DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Demosaicing

11 de noviembre de 2014

Métodos Numéricos
Trabajo Práctico Nro. 3

Integrante	LU	Correo electrónico
Martin Carreiro	45/10	martin301290@gmail.com
Kevin Kujawski	459/10	kevinkuja@gmail.com
Juan Manuel Ortíz de Zárate	403/10	jmanuoz@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Resumen

En el siguiente trabajo investigaremos el comportamiento de distintos algoritmos pensados para resolver el problema de Demosaicing. Describiremos sus distintos funcionamientos, realizaremos pruebas transversales a todos ellos y pruebas particualres a cada uno. Con el objetivo de, en primera instancia, hacer comparaciones a grandes rasgos de los resultados obtenidos para luego con las pruebas individuales evaluar el resultado en los casos potencialmente conflictivos de cada uno y así tener un panaroma más completo a la hora de concluir cual es el mejor o si sus eficiencias varian según los contextos de uso.

2. Introducción teórica

El problema de Demosaicing consta de poder construir una imagen con información de los 3 canales en cada uno de sus pixel en base a una que sólo tiene definidos los valores de un sólo canal para cada celda. Este desafío es muy común en las camáras de fotos digitales ya que en realidad cada fotosensor detecta un sólo color, por lo cual a la imagen capturada hay que aplicarle algún procedimiento lógico para que el usuario final pueda verla efectivamente con todos los colores.

Particularmente estaremos analizando 4 algoritmos distintos que intentan solucionar esto. Ellos son: Vecinos, Bilineal, Direccional y High Quality. Todos estos deben aplicarse sobre imágenes en formato Bayer array, es decir, imágenes que en cada pixel tienen información sólo de un canal (Verde, Rojo o Azul) y estos además tienen una distribución especial (en la que por ejemplo el verde predomina en cantidad ya que es el color que mejor recepción tiene en el ojo humano).

Para nuestras pruebas lo que haremos es tomar imágenes con todos sus canales completos en todos lo pixeles y pasarlas al formato Bayer array. Como en lo que queremos hacer foco en este trabajo es la calidad de estos procedimientos y sus resultados, no nos interesa que la imagen Bayer haya sido tomada realmente por una cámara, es más, nos sirve más tener una imagen full color y transformala ya que así podemos comparar nuestros resultados contra la original.

Por esto es que desarrollamos una función muy simple que realiza esta transformación. Básicamente lo que hace es:

- Celdas en columnas y filas pares, deja sólo el canal azul
- Celdas en columnas y filas impares, deja sólo el canal rojo
- En todas las otras dejamos sólo el canal verde

3. Desarrollo

Como explicamos en la introducción estaremos analizando 4 algoritmos distintos, estos son: Vecinos, Bilineal, Direccional y High Quality. La idea será comparar sus resultados subjetivamente, es decir a simple vista como vemos la imagen resultante, objetivamente y temporalmente, osea cuanto tiempo demora en ejecutarse el procedimiento, para poder concluir finalmente las ventajas y desventajas de cada uno. A continuación explicaremos como implementamos cada uno.

3.1. Vecinos

Este es el más simple de todos. La idea es estabelecer el valor de los colores faltantes de cada pixel en base al vecino que tenga dicho valor. Por ejemplo si estamos en un pixel azul le preguntamos a algun vecino rojo y otro verde su valor y los seteamos en los correspondientes colores de nuestro pixel.

Pseudocódigo:

```
Para cada celda:
        si es roja:
                celda.verde = vecinoIzq.verde
                celda.azul  = vecinoSuperiorIzq.azul
        si es azul:
                celda.rojo = vecinoInferiorDerecho.rojo
                celda.verde = vecinoDerecho.verde
        si es verde y fila par:
                celda.rojo = vecinoDerecho.rojo
                celda.azul = vecinoSuperior.azul
        si es verde y fila impar
                celda.rojo = vecinoInferior.rojo
                celda.azul = vecinoIzquierdo.azul
```

Se diferencia entre fila par e impar cuando la celda es verde ya que es distinta la disposición de sus vecinos en cada caso, esto en cambio se mantiene inmutable en los casos de celda roja o azul. La selección del vecino izquierdo y superior izquierdo cuando es roja es porque siempre existe ese vecino, a diferencia del derecho por ejemplo ya que el rojo puede ser una celda borde y no tener dicho adyacente. La misma idea aplicamos al caso de la celda azul y de las verdes, apuntamos a elegir el vecino que siempre existe.

3.2. Bilineal

3.3. Direccional

Para este algoritmo nos basamos en lo explicado por Burden y Faires[1] para el caculo de los splines y en lo desarrollado por Ron Kimmel[2] para el algoritmo en sí. El método lo aplicamos sólo para el color verde mientras que para los otros utilizamos bilineal.

Lo que hace es:

```
Para cada celda:
        Interpolo mediante spline su fila y columna
        Calculo sus derivadas aproximadas en direcci\'on horizontal y vertical
        Si la derivada horizontal es mayor:
```

```
                celda.verde = a
sino                celda.verde = a
```

Dado que las interpolaciones mediante splines no son nada triviales lo explicaremos mas adelante en detalle. Las derivadas en x la aproximamos haciendo $|G(x-1,y) - G(x+1,y)|$ donde G es el valor del color verde en ese punto, la derivada en y es análoga. Dado que un mayor valor en la derivada puede estar indicándonos un potencial borde le damos mayor peso a la derivada cuyo valor es más chico multiplicando a este por 0.7 y a la otra por 0.3. Finalmente las sumamos para obtener el verde correspondiente en nuestra celda.

3.4. High Quality

Todos los algoritmos anteriores aproximaban el valor mediante un color, pero esto no era suficiente para darle definición ya que muchas veces los colores tienen un brillo o luz que impacta en todos, por lo tanto...

El algoritmo de demosaicing que denominamos "High Quality"se basa en el paper de Malvar, He y Cutler. Su principal atractivo es que realiza una corrección de la imagen luego de aplicar el algoritmo de Interpolación Bilineal pero teniendo en cuenta para calcular un color a los otros dos, en diferencia a los otros algoritmos que solo tienen en cuenta los valores cercanos con respecto a su mismo color.

Para el algoritmo highquality decidimos solamente procesar el color verde de los pixeles ya que solamente medimos la calidad de ese color en el tp.

```
Primero hacemos bilineal sobre todos los colores de la imagen
Por cada pixel de la imagen (exceptuando los bordes):
        Si la imagen cae en verde la ignoro
        Si cae en rojo o azul:
                Al color verde de ese pixel le sumo el valor del color actual
                   calculado en la bilineal
```

4. Experimentación Y Resultados

5. Discusión

6. Conclusiones

Referencias

[1] $R.BurdenyJ.D.Faires, Analisisnumerico, InternationalThomsonEditors$, 1998

[2] $RonKimmel.Demosaicing : imagereconstructionfromcolorccdsamples.IMAGEPROCESSING, IEEETRANSACTIONSON$, 1999.