Image Segmentation and Parameter Estimation in a Markovian Framework

Zoltan Kato

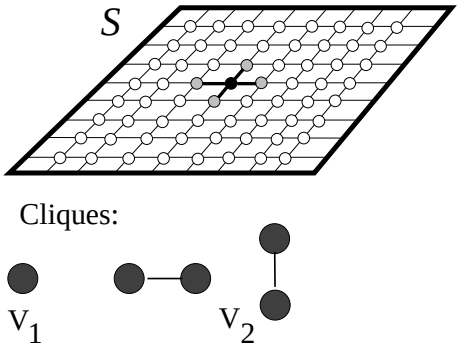
Image Segmentation Problem

The objective of segmentation is to partition an *image* into homogeneous regions such that:

- The segmentation must be complete (i.e. every pixel must be in a region).
- The pixels in a region must be connected.
- The regions must be disjoint.

In real scenes, neighboring pixels usually have similar intensities. In a probabilistic framework, such regularities are well expressed by Markov Random Fields (MRF). On the other hand, the local behavior of MRF permits to develop highly parallel algorithms.

MRF − Definitions



$\mathcal{S}$ - set of pixels (or sites)
$\mathcal{F}$ - observed image (data set)
$\Lambda = \{0, 1, \ldots, L-1\}$ - labels (or classes,
$\omega \in \Omega$ — global labeling
$\mathcal{X}$—label process (MRF)

MRF: $\mathcal{X}$ is a MRF with respect to $\mathcal{V}$ if

1. for all $\omega \in \Omega$: $P(\mathcal{X} = \omega) > 0$,

2. for every $s \in \mathcal{S}$ and $\omega \in \Omega$:
$P(X_s = \omega_s \mid X_r = \omega_r, r \neq s) = P(X_s = \omega_s \mid X_r = \omega_r, r \in \mathcal{V}_s)$.

Hammersley-Clifford theorem: $\mathcal{X}$ is a MRF with respect to the neighborhood system $\mathcal{V}$ if and only if $\pi(\omega) = P(\mathcal{X} = \omega)$ is a Gibbs distribution:

$$\pi(\omega) = \frac{1}{Z} \exp\left(-U(\omega)\right) = \frac{1}{Z} \exp\left(-\sum_{C \in \mathcal{C}} V_C(\omega)\right) \qquad (1)$$

where Z is the normalizing constant or partition function:

$$Z = \sum_{\omega} \exp\left(-U(\omega)\right),$$

$U(\omega) = \sum_{C \in \mathcal{C}} V_C(\omega)$ is the energy function and V_C denotes the clique potentials.

This equivalence provides a simple way to specify MRF's through clique-potentials instead of local characteristics, which is usually very difficult.

Supervised Image Segmentation

- $\mathcal{F} = \{f_s\}_{s \in \mathcal{S}}$ is a set of image data (f_s = the greylevel value at pixel s).

- **MAP estimation:** We want to find the **labeling** $\widehat{\omega} \in \Omega$ which maximizes

$$P(\omega \mid \mathcal{F}) \overset{Bayes}{\propto} P(\mathcal{F} \mid \omega) P(\omega)$$

$\Omega =$ set of all possible global labelings.

- **Hypothesis:**

  - $P(\mathcal{F} \mid \omega)$ is **Gaussian**,

  - $P(\omega)$ is **Markovian**.

Monogrid Model

$$\widehat{\omega} = \min_{\omega \in \Omega} \sum_{s \in \mathcal{S}} V_1(\omega_s, f_s) + \sum_{C \in \mathcal{C}} V_2(\omega_C)$$

with $V_2(\omega_C)$ = second order clique-potentials, which favour **similar classes for neighboring pixels**:

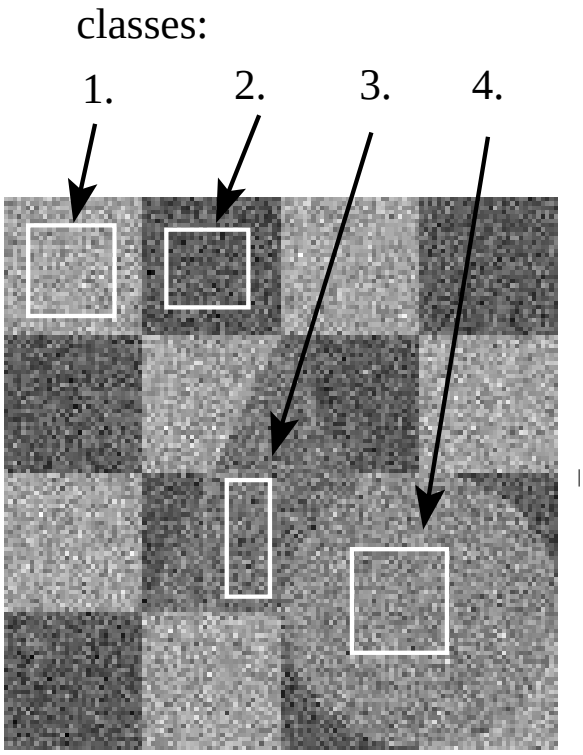
$$V_2(\omega_C) = V_{\{s,r\}}(\omega_s, \omega_r) = \begin{cases} -\beta & \text{if } \omega_s = \omega_r \\ +\beta & \text{if } \omega_s \neq \omega_r \end{cases}$$

and

$$V_1(\omega_s, f_s) = \log(\sqrt{2\pi}\sigma_{\omega_s}) + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2}$$

μ_{ω_s} and σ_{ω_s} are learned for each class (**supervised** segmentation).

Supervised Parameter Learning



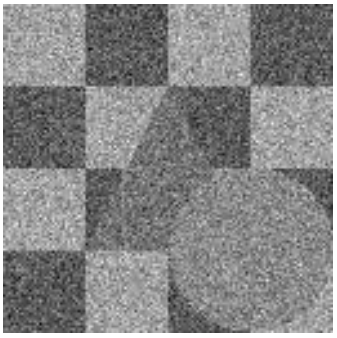
The class statistics (mean and variance) are estimated through the *empirical mean and variance*:

$$\forall \lambda \in \Lambda : \qquad \mu_\lambda = \frac{1}{\mid S_\lambda \mid} \sum_{s \in S_\lambda} f_s, \tag{2}$$

$$\sigma_\lambda^2 = \frac{1}{\mid S_\lambda \mid} \sum_{s \in S_\lambda} (f_s - \mu_\lambda)^2, \tag{3}$$

where S_λ denotes the set of pixels in the training set of class λ.

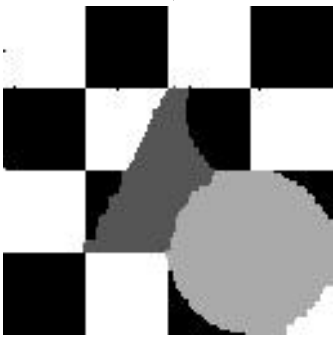
Supervised Segmentation (Monogrid case)



Parameters Θ

MRF image segmentation model

Find MAP estimate
(Simulated Annealing, for instance)



Optimization

- Problem reduced to the minimisation of a **non-convex** energy function $\Longrightarrow$ many **local** minima.

- Solution: **relaxation** methods

  - **stochastic** techniques: Simulated Annealing (Metropolis et al 53, Geman and Geman 84).

  - **deterministic** techniques: Graduated Non Convexity (Blake and Zisserman 87, Rangarajan and Chellappa 90), Iterated Conditional Mode (Besag 86, Jeng and Woods 88), Mean Field Annealing (Geiger and Girosi 89, Zerubia and Chellappa 90)...

Metropolis Algorithm (Simulated Annealing)

Algorithm 1 (Simulated Annealing)

① *Set* $k = 0$ *and initialize* ω *randomly. Choose a sufficiently high initial temperature* $T = T_0$.

② *Construct a trial perturbation* η *from the current configuration* ω *such that* η *differs only in one element from* ω.

③ **(Metropolis criteria)** *Compute* $\Delta U = U(\eta) - U(\omega)$ *and accept* η *if* $\Delta U < 0$ *else accept with probability* $\exp(-\Delta U/T)$ *(analogy with thermodynamics):*

$$\omega = \begin{cases} \eta & \text{if } \Delta U \leq 0, \\ \eta & \text{if } \Delta U > 0 \text{ and } \xi < \exp(-\Delta U/T), \\ \omega & \text{otherwise} \end{cases} \quad (4)$$

where ξ *is a uniform random number in* $[0, 1)$.

④ *Decrease the temperature:* $T = T_{k+1}$ *and goto Step ② with* $k = k + 1$ *until the system is* frozen.

Gibbs Sampler

Algorithm 2 (Gibbs Sampler)

① *Set* $k = 0$, *assign an arbitrary initial configuration* ω *and let* $T = T_0$ *be a sufficiently high initial temperature.*

② *For each configuration which differs at most in one element from the current configuration* ω *(they are denoted by* $\mathcal{N}_\omega$*), compute the energy* $U(\eta)$ *(*$\eta \in \mathcal{N}_\omega$*).*

③ **(Gibbs Sampler)** *From the configurations in* $\mathcal{N}_\omega$*, a sample is drawn such that* η *is accepted with probability*

$$\frac{\exp(-U(\eta))}{\sum_{\zeta \in \mathcal{N}_\omega} \exp(-U(\zeta))} \tag{5}$$

as the new configuration.

④ *Decrease the temperature:* $T = T_{k+1}$ *and goto Step ② with* $k = k + 1$ *until the system is* frozen.

Iterated Conditional Modes (ICM)

Algorithm 3 (ICM)

① *Start at a "good" initial configuration* ω^0 *and set* $k = 0$.

② *For each configuration which differs at most in one element from the current configuration* ω^k *(they are denoted by* $\mathcal{N}_{\omega^k}$*), compute the energy* $U(\eta)$ *(*$\eta \in \mathcal{N}_{\omega^k}$*).*

③ *From the configurations in* $\mathcal{N}_{\omega^k}$*, select the one which has a minimal energy:*

$$\omega^{k+1} = \arg \min_{\eta \in \mathcal{N}_{\omega^k}} U(\eta). \tag{6}$$

④ *Goto Step ② with* $k = k + 1$ *until convergence is obtained (for example, the energy change is less than a certain threshold).*

Modified Metropolis Dynamics

Algorithm 4 (MMD)

① *Pick up randomly an initial configuration* ω^0*, with* $k = 0$ *and* $T = T_0$*.*

② *Using a uniform distribution, pick up a global state* η *which differs only in one element from* ω^k*.*

③ **(Modified Metropolis Dynamics)** *Compute* $\Delta U = U(\eta) - U(\omega)$ *and accept* η *according to the following rule:*

$$\omega^{k+1} = \begin{cases} \eta & \text{if } \Delta U \leq 0, \\ \eta & \text{if } \Delta U > 0 \text{ and } \ln(\alpha) \leq \left(-\frac{\Delta U}{T}\right), \\ \omega^k & \text{otherwise} \end{cases} \tag{7}$$

where α *is a constant threshold (*$\alpha \in (0, 1)$*), chosen at the beginning of the algorithm.*

④ *Decrease the temperature* $T = T_{k+1}$ *and goto Step ② until convergence is obtained (*ΔU *less than a certain threshold, for example).*

Behavior of MMD

- At **high temperature**, an energy increase is permitted (**"pseudo-stochastic"** phase).

- **Under** a certain **temperature-threshold**, it becomes **deterministic**.

Cellular Neural Network Implementation of MMD

- CNN is basically a deterministic analog circuit.

- Its VLSI implementation takes place on a single analog chip containing several thousands (recently about 10,000 to 40,000) cells, each cell beeing connected to its neighbors.

- It has only simple arithmetic functions (addition, multiplication) and very simple nonlinear output functions (step, jigsaw).

- It's precision is limited to a few digits (8 bit).

- MMD is an ideal algorithm for CNN implementation

- A real VLSI CNN chip can execute an MMD relaxation algorithm of about 100 iterations in about 1msec.

Simulated Annealing

We have three ways of annealing:

- **Homogeneous annealing:** relaxation is performed at a fixed temperature until an equilibrium is reached.

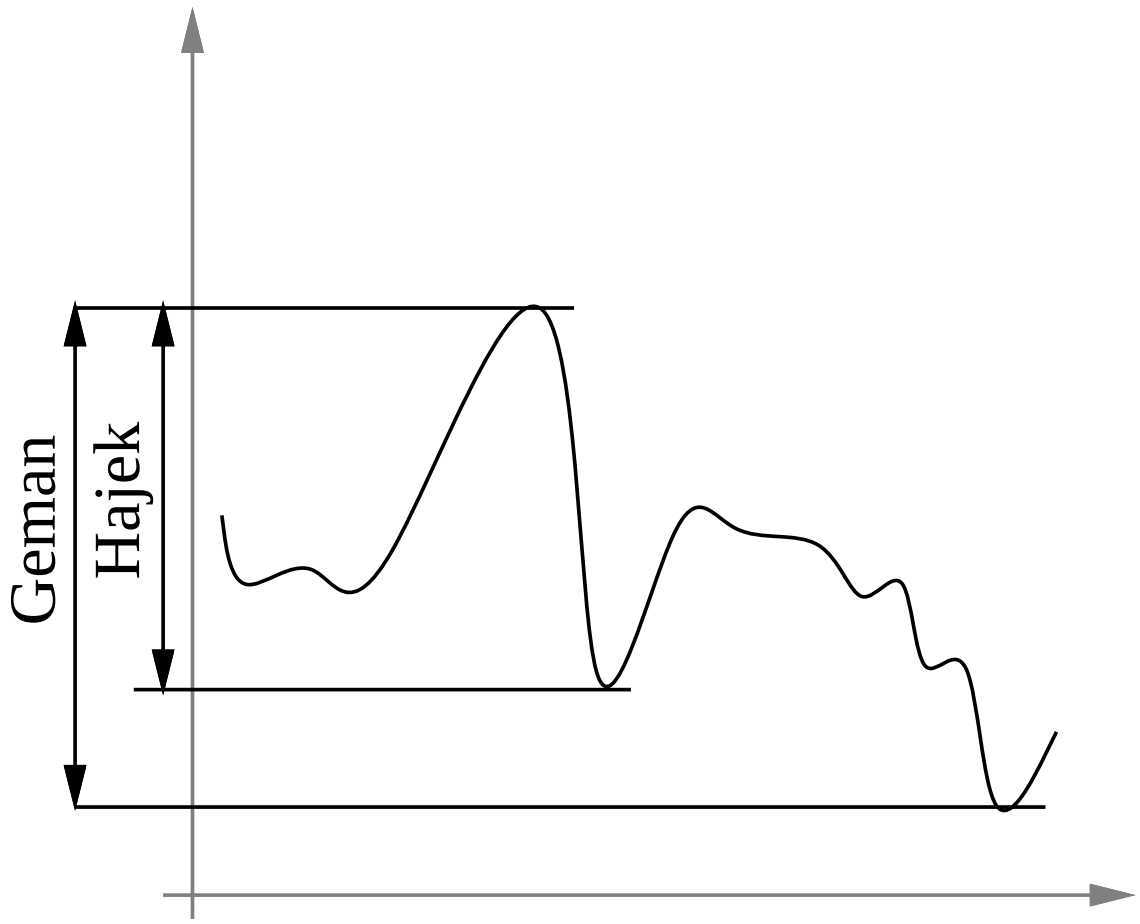
- **Inhomogeneous annealing:** the temperature is lowered after each iteration.

- **Multi-Temperature Annealing (MTA):**

  - To the higher levels, we associate higher temperatures which enable to be less sensitive to local minima.

  - At a finer resolution, the relaxation is performed at a lower temperature (at the bottom level, it is closed to 0).

Temperature Schedule Theory

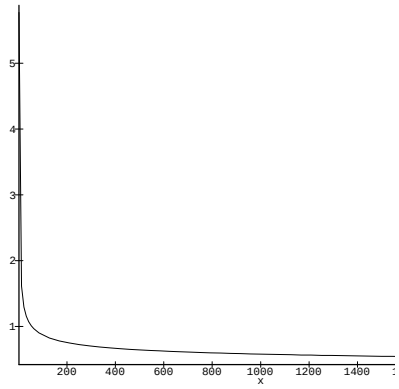


$$T_k \geq \frac{\Gamma}{\ln(k)} \tag{8}$$

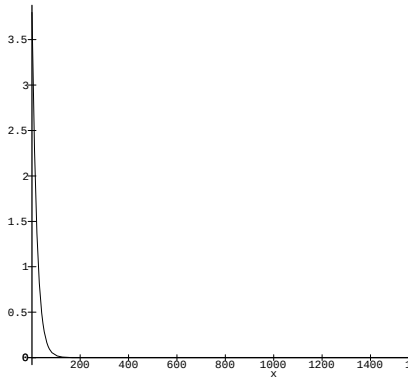
with

$$\Gamma > \max_{\omega \in \Omega} U(\omega) - \min_{\omega \in \Omega} U(\omega) \tag{9}$$

Temperature Schedule Practice



Logarithmic schedule ($4/\ln(k)$).



Exponential schedule ($0.95^k \cdot 4$).

Initial temperature: One usually set T_0 to a relatively low value ($T_0 = 4$) resulting in a faster execution of the algorithm.

Final temperature We can simply fix the number of iterations or terminate the execution of the algorithm if the last few configurations obtained by SA have nearly the same energy (i.e. ΔU is less than a certain threshold).

Temperature schedule:

$$T_{k+1} = c \cdot T_k, \qquad k = 0, 1, 2, \ldots \tag{10}$$

where $c < 1$ is a constant close to 1.