Eliminación de ruido en imágenes y audios

Métodos Numéricos, Departamento de Computación, Universidad de Buenos Aires

Guillermo Gallardo Diez, Damian Eliel Aleman, y Luis Scoccola

gagdiez.c@gmail.com
damianealeman@gmail.com
luis.scoccola@gmail.com

Eliminación de ruido en imágenes y audios

Métodos Numéricos, Departamento de Computación, Universidad de Buenos Aires

Índice

Resumen

En el presente trabajo se estudian algunas ventajas de procesar una señal digital en el dominio transformado. En particular, se usa la Transformada del Coseno Discreta (DCT).

Se programaron filtros básicos de eliminación de ruido en imagenes y audios y se midió el rango dinámico efectivo (*peak signal-to-noise ratio*) comparando el de una señal ruidosa y esa misma señal filtrada. Se comparan los resultados de los distintos filtros de manera visual y de manera objetiva usando el *peak signal-to-noise ratio*.

Index Terms

DCT, filtro, ruido normal, PSNR, factorización PLU

I. Introduccíon teórica

GRAN parte del procesamiento de señales se realiza en el dominio transformado. Las ventajas son varias y pueden resumirse a:

- manipular el espectro de una señal de forma directa
- realizar convoluciones en tiempo lineal en lugar de cuadrático [1]

En este caso estudiamos el primer punto, la manupulación de las componentes armónicas de una señal discretizada. Usamos la DCT para obtener nuestra señal en la base ortogonal compuesta por cosenos con distintas frecuencias. Estas frecuencias son divisiones enteras de la frecuencia de muestreo[2].

La transformada se obtiene realizando una simple multiplicación entre la matriz cambio de base y la señal original.

Para reducir el ruido de la señal usamos dos filtros:

LPF: atenua frecuencias agudas
multiband gate: atenua frecuencias por debajo de un cierto umbral

Para volver al dominio de las señales aplicamos el cambio de base inverso[2]. En este caso usamos la tecnica de plantear y resolver el sistema lineal dado por la matriz cambio de base y las componentes armónica de la señal filtrada como se pide en [1].

Utilizamos ruido normal para obtener imagenes ruidosas y poder calcular el PSNR logrado.

II. Desarrollo

DESEAMOS eliminar ruido tanto de imágenes como de audios. Si bien el proceso es similar debemos realizar algunas operaciones de manera particular en cada caso.

[1] Esta mejora es realmente notable cuando se usa una transformada rápida.

A. *Procesando audio*

Como se explica en [1], a partir del vector v que representa el audio sampleado, obtenemos su tranformado t realizando la multiplicación $M \times v = t$, donde M representa la matriz cambio de base entre la canónica y la base $\{1, cos(x), \ldots, cos(\frac{x}{n-1})\}$. Aquí x es la frecuencia de sampleo.

Una vez obtenido el espectro t se aplica un filtro, que es una función de aridad $f : T \to T$, dónde T representa el espacio vectorial al cual pertenece t, en este caso $\mathbb{R}^n$. De esta manera obtenemos nuestro espectro filtrado $e = f(t)$.

Finalmente realizamos la transformada inversa y obtenemos nuestro audio filtrado a resolviendo el sistema $M \times a = e$. Este sistema es resuelto mediante el método de triangulación y sustitución para atrás, *gauss con pivoteo*.

B. *Procesando imágenes*

En [1] se explica que la transformada T de una señal S, una matriz cuadrada que representa los pixeles de una imágen, se obtiene realizando: $M \times S \times M^t = T$. Dado que la matriz asociada a una imagen no suele ser cuadrada recortamos la imágen para obtener una imagen cuadrada[2].

Luego transformamos y aplicamos una función similar a la explicada anteriormente para obtener el espectro filtrado E.

Finalmente debemos realizar la transformada inversa para regresar al dominio original y obtener nuestra señal filtrada A. Para esto debe resolverse el sistema: $M \times A \times M^t = E$.

Esto se logra notando que podemos resolverlo de a poco. Planteamos $Y = A \times M^t$ y obtenemos el sistema: $M \times Y = E$. Este lo resolvemos realizando n veces *gauss*, uno por cada columna de E e Y.

Luego resolvemos la ecuación $Y = A \times M^t$. Para esto notamos que es equivalente a: $Y^t = M \times A^t$. Este sistema se resuelve de manera análoga al explicado anteriormente.

Vimos que el proceso de realizar *gauss* n tomaba demasiado tiempo. De hecho, ya que *gauss* toma un tiempo cúbico, estaremos hablando de una complejidad de $O(n^4)$. Una complejidad como esta suele no ser admisible, sobre todo si puede mejorarse. Por este motivo realizamos la optimización de factorizar a la matriz cambio de base mediante la factorización PLU. De esta manera estaremos triangulando una única vez, y la complejidad permanecerá cúbica.

Finalmente, luego de resolver ambos sistemas, obtenemos la matriz A, nuestra imagen filtrada.

C. *Filtrando una señal*

Implementamos dos filtros que trabajan sobre el espectro de maneras distintas.

[2] Veremos que uno de los filtros implementados no impondrá esta precondición en la entrada.

C.1 Filtro tipo Gate

Este filtro se comporta como una compuerta multibanda. A partir de un umbral dado en decibeles, se reducen, por un factor dado, todas las componentes armónicas que se encuentran por debajo del umbral.

La idea es que frecuencias con una baja amplitud contribuyen más al ruido que a la señal en si.

C.2 Filtro tipo LPF

Este filtro atenua por factor dado, las frecuencias agudas, a partir de cierta frecuencia inicial. Para esto se elige un porcentaje que simboliza el punto del espectro a partir del cual deben filtrarse las componentes armónicas. Esto es análogo a realizar una convolución entre la señal y una *sinc function*, es decir, es el filtro pasa-bajos más simple que puede lograrse[3, The Ideal Lowpass Filter].

La idea detrás de este filtro es que la frecuencias agudas tienden a ser percibidas más por el ser humano, tanto visual como auditivamente. Por ende, si hubiese ruido, será percibido más en esta parte del espectro.

Las frecuencias agudas se asocian, visualmente, con transiciones más frecuentes en la escala de grises. Por lo tanto reducir las frecuencias agudas resultará en una imagen con cambios más suaves.

Auditivamente, la reducción de frecuencias agudas suele utilizarse para disminuir el *hiss* de casettes, vinilos e incluso, en etapas de edición y mezcla de audio, para limpiar guitarras electricas y voces de componentes armónicas innecesarias, que contribuyen al piso de ruido general.

C.3 Filtro por bloques

En el caso de las imagenes notamos que, por más que usemos la factorización PLU para reducir el tiempo de cómputo, el mismo sigue siendo muy significativo.

Por esta razón implementamos los filtros previamente explicados por bloques. Es decir, dada una imagen, tomaremos submatrices cuadradas de la misma y las filtraremos una por una. La dimensión de estas matrices será elegida en un *trade-off* entre calidad del resultado y tiempo de cómputo.

Notemos además, que este filtro no tendrá restricciones en cuanto a las dimensiones de la imagen.

D. Midiendo las mejoras

Si bien las mejoras terminan siendo útiles en la medida en que sean agradables al ser humano, es de suma utilidad poder cuantificarlas. Para esto implementamos una función que calcula el *peak signal-to-noise ratio* en función de la señal original y la filtrada.

E. Ensuciando una señal

Para poder experimentar más cómodamente realizamos una función para aplicar ruido a una señal. El tipo de ruido elegido es el ruido gaussiano.

Cada muestra del ruido se genera con la fórmula:

$$guilletiralaformula$$

TESTEAMOS los algoritmos en tres etapas.

F. Efectos en la señal y el espectro

Graficamos el espectro original y el filtrado para señales de audio. La idea, es mostrar que, por más que ganemos cierto rango dinámico, no estaremos filtrando el ruido en si, sino frecuencias que lo hacen más notable.

G. Buenos parámetros para señales específicas

Tomando un conjunto de audios e imágenes buscamos parámetros de ambos filtros que maximicen el rango dinámico ganado para distintos niveles de ruido agregado. Queremos ver cuando los filtros resultan más efectivos. Es decir, si lo son en la presencia de más o menos ruido.

H. Tiempo de corrida

Para el caso de las imágenes comparamos el tiempo de ejecución de los filtros estándar con los filtros por bloques para distintos tamaños de bloque. Esperamos ver una complejidad cuadrática para bloques pequeños e imagenes grandes ya que las operaciones de resolución de sistemas serán despreciables y se notará más que nada el recorrido de la matriz principal, de tamaño cuadrático en función del lado.

III. Resultados

A. Efectos en la señal y el espectro

B. Buenos parámetros para señales específicas

B.1 Sonidos

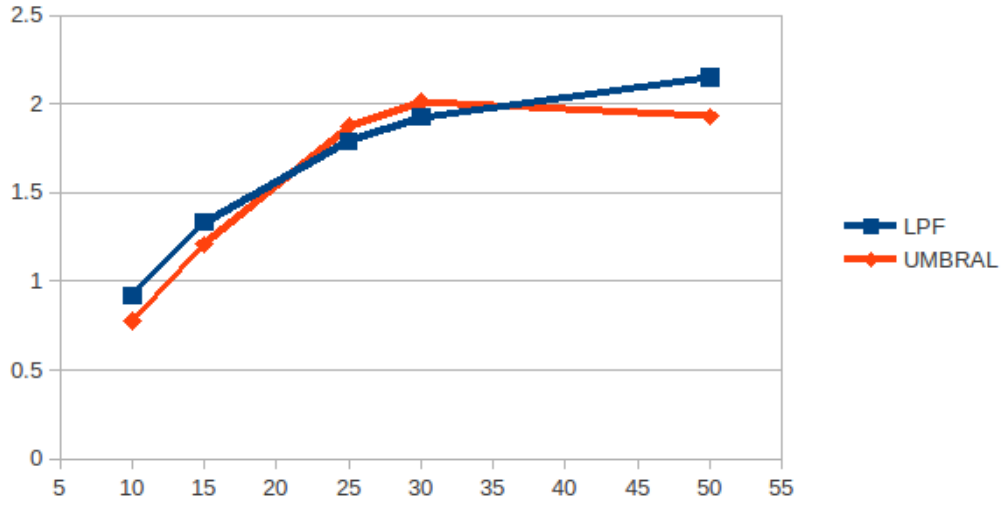
!!!!!!!!!!!!!!!!!!!!!!!!!!!1

B.2 Imágenes

Aquí presentamos un gráfico que muestra la relación entre ruido introducido y mejora del PSNR. Es importante notar que cada uno de estos resultados fue el trabajo de una optimización manual[3], y que si bien pensamos que puede ser automatizada, no nos resultó evidente la manera de hacerlo. Dado que deben optimizarse dos parámetros simultaneamente y las funciones a optimizar parecen registrar varios mínimos locales.

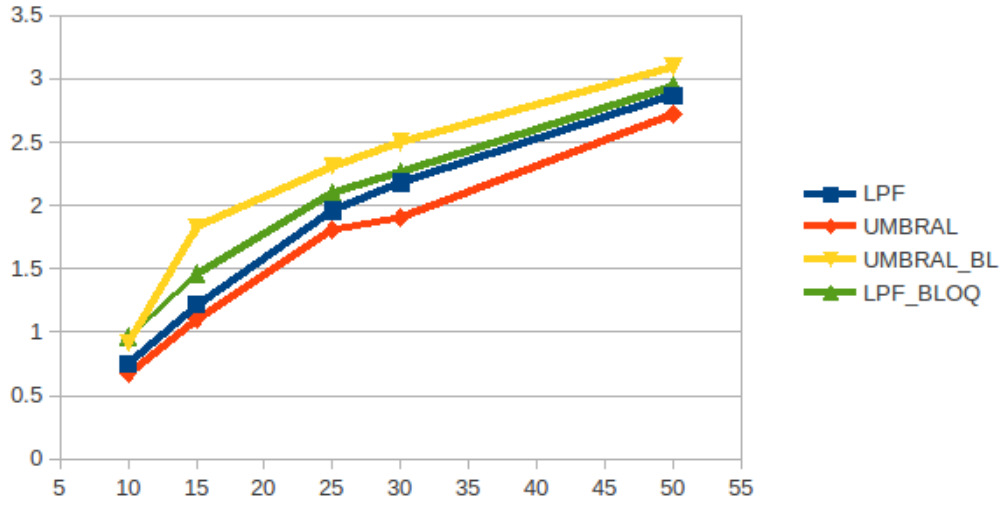
[3] Ver el apéndice de tablas para los resultados numéricos y los parámetros utilizados

Las imágenes usadas fueron `Red_cuadrada.pgm` y `masterVoice_cuadrada.pgm`. Dado que la primera es de un tamaño pequeño utilizamos unicamente los filtros estándar. Para la segunda utilizamos también los algoritmos por bloques.



Decibeles ganados - porcentaje de ruido introducido

Imágen `Red_cuadrada.png`



Decibeles ganados - porcentaje de ruido introducido

Imágen `masterVoice_cuadrada.png`

C. Tiempo de corrida

La diferencia, en tiempo, entre trabajar con el espectro de toda la imágen y filtrar de a bloques se ve en esta imágen:

AKSJDAKJSDNASKJN

Aquí se comparan los tiempos de ejecución de un filtro por bloques para distintos tamaños de bloque[4]:

JANSDKAJNDKAJNDKASJDN

IV. Discusión

A. Error en ciertos píxeles

Si bien implementamos los algoritmos como es explicado y los mismos se comportan como era de esperarse, notamos que, al aplicar tanto el filtro pasa-bajos como el de tipo compuerta algunos pixeles se saturan.

Esto se hace evidente en la esquina izquierda-superior de las imágenes. Luego de investigar el problema, estamos seguros de que el fenómeno proviene de el filtrado en sí, y no de un problema numérico de la transformada.

Conjeturamos que se debe al uso de un corte muy repentino entre señales sin atenuación y señales con atenuación. Como se explica en [3, Window Method for FIR Filter Design], suele transformarse la respuesta en frecuencia deseada, aplicarsele una ventana, y luego transformarse otra vez, ahora si para modificar el espectro de la señal. Este procedimiento minimiza los artefactos creados por el uso de una respuesta en frecuencia poco suave, como es nuestro caso.

V. Conclusiones

Referencias

[1] Cátedra de Métodos numéricos,
Segundo Trabajo Práctico,
Primer cuatrimestre 2013
[2] Syed Ali Khayam,
The Discrete Cosine Transform (DCT): Theory and Application,
March 10th 2003
[3] Julius O. Smith III,
Spectral Audio Signal Processing,
W3K Publishing, 2011

[4] En ambos casos no se hace distinción entre los filtros de umbral o pasa-bajos, ya que esto no es relevante para la toma de tiempos.

VI. Apéndices

A. Enunciado

Introducción

La Transformada Discreta del Coseno (DCT, por sus siglas en inglés) es una herramienta que nos permite representar cualquier señal en el plano de las frecuencias. Dado que es utilizada por el estándar de compresión de imágenes JPEG y formato de video MPEG, se encuentra implementada en más lugares de lo que pensamos: en cada cámara digital o teléfono móvil. La DCT no solo tiene aplicaciones al mundo de la compresión (donde los valores transformados pueden ser codificados de forma eficiente), sino también al procesamiento: el análisis de qué frecuencias están presentes en las señales es esencial en ciertos contextos de aplicación.

La idea intuitiva de esta transformada, en el plano continuo, consiste en representar una función $f : \mathbb{R} \to \mathbb{R}$ en la base de funciones $\mathcal{B} = \{1, \cos(x), \cos(2x), ...\}$. En el plano discreto, la DCT se corresponde a un cambio de base: cada una de las funciones de la base $\mathcal{B}$ se discretiza en ciertos puntos pasando a ser una base de vectores en $\mathbb{R}^n$, (donde n es la dimensión del vector o señal a transformar). Es decir, dado un vector o señal $x \in \mathbb{R}^n$ existe una matriz $M \in \mathbb{R}^{n \times n}$ de cambio de base que define la transformada DCT, donde $y = Mx$ es el vector o señal transformado al espacio de frecuencias por la DCT (ver apéndice). Esta operación es fácilmente extensible a señales de dos dimensiones (ver apéndice).

Enunciado

El objetivo del trabajo es eliminar ruido sobre una señal ruidosa $x \in \mathbb{R}^n$. Para ello se realiza el siguiente proceso:

1. $y := Mx$ [Transformar usando Ec. () de Ap.]
2. $\tilde{y} := f(y)$ [Modificar]
3. Resolver $M\tilde{x} = \tilde{y}$ [Reconstruir]

Una forma de medir la calidad visual de la señal reconstruida $\tilde{x}$, es a través del PSNR (*Peak Signal-to-Noise Ratio*). EL PSNR es una métrica 'perceptual' (acorde a lo que perciben los humanos) y nos da una forma de medir la calidad de una imagen perturbada, siempre y cuando se cuente con la señal original. Cuanto mayor es el PSNR, mayor es la calidad de la imagen. La unidad de medida es el decibel (db) y se considera que una diferencia de 0.5 db ya es notada por la vista humana. El PSNR se define como:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_x^2}{ECM} \right)$$

donde MAX_x define el rango máximo de la señal (en caso de entradas de 8 bits sin signo, sería 255) y ECM es el *error cuadrático medio*, definido como: $\frac{1}{n} \sum_i (x_i - \tilde{x}_i)^2$, donde n es la cantidad de elementos de la señal, x es la señal original y $\tilde{x}$ es la señal recuperada.

En la implementación realizada deben llevar a cabo los siguientes experimentos:

- Para varias señales con distintos niveles de ruido, se deberán experimentar con al menos 2 estrategias (definiendo f de forma conveniente) para modificar la señal transformada y (paso 2) con el objetivo de que la señal recuperada $\tilde{x}$ contenga menos ruido; se deberán

extraer conclusiones en cuanto a la calidad de la señal recuperada, en función de la estrategia utilizada.
- Se deberán repetir los anteriores experimentos también sobre imágenes adaptando el método para aplicar la transformada DCT en dos dimensiones según se explica en la apéndice.
- **(Opcional)** Se deberá analizar la aplicación de la DCT 'por bloques' sobre imágenes. Por ejemplo, si tenemos una imagen de 64×64 píxeles podemos subdividirla en: 4 bloques de 32×32, o 16 bloques de 16×16, o 64 bloques de 8×8, y aplicar la DCT en 2D sobre cada uno de los bloques (considerar un tamaño mínimo de 8×8 para cada bloque).
  Elegir una estrategia utilizada para señales unidimiensionales y sacar conclusiones respondiendo a los siguientes interrogantes (realizando experimentos que justifiquen la respuesta): ¿Es lo mismo eliminar ruido sobre la imagen entera que de a bloques? ¿Qué forma es más conveniente en cuanto a la calidad visual? ¿Qué forma es más rápida?

Formatos de archivos de entrada

Las señales serán leídas de un archivo de texto en cuya primer línea figuran la cantidad de datos y en la línea siguiente se encuentran los datos en ASCII separados por espacios. Para leer y escribir imágenes sugerimos utilizar el formato *raw* binario `.pgm`[5]. El mismo es muy sencillo de implementar y compatible con muchos gestores de fotos[6] y Matlab.

Fecha de entrega:

- *Formato electrónico:* jueves 16 de mayo de 2013, hasta las 23:59 hs., enviando el trabajo (informe+código) a `metnum.lab@gmail.com`. El subject del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de los integrantes del grupo.
- *Formato físico:* viernes 17 de mayo de 2013, de 18 a 20hs (en la clase del labo).

Transformada Coseno Discreta

Para generar la matriz $M \in \mathbb{R}^{n \times n}$ que define la transformada de Coseno Discreta definimos:

- Vector de frecuencias: $g = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ n-1 \end{pmatrix}$
- Vector de muestreo: $s = \frac{\pi}{n} \begin{pmatrix} \frac{1}{2} \\ 1+\frac{1}{2} \\ \vdots \\ (n-1)+\frac{1}{2} \end{pmatrix}$
- Constante de normalización: $C(k) = \begin{cases} \sqrt{\frac{1}{n}} & k = 1 \\ \sqrt{\frac{2}{n}} & k > 1 \end{cases}$

Siendo $T = \cos(g \cdot s^t)$ la matriz resultante de aplicarle el coseno a cada elemento de la matriz $g \cdot s^t$, finalmente definimos, $\widehat{M}_{i,j} = C(i) \cdot T_{i,j}$

[5] `http://netpbm.sourceforge.net/doc/pgm.html`
[6] XnView `http://www.xnview.com/`

Para obtener una versión entera de la transformación que define la matriz M, la cual será aplicada a señales (o vectores) enteras en el rango $[0, q]$, definimos:

$$M = \left\lfloor \frac{q\widehat{M} + 1}{2} \right\rfloor \tag{1}$$

donde $\lfloor \cdot \rfloor$ indica la parte entera inferior[7]. (Es decir, escalamos los elementos de la matriz M por $q/2$ y luego redondeamos los valores.)

Extensión a 2D Dada una matriz $B \in \mathbb{R}^{n \times n}$, podemos extender fácilmente la transformada DCT a señales de dos dimensiones. Para ello, aplicamos la transformación por filas y por columnas: $M\,B\,M^t$

[7] El redondeo de un número m pude definirse como $\lfloor m + 1/2 \rfloor$. Luego, M se define como el redondeo de $\widehat{M} \cdot (q/2)$.

B. Código relevante

C. Tablas