Bauhaus-Universität Weimar

# Image Analysis and Object Recognition

## Assignment 5

## Clustering using k-means and mean shift or normalized-cut

SS 2018

(Course notes for internal use only!)

# Sample solution task A

```
function A_fft_filtering
%        ===============
sigma = 1.4;                                              % standard deviation
Img = double(mean(imread('taskA.png'), 3)) / 255;           % read input image
Nsy = imnoise(Img, 'gaussian', 0, 0.01);                           % add noise
NsyFFT = fft2(Nsy);                                         % Fourier transform

kernel = Gauss1d(sigma)' * Gauss1d(sigma);                % 2D Gaussian kernel
filter = zeros(size(Nsy));
filter(1:size(kernel, 1), 1:size(kernel, 2)) = kernel;        % Filter padding
filter = circshift(filter, -floor(size(kernel)/2));            % Center filter
FilFFT = fft2(filter);                                      % Fourier transform

MulFFT = NsyFFT .* FilFFT;                          % Multiply image with filter
Res = ifft2(MulFFT);                                 % Inverse Fourier transform

figure, imshow(Img), title('Original image');
figure, imshow(Nsy), title('Noisy image');
figure, imagesc(log(abs(fftshift(NsyFFT)))), title('Spectrum of noisy image');
figure, imagesc(log(abs(fftshift(FilFFT)))), title('Spectrum of Gaussian filter');
figure, imagesc(log(abs(fftshift(MulFFT)))), title('Spectrum of filtered image');
figure, imshow(Res), title('Filtered image');


function g = Gauss1d(sigma)
%            ==============
r = round(3*sigma); i = -r:r;
g = exp(-i.^2 / (2*sigma^2)) / (sigma*sqrt(2*pi));
```

# Sample solution task B

```
function B_fourier_descr
%        ===============
train_image = double(mean(imread('trainB.png'), 3)) / 255;     % read train image
[D_t, ~] = fourier_descr(Thresholding(train_image));     % build model descriptor

name = 'test1B.jpg';                                           % first test image
for k = 1:2
    image = double(mean(imread(name), 3)) / 255;                % read test image
    mask = Thresholding(image);                                     % binary mask
    [D, B] = fourier_descr(mask);                          % build all descriptors
    figure, imshow(mask), hold on;
    for i = 1 : size(D, 1)                                   % for each descriptor
        dist = norm(D_t - D(i, :));              % if descriptor is similar to model
        if dist < 0.07
            plot(B{i}(:, 2), B{i}(:, 1), 'r', 'LineWidth', 1);    % plot boundary
        end
    end
    name = 'test2B.jpg';                                      % second test image
end

function [fd, B] = fourier_descr(Img)
%                  ==================
n = 25;                                         % number of descriptor elements
B = bwboundaries(Img);                                 % extract all boundaries
fd = zeros(length(B), n-1);
for i = 1 : length(B)                                       % for each boundary
    if length(B{i}) > n
        desc = fft(B{i}(:,2) + j*B{i}(:,1));        % points as imaginary numbers
        fd(i, :) = abs(desc(2:n) / desc(2));               % normalize descriptor
    end
end

function mask = Thresholding(Image)                        % image thresholding
%              ===================
mask = im2bw(Image, graythresh(Image));
```

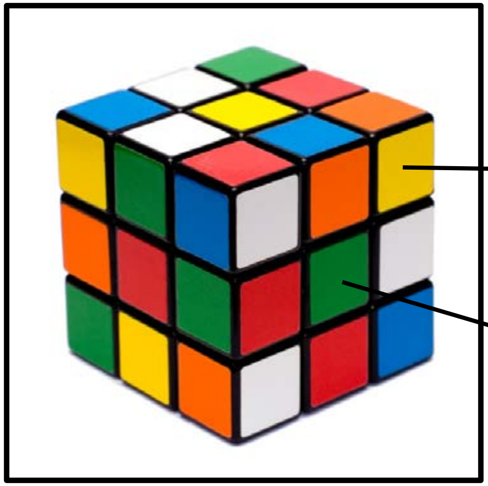
# Assignment 6

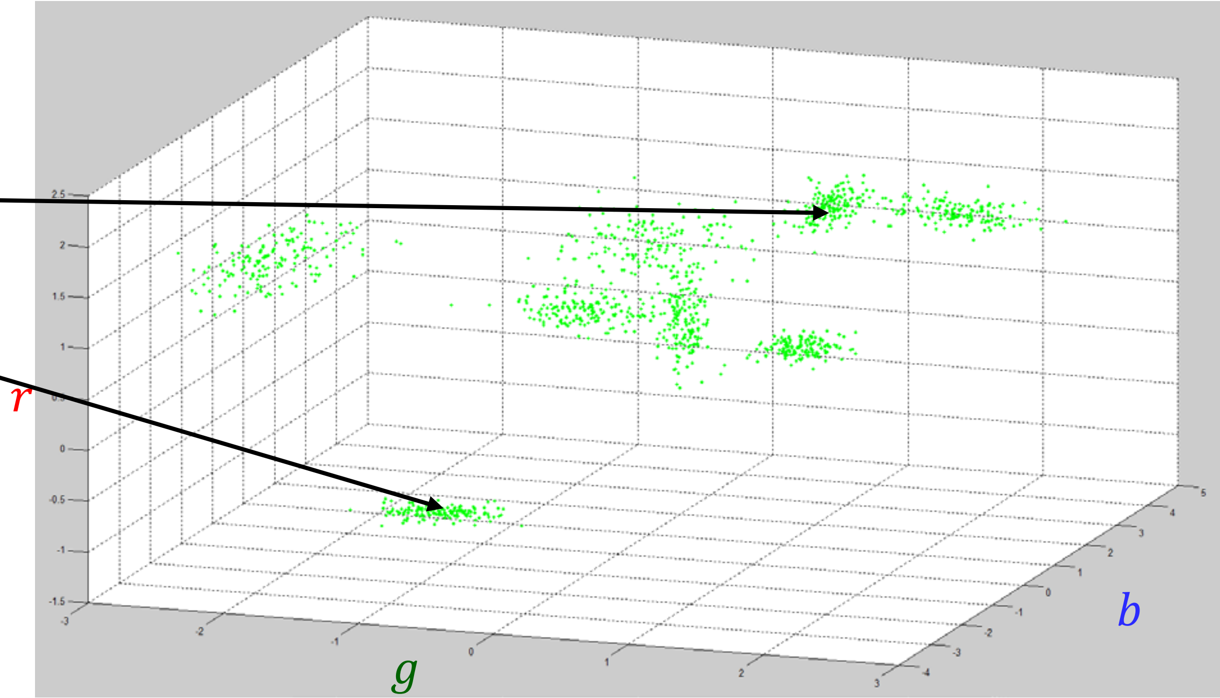
**A:** Clustering using k-means

**B:** Clustering using mean shift or normalized-cut

# Feature space

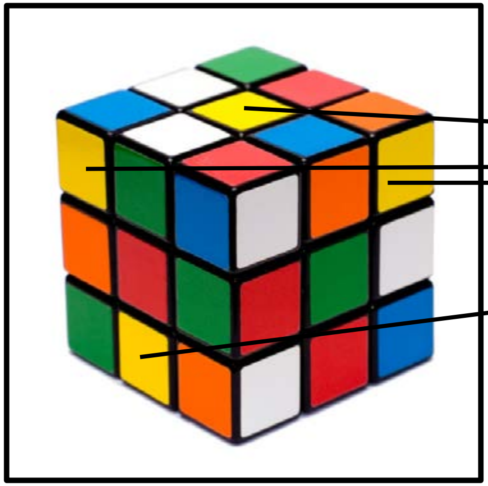
- Given: 3-channel color image
  - Each channel (r, g, b) represents one dimension of a feature space
  - Each pixel of the image maps to a point in that space
  - Additional spatial support is given by the position (x, y) in the image

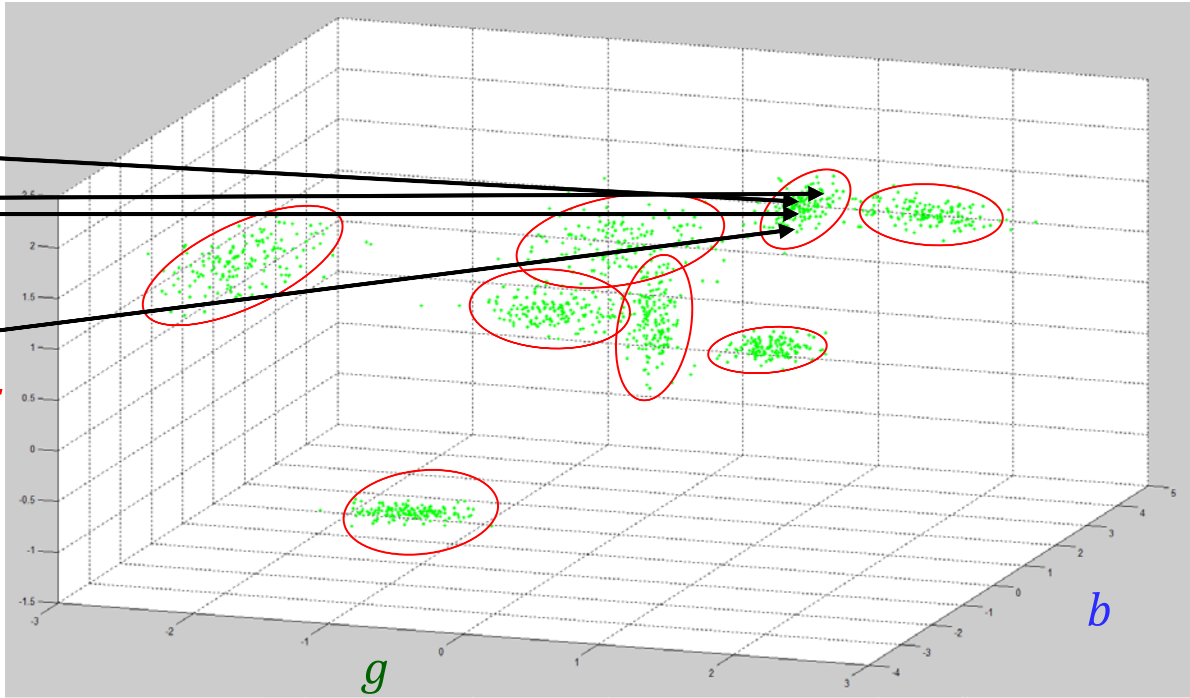




# Idea of clustering

- Group all similar points in feature space to „clusters“
- Each cluster contains pixels with similar spectral properties
  - → Members of a cluster belong to the same segment (or segment class)





# Thank you!