Image Segmentation with MRF model

Task: segment a noisy image into N graylevel-classes with a Markov Random Field model (MRF).

Assume that we have N classes (e.g. obtained by training), each of them generates the pixels’ gray levels according to a unique Gaussian distribution. Our goal is to create a segmented label map using N labels representing the N classes, considering the following two soft constraints:

- Observed gray level of a given pixel should be generated with high probability by the Gaussian distribution of the corresponding class
- Neighbouring pixels usually correspond to the same class (homogeneous label map)

Notations:
S – set of pixels (image)
s ∈ S – given pixel
X = {x_s} – measured pixel values (x_s is the gray level of pixel s).
L= {1,2,...} – labels corresponding to the different segmentation classes
ω= {ω_s} – global labelling/segmentation (ω_s is the label of pixel s)

Goal: find among all the possible ω global labellings the optimal ω̂ labelling which maximizes the following probability value:

\[ P(\hat{\omega} \mid X) = P(X \mid \hat{\omega}) \cdot P(\hat{\omega}) / P(X) \]

Using the logarithm function is monotonous, and the fact the P(X) does not depend on ω, the above probability maximization is equivalent to the following energy minimization step:

\[ \hat{\omega} = \arg \min_{\omega} \left[ -\log P(X \mid \omega) - \log P(\omega) \right] \]

We will use the Potts-model where the (1) and (2) terms of the above energy functions have the following form:

(1) posterior term (depends on the measured pixel gray values):

\[ - \log P(X \mid \omega) = \sum_{s \in S} - \log P(x_s \mid \omega_s) \]

where we assume that in the k-th class the gray level distribution is Gaussian with \( \mu_k \) expected value and \( \sigma_k \) standard deviation. Definition of the energy term:

\[ - \log P(x_s \mid \omega_s = k) = \log(\sqrt{2\pi \cdot \sigma_k}) + \frac{1}{2} \left( \frac{x_s - \mu_k}{\sigma_k} \right)^2 \] (1)
prior term:

If \( r \) and \( s \) are neighbouring pixels:

\[
V_2(\omega_r, \omega_s) = \begin{cases} 
-\beta & \text{if } \omega_s = \omega_r \\
\beta & \text{if } \omega_s \neq \omega_r 
\end{cases}
\]

Otherwise \( V_2(\omega_r, \omega_s) = 0 \) for all cases when \( s \) and \( r \) are not neighbours. The prior term penalizes neighbouring pixels with different labels. As a consequence, the segmented map will contain smooth connected regions, which are influenced by the \( \beta \) parameter.

In summary, the complete energy function is defined by the following formula:

\[
\hat{\omega} = \arg \min_{\omega} \sum_{s \in S} -\log P(x_s | \omega_s) + \sum_{r,s \in S} V_2(\omega_r, \omega_s)
\]

Optimization is in general NP-hard. The proposed optimization algorithms (ICM, Metropolis, MMD, Gibbs Sampler etc) provide suboptimal solutions with different qualities, which can be compared in this assignment task.

**Usage of the demo Matlab program**

**Start:** MRFSegmenter command

**Load image:** load a given image

**Add noise:** add Gaussian white noise to the gray values of the pixels (0 expected value, given deviation)

**Define regions:** For the estimation of the gray value distributions of the different classes, we need to assign training areas to each class. Training areas have a rectangular shape, which can be defined by their left upper and right bottom corners (or any other 2 “opposite” corners)

Defining corner points:
Press „Define regions“ button, then move the mouse cursor above the “Input image”, and assign the corners by pressing the left mouse button. The “2k-1”-th and “2k”-th points are the left upper and bottom right corners of the training region of the k-th class (k=1,2,...). The rectangle of the training regions also appears on the image. The program allows us to define “Classnum” number of regions.

After defining the regions, the program automatically calculates the empirical mean and deviation values of each class, and stores the obtained parameters in the class' model. The parameters also appear in the “Class parameters” list in the dialog window.

**Init mask (before: set the „Initial labeling“ popup)**
Assigns initial labels to the pixels according to different strategies:
Types:
- **Rand**: random label assignment following uniform distribution (used by MMD, Metropolis and Gibbs Sampler optimization)
- **MAP**: for each pixel, assign the class, which generates the observed pixel value with the highest probability (used for the ICM optimization method):
  \[
  \omega^0_s = \arg\min_{k \in L} \{-\log P(x_s \mid \omega_s = k)\}
  \]

Watch out: the „Init mask” command can be only used for already initialized class probability density functions (see „Define regions” function)!

**Run**

Start the MRF optimization running, our goal is to obtain an „appropriate” segmentation. Choosing the optimization type is possible through the „Optimization popup” item. The optimization starts from an initial segmentation (see Initial mask section), which is changed in consecutive iterative steps. The process stops when convergence is obtained (ie. between the consecutive iteration no change can be observed).

**Optimization parameters:**

- **Beta**: responsible for the size of the homogeneous regions
- **DeltaUmin**: parameter defining the stopping condition of the iterations: if the global energy change between two consecutive iterations is smaller then DeltaUmin, the optimization stops.
- **T0**: initial temperature used by simulated annealing (no use for ICM)
- **c**: constant defining the speed of temperature change during simulated annealing: 
  \[T_{k+1} = c \cdot T_k\] (used by MMD)
- **Alpha**: only for MMD, see algorithm description

**Implementation**

**MRF structure:**

- **Beta**: 0.9000
- **DeltaUmin**: 0.0500
- **T0**: 4
- **c**: 0.9800
- **initmethod**: 1
- **optmethod**: 1
- **imagesize**: [128 128]
- **classnum**: 4
- **Alpha**: 0.1000
- **gauss_params**: ([1x1 struct] [1x1 struct] [1x1 struct] [1x1 struct])
- **logProbs**: ([128x128 double] [128x128 double] [128x128 double] [128x128 double])
- **classmask**: [128x128 int]

Accessing the different fields: e.g. „mrf.Beta” gives the value of the Beta parameter
Meaning of the different fields of the MRF structure:

- **classnum**: number of segmentation classes
- **classmask**: output labelling: array of class labels, where classmask(y,x) is the label of pixel (y,x), each label can be: (1,…,classnum)
- **initmethod**: 1=RAND, 2=MAP
- **optmethod**: 1=MMD, 2=ICM, 3=Gibbs Sampler, 4=Metropolis
- **gauss_params**: structure array with “classnum” elements, eg.
  - „gauss_params{1}.mean” mean value for the first segmentation class
  - „gauss_params{2}.dev” deviation value for the second segmentation class
- **logProbs**: again, a structure array with “classnum” elements --- to each class, we assign a 2D array of probability values, with the same size as the input image. logProbs is a structure array of these arrays, ex:
  - let be the coordinates of pixel s: s^y, s^x. In this case logProbs{i}(s^y,s^x)=-log P(x_s | omega= i) (this is Eq. (1), the posterior term)

**Task 1**: try the already implemented ICM optimization algorithm (gmrf_doICM.m) with RAND and MAP label initialization methods (see gmrf_initClassMaskRand.m, gmrf_initClassMaskMAP.m). Conclude the experiences and consult the implemented source code of the algorithm.

**Task 2**: Implement the MMD optimization (gmrf_doMMD.m) technique, and test it with the RAND and MAP label initialization methods. Conclude the experiences.

Note Metropolis and Gibbs Sampler optimization should not be implemented.

**Help for MMD:**

For practical reasons, we suggest some differences from the general MMD schema presented in the MRF lecture.

**Init mask**
fill “classmask” with random label values

**Optimization’s pseudo code**

temperature = T0 (initial temperature, parameter of the mrf model, T0=4 above)

*Alpha* is a parameter of the mrf model (0.1 above)

\[ \Delta E = \text{BIGNUMBER} \] (just for initialization)

while \( \Delta E > \text{DeltaUmin} \)

\[ \Delta E = 0 \]

**for each pixel** \( s = (s^y, s^x) \)

- let \( \xi \) be a random new label candidate for pixel \( s \)
- calculate the possible \( \Delta U \) global energy change, that we get if we exchange the label of \( s \) from the original \( \omega_s \) to \( \xi \)
\[ \Delta U = \left( -\log P(x_s \mid \xi) + \sum_{r \in N(s)} V_2(\omega_r, \xi) \right) - \left( -\log P(x_s \mid \omega_s) + \sum_{r \in N(s)} V_2(\omega_r, \omega_s) \right) \]

(here log means natural logarithm, but you can use the precalculated values from the MRF structure)

- Apply or decline label change:
  - if \( \left( \log(\text{Alpha}) < \frac{-\Delta U}{\text{temperature}} \right) \)
    - then set \( \xi \) as the new label of \( s \) instead of \( \omega_s \) and increase the global energy change accumulator \( \Delta E = \Delta E + \text{abs}(\Delta U) \),
    - otherwise keep the original label of \( s \)

end for each pixel

\[ \text{temperature} = \text{temperature} \times c \ (\text{used } c = 0.9800) \]

show the actual label map.

**Submission requirements**

You should upload the completed “gmrf_doMMD.m” file, and a documentation which briefly summarizes the main decision points during the development, the experiences and the test/evaluation results regarding Task1 and Task2 (some resulting images etc).