Image Processing and Pattern Recognition

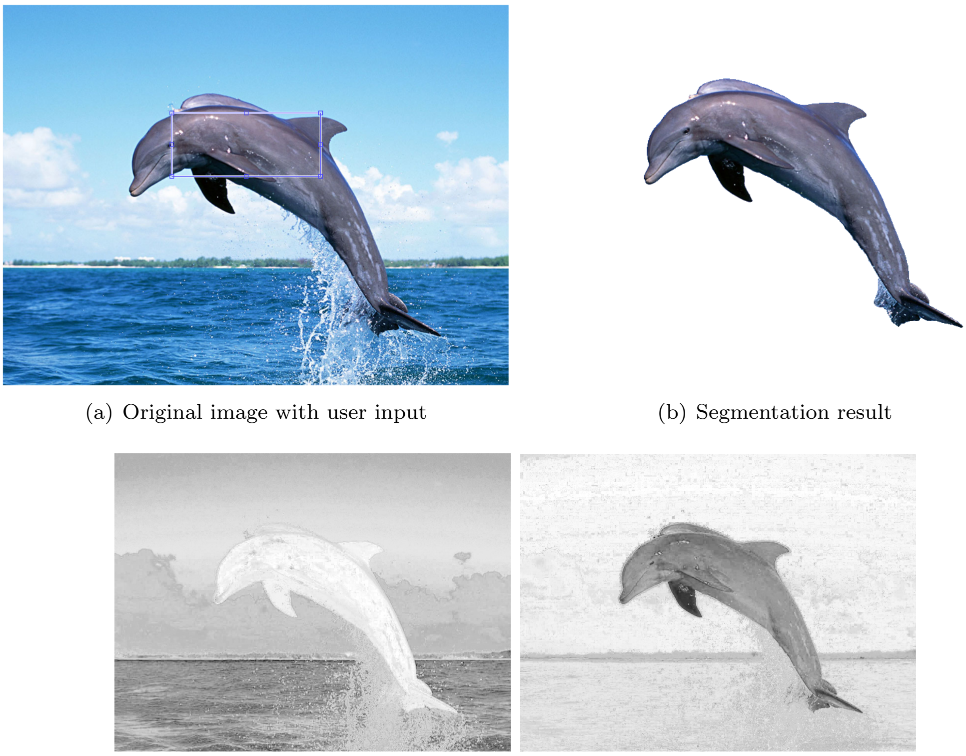
Assignment 4 - Image Segmentation with Gaussian Mixture Model

December 5, 2016

Deadline: December 19, 2016 at 23:59h.
Submission: Upload report and implementation zipped (*[surname].zip*) to the TeachCenter using the "My Files" extension.

1 Goal

This exercise focuses on binary image segmentation, which is a core problem in computer vision. The task of binary image segmentation is to partition the image into foreground and background. Typically, the user is required to mark a region of interest (*e.g.* via drawing scribbles or defining a polygon) and the algorithm then computes the segmentation based on a foreground and a background model (see fig. 1). In this exercise, we will use a Gaussian mixture model (GMM) to represent foreground and background.



(a) Original image with user input

(b) Segmentation result

(c) GMM weights foreground

(d) GMM weights background

Figure 1: Input image and result of the image segmentation

2 Method

The method consists of two steps, which are executed iteratively:

1. Fit a Gaussian mixture model to current foreground and background region
2. Segment the image using a graph-cut algorithm, where the unary weights are computed from the log-likelihood of the GMMs. Use the segmentation to update the foreground and background region and go to 1.

2.1 Fitting the Gaussian mixture model

Your task is to implement step 1, *i.e.* the fitting of the GMM. A Gaussian mixture is a weighted sum of Gaussians with different means and standard deviations, which is defined as

$$p(x;\theta) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x;\mu_k,\Sigma_k) \tag{1}$$

Here, $\theta = \{\alpha_k, \mu_k, \Sigma_k\},\ k = 1 \dots K$ are the parameters of the Gaussian mixture and $\mathcal{N}(x;\mu_k,\Sigma_k)$ denotes a multivariate Gaussian distribution. The equation for the multivariate Gaussian reads

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{2}$$

where $x \in \mathbb{R}^d$ is a vector in a d-dimensional feature space and $|\Sigma|$ denotes the determinant of Σ. For this exercise, the feature space will consist of the RGB color of pixels, *i.e.* $d = 3$.
To fit the model to a given set of input features $x_i,\ i = 1 \dots n$, we maximize the log-likelihood of the GMM using the EM-algorithm. The update equations are given as

$$\begin{aligned}
\alpha_k^{j+1} &= \frac{1}{n}\sum_{i=1}^{n} \gamma_k^j(x_i) \\
\mu_k^{j+1} &= \frac{\sum_{i=1}^{n} \gamma_k^j(x_i) x_i}{\sum_{i=1}^{n} \gamma_k^j(x_i)} \\
\Sigma_k^{j+1} &= \frac{\sum_{i=1}^{n} \gamma_k^j(x_i)(x_i-\mu_k^{j+1})(x_i-\mu_k^{j+1})^T}{\sum_{i=1}^{n} \gamma_k^j(x_i)},
\end{aligned} \tag{3}$$

with the auxiliary variable

$$\gamma_k^j(x_i) = \frac{\alpha_k^j \mathcal{N}(x_i;\mu_k^j,\Sigma_k^j)}{\sum_{k=1}^{K} \alpha_k^j \mathcal{N}(x_i;\mu_k^j,\Sigma_k^j)} \tag{4}$$

and the superscript j denoting the iteration number.

2.2 Computing a segmentation

Once we have the GMM for the foreground and background, we compute a segmentation using the log-likelihood of the foreground and background GMM as unaries in a graph-cut algorithm. This step is already implemented in the framework. The graph cut is computed via solving the ROF model for image denoising and subsequent thresholding. As a measure of convergence, we can inspect the primal-dual gap, which, at the optimal solution, is zero.

3 Framework

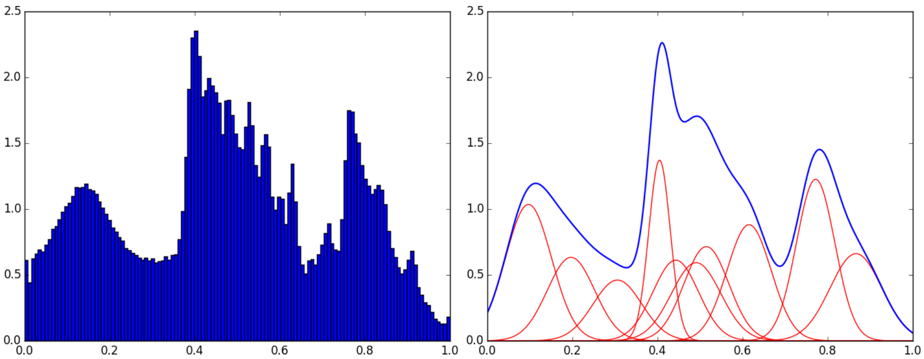
The framework consists of the basic structure of the segmentation algorithm. Your task is to implement the fitting of the GMM (function `fit_gmm()`).

Hints

- In order to compute $\gamma_k^j(x_i)$ (eq. (4)), you will have to evaluate a multivariate Gaussian distribution (eq. (2)). To implement $(x-\mu)^T\Sigma^{-1}(x-\mu)$ efficiently in a vectorized form, use the Cholesky decomposition[1] to factor the covariance matrix into $\Sigma = U^TU$, where U is a upper triangular matrix. With the abbreviation $z = x - \mu$, the argument of the exponential function in eq. (2) can then be rewritten

$$\begin{aligned} z^T\Sigma^{-1}z &= z^T(U^TU)^{-1}z \\ &= \underbrace{z^TU^{-1}}_{q^T}\underbrace{(U^T)^{-1}z}_{q} \\ &= q^Tq \end{aligned} \tag{5}$$

- As the 3-dimensional feature space is hard to visualize, verify the implementation of your method using a one-dimensional feature space, see fig 2.



(a) Histogram of the values of the blue channel

(b) Fitted GMM. Individual Gaussians are shown in red, the final mixture is shown blue.

Figure 2: Histogram of the data and corresponding GMM with 10 components

Parameters

- `K`: number of conponents of the GMM
- `gmm_iters`: number of iterations to run the gradient descend for fitting the GMM
- `rof_iters`: number of iterations for solving the segmentation. (The method prints the primal-dual gap, which can be used as a measure of convergence. In the true solution, the gap reduces to zero.)
- `N`: number of times to iterate the procedure (see sec. 2)

Play with the parameters and see how it influences the output! Investigate what changing K means and describe it in your report. Is it necessary to solve the graph cut exactly, *i.e.* to a primal-dual gap of zero? What is the most important parameter *w.r.t.* segmentation quality?

Experiment with your own images and include your findings in the report!

[1] python: `scipy.linalg.cholesky()`