Desafío 8

Mikel Dalmau

23 de Diciembre de 2018

Indice

1 Enunciado

El objetivo de este desafío es realizar un sistema que distinga las texturas provenientes de la base de datos original de Brodatz usando un banco de filtros de Gabor. La validación se realizará sobre la imagen compuesta de varias texturas intentando distinguir las regiones correspondientes a cada textura.

La idea general consiste en que cada textura se caracteriza de la siguiente manera:

1. se aplica un banco de filtros de Gabor obteniendo una imagen multicapa resultante (para cada filtro obtenemos la fase y la magnitud de la respuesta), donde cada pixel está representado por un vector de respuestas en ese pixel. Cada capa se normaliza independientemente usando zscore.

2. Se aplica clustering (K-means) a los vectores de caracteristicas de los pixeles de la imagen (fijamos arbitrariamente K)

2 Resolución

El siguiente proceso de segmentación de texturas utilizando filtros de Gabor está extraído de [MatLab 1], que a su vez se basa en la solucción expuesta en [Jain, 1991] y consiste en los siguentes pasos.

1. Diseño del vector de filtros de Gabor

Año: 2018 - 2019
Profesor(a): Manuel Graña
Teléfono:
E-mail: mdalmau002@ikasle.ehu.es

GIPUZKOAKO CAMPUSA
CAMPUS DE GUIPUZCOA
Paseo Manuel de Lardizabal, 1
20018 Donostia (Gipuzkoa)

2. Post procesado de las imágenes de Magnitudes de Gabor en Características de Gabor.
3. Clasificación de las características de Gabor.

2.1 Diseño del vector de filtros de Gabor

El primer paso consiste en diseñar un array de filtros de Gabor calibrados a diferentes frecuencias y orientaciones. El conjunto de frecuencias y orientaciones está diseñado para localizar subconjuntos de información de frecuencia y orientación diferentes, aproximadamente ortogonales, en la imagen de entrada. Se crea un muestreo de orientaciones entre [0,135] en intervalos de 45 grados y se crea un muestreo de longitudes de onda en potencias crecientes de 4/sqrt(2) hasta la longitud de la hipotenusa de la imagen de entrada. Estas combinaciones de frecuencia y orientación se han tomado de [Jain, 1991].

```
% longitudes de onda
wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

% Orientaciones
deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);

% Banco de filtros de Gabor
g = gabor(wavelength, orientation);
```

Finalmente se extraen las características de la magnitud de Gabor de la imagen de origen. Cuando se trabaja con filtros de Gabor, es común trabajar con la respuesta de magnitud de cada filtro. La respuesta de magnitud de Gabor también se denomina a veces "Energía de Gabor". Cada imagen de salida de magnitud MxN Gabor en gabormag (:,:, ind) es la salida del filtro Gabor correspondiente g (ind).

```
gabormag = imgaborfilt(imagen_validacion,g);
```

2.2 Post procesado de las imágenes de Magnitudes de Gabor en Características de Gabor

De cara a utilizar las magnitudes de Gabor como características para su uso en la clasificación, se requiere algo de procesamiento posterior. Este procesamiento posterior incluye:

1. Suavizado Gaussiano.
2. Adición de información espacial adicional al conjunto de características.
3. Remodelación del conjunto de características a la forma esperada por las funciones pca y kmeans.
4. Normalización de la información de las características a una varianza y media comunes.

2.2.1 Suavizado Gaussiano

Cada imagen de magnitud de Gabor contiene algunas variaciones locales, incluso dentro de regiones bien segmentadas de textura constante. Podemos compensar estas variaciones utilizando un filtro gaussiano de paso bajo para suavizar la información de magnitud de Gabor. Para ello se elige un sigma que coincida con la longitud de onda del filtro de Gabor que extrajo cada característica e introducimos un término de suavizado K que controla la cantidad de suavizado que se aplica a las respuestas de magnitud de Gabor.

```
for i = 1:length(g)
    sigma = 0.5*g(i).Wavelength;
    K = 3;
    gabormag(:,:,i) = imgaussfilt(gabormag(:,:,i),K*sigma);
end
```

2.2.2 Adición de información espacial

Al construir los conjuntos de características de Gabor para la clasificación, es útil agregar un mapa de información de ubicación espacial tanto en X como en Y. Esta información adicional permite al clasificador preferir agrupaciones que están muy cerca espacialmente. Las características se añaden como una capa extra a los filtros de Gabor.

```
X = 1:numCols;
Y = 1:numRows;
%   [X,Y] = MESHGRID(xgv,ygv) replicates the grid vectors xgv and ygv to
%   produce the coordinates of a rectangular grid (X, Y). The grid vector
%   xgv is replicated numel(ygv) times to form the columns of X. The grid
%   vector ygv is replicated numel(xgv) times to form the rows of Y.
[X,Y] = meshgrid(X,Y);
%   CAT(DIM,A,B) concatenates the arrays A and B along
%   the dimension DIM.
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);
% 1200x1200x32 ->  1200x1200x34
```

2.2.3 Remodelación del conjunto de características para utilizarlas en kmeans

Cada píxel en la cuadrícula de imagen es un punto de datos independiente, y cada plano representa una característica diferente. En este ejemplo, hay una característica separada para cada filtro del banco de filtros de Gabor y dos características adicionales de la información espacial que se agregó en el paso anterior. En total, hay 32 características de Gabor y 2 características espaciales para cada píxel en la imagen de entrada.

```
numPoints = numRows*numCols;
%   RESHAPE(X,M,N) or RESHAPE(X,[M,N]) returns the M-by-N matrix
%   whose elements are taken columnwise from X. An error results
%   if X does not have M*N elements.
X = reshape(featureSet,numPoints,[]);
% Matriz de carateristicas -> vector de caracteristicas
% 1200x1200x34 -> 1440000x34
```

2.2.4 Normalización de las características

Normalizar las características para que tengan media zero y varianza unitaria.

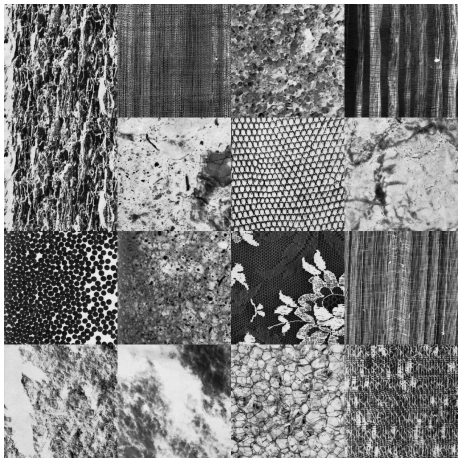
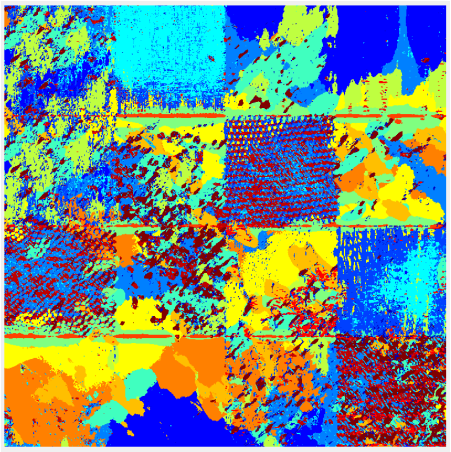
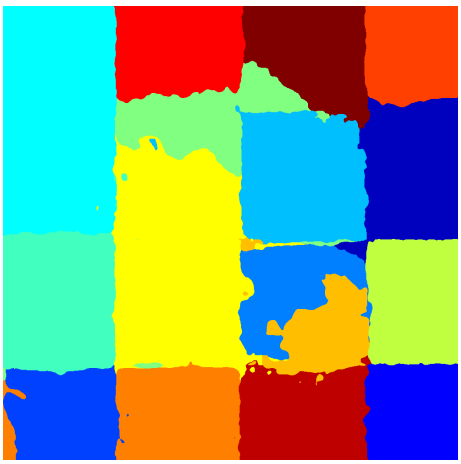
```
%   C = BSXFUN(FUNC,A,B) applies the element-by-element binary operation
%   specified by the function handle FUNC to arrays A and B, with implicit
%   expansion enabled.
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));
```

2.3 Clasificación utilizando K-means

Repetimos k-means clustering varias veces para evitar los mínimos locales cuando busque medias que minimicen la función objetivo. La única información previa que asumimos es cuántas regiones distintas de textura están presentes en la imagen que se segmenta. Hay 15 regiones distintas en este caso.

```
L = kmeans(X,16,'Replicates',5);
```

3 Resultados

Imágen de texturas original		Resultados k=16, 5 iteraciones Sin inforamción espacial adicional ver seccion 2.2.2.	
Resultados k=15, 5 iteraciones		Resultados k=16, 5 iteraciones	

4 Anexo

4.1 Código completo desafío 8

```
imagen_validacion = imread('validacion.gif');
numRows = 1200;
numCols = 1200;

% longitudes de onda
wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

% Orientaciones
deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);

% Banco de filtros de Gabor
g = gabor(wavelength, orientation);

% Extraer las magnitudes de gabor,
% Cada magnitud MxN de la imagen resultante de gabormag(:,:,ind)
% es el resultado del correspondiente filtro de gabor g(ind).
gabormag = imgaborfilt(imagen_validacion,g);

% Aplicacion de un filtro gaussiano para eliminar las variaciones locales
% en la magnitud del filtro de gabor.
for i = 1:length(g)
    %Calculo del paso del filtro gausiano en funcion de cada longitud de
    %onda y suavizado por K
    sigma = 0.5*g(i).Wavelength;
    K = 3;
    gabormag(:,:,i) = imgaussfilt(gabormag(:,:,i),K*sigma);
end

%  Agregar un mapa de informacion de ubicacion espacial tanto en X como en Y.
X = 1:numCols;
Y = 1:numRows;
%   [X,Y] = MESHGRID(xgv,ygv) replicates the grid vectors xgv and ygv to
%   produce the coordinates of a rectangular grid (X, Y). The grid vector
%   xgv is replicated numel(ygv) times to form the columns of X. The grid
%   vector ygv is replicated numel(xgv) times to form the rows of Y.
[X,Y] = meshgrid(X,Y);
%   CAT(DIM,A,B) concatenates the arrays A and B along
%   the dimension DIM.
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);
%  1200x1200x32 ->  1200x1200x34

numPoints = numRows*numCols;
%   RESHAPE(X,M,N) or RESHAPE(X,[M,N]) returns the M-by-N matrix
%   whose elements are taken columnwise from X. An error results
```

```
%   if X does not have M*N elements.
X = reshape(featureSet,numPoints,[]);
% Matriz de carateristicas -> vector de caracteristicas
% 1200x1200x34 -> 1440000x34

%   C = BSXFUN(FUNC,A,B) applies the element-by-element binary operation
%   specified by the function handle FUNC to arrays A and B, with implicit
%   expansion enabled.
% Normalizar las caracteristicas para que tengan media zero y varianza
% unitaria
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));

% Aplicar clustering k means
% iterations = 5;
% s = (numRows/4);% 300 px
% centroids = zeros(16,34);
% k = 1;
% for i = 0:3
%     for j = 0:3
%         centroids(k,:) = X(i*s+(s/2) + j*s+(s/2),:);
%         k = k +1 ;
%     end
% end

% centroids = repmat(centroids, [ 1 1 iterations]);
L = kmeans(X,16,'Replicates',5);

%% MOstrar resultado
L = reshape(L,[numRows numCols]);
figure
imshow(label2rgb(L))
```

Bibliografía

[MatLab 1] Matlab documentación oficial:*https://www.mathworks.com/help/images/texture-segmentation-using-gabor-filters.html*

[Brodatz] Brodatz textures database. *http://multibandtexture.recherche.usherbrooke.ca/images/Original_Brodatz.zip*

[Jain, 1991] A. K. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters",1991