TAREA 3: PROCESAMIENTO DE IMÁGENES

INTRODUCCIÓN

En esta tarea se proponen 5 algoritmos para la restauración de 10 imágenes degradas (borrosas). Estos algoritmos son: Inversa, Wiener, Tikhonov (más conocido como Filtro de Mínimos Cuadrados Restringido), Lucy-Richardson y Tony. Siendo este último uno propuesto por mi. Los 5 algoritmos son comparados a través del error promedio de la restauración (EPR) definido como el promedio del absoluto de la diferencia de cada pixel entre la imagen original (C0) y la imagen restaurada.

DESCRIPCIÓN ALGORITMOS

Toda las imágenes fueron degradas con máscaras gaussianas. Esto puede ser modelado como:

$$g = h * f \qquad (1)$$

*Donde f es la imagen original, h la máscara gaussiana y g la imagen degrada. " * " Denota la operación convolución*

Tenemos que una convolución en el dominio del espacio es equivalente a una multiplicación por elemento en el dominio de la frecuencia. Por ende podemos reescribir la formula 1 a la siguiente:

$$G = H .* F \qquad (2)$$

Dónde G, H y F son la transformada de Fourier de g, h y f. Además . denota la multiplicación por elemento.*

Dado que .* tiene una función inversa, denotada como: ./, podemos reconstruir la imagen original F a partir de G haciendo:

$$G ./ H = F \qquad (3)$$

./ denota la función inversa a la multiplicación por elemento

Luego a (3) se le calcula la función inversa a Fourier y se llega a f. Esta metodología corresponde al algoritmo de la inversa, el cuál es el primero en ser evaluado. Sin embargo este algoritmo tiene dos principales problemas, el primero es que H podría tener valores nulos.

Para solucionar esto surge Wiener, dónde cambia la formula (3) por la siguiente:

$$G ./ (H + k) = F \qquad (4)$$

siendo k un número real entero mayor a cero

Al agregar una constante k jamás se dividirá por cero. Sin embargo hay que determinar cómo encontrar ese valor de k. Para determinar este valor se debe hacer una pequeña corrección a las fórmulas expuestas, las cuales, a modo de ejemplo, son extremadamente simplistas y no considerar la existencia de ruido, este es el segundo problema con el algoritmo de la inversa. En estricto rigor, (1) debiese ser:

$$g = h * f + n \qquad (5)$$

Dónde n representa el ruido existente en la imagen.

Usando esta modelación es que Wiener determina que el valor óptimo de k es:

$$k = S_n / S_f$$

Donde S_n y S_f corresponden a la densidad del espectro de potencia de n y f respectivamente.

Muchas veces no sé conocen los valores S_n y S_f por lo que k se determina empíricamente.

El tercer algoritmo utilizado, Filtro de Mínimos Cuadrados Restringido, es bastante similar a Wiener. Sin embargo difiere en la forma de calculo de k. En vez de hacer un k constante para todos los elementos hace que este sea una matriz, variando para cada elemento. La fórmula 4 queda de la siguiente forma:

$$G \,./\, (H + yP) = F \qquad (6)$$

En este caso P corresponde a la transformada de Fourier de un filtro laplaciano de 3x3. Esto surge ya que este filtro permite analizar los bordes (y la suavidad de una imagen) y es exactamente en los bordes donde una imagen nítida y una borrosa difieren. Por el otro lado agrega un parámetro y. Este puede ser calculado si se conoce el ruido y otros parámetros más, pero se puede estimar ajustándolo manualmente.

Cabe señalizar que las ecuaciones 4 y 6 fueron simplificadas, dado que la transformada de Fourier genera números complejos la verdadera formula de ambos son, respectivamente, las siguientes:

$$(\, H^* ./\, (|H|^2 + k)\,) .* G = F \qquad (7)$$

$$(\, H^* ./\, (|H|^2 + y|P|^2)\,) .* G = F \qquad (8)$$

El cuarto algoritmo que se usó corresponde a Lucy-Richardson. Este, a diferencia de los anteriores, se calcula en el dominio del espacio y no en el dominio de las frecuencias. Lo que hace es determinar que un punto en la imagen degradada corresponde a la sumatoria de los valores de la imagen original multiplicada por un

valor. Estos valores vienen dado por lo que se llama la Point Spread Function o PSF. Es decir tenemos la siguiente fórmula:

$$g_i = \sum_j PFS_{i,j} * f_j \quad (9)$$

Dónde g_i es el pixel i de g. Análogo para f_i. $PSF_{i,j}$ es la Point Spread Function para los pixeles i e j.

De esta forma vemos como cada pixel de g está compuesto por una promedio ponderada de todos los pixeles de f. Esto puede ser modelado como un proceso de poisson, donde cada pixel de g distribuye poisson con un lamba que depende del verdadero valor de ese pixel (el valor de ese pixel en f) y de la función PSF. Usando esta distribución, Lucy-Richardson genera una ecuación iterativa para resolver el problema. Esto nos da, nuevamente, un parámetro que se puede ajustar para dar un mejor resultado, el número de iteraciones a hacer.

El último algoritmo que se uso es Tony, uno inventado por mi. Este es un algoritmo muy simple y de creación mía tiene bastante poco. Este algoritmo hace exactamente lo mismo que la máscara Sharpen del ejemplo IMG04_SharpenMoon.m del profesor Domingo Mery. Al igual que el algoritmo de Tikhonov este surge de la premisa que una imagen es borrosa si tiene los bordes borrosos. Entonces busca marcar más los bordes. Para hacer esto pasa una máscara promedio por la imagen, suavizando aún más los bordes. Luego la resta de ambos da los bordes. Esta resta es incrementada por algún escalar y sumada a la imagen difusa original para exaltar los bordes. Con lo que la ecuación queda:

$$b = g * Prom_n \quad (10)$$

$$f = g + y\,(g - b) \quad (11)$$

Dónde $Prom_n$ es una mascara promedio de nxn, y es el escalar.

Al igual que los algoritmos anteriores este algoritmo también tiene variables de ajuste. Se debe determinar de que tamaño será la máscara (n) y de que magnitud será el escalar que incrementa los bordes (y).

DETERMINAR VARIABLES DE AJUSTE

Todos los algoritmos, excepto el de la inversa, tienen una o más variables de ajustes. Para determinar el valor de esta se usaron los archivos *findParam.m* y *findTonyParam.m*. El archivo *findParam.m* es para encontrar los valores de ajuste de los algoritmos Wiener, Tikhonov y Lucy-Richardson, mientras que el archivo *findTonyParam.m* es para los valores de Tony. Se hace esta distinción dado que Tony requiere estimar dos parámetros mientras que los otros algoritmos solo 1.

Al ejecutar estos archivos se pregunta por consola qué imagen es en la que se busca trabajar, en el caso del archivo *findParam.m* que algoritmo probar y, por último, cuantas iteraciones, cantidad de valores que se busca probar. Una vez que se ingresan estos parámetros el programa responde con 5 figuras, una por cada imagen degrada, en que se muestra la imagen no degradada, la imagen degrada, la imagen restaurada con el mejor parámetro y un gráfico de EPR para los distintos valores probados. El título de las imágenes incluye el EPR correspondiente mientras que el título del gráfico es el valor de la variable de ajuste en que el EPR fue mínimo. El único caso ligeramente distinto es el de *findTonyParam.m* donde el título del gráfico entrega los valores mínimos de la forma *n:y.* Cabe señalar que el gráfico es respecto a n y el punto que se muestra es con el valor de y que minimiza, para ese n, el EPR.

PRUEBAS Y RESULTADOS

Una vez que se tienen los valores óptimos para las dos imágenes en sus 5 degradaciones, para todos los algoritmos, se procede a hacerlos competir. Para esto se debe correr el archivo *tarea3.m* e indicar sobre que rostro se desea hacer la prueba. Al igual que los archivos anteriores, este responde con 5 figuras, una por cada imagen degrada. Estas figuras tienen la imagen original, no degradada, la imagen degradada y varias imagen restauradas con distintos algoritmos. Al igual que antes, en el título de cada imagen se señala con que algoritmo fueron restauradas y el EPR correspondiente.

Además cada figura consta con un gráfico que muestra el EPR para cada algoritmo. La línea roja determina el EPR de la imagen degrada sin restauración, por lo que los puntos sobre esta líneas son más distantes que la imagen degradada y los bajo esta línea presentan una mejora. Notamos que este gráfico omite la presencia del algoritmo de la inversa debido a que este presenta un EPR muy grande y distorsiona la escala.

Hasta el momento hemos hablado de 5 algoritmos, sin embargo en estas figuras se muestran 10 algoritmos. La imágenes original y la degrada tiene sus valores entre 0 y 1. Sin embargo, los algoritmos de restauración, para ciertos pixeles, arrojan valores menores a 0 y, en otros casos, mayores a 1. Por lo que los algoritmos que tiene un "++" buscar arreglar este error matemático. Lo que hacen es calcular el mínimo y máximo de la imagen degradada, luego la restauran, según el algoritmo correspondiente, y cambian todos los valores menores al mínimo por el valor mínimo y todos los valores mayores al máximo por el valor máximo, de esta manera la imagen restaurada está entre los mismos valores que la imagen degradada. Al único algoritmo que no se le hace esto es a Tony, el cual ya lo tiene incorporado como parte del algoritmo mismo.

Analizando los resultados se puede ver como entre más degradada está la imagen más difícil resulta restaurarla (presentan mayor EPR), esto se puede apreciar particularmente en el caso de la inversa en la cual pasado la degradación de 7x7 no se puede ni vislumbrar la presencia del rosto.

Dentro de los algoritmos siempre se da la misma tendencia, donde la inversa presenta el mayor EPR, seguido por Wiener, Tikhonov, Lucy-Richardson y, presentando el mejor EPR, Tony. Además tenemos que Wiener y Tikhonov siempre presentan un mayor EPR que la imagen degradada para la imagen 1, mientras que Lucy-Richardson y Tony siempre están bajo el EPR de la imagen degradada para el rostro 1.

Podemos notar que el ajuste "++" presenta una mejora en casi todos los casos (exceptuando un par de casos de la imagen 51). Incluso, Wiener++ y Tikhonov++ tienen mejor EPR que la imagen 1 degradada por una máscara de 7x7. Esto tiene un sustento matemático bastante simple, si un pixel tiene un tono de gris mayor que el máximo, la diferencia con la imagen original va a ser grande, si se lleva este valor a el máximo, de todas maneras se va a achicar esa diferencia reduciendo el EPR. Además tiene un efecto visual muy importante, al haber valores negativos matlab calibra la escala de gris de manera distinta, por lo que los rostros sin el "++" se ven más grises, sin embargo esto es solo un efecto visual.

Dado que los dos mejores algoritmos son Lucy-Richardson y Tony, surge T-Lucy o Tony-Lucy. Este último busca combinar ambos para ver de hacer una mejor restauración. Donde primero se hace una restauración de tipo Tony y luego se le hace una restauración usando Lucy-Richardson. Este corresponde al último algoritmo en cuestión, al cual también se le hace el ajuste "++". Este algoritmo presenta mejores sustanciales a Lucy-Richardson común, sin embargo Tony por si solo obtuvo mejores resultados en todos los casos.

La gran pregunta que surge, es, por qué un algoritmo tan simple como Tony sistemáticamente muestra mejores resultados que los otros algoritmos. La respuesta a esto viene dado principalmente por la manera en que se calcula el puntaje de EPR.

Una imagen degradada de por si tiene un bajo EPR, es, principalmente, el color del pixel que le corresponde ligeramente modificado por sus vecinos. Los vecinos tienden a ser bastante similares al pixel en cuestión excepto en los bordes, es decir los píxeles que elevan el EPR en una imagen borrosas corresponden a los píxeles del borde. Es estos pixeles en que se hace cargo Tony. Una imagen borrosa los va a tener más difumados, Tony obliga a incrementar ese cambio acercando el valor de esos píxeles al valor real, reduciendo el EPR.

Por el otro lado, los otros algoritmos suelen modificar bastante la imagen, siendo los bordes de la imagen los que más sufren. Estos suben el ERP aún cuando puede que la imagen sea más reconocible que la borrosa.

BIBLIGROFÍA

http://yuzhikov.com/articles/BlurredImagesRestoration1.htm

http://www.robots.ox.ac.uk/~az/lectures/ia/lect3.pdf

http://www.eecs.wsu.edu/~cs445/Lecture_13.pdf

https://www.isid.ac.in/~deepayan/SC2010/project-sub/RLA/report.pdf

https://en.wikipedia.org/wiki/Wiener_filter

http://www8.cs.umu.se/kurser/TDBC30/VT05/material/lecture5.pdf