Grayscale image segmentation using reversible jump MCMC.

Pavel V. Senin, *UH, Student*

***Abstract*— While the ultimate goal of this project was to build a Reversible Jump MCMC sampler for image segmentation the others methods such as ICM, Gibbs sampler, Metropolis sampler and Simulated Annealing arised as a necessary milestones in order to meet the goal. So far all milestones are implemented while RJMCMC sampler itself is not done - the acceptance probability of the split or joining of classes is not finished. Being crucial in RJMCMC method, this part prevents from successful project closure. Nevertheless author believe that it could and would be done in short time after 5th of December of 2006. If not, the way and idea are well expressed in cited articles along with this report and provided to the public by source code repository at http://code.google.com/p/rjimage/. Repositoty contains this documents along with software implementation.**

***Index Terms*— RJMCMC, image segmentation, Gibbs sampler, ICM, Metropolis sampler.**

I. Model design

A. Introduction

IN the core of this work lies my own interest for the RJMCMC and publication [1] by Zoltan Kato "Bayesian Color image segmentation using reversible jump Markov Chain Monte-Carlo". I have chosed easiest way to go - grayscale, but as could be seen it doesn't really matter, the results could be easily elevated to color images encoded in any chosen color space. RJMCMC method described by Green [5] and used by Kato [1] deals with the problem of unsupervised image segmentation. It means that it should accept an image as the only input parameter and produce a segmented image representation as output. The method itself relies on the use of a firs-order Markov Field (MRF, Plotts model) and employs ICM algorithm [2], Gibbs sampler [3], and Metropolis sampler [4] along with optimization criteria based on Simulated annealing algorithm.

As I already written in the project proposal and intermediate report, the image segmentation is one of the fundamental problems in computer vision and lies an the first-row low-level tasks for full line of image processing like surface description, object recognition, content bases indexing etc.

I believe that Reversible Jump mimics natural human ability to segment images that are noisy or have very little features to do segmentation exactly. I think that humans also just trying some segmentation and choose the most probable one using their experience. Basically RJMCMC method search within the multidimensional variants space trying to maximize classes separation. The approach itself seems to be heavyweight and computationally costly, but second reason to choose this as I mentioned is my own interest in understanding of unsupervised learning and ICA.

The work was not supported by the IEEE.
Last revised at .
Thanks to my kids for constant disturbing.

At the beginning I was considering R (or 'GNU S'), a freely available language and environment for statistical computing and graphics as the programming environment. The taken choice was erroneous, despite being well supported and having numerous libraries for image IO and raster manipulations along with unlimited statistical tools, the R is extremely slow in matrix processing and has certain limitations in software design due to common namespace, plain code execution and lack of an IDE and debugger. The lesson learned from this failure is that while R is the ultimate "statistic" calculator (due to numerous packages) along with great plotting capabilities, it is more appropriate for single task solving, or conceptual coding, not for the routine program development, debug and evolution of software.

My second choice was Java and so far it provides desired functionality and flexibility. The software that done for the project is perfectly working, has nice GUI and could be easily debugged and extended in any desired way.

B. Model design

According [1], the observed image is: $F = \left\{ \vec{f}_s \,\middle|\, s \in S, \forall i : 0 < \vec{f}_s^i < 1 \right\}$, where vector $\vec{f}_s$ is vector that carried intensity of color for pixel s. The segmentation itself is just labeling of each pixel $s \in S$ by label $\omega_s \in \Lambda = \{1, 2, ..., L\}$. $\omega \in \Omega$ denotes a labeling (or segmentation), Ω is a set of all possible labeling.

We regard our image as a sample drawn from unknown Gaussian mixture distribution. The goal of our analysis is inference about the number L components:

- the component parameter

$$\Theta = \{\forall \lambda : 1 \leq \lambda \leq L, \Theta_\lambda = (\vec{\mu_\lambda}, \Sigma_\lambda)\};$$

- the component weights $p_\lambda \, (1 \leq \lambda \leq L)$ summing to 1;
- the clique potential (or inter-pixel interaction strength), β;
- the segmentation ω.

The joint distribution of above variables $L, p, \beta, \omega, \Theta, F$ given by formula :

$$P(L, p, \beta, \omega, \Theta, F) = P(\omega, F \,|\Theta, \beta, p, L\,)\, P(\Theta, \beta, p, L) \tag{1}$$

Impose the independence of labeling Θ, inter-pixel composition β, component weights p, and labels set power L as parameters that given randomly in every processed image. Therefore their joint probability reduces to

$$P(\Theta, \beta, p, L) = P(\Theta)\, P(\beta)\, P(p)\, P(L) \tag{2}$$

The posterior distribution of (F, ω) may be expressed as:

$$P(F, \omega \mid \Theta, \beta, p, L) = P(F \mid \omega, \Theta, \beta, p, L)\, P(\omega \mid \Theta, \beta, p, L) \quad (3)$$

The pixel classes (segmentation itself) represented as a multivariate Gaussian distribution and the underlied MRF (Markov Random Field) process follows a Gibbs distribution defined over a first order neighborhood system.

Previous equation could be factored out as:

$$(F \mid \omega, \Theta, \beta, p, L) = P(F \mid \omega, \Theta) = \prod_{s \in S} \left(\frac{1}{\sqrt{(2\pi)^3 \left|\sum_{\omega_s}\right|}} \exp\left(-\frac{1}{2} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right) \times \sum_{\omega_s}^{-1} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right)^T \right) \right) \quad (4)$$

Proceeding further we have:

$$P(\omega \mid \Theta, \beta, p, L) = P(\omega \mid \beta, p, L) = \frac{1}{Z(\beta, p, L)} \exp\left(-U(\omega \mid \beta p, L)\right) \quad (5)$$

where $U(\omega \mid \beta, p, L)$ is energy function:

$$U(\omega \mid \beta, p, L) = \sum_{s \in S} -\log(p_{\omega_s}) + \beta \sum_{\{s,r\} \in C} \delta(\omega_s, \omega_r) \quad (6)$$

$\delta(\omega_s, \omega_r) = 1$ if ω_s and ω_r are different and -1 otherwise. $Z(\beta, p, L) = \sum_{\omega \in \Omega} \exp(-U(\omega \mid \beta, p, L))$ is normalization constant or partition function. C denotes the set of cliques and $\{s, r\}$ is doubleton containing the neighboring pixel sites r and s.

Note that the whole posterior distribution could be derived from Gibbs distribution where the Gaussian distribution taken in account in the energy of the external field:

$$U(F \mid \omega, \Theta) = -\log(P(F \mid \omega, \Theta)) = \sum_{s \in S} \left(\ln\left(\sqrt{(2\pi)^3} \right) + \frac{1}{2} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right) \sum_{\omega_s}^{-1} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right)^T \right) \quad (7)$$

Now, we are using facts that $P(F)$ is constant for any particular image and Equation (1), Equation (2), Equation (4) and Equation (7) we are able to approximate the posterior density $P(L, p, \beta, \omega, \Theta \mid F) = P(L, p, \beta, \omega, \Theta, F) / P(F)$:

$$\begin{aligned} & P(L, p, \beta, \omega, \Theta \mid F) = \\ & P(F \mid \omega\Theta)\, P(\omega \mid \beta, p, L)\, P(\Theta)\, P(\beta)\, P(p)\, P(L) \\ & \approx \prod_{s \in S} \left(\frac{1}{\sqrt{(2\pi)^3 \left|\sum_{\omega_s}\right|}} \exp\left(-\tfrac{1}{2} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right) \times \right. \right. \\ & \left. \left. \sum_{\omega_s}^{-1} \left(\vec{f}_s - \vec{\mu}_{\omega_s} \right)^T \right) \right) \times \\ & \prod_{s \in S} \frac{p_{\omega_s} \exp\left(-\beta \sum_{\forall r : \{s,r\} \in C} \delta(\omega_s, \omega_r)\right)}{\sum_{\lambda \in \Lambda} p_\lambda \exp\left(-\beta \sum_{\forall r : \{s,r\} \in C} \delta(\lambda, \omega_r)\right)} \\ & \times P(\beta)\, P(L) \prod_{\lambda \in \Lambda} P(\vec{\mu_\lambda})\, P(\Sigma_\lambda)\, P(p_\lambda) \end{aligned} \quad (8)$$

Concerning the priors, the published model follows [6] and [7] by choosing uniform reference for priors for L, $\vec{\mu_\lambda}$, Σ_λ, p_λ where $\lambda \in \Lambda$.

C. Sampling from Posterior

Having Equation (8) as the base point the original article proceed to the MCMC algorithm that is used to sample from the whole posterior distribution in order to obtain a MAP estimate via simulated annealing [8]. However, classical MCMC methods are restricted to problems where the dimensionality of the parameter vector is fixed. Therefore, the estimation of the number of mixture components is not possible. In order to overcome this obstacle model uses Reversible Jump MCMC (RJMCMC), that has been proposed by Peter Green in [5]. This method makes it possible to construct reversible Markov chain samplers that jump between parameter subspaces of different dimensionality. RJMCMC allows the direct sampling of the whole posterior distribution defined over the combined model space thus reducing the optimization process to a single simulated annealing run. Another advantage is that no coarse segmentation neither exhaustive search over a parameter subspace is required. Our set of unknowns is $\{L, p, \beta, \omega, \Theta\}$, lets denote it as X and let $\pi(X)$ be the target probability measure (the posterior distribution). The wildly used tool to sample from such distribution is Metropolis-Hasting method [3, 4]. When the current state is X the new state is drawn from an arbitrary joint distribution $q(X, X')$. The new state is accepted with probability

$$A(X, X') = \min\left(1, \frac{\pi(X')\, q(X, X')}{\pi(X)\, q(X', X)}\right) \quad (9)$$

The presented model has multiple parameter subspaces of different dimensionality and it is necessary to devise move types between subspaces authors made decision to build the hybrid sampler in the five moves:

1) Sampling the class labels ω, i.e. resegment the image;
2) Sampling Gaussian parameters $\Theta = \left\{ \left(\mu_\lambda, \vec{\Sigma_\lambda} \right) \mid \lambda \in \Lambda \right\}$;
3) Sampling the mixture weights p_λ $(\lambda \in \Lambda)$;
4) Sampling the MRF hyperparameter β;
5) Sampling the number of classes L, i.e. splitting one of mixture components into two or merge two of them into one.

The only randomness left in there is the choice between splitting and merging in move (5). One iteration of the hybrid sampler, also called a sweep by authors, consists in a complete pass over these moves. The first four move types are conventional in the sense that they do not alter the dimension of the parameter space. Concerning the fifth move type, the reversible jump mechanism is needed.

II. Hybrid sampler

From here I will point out the moves that my implementation doing and problem that prevents task from completion.

A. Move 1 - image segmentation

This move is just classical image segmentation with known parameters, i.e. we have fixed parameters - their estimates and segmentation reduces to:

$$\mathrm{P}(L,p,\beta,\omega,\Theta \mid F) \approx P\left(F \mid \omega,\widehat{\Theta}\right) P\left(\omega \mid \widehat{\beta},\widehat{p},\widehat{L}\right) P(\Theta)\,P(\beta)\,P(p)\,P(L) \approx \prod_{s\in S}\left(\frac{1}{\sqrt{(2\pi)^3\left|\widehat{\Sigma}_{\omega_s}\right|}}\exp\left(-\frac{1}{2}\left(\vec{f}_s-\vec{\widehat{\mu}}_{\omega_s}\right)\widehat{\Sigma}_{\omega_s}^{-1}\left(\vec{f}_s-\vec{\widehat{\mu}}_{\omega_s}\right)^T\right)\right) \times \prod_{s\in S}\widehat{p}_{\omega_s}\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\omega_s,\omega_r)\right) \quad (10)$$

The author points here that this sub-chain of classical segmentation parameters should be obtained by Gibbs sampler [9] since ω takes discrete values over finite set Λ. In order to proceed further I have implemented Gibbs sampler that does pixel labeling once given the set of Gaussians. This part could be run independently in my demo software. The results of Gibbs sampler optimized using Simulated Annealing with a logarithmic cooling schedule that chosen so that the algorithm would converge after a reasonable number of iterations. The schedule is given by: $T_{k+1} = 0.98T_k$ with an initial temperature set to 20.0.

B. Move 2 - estimating Gaussian parameters

This move is aiming mean and covariance matrix of the pixel classes. By setting variables L,p,β,ω to their estimates $\widehat{L},\widehat{p},\widehat{\beta},\widehat{\omega}$ Equation (8) reduces to the form:

$$P(L,p,\beta,\omega,\Theta \mid F) \approx P(F,\widehat{\omega} \mid \Theta)\,P(\Theta) = \prod_{\lambda\in\Lambda}\prod_{s:\widehat{\omega}_s=\lambda} P\left(\vec{f}_s \,\middle|\, \vec{\mu}_\lambda,\Sigma_{lambda}\right)P(\vec{\mu}_\lambda)\,P(\Sigma_\lambda) = \times \prod_{s\in S}\frac{1}{\left((2\pi)^3|\Sigma_\lambda|\right)^{|\Sigma_\lambda|/2}}\exp\left(-\frac{1}{2}\sum_{s:\widehat{\omega}_s=\lambda}\left(\vec{f}_s-\vec{\mu}_\lambda\right)\Sigma_\lambda^{-1}\left(\vec{f}_s-\vec{\mu}_\lambda\right)^T\right)\times\prod_{\lambda\in\Lambda}P(\vec{\mu}_\lambda)\,P(\Sigma_\lambda)\,P(p_\lambda) \quad (11)$$

where $|\Sigma_\lambda|$ is number of sites labeled by λ.

C. Move 3 - sampling Mixture Weights

This move type aims at estimating the mixture weights p. These weights are incorporated into the Gibbs distribution of the underlying label process as the external field strength. In this model the weights required to be normalized by imposing the following constraint:

$$\sum_{\lambda\in\Lambda} p_\lambda = 1 \quad (12)$$

having this constraint it is possible to fix hyperparameter β a priori and authors fixed it to 2.5. Using these conditions and setting variables L,β,ω,Θ to their estimates $\widehat{L},\widehat{\beta},\widehat{\omega},\widehat{\Theta}$

$$P(L,p,\beta,\omega,\Theta \mid F) \approx P\left(\widehat{\omega} \mid \widehat{\beta},\widehat{p},\widehat{L}\right)P(p) = \prod_{\lambda\in\Lambda}P(p_\lambda)\left(\frac{p_\lambda}{\sum_{\lambda\in\Lambda}p_\lambda}\right)^{|S_\lambda|}\times \prod_{s\in S}\frac{\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\widehat{\omega}_s,\widehat{\omega}_r)\right)\sum_{\lambda\in\Lambda}p_\lambda}{\sum_{\lambda\in\Lambda}p_\lambda\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\lambda,\widehat{\omega}_r)\right)} = \prod_{\lambda\in\Lambda}P(p_\lambda)\,p_\lambda^{|S_\lambda|}\times\prod_{s\in S}\frac{\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\widehat{\omega}_s,\widehat{\omega}_r)\right)}{\sum_{\lambda\in\Lambda}p_\lambda\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\lambda,\widehat{\omega}_r)\right)} \quad (13)$$

D. Move 4 - sampling Hyperparameter β

As already mentioned this model has β parameter fixed a priory. The value chosen is 2.5.

III. Estimating Number of classes

Herein, the cornerstone of the method goes. This move type involves changing L by 1 and making necessary corresponding changes to ω,Θ and p. Should be noted that right before this move all posterior distribution from equation (8) is already sampled, β is set to its value and we have following distribution (from Equation (8)):

$$P(L,p,\beta,\omega,\Theta \mid F) = P(F \mid \omega\Theta)\,P\left(\omega \mid \widehat{\beta},p,L\right)P(\Theta)\,P(\beta)\,P(p)\,P(L) \approx \prod_{s\in S}\left(\frac{1}{\sqrt{(2\pi)^3\left|\sum_{\omega_s}\right|}}\exp\left(-\frac{1}{2}\left(\vec{f}_s-\vec{\mu}_{\omega_s}\right)\times\sum\nolimits_{\omega_s}^{-1}\left(\vec{f}_s-\vec{\mu}_{\omega_s}\right)^T\right)\right) = \prod_{s\in S}\frac{p_{\omega_s}\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\omega_s,\omega_r)\right)}{\sum_{\lambda\in\Lambda}p_\lambda\exp\left(-\widehat{\beta}\sum_{\forall r:\{s,r\}\in C}\delta(\lambda,\omega_r)\right)}\times P(L)\prod_{\lambda\in\Lambda}P(\vec{\mu_\lambda})\,P(\Sigma_\lambda)\,P(p_\lambda) \quad (14)$$

Since the dimensionality of the parameter space is altered, the reversible jump technique is needed for sampling from the posterior distribution. I will not go into the deepness of mathematical formulas in this report due to space and IEEE format constraints, all of those could be found within original article [1].

But I would try to give the idea of RJMCMC using plain English language. The rule that should be obeyed in the reversible jump is reversibility, i.e. if the system could move from one dimensionality to another it must have the same probability to move back. Underlying rule of equality of splitting/joining probabilities allow the system to do exhaustive search over all possible combinations (segmentations in this case). Therefore probabilities of split and join are calculated at first, than acceptance probability of split and joining compared and final decision made.

Splitting the class. The split proposal begins by choosing a class λ *at random* with a uniform probability $P_{select}^{split}(\lambda) = 1/L$, Then L is increased by 1 and λ is splitted into λ_1 and λ_2. In doing so, a new set of parameters need to be generated. The generation is driven by random variables that are chosen from the interval (0; 1]. In order to favor splitting the class into roughly equal portions, beta distributions ($beta(1.1,1.1)$) is used.

After splitting the problem of reallocation of those sites s where $\widehat{\omega_s} = \lambda$ arises. A tentative labeling of the sites could be sampled using Gibbs sampler. However, a labeling

which has a relatively high posterior probability needed to meet acceptance probability. To achieve this goal, author uses ICM [2] algorithm to obtain a suboptimal segmentation. The obtained ω^+ has a relatively high posterior probability since the ICM-ed tentative labeling is close to the optimal labeling. To follow the author in this move I have implemented ICM algorithm that uses The same Simulated Annealing with the same logarithmic schedule. In my demo software it could be run by button "ICM".

Merging two classes. To decide which pair of classes has to be merged all pairwise distances $d(\lambda_1, \lambda_2)$ computed using Mahalanobis distance and probability of selecting pair for merging is relative to their distance:

$$P_{select}^{merge}(\lambda_1, \lambda_2) = \frac{d(\lambda_1, \lambda_2)}{\sum\limits_{\in\Lambda}\sum\limits_{k\in\Lambda} d(\lambda, k)} \tag{15}$$

The merge proposal is deterministic, once the choice of λ_1 and λ_2 has been made. The two components are merged reducing L by 1. The reallocation is simply done by setting the label at sites $s \in S_{\lambda_1,\lambda_2}$ *In order to calculate acceptance probability author says that 5 equations should be solved to get random variables that used for its calculation. Here I am facing hard time - I cannot solve those so far . I am still working figuring it out which way it could be done*.

Acceptance probability. Once proposal for the split or merging are done acceptance probabilities are computed and compared in order to do the image segmentation.

$$A_{split}\left(L, \widehat{p}, \widehat{\beta}, \widehat{\omega}, \widehat{\Theta}; L+1, p^+, \widehat{\beta}, \widehat{\omega}^+, \widehat{\Theta}^+\right) = min(1, A) \tag{16}$$

where

$$A = \frac{P\left(L+1, p^+, \widehat{\beta}, \widehat{\omega}^+, \widehat{\Theta}^+ \mid F\right)}{P\left(L, \widehat{p}, \widehat{\beta}, \widehat{\omega}, \widehat{\Theta} \mid F\right)} \times$$

$$\frac{P_{merge}(L+1)P_{select}^{merge}(\lambda_1,\lambda_2)}{P_{split}(L)P_{select}^{split}(\lambda)P_{realloc}} \times$$

$$\frac{1}{P_{beta(1.1,1.1)}(u_1)P_{beta(1.1,1.1)}(u_2)P_{beta(1.1,1.1)}(u_3)} \times$$

$$\left|\frac{\partial\varphi}{\partial(\Theta_\lambda, p_\lambda, u)}\right| \tag{17}$$

$P_{realloc}$ denotes the probability of reallocating pixels labeled by λ into regions labeled by λ_1 and λ_2. The last factor is Jacobian of the transformation. The acceptance probability now could be expressed as:

$$A_{merge}\left(L, \widehat{p}, \widehat{\beta}, \widehat{\omega}, \widehat{\Theta}; L-1, p^-, \widehat{\beta}, \widehat{\omega}^-, \widehat{\Theta}^-\right) = min\left(1, \frac{1}{A}\right) \tag{18}$$

IV. Optimization according to the MAP criteria, Simulated annealing

In the article MAP estimator build using Simulating annealing and provides us with an image segmentation $\widehat{\omega}$ and model parameters $\widehat{L}, \widehat{p}, \widehat{\beta}, \widehat{\Theta}$. The MAP estimator of unknowns is given by combinatorial optimization problem:d using Simulating annealing and provides us with an image segmentation $\widehat{\omega}$ and model parameters $\widehat{L}, \widehat{p}, \widehat{\beta}, \widehat{\Theta}$. The MAP estimator of unknowns is given by combinatorial optimization problem:

$$\left(\widehat{\omega}, \widehat{L}, \widehat{p}, \widehat{\beta}, \widehat{\Theta}\right)^{(MAP)} = \arg\max_{L,p,\beta,\omega,\Theta} P(L, p, \beta, \omega, \Theta \mid F) \tag{19}$$

with the following constraints:

$$\omega \in \Omega, \tag{20}$$

$$L_{min} \le L \le L_{max}, \tag{21}$$

$$\sum_{\lambda\in\Lambda} p_{lambda} = 1, \tag{22}$$

$$\forall \in \Lambda : 0 \le \mu_{,i} \le 1, \tag{23}$$

$$\forall \in \Lambda : 0 \le \Sigma_{,i,i} \le 1, -1 \le \Sigma_{,i,j} \le 1 \tag{24}$$

V. Algorithm, RJMCMC segmentation

In this section I will describe in short algorithm that summarizes all the above:

1) Set $k = 0$ and initialize $\widehat{L}^0, \widehat{p}^0, \widehat{\beta}^0, \widehat{\Theta}^0$, and set temperature τ^0.
2) A sample $\left(\widehat{\omega}^k, \widehat{L}^k, \widehat{p}^k, \widehat{\beta}^k, \widehat{\Theta}^k\right)$ is drawn from the modified posterior distribution using the hybrid sampler defined in section (NUMBER NEEDED) before. The modification is due to simulated annealing constraint - temperature τ_k:

$$\prod_{s\in S} \frac{1}{\left((2\pi)^3 |\Sigma_{\omega_s}|\right)^{1/2\tau_k}} \exp\left(-\frac{1}{2\tau_k}\left(\vec{f}_s - \vec{\mu}_{\omega_s}\right)\Sigma_{\omega_s}^{-1}\left(\vec{f}_s - \vec{\mu}_{\omega_s}\right)^T\right.$$

$$\times \prod_{s\in S} \frac{\exp\left(\frac{\log(p_{\omega_s})}{\tau_k} - \frac{\beta}{\tau_k}\sum\limits_{\forall r:\{s,r\}\in C}\delta(\omega_s,\omega_r)\right)}{\sum\limits_{\lambda\in\Lambda}\exp\left(\frac{\log(p_\lambda)}{\tau_k} - \frac{\beta}{\tau_k}\sum\limits_{\forall r:\{s,r\}\in C}\delta(\lambda,\omega_r)\right)} \tag{25}$$

   a) $\widehat{\omega}^k$ is drawn from distribution in Equation (10).
   b) $\widehat{\Theta}^k$ is drawn from distribution in Equation (11).
   c) $\widehat{p}^k$ is drawn from distribution in Equation (13).
   d) $\widehat{L}^k$ is estimated using the reversible jump techniques from the section (III).
3) GOTO to step 2 with $k = k+1$ and T_{k+1} until $k \le K$.

VI. Conclusion

As the result of above the methods of ICM, Simulated Annealing, Gibbs sampler based image segmentation and Metropolis-Hasting sampler based image segmentation are coded and tested in Java. The aim of understanding and impelmenting the RJ MCMC method is achieved and only lack of time prevents project from complete finishing. As could be seen in experimental results section ICM works with noised and damaged images restoring them back. Gibbs sampler is less effective than Metropolis-Hasting sampler from time performance point of view, but it is better from final result point of view because it serach over all available combinations of parameters.

APPENDIX

The coded software is accessible with source codes at http://code.google.com/p/rjimage/. For the compilation and correct work it needs at least Java 1.5 and Ant 1.7 installed at system. So far it correctly works with any image, but there is some problems with converting any image that not in the grayscale (byte for pixel). I think because I lost luminance factor when converting into grayscale. Baisically the software does conversion of any color space image into grayscale by itself. The demo software demonstrates the the ICM algorithm, Gibbs sampler and Metropolis-Hasting sampler. The RJMCMC methos as I have mentioned is almost done, I am stuck figuring out those acceptance probability calculationused , once it would be done the joy of running RJMCMC algorithm would be provided to community.

REFERENCES

[1] Z. Kato *Bayesian color image segmentation using reversible jump Markov chain Monte Carlo*. Probability, Networks and Algorithms, February, 28, 1999. CWI, Amsterdam.

[2] J. Besag *On the statistical analysis of dirty pictures*. Jl. Roy. Statist. Soc., ser. B, 1986.

[3] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller *Equation of state calculation by fast computing machines*. J. of Chem. Physics, Vol. 21, pp 1087-1092, 1953.

[4] W. K. Hasting *Monte carlo sampling methods using Markov chains and their application*. Biometrika, vol. 57, pp. 1701-1762, 1994.

[5] P. J. Green *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*. Biometrika, vol. 82, no. 4, pp. 711-731, 1995.

[6] S. Richardson and P. J. Green, *Bayesian analysis of mixtures with an unknown number of components*, Jl. Roy. Statist. Soc., ser. B, vol. 59, no. 4, pp. 731-792, 1997.

[7] S. A. Barker and P. J. W. Rayner, *Unsupervised image segmentation using Markov random feld models*, Energy Minimization Methods in Computer Vision and Pattern Recognition. 1997, pp. 165-178, Springer.

[8] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images.* IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 6, pp. 721-741, 1984.