Bauhaus-Universität Weimar

Image Analysis and Object Recognition

Assignment 5

Clustering using Expectation Maximization (EM) and Maximum-Likelihood (ML) image segmentation

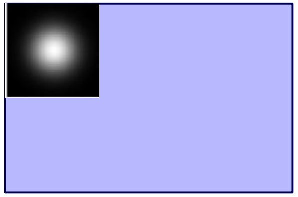
SS 2017

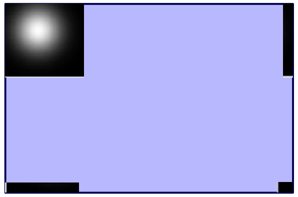
(Course notes for internal use only!)

- Assignment dates:
  - 13.04.2017 → Introduction + Assignment 1
  - 04.05.2017 → Assignment 1+2
  - 18.05.2017 → Assignment 2+3
  - 01.06.2017 → Assignment 3+4
  - **15.06.2017** → Assignment 4+5
  - **29.06.2017** → Assignment 5+6
  - 13.07.2017 → Final Meeting / Summary / Discussion

A4: Filter Computation

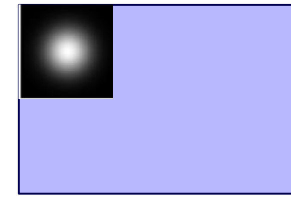
```
[X, Y] = meshgrid(1:N); X-round(N/2); Y-round(N/2);
G = (1/(2*pi*S^2))*exp(-((X).^2+(Y).^2)/(2*S^2));
Filter = zeros(size(Img));
Filter(1:N,1:N) = G;
Filter = circshift(Filter, [-1 -1]);
```

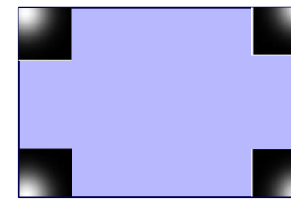




A4: Filter Computation

```
[X, Y] = meshgrid(1:N); X-round(N/2); Y-round(N/2);
G = (1/(2*pi*S^2))*exp(-((X).^2+(Y).^2)/(2*S^2));
Filter = zeros(size(Img));
Filter(1:N,1:N) = G;
Filter = circshift(Filter, round([-N/2 -N/2]));
```

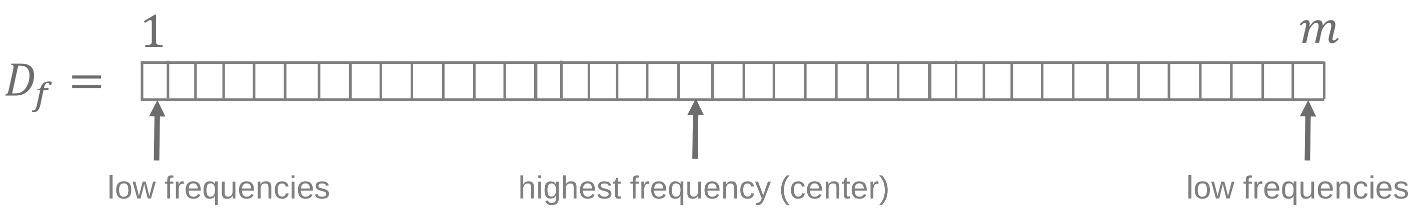




A4: Fourier Descriptor

Generalization of descriptor by using $n = 24$ elements

- Number of elements m in $D = [(x_1 + jy_1), \ldots, (x_m + jy_m)]^T$
- Fourier transform $D_f = fft(D)$

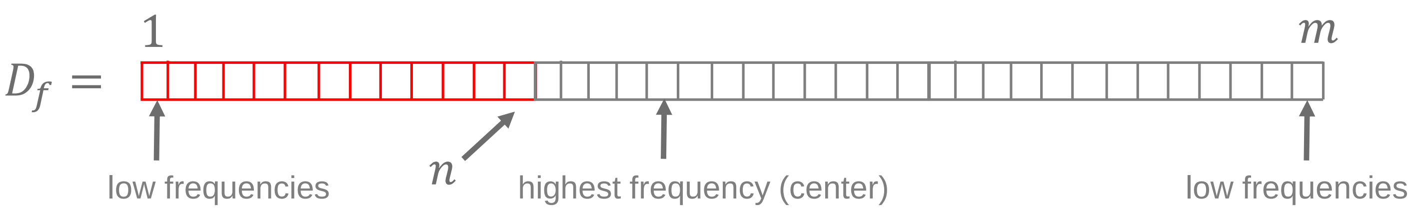


→ **Symmetric vector**

A4: Fourier Descriptor

Generalization of descriptor by using $n = 24$ elements

- Number of elements m in $D = [(x_1 + jy_1), \ldots, (x_m + jy_m)]^T$
- Fourier transform $D_f = fft(D)$



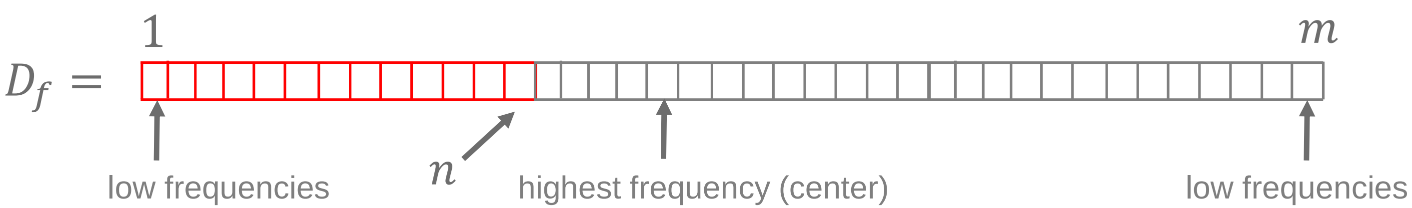
→ **Symmetric vector**

→ Extract first n values → generalization

A4: Fourier Descriptor

Generalization of descriptor by using $n = 24$ elements

- Number of elements m in $D = [(x_1 + jy_1), \dots, (x_m + jy_m)]^T$
- Fourier transform $D_f = fft(D)$



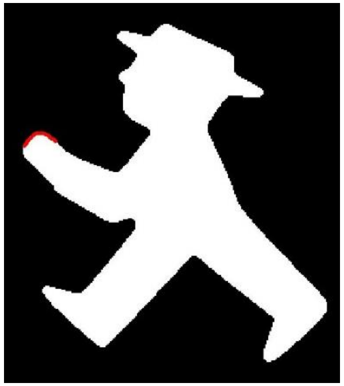
→ **Symmetric vector**

→ Extract first n values → generalization

- Fourier transform $D_f = fft(D, n)$ → **transforms n elements of D!**

$n = 842$

$n = 421$



$n = 24$

Solution A4

```
function taskA
Img = rgb2gray(imread('taskA.png'));
Img = im2double(Img);
Org = fft2(Img);
Img = imnoise(Img,'gaussian',0,0.01);

N = 64;
G = zeros(N);
S = 3;
M = N/2;
[X, Y] = meshgrid(1:N);
G = (1 /(2*pi*S^2))*exp(-((X-M).^2+(Y-M).^2)/
Filter = zeros(size(Img));
Filter(1:N,1:N) = G;
Filter = circshift(Filter, [-N/2 -N/2]);

FilterFFT = fft2(Filter);
ImgFFT = fft2(Img);
figure;
imagesc(log(abs(ImgFFT)));
figure;
imagesc(log(abs(FilterFFT)));
figure();
imagesc(log(abs(Org)));
Res = ImgFFT.*FilterFFT;
figure();
imagesc(log(abs(Res)));
figure();
imshow(ifft2(Res));
end
```

```
function TaskB
Img = im2bw(rgb2gray(imread('trainingB.png')), 0.1);
B = bwboundaries(Img);
D = complex(B{1}(:,1),B{1}(:,2));
Df = fft(D);
Df = Df(2:size(Df, 1));
Df = Df / abs(Df(1));
Df = abs(Df);
Df24 = Df(1:24);

ImgName = 'test1B.jpg';
for i = 1:2
    Img = im2bw(rgb2gray(imread(ImgName)), 0.15);
    imshow(imread(ImgName));
    B = bwboundaries(Img);
    for k = 1:length(B)
        Dk = complex(B{k}(:,1),B{k}(:,2));
        Dfk = fft(Dk);
        Dfk = Dfk(2:size(Dfk, 1));
        Dfk = Dfk / abs(Dfk(1));
        Dfk = abs(Dfk);
        if(size(Dfk, 1) >= 24)
            Dfk24 = Dfk(1:24);
            Dis = norm(Df24 - Dfk24);
            if(Dis < 1)
                hold on;
                plot(B{k}(:,2),B{k}(:,1),'b','LineWidth',3);
            end
        end
    end
    if(i == 1)
        figure;
    end
    ImgName = 'test2B.jpg';
end
end
```

Bauhaus-Universität Weimar

Assignment 6

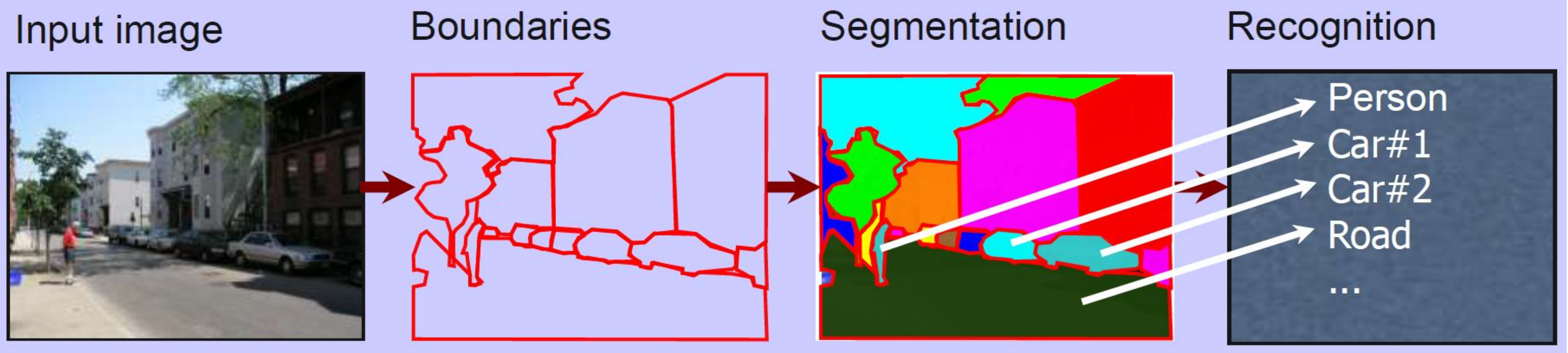
A: Expectation Maximization (EM) Clustering

B: Maximum Likelihood (ML) image segmentation (**Application Task**)

Motivation

Segmentation

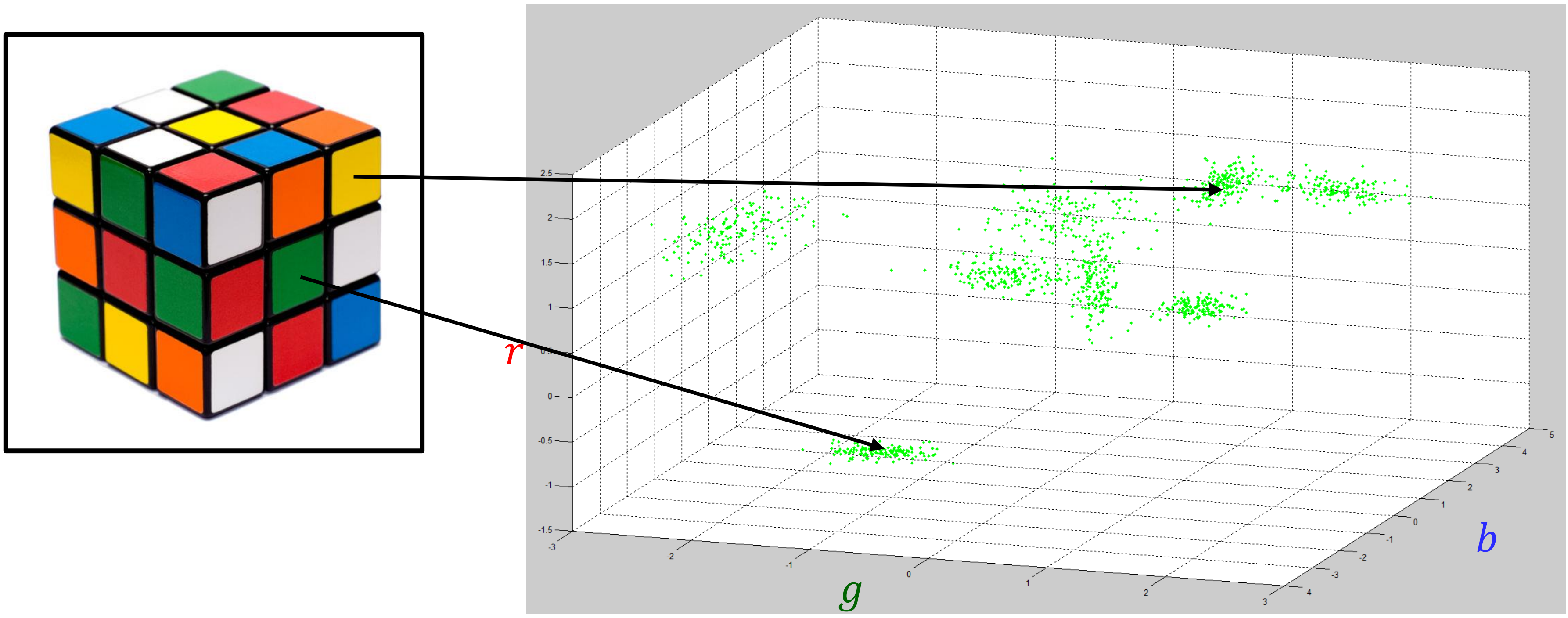
- Partitioning an image into a collection of connected set of pixels



- Needed for understanding the content of a scene

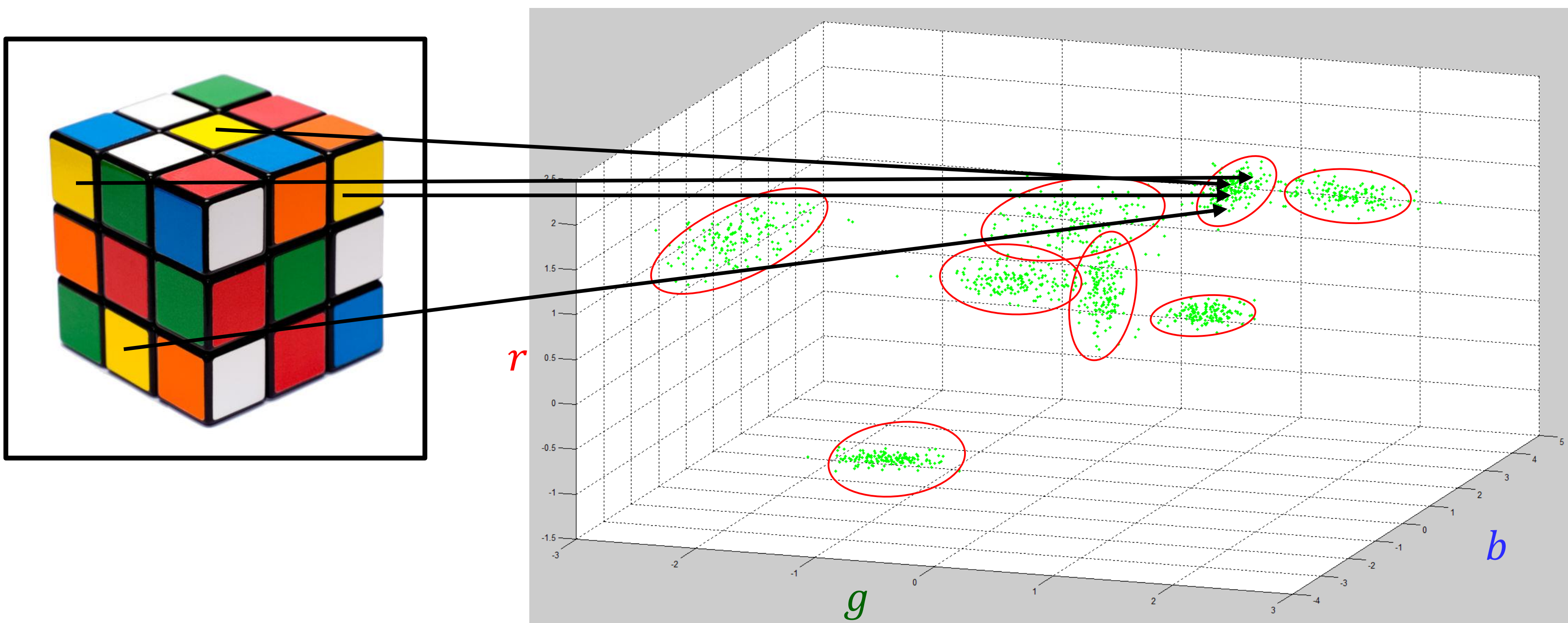
Feature Spaces

- Given: 3-channel image (rgb)
  - Each channel represents one dimension of a feature space
  - Each pixel of the image maps to a point in that space



Idea of Clustering

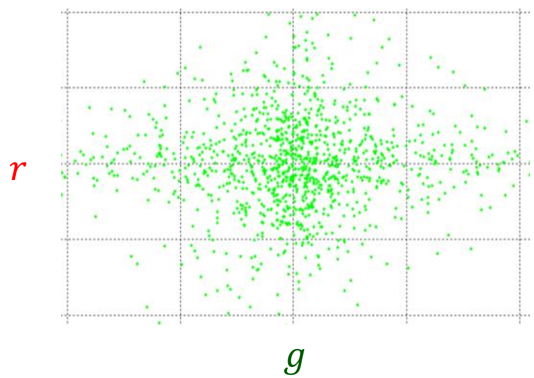
- Group all similar points in feature space to „clusters“
- Each cluster contains pixels with similar spectral properties
  → Members of a cluster belong to the same segment or segment class



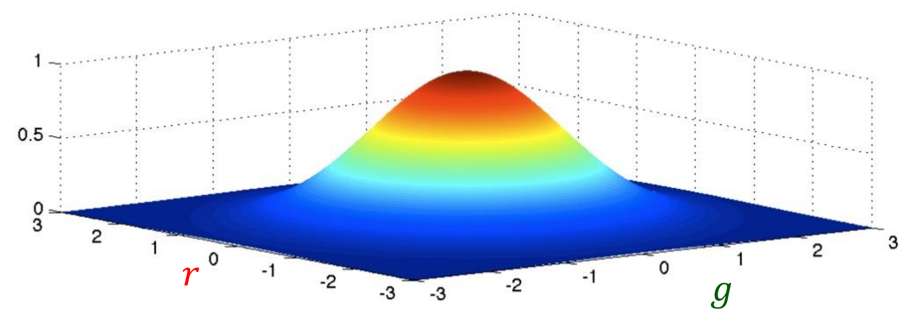
Parameterization of Data

- **Cluster representation** using multivariate Gaussian functions

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} \cdot |\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$



$\longrightarrow$ $p(x|\mu,\Sigma)$



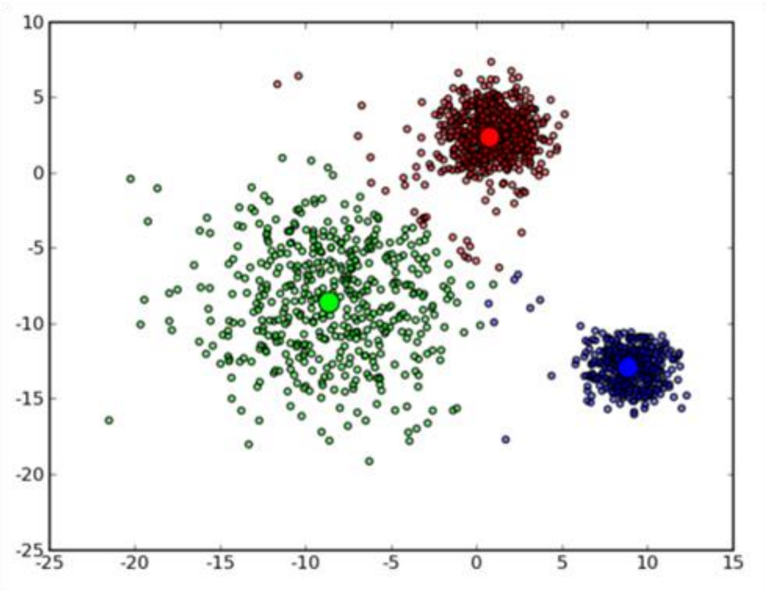
- $p(x|\mu,\Sigma)$: „Probability of x given the patameters μ,Σ“
- x: Feature vector, e.g. rg-information of a pixel $x = (r, g)$
- $\boldsymbol{\mu}$: **Center of the Gaussian function, here** $\boldsymbol{\mu} = (\mathbf{0}, \mathbf{0})$
- $\boldsymbol{\Sigma}$: **Covariance matrix (symmetric), here of size** $\mathbf{2} \times \mathbf{2}$ **(2 dimensions)**
- d: Number of feature-space dimensions, here $d = 2$
- $|\Sigma|$: Determinant of the covariance matrix
- Σ^{-1}: Inverted covariance matrix

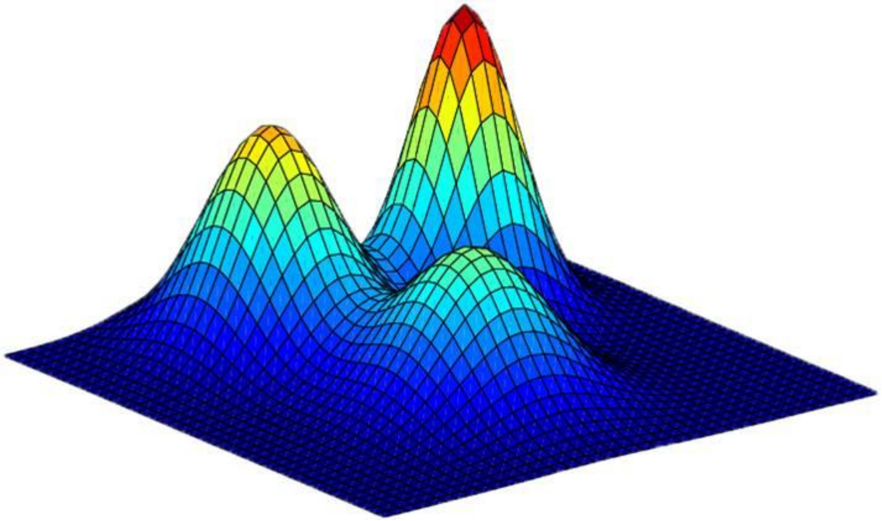
Gaussian Mixture Models

- Mixture-Model: Weighted sum of N_c elementary Gaussians

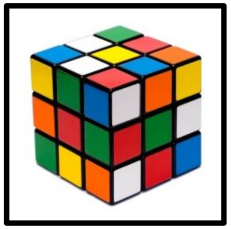
$$p(x|\Omega) = \sum_{c=1}^{N_c} \alpha_c \, p(x|\mu_c, \sigma_c), \qquad \Omega = \{\alpha_1, \mu_1, \sigma_1, \dots, \alpha_{N_c}, \mu_{N_c}, \sigma_{N_c}\}$$

- $N_c = 3$ example in 2D:

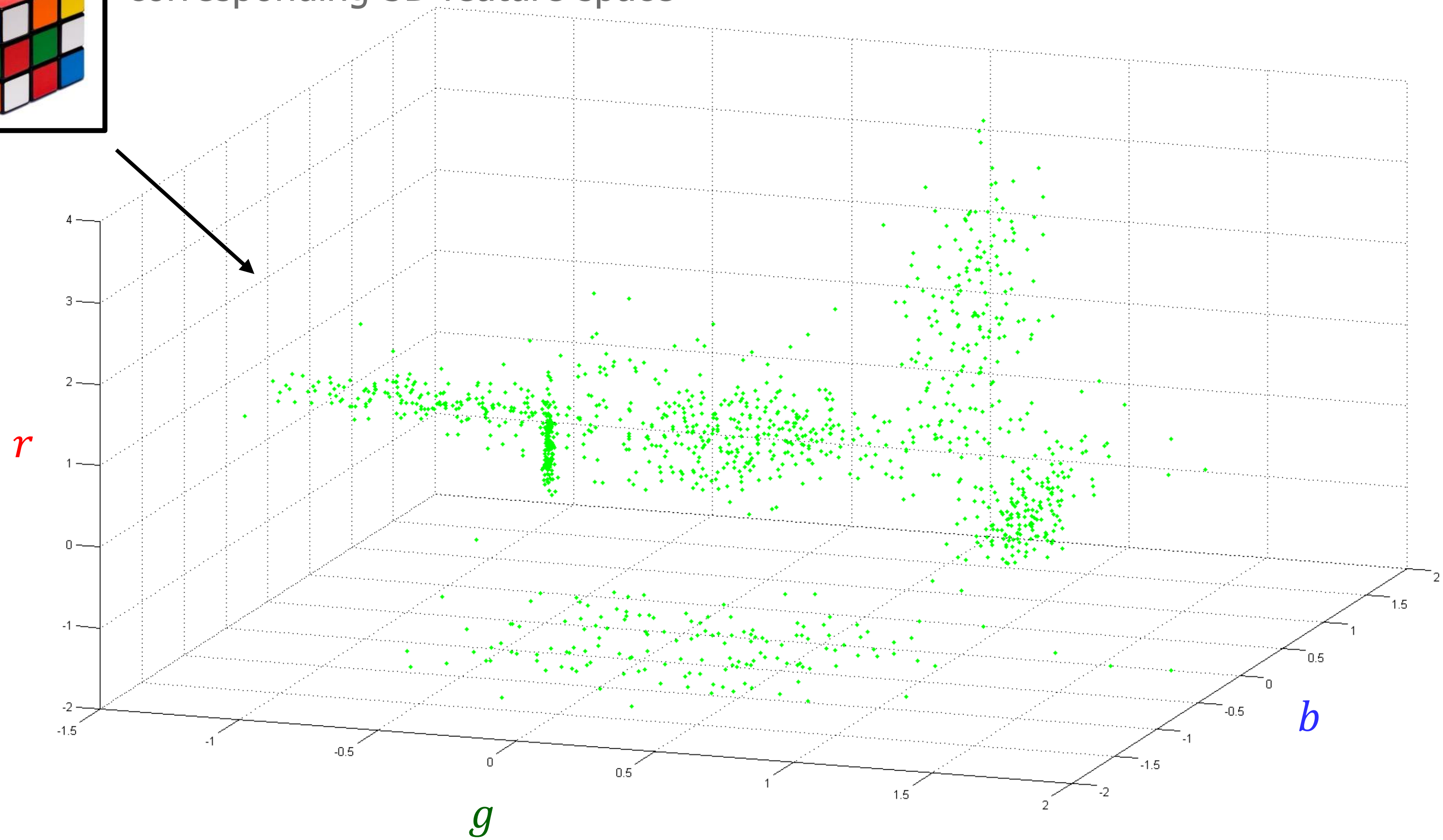




3D Gaussian Mixture Model Example



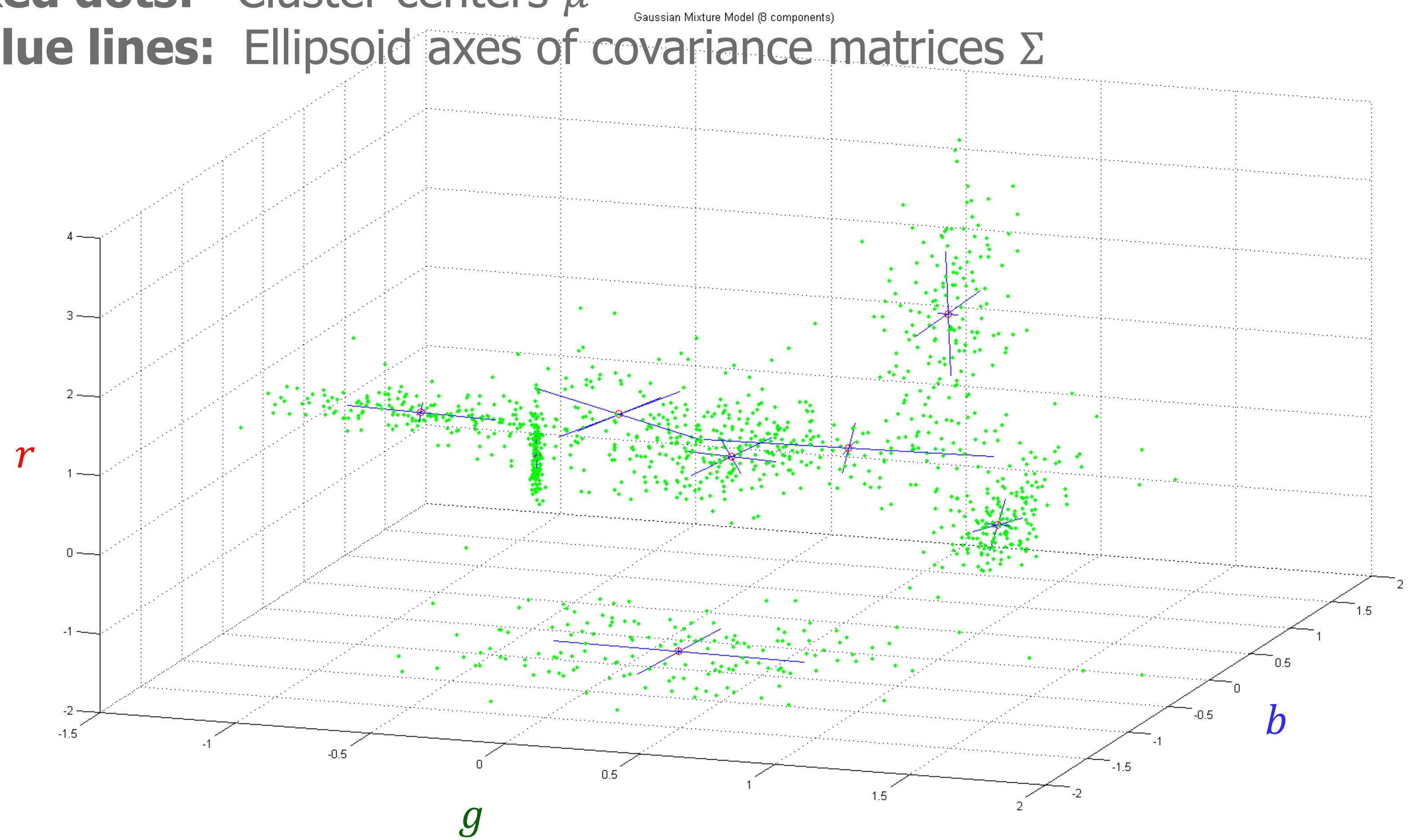
Remember: Each pixel of an rgb-image represents a point in the corresponding 3D feature space



3D Gaussian Mixture Model Example

Red dots: Cluster centers μ

Blue lines: Ellipsoid axes of covariance matrices Σ



Parameter Estimation for GMMs

- **Given**: d-dimensional feature space consisting of N_x feature vectors $\{x_1, \dots, x_{N_x}\}$

- **Wanted**: Set of parameters $\Omega = \{\alpha_1, \mu_1, \Sigma_1, \dots, \alpha_{N_c}, \mu_{N_c}, \Sigma_{N_c}\}$ for mixture model with N_c components
  → α_c: Weight for a cluster, where $\sum_{c=1,\dots,N_c} a_i = 1.0$
  → μ_c: Mean vector for a cluster
  → Σ_c: Covariance matrix for a cluster

- **Approach**: Use probabilities

$$p(y_i = c | x_i, \Omega_c), \qquad y_i \epsilon \{1, \dots, N_c\},$$

which describe the membership of a feature space point x_i to a cluster y_i
  → This enables parameter estimation!!

Parameter Estimation

Expectation-Maximization (EM):

- Iterative Algorithm (chicken-egg problem!)

E-Step: computation of $p(y_i = c|x_i, \Omega)$ based on Ω^k

Initialization $\Omega^0 \longrightarrow$ $\Omega = \{\alpha_1, \mu_1, \Sigma_1, \dots, \alpha_{N_c}, \mu_{N_c}, \Sigma_{N_c}\}$

$p(y_i = c|x_i, \Omega)$

M-Step: computation of Ω^{k+1}

- **E-Step:** Compute probabilities $p(y_i = c|x_i, \Omega^k)$ for each vector x using the current model parameters Ω^k
- **M-Step:** Computation of new model parameters Ω_c^{k+1} using the probabilities $p\left(y_i = c|x_i, \Omega_c^k\right)$ from E-Step

E-Step

Compute membership probabilities for each vector x_i and component c using the current model parameters Ω^k

„Prior probability" (scalar value: weight)

„Likelihood" (Gaussian function)

$$p\left(y_i = c \middle| x_i, \Omega_c^k\right) = \frac{\alpha_c p\left(x_i \middle| \mu_c^k, \Sigma_c^k\right)}{\sum_{j=1}^{N_c} \alpha_j p\left(x_i \middle| \mu_j^k, \Sigma_j^k\right)}$$

„Evidence" (normalization)

- Bayes' theorem
  - „Probability of a cluster c given x_i and the parameters Ω_c"
- Result: Matrix of size $N_c \times N_x$
  - N_c: Number of clusters
  - N_x: Number of feature vectors

Computations in E-Step 1/2

Computation of: $p(y_i = c|x_i, \Omega_c^k) = \frac{\alpha_c p(x_i|\mu_c^k, \Sigma_c^k)}{\sum_{j=1}^{N_c} \alpha_j p(x_i|\mu_j^k, \Sigma_j^k)}$

numerator

denominator

Problem: Values $p(x_i|\mu_c^k, \Sigma_c^k)$ often very small → computational problems!

→ Use log-values: $\log\left(p(y_i = c|x_i, \Omega_c^k)\right)$

→ Gaussian parametrization

$$p(x_i|\mu_c, \Sigma_c) = \frac{1}{(2\pi)^{\frac{d}{2}} \cdot |\Sigma_c|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(x_i - \mu_c)^T {\Sigma_c}^{-1}(x_i - \mu_c)\right)$$

→ The log-value for the numerator is:

$$log\left(\alpha_c \cdot p(x_i|\mu_c^k, \Sigma_c^k)\right) = log(\alpha_c) - \frac{1}{2}\left[log(|\Sigma_c^k|) + (x_i - \mu_c^k)^T {\Sigma_c}^{-1}(x_i - \mu_c^k)\right] + t$$

→ Value t constant → can be ignored in all computations

Computations in E-Step 2/2

Computation of: $\log\left(p(y_i = c|x_i, \Omega_c^k)\right) = \log\left(\frac{\alpha_c p(x_i|\mu_c^k, \Sigma_c^k)}{\sum_{j=1}^{N_c} \alpha_j p(x_i|\mu_j^k, \Sigma_j^k)}\right)$

$$= \underbrace{\log\left(\alpha_c p(x_i|\mu_c^k, \Sigma_c^k)\right)}_{\text{numerator}} - \underbrace{\log\left(\sum_{j=1}^{N_c} \alpha_j p(x_i|\mu_j^k, \Sigma_j^k)\right)}_{\text{denominator}}$$

- The log-value for the numerator is:

$$log\left(\alpha_c \cdot p(x_i|\mu_c^k, \Sigma_c^k)\right) = log(\alpha_c) - \frac{1}{2}\left[log\left(\left|\Sigma_c^k\right|\right) + \left(x_i - \mu_c^k\right)^T {\Sigma_c}^{-1}\left(x_i - \mu_c^k\right)\right]$$

- Computation of denominator: no rule for sum of log-values ☹
- Use numerator in the following way:

$$Num_{c,i} = log\left(\alpha_c \cdot p(x_i|\mu_c^k, \Sigma_c^k)\right)$$

$$Denom_i = \log\left(\sum_{j=1}^{N_c} \alpha_j p(x_i|\mu_j^k, \Sigma_j^k)\right) = log\left(\sum_{j=1}^{N_c} \exp(Num_{j,i})\right)$$

- Final result: $log\left(p(y_i = c|x_i, \Omega_c^k)\right) = Num_{c,i} - Denom_i$

M-Step

Compute new model parameters Ω^{k+1} for each cluster c based on the probabilities $p\left(y_i = c | x_i, \Omega_c^k\right)$ of E-Step **(no *log*-values!)**:

$$N_p = \sum_{i=1}^{N_x} p\left(y_c = c | x_i, \Omega_c^k\right)$$

→ Number of feature points in cluster c

$$\alpha_c = \frac{N_p}{N_x}$$

→ Weight for each cluster

$$\mu_c^{k+1} = \frac{1}{N_p} \sum_{i=1}^{N_x} x_i p\left(y_i = c | x_i, \Omega_c^k\right)$$

→ Mean vector (centroid) for each cluster

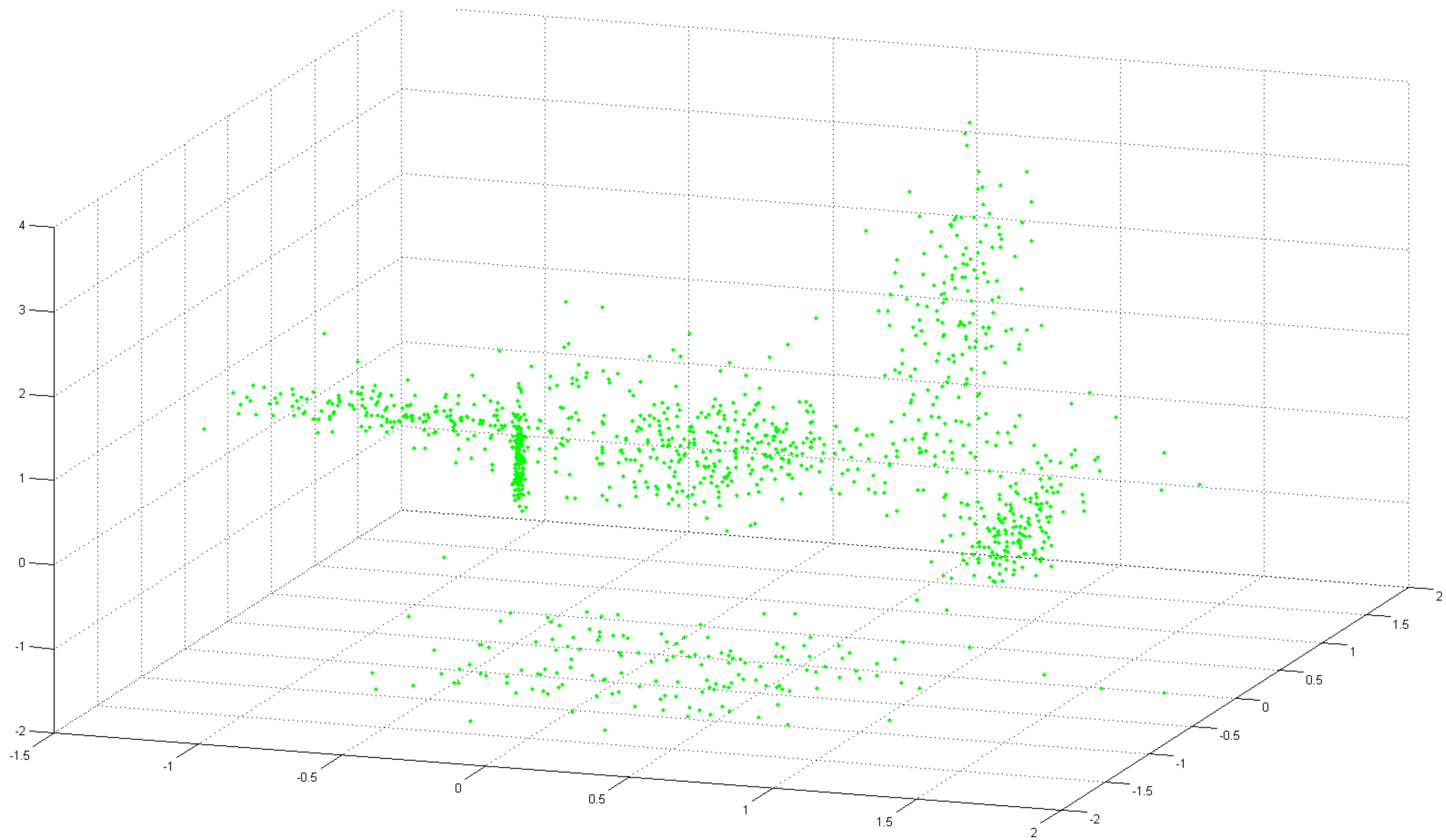
$$\Sigma_c^{k+1} = \frac{1}{N_p} \sum_{i=1}^{N_x} \left(x_i - \mu_c^{k+1}\right)\left(x_i - \mu_c^{k+1}\right)^T p\left(y_i = c | x_i, \Omega_c^k\right)$$

→ Covariance matrix for each cluster

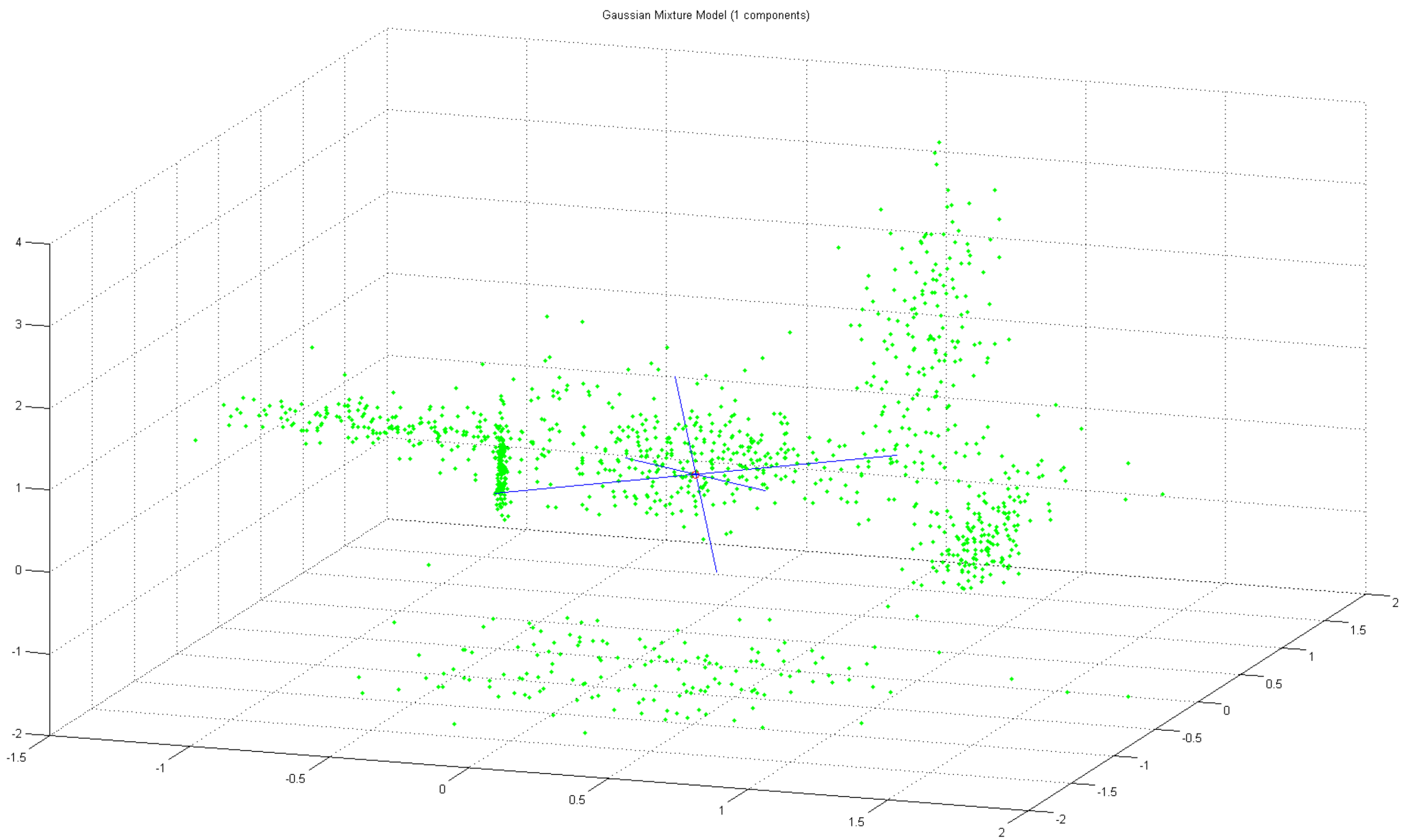
Model Initialization

- EM derived using iterative gradient climbing
  - Converges to a **local** maximum
  - Quality of model depends on initialization
- Often applied: Random initialization
  - Estimation repeated several times
  - Problematic when number of components is large
- **Here**: Iterative splitting of components
  - Algorithm starts with one component (trivial)
  - After a local maximum is found a new component is added by splitting the „biggest“ cluster into two new ones
  - Repeated, till the desired number of components is reached

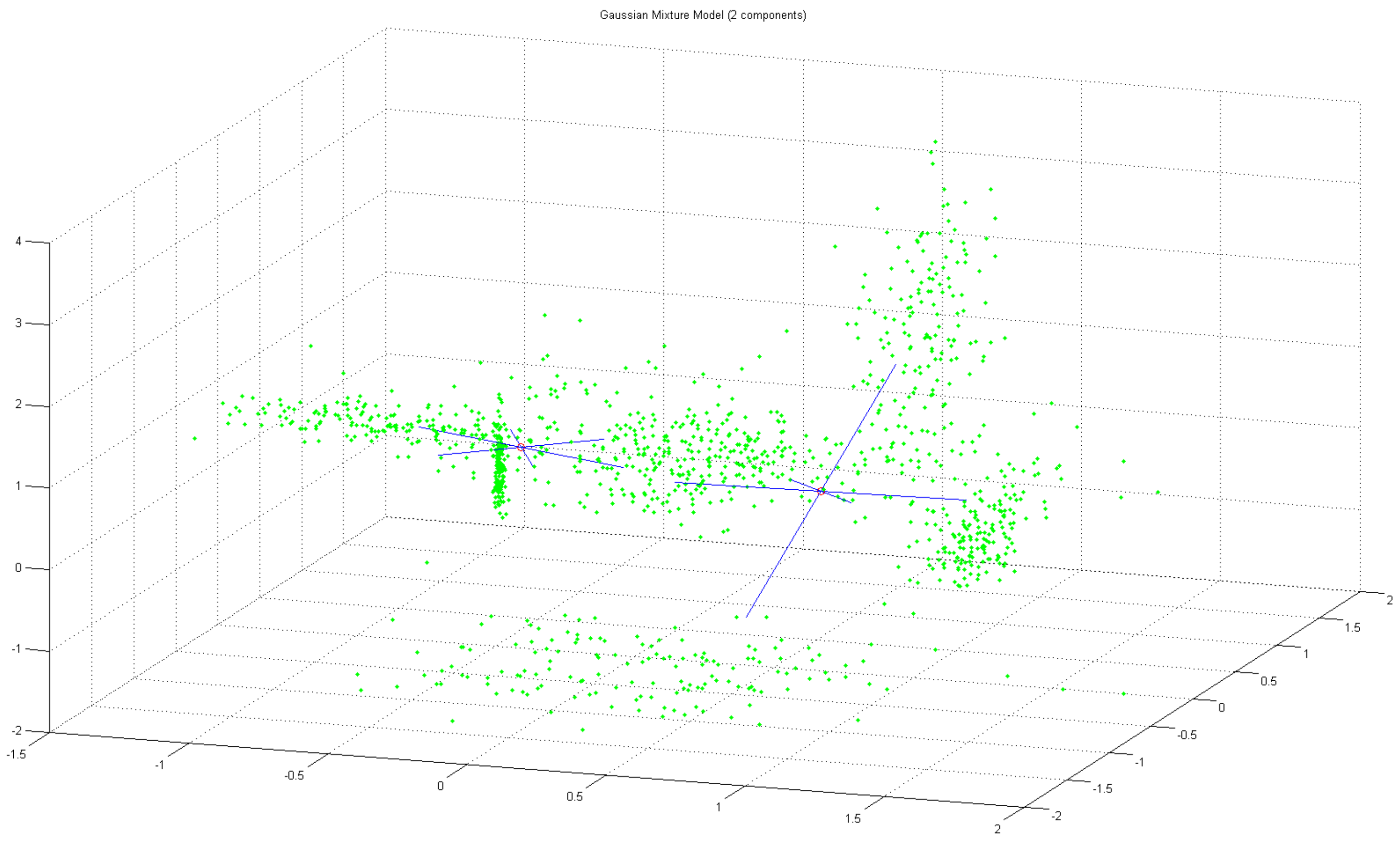
Model Initialization: Feature Space



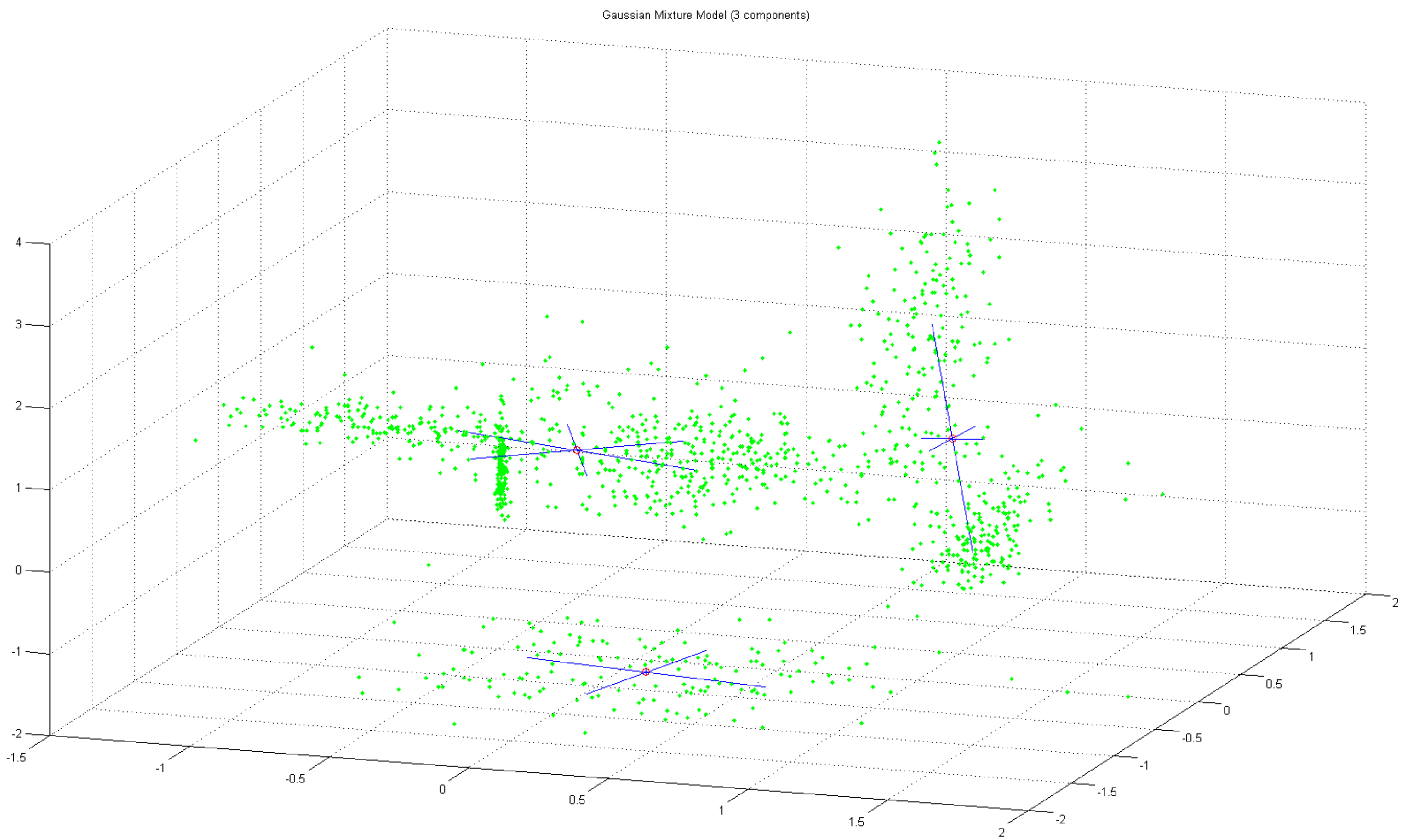
Result for 1 Cluster



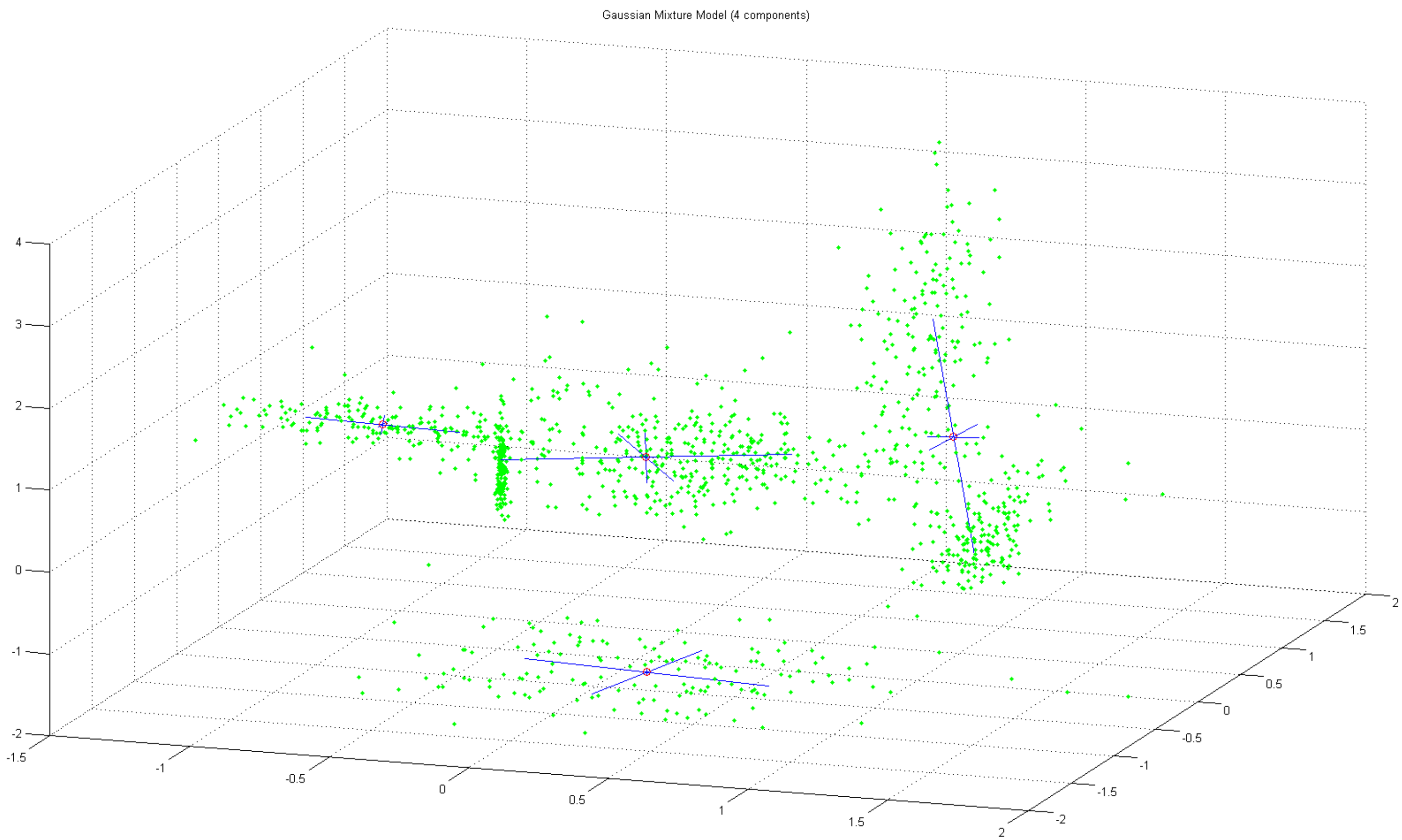
Result for 2 Cluster



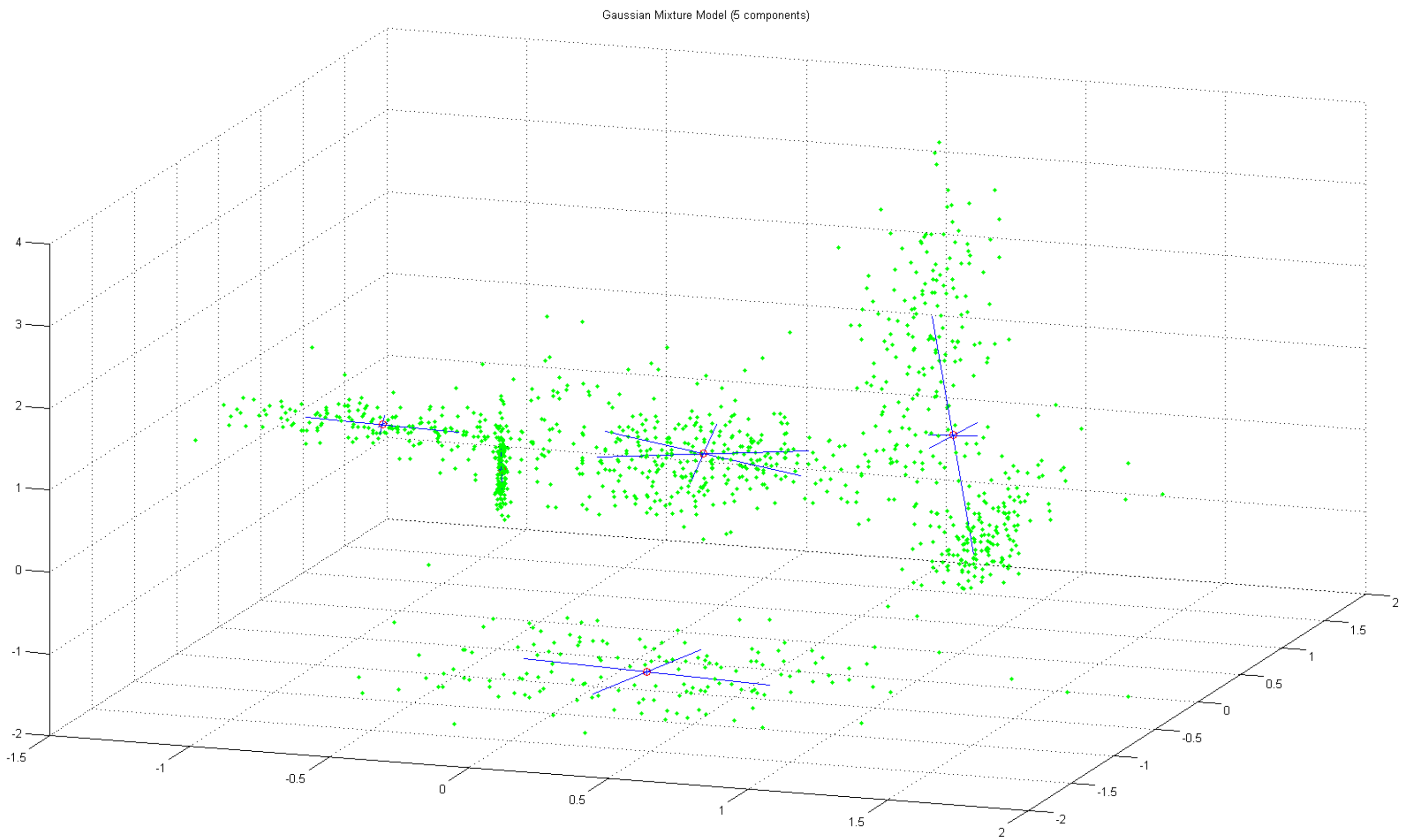
Result for 3 Cluster



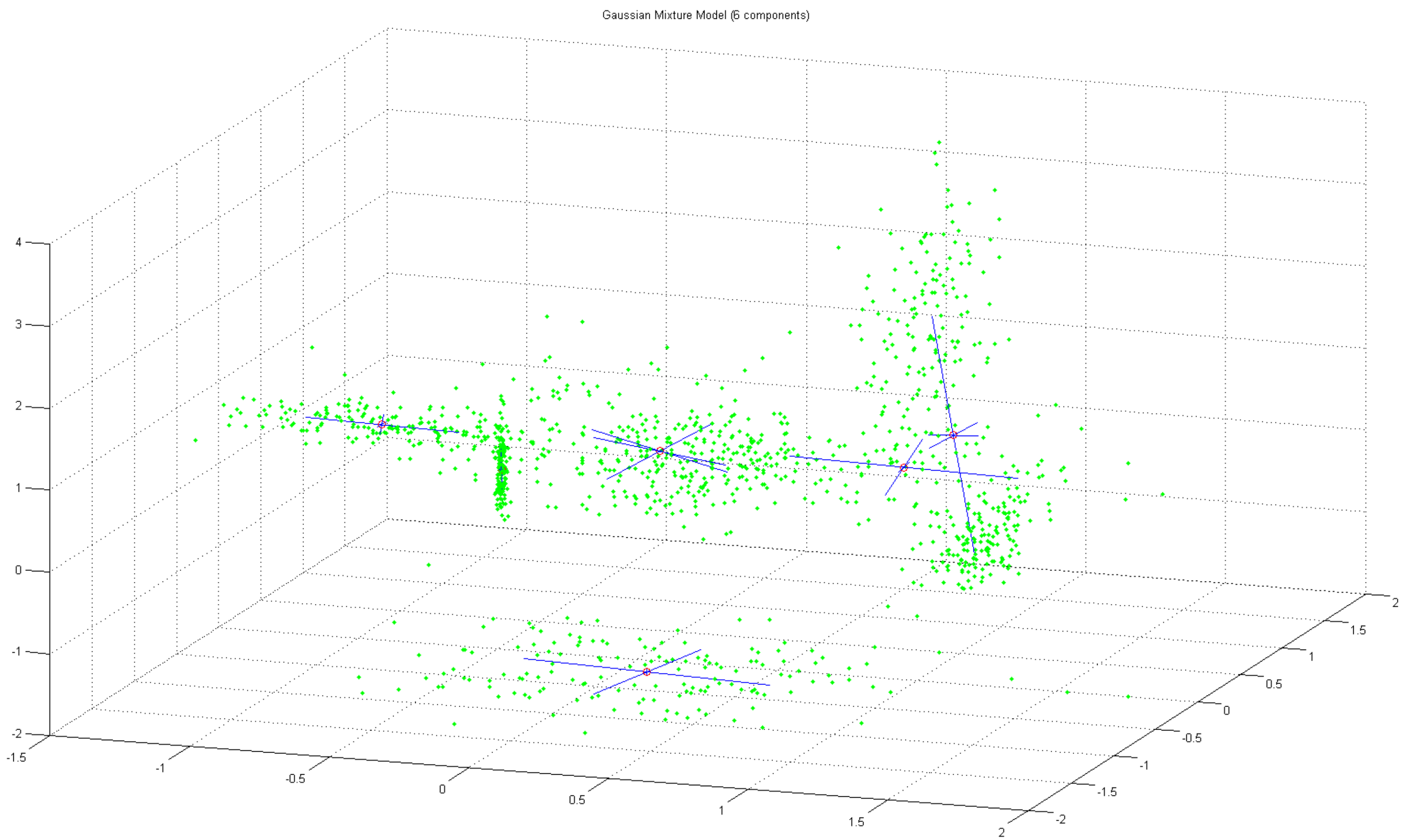
Result for 4 Cluster



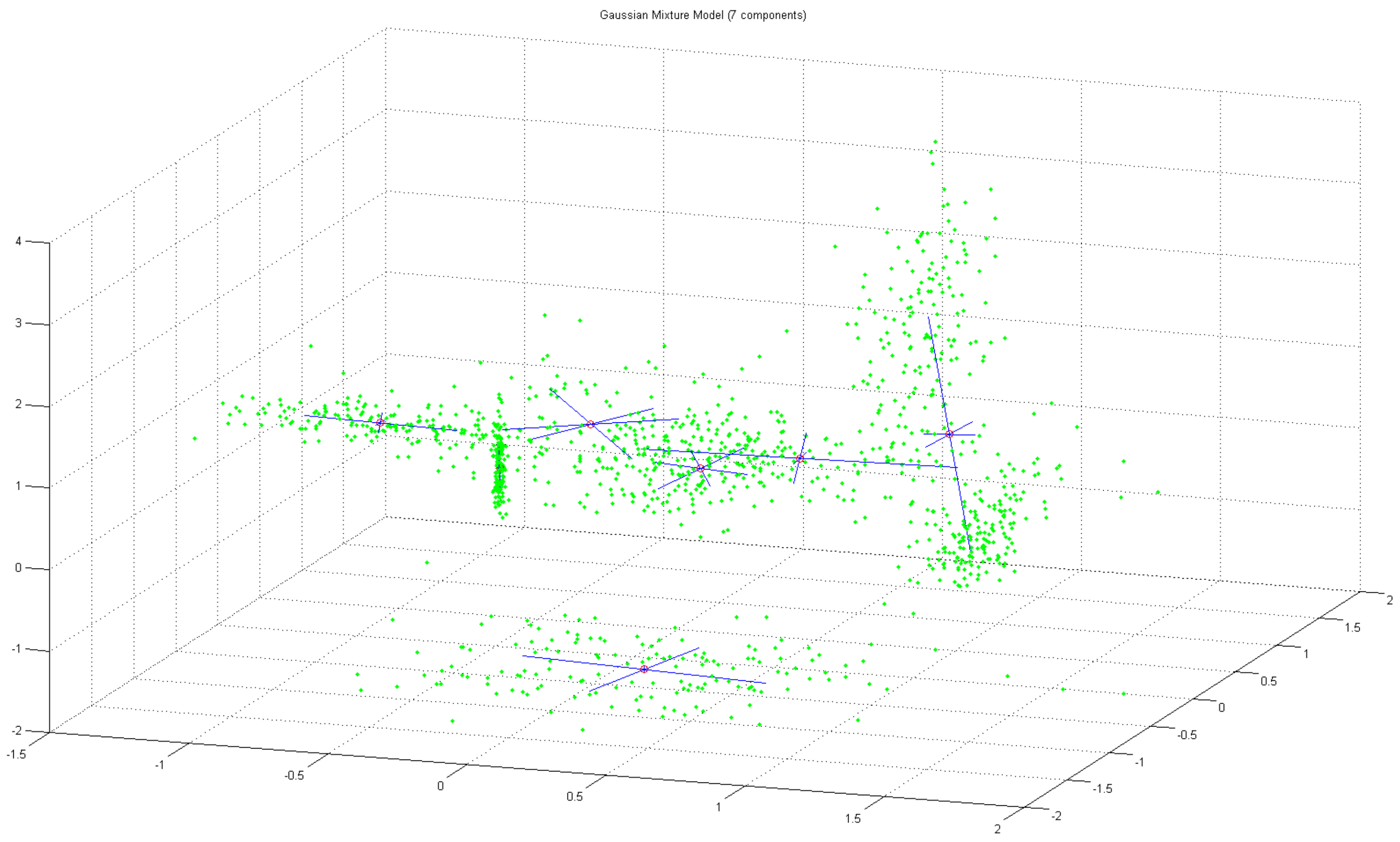
Result for 5 Cluster



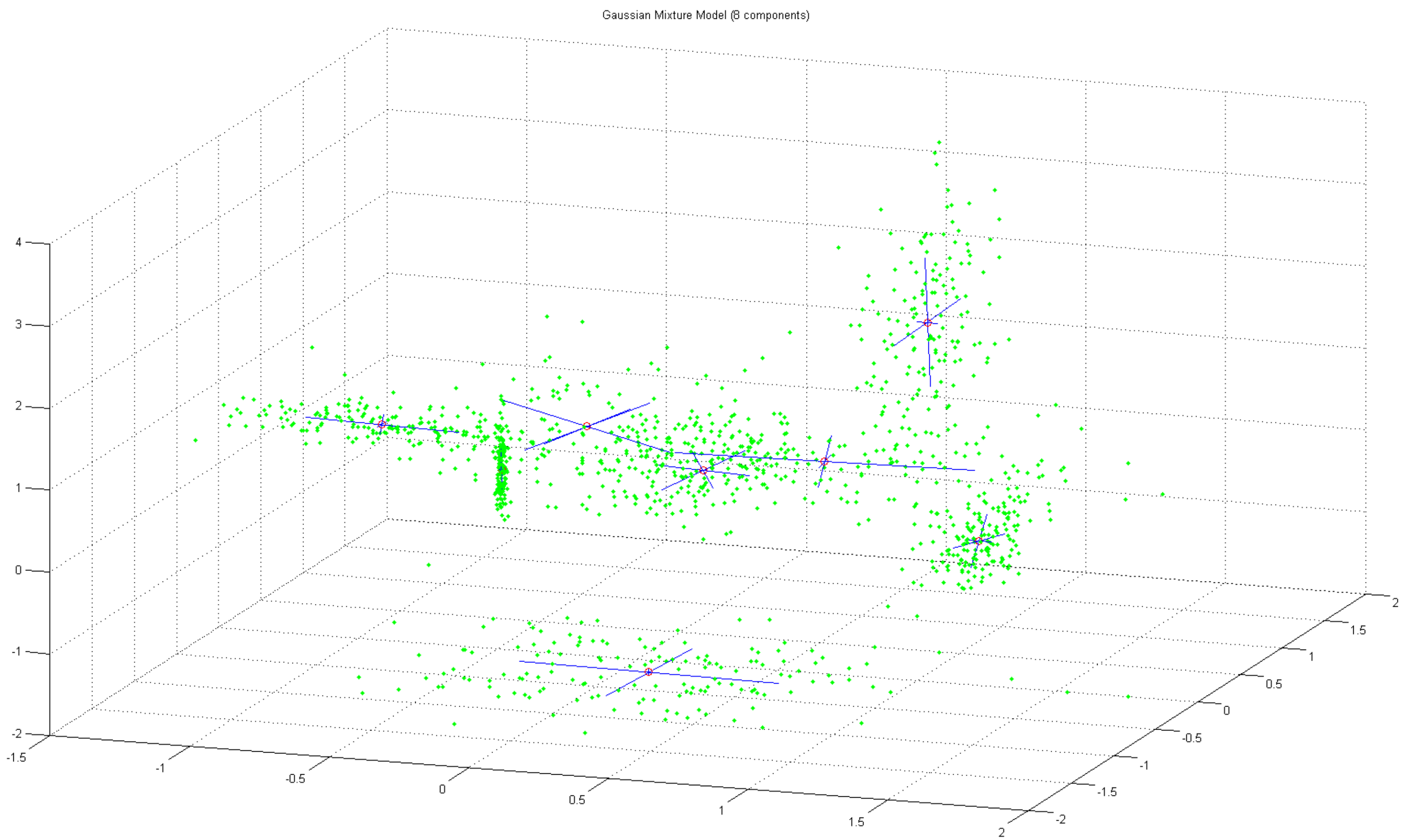
Result for 6 Cluster



Result for 7 Cluster



Result for 8 Cluster



Algorithm Outline (Just an information)

- **Initialization of $\boldsymbol{\Omega_0}$:** $\mu \leftarrow [0,0,0]^T$, $\Sigma \leftarrow \mathrm{I}$ (identity matrix)
- **For** $c = 1, \dots, N_c$
  - $lastPX \leftarrow -\infty$
  - **While** $(\log(PX) - lastPX > \varepsilon)$
    - $lastPX \leftarrow PX$
    - $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Omega})$ $\leftarrow$**E-Step**
    - Ω $\leftarrow$**M-Step**
  - **Endwhile**
  - Ω $\leftarrow$Initialize a new component
- **Endfor**

$$PX: p(\boldsymbol{x}|\boldsymbol{\Omega}) = \prod_{i=1}^{N_x} \sum_{c=1}^{N_c} \alpha_i p(x_i|\Omega_c)$$

→ Scalar value: Probability of a dataset $\boldsymbol{x}$

→ This value will be maximized during EM

→ Used for decision when to stop iterating

Task A

Algorithm is already implemented, but there are 3 functions missing. **Implement these functions:**

a. Function

LnVectorProb = ***CalcLnVectorProb***(*model*, *trainVect*);

calcutates

$$log\left(\alpha_c \cdot p\left(x_i|\mu_c^k, \Sigma_c^k\right)\right) = log(\alpha_c) - \frac{1}{2}\left[log\left(\left|\Sigma_c^k\right|\right) + (x_i - \mu_c)^T {\Sigma_c}^{-1}(x_i - \mu_c)\right]$$

→ *Log*-probability of all feature vectors in all components

→ *LnVectorProb:* $N_c \times N_x$ array containing all possible $\log\left(\alpha_c p\left(x_i|\mu_c^k, \Sigma_c^k\right)\right)$

→ Needed in E-step and also for other computations

Task A

Algorithm is already implemented, but there are 3 functions missing. **Implement these functions:**

b. Function

LnCompProb = ***GmmEStep****(model, trainVect)*

calcutates

$$\log\left(p\left(y_i = c | x_i, \Omega_c^k\right)\right) = \log\left(\frac{\alpha_c p\left(x_i | \mu_c^k, \Sigma_c^k\right)}{\sum_{j=1}^{N_c} \alpha_j p\left(x_i | \mu_j^k, \Sigma_j^k\right)}\right)$$

→ Also $N_c \times N_x$ array

→ Use ***CalcLnVectorProb*** here (*model, trainVect* only needed for this call!)

→ Use *log*-values for computation and for the result!

→ *LnCompProb* needed in M-step (function *GmmMStep*)

Task A

Algorithm is already implemented, but there are 3 functions missing. **Implement these functions:**

c. Function

model = ***GmmMStep****(model, trainVect, LnCompProb)*

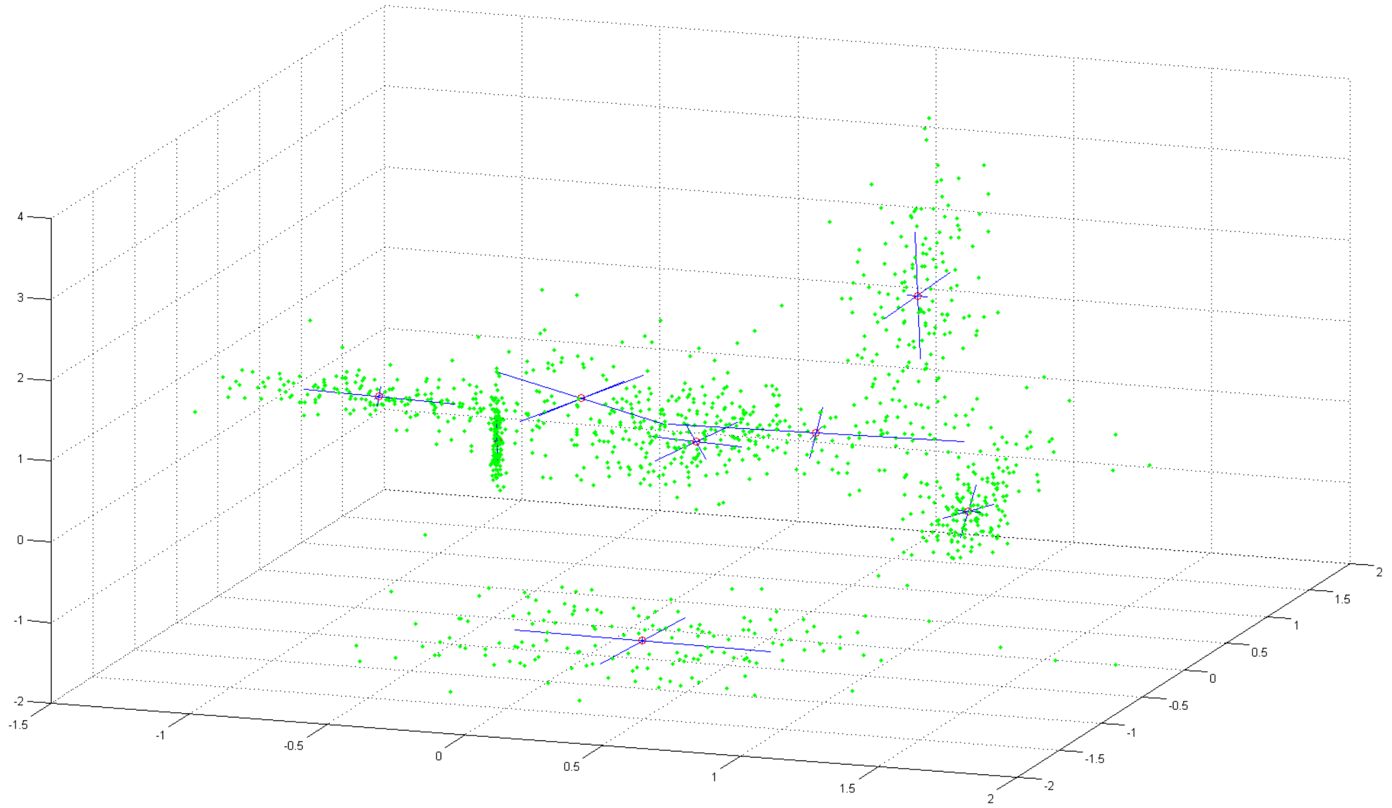
→ calcutates the new model parameters Ω^{k+1} (see slide 22)

→ Use *exp*-function to convert the values $\log\left(p\left(y_i = c|x_i, \Omega_c^k\right)\right)$ (*LnCompProb*) before computations

Task A

d. Test your implementation using provided function *TestGaussMixEM:*

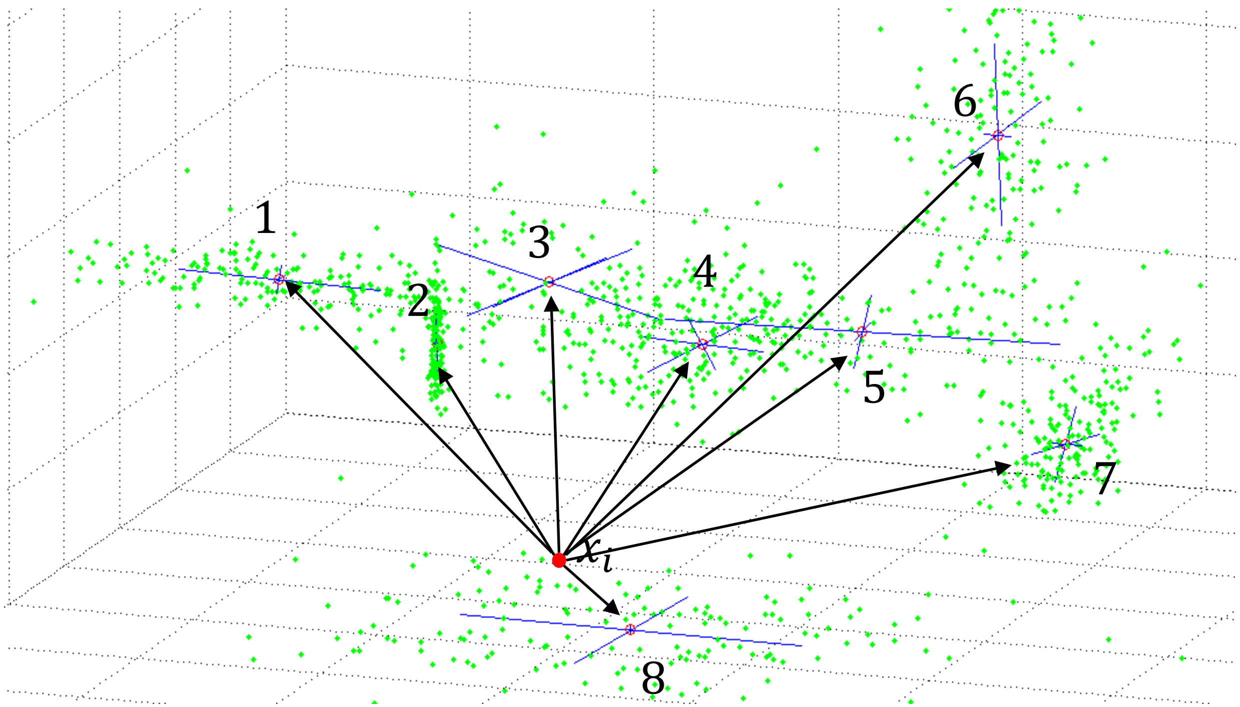
- *Generates* n_{comp} normally distributed point clouds
- Applys *LearnGaussMixModel.m*
- Example Result:



Task B: ML-Image Segmentation

Idea: computation of *log*-Likelihood $log\left(\alpha_c p(x_i|\mu_c,\Sigma_c)\right)$ for each pixel (feature vector)
→ store the index of the cluster with the maximum value for each pixel

$$\log\left(\alpha_1 p(x_i|\mu_1,\Sigma_1)\right) = -130$$
$$\log\left(\alpha_2 p(x_i|\mu_2,\Sigma_2)\right) = -190$$
$$\log\left(\alpha_3 p(x_i|\mu_3,\Sigma_3)\right) = -110$$
$$\log\left(\alpha_4 p(x_i|\mu_4,\Sigma_4)\right) = -90$$
$$\log\left(\alpha_5 p(x_i|\mu_5,\Sigma_5)\right) = -91$$
$$\log\left(\alpha_6 p(x_i|\mu_6,\Sigma_6)\right) = -221$$
$$\log\left(\alpha_7 p(x_i|\mu_7,\Sigma_7)\right) = -111$$
$$\log\left(\alpha_8 p(x_i|\mu_1,\Sigma_8)\right) = -16$$



→ According to ML-criterion, pixel x_i is member of cluster 8!

Task B: ML-Image Segmentation

a. **Copy** your implemented function *CalcLnVectorProb* from Task A into the given file *ApplyGaussMixEM.m*

→ Needed for the ML-Segmentation

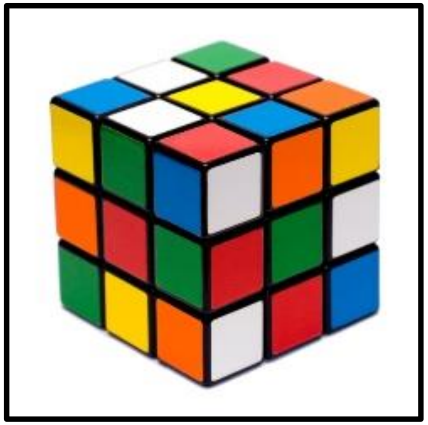
b. **Run** *ApplyGaussMixEM* and use the provided image *inputEx6.jpg* as input

→ Generates a subset of feature vectors from image

→ Learns a GMM

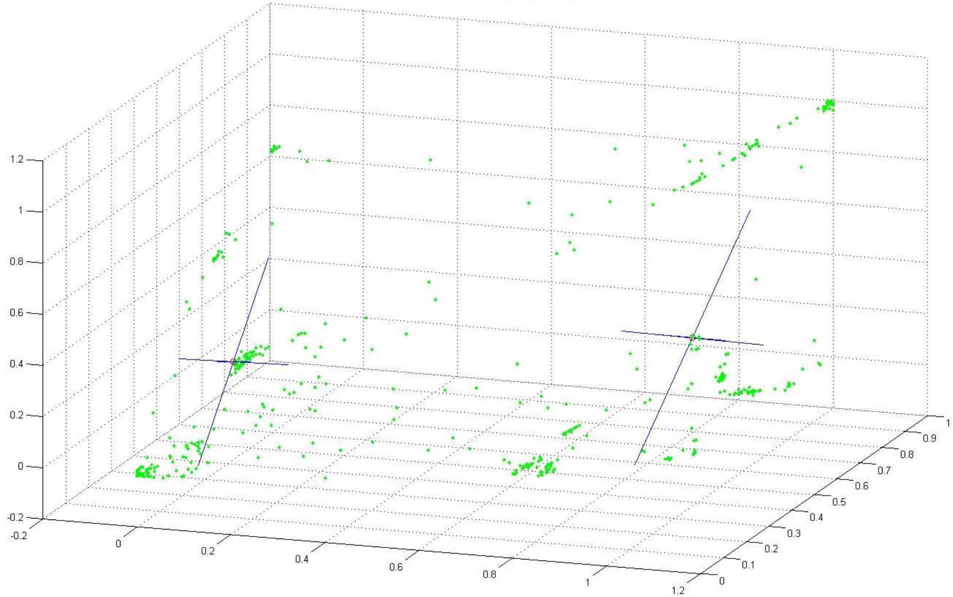
→ Classifys all pixels using ML-criterion

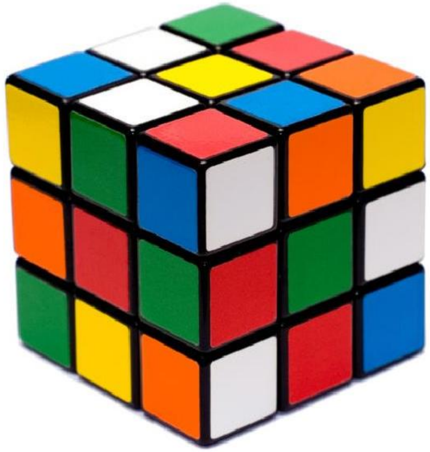
→ Plots the segmentation result

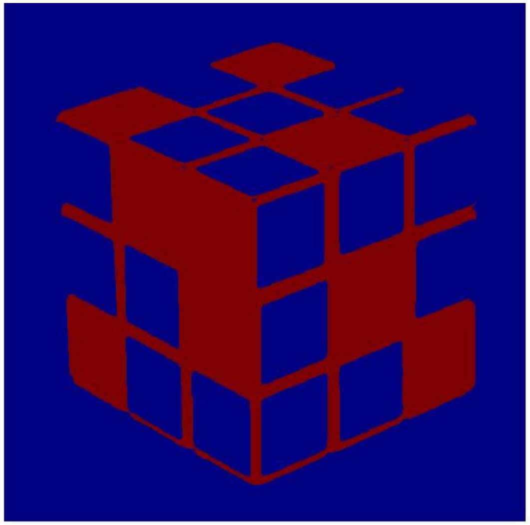


c. **Vary** the parameter n_{comp} (see source code of *ApplyGaussMixEM*). Which number is suitable for the given image? Can you observe any problems in segmenting the image?

Task B: Example result, $n_{comp} = 2$

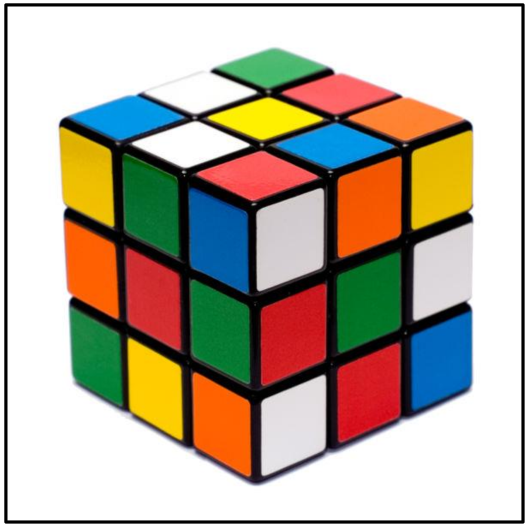






Provided Data (See CV-Homepage)

- **Matlab** source
  - *LearnGaussMixModel.m* → For implementation of A
  - *TestGaussMixEM.m* → For testing of A
  - *ApplyGaussMixEM.m* → For classification in B
- **Octave** source
  - *ApplyGaussMixEM_octave.m*
  - *LearnGaussMixModel.m, TestGaussMixEM.m* also work in Octave
- **Image**: *inputEx6.jpg*



Task A: Implementation Details

Given:

- *LearnGaussMixModel.m*
  - EM-algorithm outline
  - Initialization and Splitting
  - Visualization
  - **Function dummys for implementation**:
    - *CalcLnVectorProb*
    - *GmmEStep*
    - *GmmMStep*

Task A: Implementation Details

Inputs *CalcLnVectorProb*:

- *model*: Structure with current model parameters (**already present**)
  - *model.weight*: Vector with cluster weights
  - *model.weight(c)*: Weight α_c of component c
  - *model.mean*: Mean vectors μ of all components
  - *model.mean(c,:)*: Mean vector μ_c (3 elements) of component c
  - *model.covar*: Covariance matrices Σ of all components
  - ***squeeze****(model.covar(c,:,:))*: Covariance matrix Σ_c (3×3 elements) of component c

$$log\left(\alpha_c \cdot p\left(x_i | \mu_c^k, \Sigma_c^k\right)\right)$$

- *trainVect*:
  - Feature (or training) vectors (**given**)
  - *trainVect(i,:)*: 3-element feature vector

Task A: Implementation Details

Inputs *GmmEStep*:

- *model*: Structure with current model parameters (see above)
- *trainVect*: Feature (or training) vectors (see above)

Inputs *GmmMStep*:

- *model*: Structure with current model parameters (see above)
- *trainVect*: Feature (or training) vectors (see above)
- *LnCompProb:* Output of E-Step

Rules for Log-Value Computations

- $\log(a \cdot b) = \log(a) + \log(b)$
- $\log(a + b)$: no mathematical rule using $\log(a)$ and $\log(b)$!
  - → $\log(\exp(a) + \exp(b))$
- $\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$
- $\log(\exp(a)) = a$
- $\log\left(a^b\right) = b \cdot \log(a)$

Thank you!