en lugar de por una constante) para que sea consistente la aplicación del ruido. De otra forma, al tener valores en el vector al cuál le vamos a aplicar el ruido que se encuentran comprendidos en un rango muy amplio, si multiplicaramos por una constante estaríamos aplicando proporcionalmente demasiado ruido en algunos valores y muy poco en otros, lo que resultaría en un ruido que deformaría completamente algunas partes de la imagen y dejaría casi intactas otras, cosa que no nos es útil.

Otra alternativa que analizamos es aplicar el ruido directamente sobre los pixeles de la imagen, después de calcular los vectores de tiempos de recorrida y de velocidades exactos. Pero finalmente nos decidimos por la utilización del ruido sobre el vector de tiempos.

3. Resultados y discusion

En cuanto al ruido, para verificar lo expicado anteriormente acerca de nuestra elección de ruido multiplicativo hicimos los siguientes experimentos. Utilizamos una imagen de 100px x 100px, con $\alpha = 0.2$, dividida en 5 celdas y con 100 emisores de rayos, y 100 rayos emitidos desde cada uno de estos. Luego, sobre esto comparamos el vector de tiempos de los rayos sin ruido contra el mismo con ruido multiplicativo y aditivo.

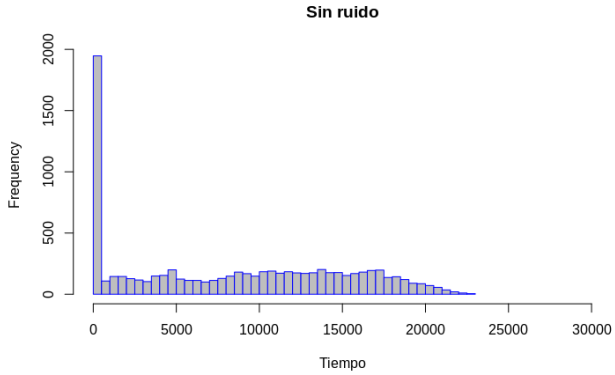


Figura 2: Vector de tiempos de rayos sin ruido

Acá podemos observar que efectivamente, el ruido aditivo modifíca mucho más algunas partes del vector mientras que otras no se aprecian muchos cambios. En este caso el histograma de ruido aditivo sufre cambios notorios respecto del vector sin ruido, mientras que en los valores más grandes casi que no notamos se notan diferencias.

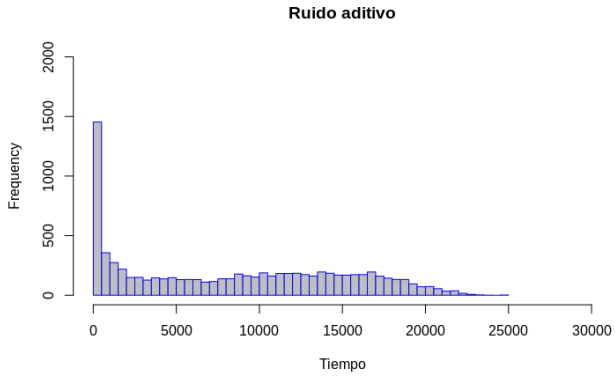


Figura 3: Vector de tiempos de rayos con ruido aditivo

En este otro caso en cambio, vemos que las diferencias entre el histograma de ruido multiplicativo y sin ruido existen pero están más uniformemente distribuídas. La única anormalidad que observamos es un incremento en los valores

máximos, no obstante se trata de unos pocos casos aislados. Esto se parece más a lo que esperabamos que hiciera nuestro ruido, por eso fue que elegimos usar este método en nuestra implementación.

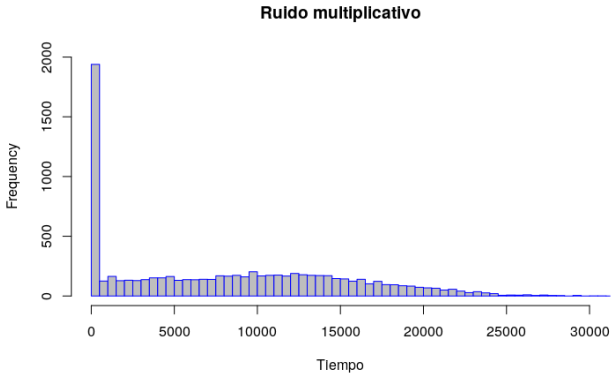


Figura 4: Vector de tiempos de rayos con ruido multiplicativo

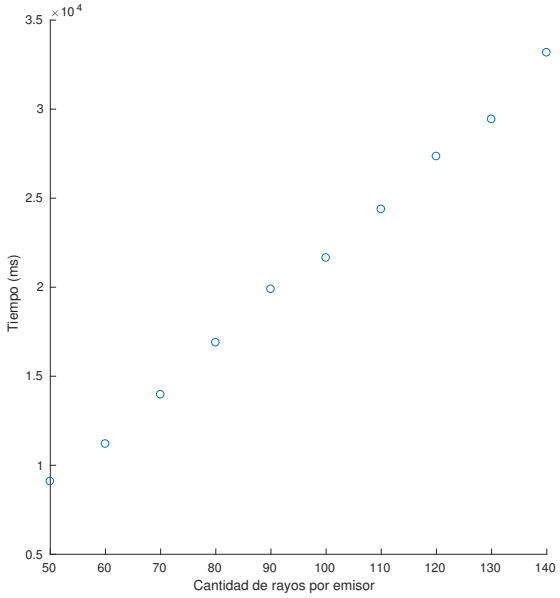


Figura 5: Tiempo en funcion de la Cantidad de rayos con granularidad, cantidad de emisores y ruido fijos

Tal como se puede apreciar en esta imagen la cantidad de rayos que parten de cada emisor afectan de manera importante el tiempo de ejecución, lo cual consideramos que se debe a que a medida que crece la cantidad de rayos, crece el tamaño de las matrices con las cuales operamos por lo que la eliminación

Gaussiana requiere más operaciones.

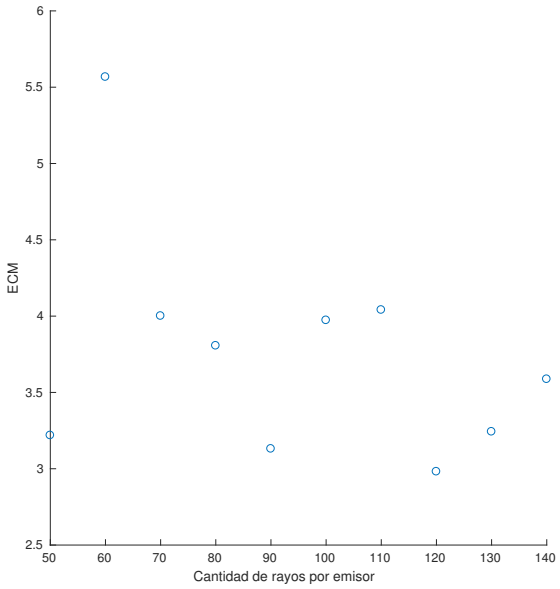


Figura 6: ECM en funcion de la Cantidad de rayos con granularidad, cantidad de emisores y ruido fijos

En este gráfico, esperamos que a medida que aumente la cantidad de informacion de la imagen, al incrementar el número de rayos por emisor, suba la calidad de la misma (baje el error cuadratico medio). Sin embargo, en el gráfico podemos ver que la cantidad de rayos no aparenta tener ninguna relación con el ECM dentro de este rango (el programa no corre adecuadamente con menor cantidad de rayos por emisor porque se presentan celdas por las que no pasa ningun rayo con una alta probabilidad). Una posible explicación para ese fenomeno es que la naturaleza aleatoria de la forma en que se construyen estos rayos lleva a que, cuando el número de rayos es suficiente para pasar por todos los pixeles de la imagen, ya se tiene suficiente información para que un incremento del número de rayos no cree una mejoría notable.

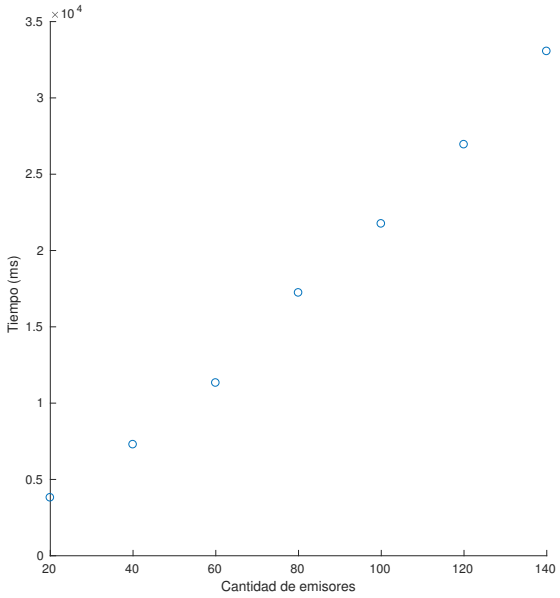


Figura 7: Tiempo en funcion de la Cantidad de emisores con granularidad, cantidad de rayos por emisor y ruido fijos

Tal como se puede observar en este gráfico la cantidad de emisores de rayos afecta de manera directa en el tiempo de ejecución. Esto se debe a que nosotros mantuvimos fijos la cantidad de rayos por emisor, con lo cual, al aumentar la cantidad de emisores estamos también aumentando la cantidad total de rayos utilizados.

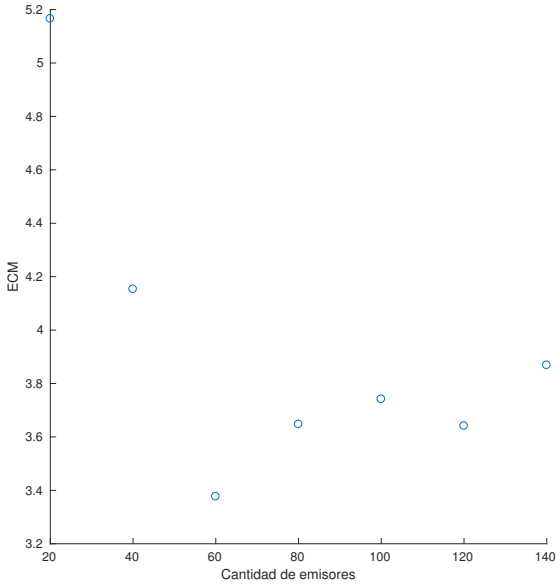


Figura 8: ECM en funcion de la Cantidad de emisores con granularidad, cantidad de rayos por emisor y ruido fijos

De forma similar al ECM en funcion de la cantidad de rayos por emisor esperabamos que baje el error cuadratico a medida que aumenta el número de de emisores. Sin embargo, como se puede ver que no hay una relación real entre la cantidad de emisores y el ECM dentro de este rango (el programa no corre con menor cantidad de emisores porque de hacerlo habría una muy alta probabilidad de que se presenten celdas por las que no pasa ningun rayo). La explicación para este fenómeno es la misma que en el caso previamente mencionado.

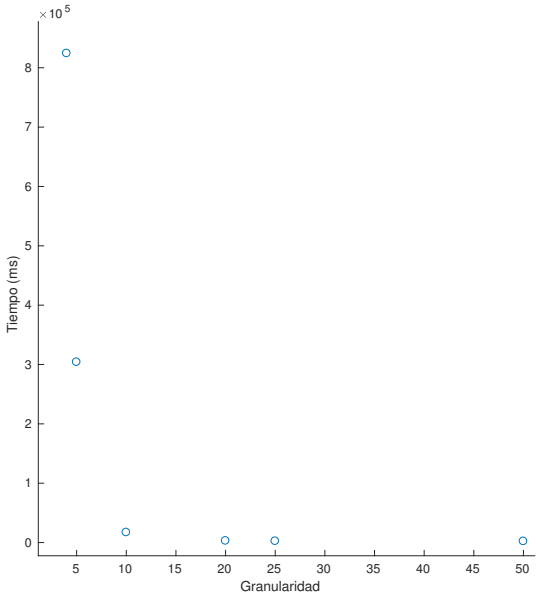


Figura 9: Tiempo en funcion de la Granularidad con cantidad de rayos, cantidad de emisores y ruido fijos

A mayor granularidad, menor resulta el tiempo de ejecución debido a que casos de granularidad se reduce el tamaño de la matriz imagen.

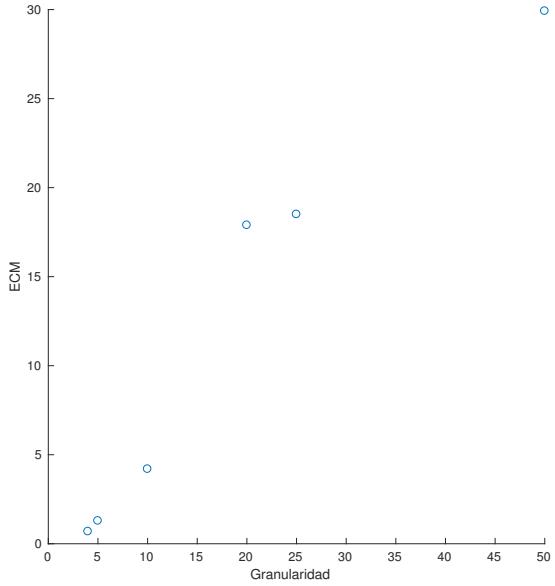


Figura 10: ECM en funcion de la Granularidad con cantidad de rayos, cantidad de emisores y ruido fijos

Se espera que el error cuadrático medio aumente a medida que el número de celdas aumenta, ya que a mayor cantidad de celdas habrá un menor número de rayos que pase por cada una de ellas. Como se puede ver se confirma nuestra hipótesis y de hecho se puede ver que hay una relación lineal entre la granularidad y el ECM.

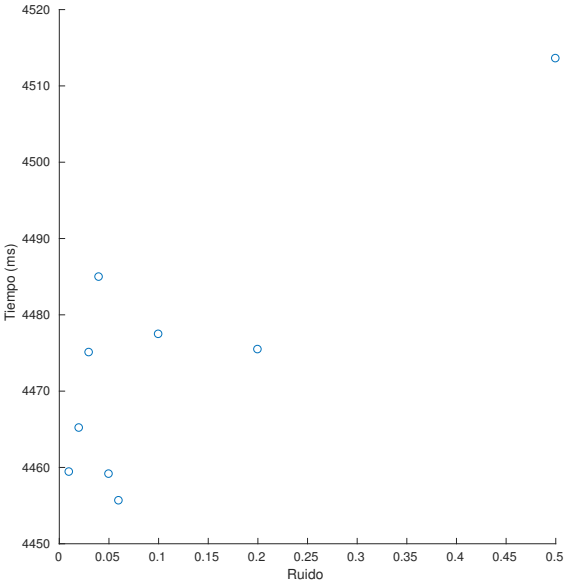


Figura 11: Tiempo en funcion del ruido con cantidad de rayos, cantidad de emisores y granularidad fijos

Tal como esperabamos, el ruido que se le agrega a la imagen no afecta de manera significativa el tiempo de ejecución.

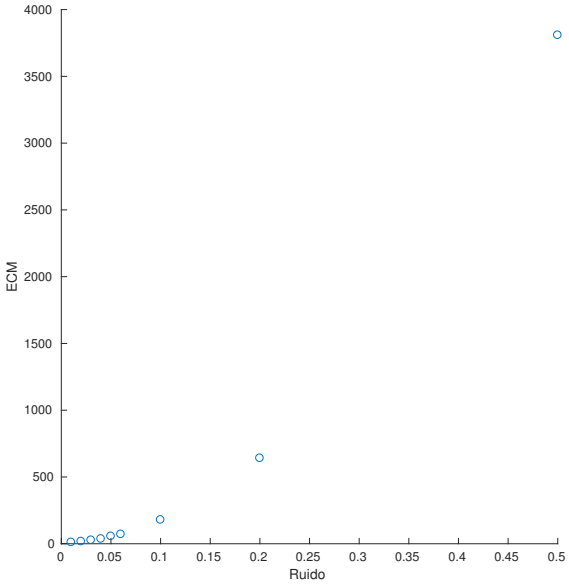


Figura 12: ECM en funcion del ruido con cantidad de rayos, cantidad de emisores y granularidad fijos

En los test que realizamos, también evaluamos distintas la robustez del sistema respecto al ruido gaussiano. Es decir, que tan buenos eran los resultados en relación al nivel de ruido que agregamos. Lo probamos con valores chicos hasta llegar al $\alpha = 0.5$, dado que por la forma en la que implementamos el ruido esto ya es un nivel alto, y de ser más alto todavía deformaría la imagen.

En los resultados de estos tests esperábamos encontrar diferencia notable entre los diferentes resultados de las diferentes ejecuciones. Sin embargo, no nos encontramos con la diferencia que esperábamos, tanto el valor más chico ($\alpha = 0.1$) como el más grande ($\alpha = 0.5$) nos parecen una reconstrucción aceptable de la imagen. Para verificar que nuestra implementación de ruido no tuviera ningún problema corrimos un nuevo test con un α aún mayor (0.8), ahora sí dando como resultado una imagen en la que no pudimos distinguir ningún objeto. Luego, pensamos que esto puede deberse a que estamos experimentando con imágenes pequeñas, o bien a que realmente nuestro sistema sea muy robusto respecto al ruido. También creemos que puede tener algo de injerencia el hecho de que el ruido que estamos agregando tenga media en cero. Estas imágenes se encuentran en el apéndice, sin embargo por su reducido tamaño puede dificultarse su análisis. De cualquier manera las adjuntamos también en los archivos.

4. Conclusiones

Conclusiones

El método utilizado para la generación de rayos permitió realizar una reconstrucción aceptable de las imágenes, no obstante, nos parece pertinente mencionar que si llegásemos a necesitar una implementación real de un tomógrafo la elección aleatoria de las ubicaciones de emisores y de los ángulos podría no ser la mejor. En este sentido sería superador a futuro encarar otra implementación que permita parámetros no aleatorios.

En este caso encontramos un buen comportamiento de nuestro sistema en relación al ruido gaussiano, nos parece interesante a futuro pensar cómo puede verse afectado por otros tipos de ruido tal vez menos comunes pero que igualmente pueden presentarse en algunas ocasiones.

La cantidad de rayos emitidos no evidenció ser un factor determinante en la calidad de la imagen reconstruída dada la forma de trazarlos usada en este trabajo ya que mientras que el método se complete la calidad de la imagen es alta en la mayoría de los casos. En cambio la granularidad elegida juega un papel clave ya que a menor tamaño de celda los cálculos permiten una mejor aproximación de los valores de intensidad a calcular. Tomando un tamaño de celda mayor se termina aproximando por un valor "promedio" de los valores de los pixeles de la celda con lo cual se pierde información particular de los mismos.

En cambio la localizacion de los emisores si parecio tener una influencioa importante en la mejora de los resultados respecto de los primeros intentos fallidos que realizamos usando la grilla y los rayos trazados desde los vértices.

Tampoco pudimos establecer una relación directa entre el error cuadrático medio y la resolución final de la imagen observada a simple vista. No nos pareció una métrica que funcionara de manera muy apropiada en este caso.

Con respecto a los tiempos de procesamiento lo que más influye en los mismos es el tiempo de resolución del sistema de ecuaciones de cuadrados mínimos el cual crece mucho de tamaño al disminuir la granularidad (como se explicó en mayor detalle en la Introducción). Esto condiciono la eleccion de las imagenes y de la granularidad.

5. Apendices

Apendices

Analisis de las imágenes reconstruídas.

En el caso de las imágenes correspondientes a las pruebas que se realizaron variando la cantidad de emisores a simple vista se puede observar que los mejores resultados se dan con 60, 120 y 140 emisores mientras que valores intermedios producen resultados menos definidos. En el caso de los 60 emisores este resultado además es consistente con un menor valor del ECM pero en el caso de 120 y 140 emisores esa relación no es clara, incluso el error con 140 es más grande que con 80 emisores sin embargo la imagen resulta más nítida. Nos cuesta establecer una relación entre la cantidad de emisores y la definición de la imagen. Es posible que al elegir los ángulos de los rayos de forma aleatoria se pueda estar generando rayos muy parecidos que no aportan información relevante.

Algo similar ocurre con las pruebas hechas variando la cntidad de rayos, en las cuales podemos observar mejores resultados con una cantidad de 60 o de 100 rayos mientras que con 140 es practicamente irreconocible.

En el caso de las imágenes correspondientes a las pruebas que se realizaron variando la granularidad se puede ver que a una granularidad más fina se observan imágenes más definidas. Esto está además en correlación con los resultados obtenidos en el cálculo del ECM.

Figura 13: Reconstruccion de una imagen de 100x100 con 20, 40, 60, 80, 100, 120 y 140 emisores de rayos respectivamente

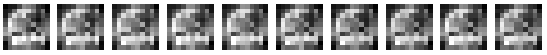


Figura 14: Reconstruccion de una imagen de 100x100 con 50, 60, 70, 80, 90, 100, 110, 120, 130 y 140 rayos por emisor respectivamente

Figura 15: Reconstruccion de una imagen de 100x100 con granularidad 5, 10, 15, 20 y 50 respectivamente

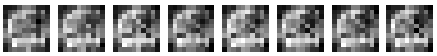


Figura 16: Reconstruccion de una imagen de 100x100 con ruido 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.1, 0.5 respectivamente

6. Referencias

Referencias

https://es.slideshare.net/sriAnkush/comparative-study-of-salt-pepper-filters-and-gaussian-filters