Electronic Notes in Theoretical Computer Science 46 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume46.html 12 pages

Stochastic Segmentation Using Gibbs Priors

Eilat Vardi [1,2]

Department of Bioenginnering
University of Pennsylvania
Philadelphia, PA, USA

Gabor T. Herman [3]

Center for Computer Science and Applied Mathematics
Temple University
Philadelphia, PA, USA

Abstract

In earlier work, a fast stochastic method for reconstructing a certain class of two-dimensional binary images from projections using Gibbs priors was presented. In the present study, we introduce a stochastic segmentation of magnetic resonance gray-scale images of trabecular bone using Gibbs priors. We show some results as well as some post-processing that can be used to clean up segmentations of noisy gray-scale images.

1 Introduction

Segmentation deals with the problem of creating a binary image from a gray-scale image (where, in the binary image, 1 represents the presence and 0 represents the absence of the object which we wish to segment from the gray-scale image). A stochastic algorithm for the reconstruction of binary images from projections using Gibbs priors presented in [1], [5] gave good results. The main goal of the work reported here is to develop a segmentation algorithm using a similar approach.

[1] The first author was supported by a U.S. National Institutes of Health (NIH) training grant T32 CA 74781. The second author was supported by the NIH grant HL28438, as well as by the U.S. National Science Foundation grant DMS9612077. We thank the Laboratory of Retinal Microcircuitry at the University of Pennsylvania for generously providing space and computer facilities to the first author.

[2] Email: eilat@seas.upenn.edu

[3] Email: gaborherman@netscape.net

In this paper, an *image* is a rectangular two-dimensional array of pixels. A gray-scale image is one where each pixel has a value inclusively between 0 and 255 (where 0 corresponds to black and 255 corresponds to white). A binary image is one where the value of each pixel is either 0 or 1. In the discussion of binary images that follows, we use black and 0, as well as white and 1, interchangeably.

Appropriate prior information regarding the class of images to be segmented may be useful for the segmentation process. In addition to the inherent information in the gray-scale image, local properties of the desired binary image can also provide useful information.

Local properties of a binary image ω, defined on (an arbitrary, but fixed array containing) H pixels ($\omega(h)$ is either black or white for each pixel indexed by an integer $1 \leq h \leq H$), can be characterized by a *Gibbs distribution* of the form

$$Q(\omega) = \frac{1}{Z} \exp(\gamma \sum_{h=1}^{H} I_h(\omega)),$$

where $Q(\omega)$ is the probability of occurrence of image ω, Z is the normalizing factor (which insures that Q is a probability function over the set of all binary images defined on the fixed set of H pixels), γ is a positive parameter defining the "peakedness" of the Gibbs distribution, and $I_h(\omega)$ is the local potential at the pixel indexed by h in image ω. Since we take $\gamma > 0$, $Q(\omega)$ increases as the sum of the local potentials increases.

In this paper, we define the *clique*, $C(h)$, centered at a pixel h to consist of the nine pixels in the 3×3 neighborhood of h. We restrict our attention to Gibbs distributions in which the local potential at a pixel h depends only on the values of the elements of $C(h)$ in the binary image ω. (To make this and later definitions apply everywhere, we treat pixels near the edge of a binary image as if the image were extended by a boundary of two layers of pixels. In the experiment described here, we use a boundary that is all zeros.)

For all pixels h in an (actual; i.e., not the extended) binary image, the color of pixel h influences the values of the local potentials only at the pixels in $C(h)$. These nine local potentials depend only on the values of the 25 pixels in the 5×5 neighborhood of h. Therefore, the change in the probability $Q(\omega)$ that results from switching the color of pixel h (in the binary image ω) depends only on the values of those 25 pixels in ω.

We assume that we are given a set of typical (to a given application area) gray-scale images and their corresponding gold-standard segmentations represented as binary images. We refer to these images as the *training set*.

Appropriate definition of the local potentials plays an important role in the success of image segmentation. The definition should reflect the characteristics of a typical image of the particular application area. The local potential is defined in such a way that it encourages certain local configurations, such as uniform white or black clusters of pixels. There are many possible ways

of defining the local potentials. One simple approach is the following. We partition the set of 512 possible 3×3 binary clique configurations into 6 clique configuration types: 0) the all black configuration, 1) the all white configuration, 2) clique configurations in which the white pixels form an edge, 3) clique configurations in which the white pixels form a convex corner, 4) clique configurations in which the white pixels form a concave corner, and 5) any other clique configurations.

The working definition of the clique configurations types is as follows. A configuration is one of the first five types only if there are k consecutive white pixels and $8-k$ consecutive black pixels among the eight external pixels of the clique. If the central pixel is black, then the configuration is a black region, a convex corner, an edge, or a concave corner, if, and only if, $k = 0$, $1 \leq k \leq 2$, $k = 3$, or $4 \leq k \leq 5$, respectively. Similarly, if the central pixel is white, then the configuration is a convex corner, an edge, a concave corner or a white region, if, and only if, $3 \leq k \leq 4$, $k = 5$, $6 \leq k \leq 7$, or $k = 8$, respectively. Otherwise, the clique configuration is of type other.

Let $t_h(\omega)$ be the clique configuration type at pixel h in image ω ($0 \leq t_h(\omega) < 6$). (For more information on local potentials and clique configuration types, see [3].) We create the prior information from the binary training images in the following way. We create a table r of size 6. We store in its t'th location ($0 \leq t < 6$) the number of internal cliques of configuration type t in the binary training images divided by the number of different clique configurations which are of type t. (By an *internal clique* we mean those cliques which are entirely included in the actual image, without the boundary discussed earlier.) The table value $r(t)$ gives us the average number of appearances of cliques of a certain type t. The local potential $I_h(\omega)$ is defined as $\ln(r(t_h(\omega)) + 1)$. These local potentials give us information about the frequency of occurrence of the different clique configuration types in the binary training images. The usefulness of the resulting distribution depends on the size of the collection of the training images and on how representative they are for the application area. As we further discuss in Section 3, we adjust the value of the parameter γ in the definition of the Gibbs distribution $Q(\omega)$ in such a way that the expected percent of white pixels in a random image of the Gibbs distribution will be the average percent of white pixels in the binary training images.

We also use the training images to correlate the characteristics of the binary images and the gray-scale images. Let $g(h)$ be the mean of the gray-scale values in $C(h)$. For each clique configuration type t ($0 \leq t < 6$), we calculate the mean and standard deviation over all pixels h, of type t in the set of given binary images, of the corresponding values of $g(h)$. We denote the mean and standard deviation for each type t by μ_t and σ_t, respectively. In this paper, we write $\mu_h(\omega)$ for $\mu_{t_h(\omega)}$ and $\sigma_h(\omega)$ for $\sigma_{t_h(\omega)}$.

The segmentation algorithm should find a binary image which is not only statistically consistent with the gray-scale image, but which is also typical of the Gibbs distribution of the binary images. Thus, if $q(g|\omega)$ is the probability

of g (the image containing the values of $g(h)$, $1 \leq h \leq H$), given a binary image ω, then Bayes' theorem says that the posterior probability of ω (given g) is proportional to

$$M(\omega) = \{Q(\omega)q(g|\omega)\}^{\beta},$$

where $Q(\omega)$ is the Gibbs distribution of the binary images as defined above and $\beta = 1$. Assuming that the $g(h)$ $(1 \leq h \leq H)$ of previous paragraph are independently and normally distributed, we have that

$$q(g|\omega) = \prod_{h=1}^{H} \frac{1}{\sqrt{2\pi}\sigma_h(\omega)} \exp\left(\frac{-(g(h) - \mu_h(\omega))^2}{2\sigma_h^2(\omega)}\right).$$

Since our task is to find the ω^* which maximizes $M(\omega)$ and this ω^* is independent of β (as long as $\beta > 0$), for algorithmic reasons we use the more general form of $M(\omega)$ with $\beta > 0$, rather than just with $\beta = 1$.

An *annealing schedule* is a schedule in which an initial inverse temperature, an incrementing schedule, and a stopping criterion are specified. Here, the parameter β is the inverse temperature. It is not held constant, instead it is incremented in the segmentation algorithm as described by the annealing schedule.

We use the iterative Metropolis Algorithm [4] to estimate the binary image ω^* that maximizes $M(\omega)$. We start the algorithm with a completely white image. In the iterative step, the current image ω_1 is changed into ω_2 by randomly picking a single pixel h' and changing its color. Let p be defined by

$$\begin{aligned} p &= \frac{M(\omega_2)}{M(\omega_1)} \\ &= \exp(\beta \sum_{h \in C(h')} \{\gamma(I_h(\omega_2) - I_h(\omega_1)) \\ &\quad -(\ln(\frac{\sigma_h(\omega_2)}{\sigma_h(\omega_1)}) + \frac{(g(h) - \mu_h(\omega_2))^2}{2\sigma_h^2(\omega_2)} - \frac{(g(h) - \mu_h(\omega_1))^2}{2\sigma_h^2(\omega_1)})\}). \end{aligned}$$

Image ω_1 is replaced by ω_2 if p is greater than or equal to one, and image ω_1 is replaced by ω_2 with probability p if p is less than one. This is equivalent to replacing image ω_1 by ω_2 with probability $\min(p, 1)$.

The program runs for $120,000$ cycles with varying values of β. In each cycle, the algorithm performs H iterations. The program outputs the image ω in the sequence with the highest value of $M(\omega)$.

2 Reducing Run-Time

We devised several ways to reduce the run-time of the segmentation program.

Create look-up tables:

Let ω be a binary image and let h' be a pixel. The value of p corresponding to changing the color of a pixel h' depends on the local potentials of pixels

whose cliques contain h'. Therefore, p depends (but not exclusively) on the twenty-five binary pixel values in the 5×5 neighborhood of h' in ω.

Consider the two images ω_1 and ω_2, where ω_2 is obtained from ω_1 by changing the color of a randomly chosen pixel h'. Note that a part of the term in the exponent of p can be written as

$$\begin{aligned}
&\frac{(g(h)-\mu_h(\omega_2))^2}{2\sigma_h^2(\omega_2)} - \frac{(g(h)-\mu_h(\omega_1))^2}{2\sigma_h^2(\omega_1)} \\
&= g^2(h)\cdot\frac{\{\sigma_h^2(\omega_1)-\sigma_h^2(\omega_2)\}}{2\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)} \\
&+g(h)\cdot\frac{\{\sigma_h^2(\omega_2)\mu_h(\omega_1)-\sigma_h^2(\omega_1)\mu_h(\omega_2)\}}{\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)} \\
&+\frac{\{\sigma_h^2(\omega_1)\mu_h^2(\omega_2)-\sigma_h^2(\omega_2)\mu_h^2(\omega_1)\}}{2\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)}.
\end{aligned}$$

We define

$$\begin{aligned}
B_{h'}(\omega_1) = \sum_{h\in C(h')} &(\gamma\{I_h(\omega_2)-I_h(\omega_1)\} - \ln(\frac{\sigma_h(\omega_2)}{\sigma_h(\omega_1)}) \\
&-\frac{\{\sigma_h^2(\omega_1)\mu_h^2(\omega_2)-\sigma_h^2(\omega_2)\mu_h^2(\omega_1)\}}{2\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)}).
\end{aligned}$$

Let B be the value of $B_{h'}(\omega_1)$ and let B' be the value of $B_{h'}(\omega_2)$. It is easy to check that $B = -B'$. Furthermore, the pair of values $\{B, B'\}$ does not depend on the color of h'. If we are given the colors of the 24 pixels in the 5×5 neighborhood of h' (excluding the value of h'), we can calculate both B and B'. However, due to the equality $B = -B'$, we need to store only one of these two values (we discuss in the next paragraph how we do this).

Let ω_b be the image ω with the color of h' changed to black (if need be) and ω_w be the image ω with the color of h' changed to white (if need be). Let

$$\begin{aligned}
T_{h'}^0(\omega) = [\alpha \sum_{h\in C(h')} &(\gamma\{I_h(\omega_w)-I_h(\omega_b)\} - \ln(\frac{\sigma_h(\omega_w)}{\sigma_h(\omega_b)}) \\
&-\frac{\{\sigma_h^2(\omega_b)\mu_h^2(\omega_w)-\sigma_h^2(\omega_w)\mu_h^2(\omega_b)\}}{2\sigma_h^2(\omega_b)\sigma_h^2(\omega_w)})],
\end{aligned}$$

where α is a large positive integer such that $T_{h'}^0(\omega)$ can be stored as a 4-byte integer for all clique configurations (it is used to increase accuracy of the value) and $[\ \ ]$ denotes rounding to the nearest integer. For all 24-element neighborhood configurations, we store the value of $T_{h'}^0(\omega)$ in a look-up table. The look-up table has 2^{24} entries of type 4-byte integers, therefore it requires only 2^6 megabytes. Since the value $T_{h'}^0(\omega)$ does not depend on the color of the pixel h', note that in an iterative step of the Metropolis Algorithm we will always have that $T_{h'}^0(\omega_1) = T_{h'}^0(\omega_2)$.

We define two more variables,

$$T_h^1(\omega) = [\alpha \frac{\sigma_h^2(\omega_2)\mu_h(\omega_1) - \sigma_h^2(\omega_1)\mu_h(\omega_2)}{\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)}],$$

$$T_h^2(\omega) = [\alpha \frac{\sigma_h^2(\omega_1) - \sigma_h^2(\omega_2)}{2\sigma_h^2(\omega_1)\sigma_h^2(\omega_2)}].$$

These values depend only on the values of $t_h(\omega_1)$, $t_h(\omega_2)$ (the clique configuration type of pixel h' in images ω_1, ω_2, respectively). We store the values of $T_h^1(\omega)$ and $T_h^2(\omega)$ in two additional look up tables. These tables are relatively small; they have 6×6 entries each of type 4-byte integer. Note that the values of $T_h^0(\omega)$, $T_h^1(\omega)$, and $T_h^2(\omega)$ do not depend on the values of $g(h)$, $1 \leq h \leq H$.

Create random number files:

We create the following random number tables: one table for row picking, one table for column picking, and one table for each value of β in the annealing schedule containing integers of type $\left[\frac{\alpha}{\beta} \ln x\right]$, where x is a random number uniformly distributed in the range $(0, 1]$. The size of these files will be further discussed in the Section 4.

Integer arithmetic:

The values of $g(h)$ and $g^2(h)$ are used in every iteration several times. Therefore, the integer values of $[g(h)]$ and $[g(h)]^2$ for the gray-scale image to be segmented are calculated once at the beginning of the segmentation program and stored for future use. We save the table values introduced above and the values of $[g(h)]$ and $[g(h)]^2$ as integers, so that all arithmetic within the segmentation program is exact. This way, no rounding errors arise and no re-calibrations are necessary.

Reduce calculations:

We are interested in finding the binary image ω^* which maximizes the function $M(\omega)$, not in the actual value of $M(\omega^*)$. Let ω_W be the binary white image ($\omega_W(h) = 1$ for all $1 \leq h \leq H$). Notice that searching for the binary image ω^* which maximizes the function $M(\omega)$ is equivalent to searching for the binary image $\omega^\wedge$ which maximizes the function $M^\wedge(\omega) = \frac{M(\omega)}{M(\omega_W)}$, which is equivalent to searching for the binary image $\omega^\bullet$ which maximizes the function $\frac{1}{\beta} \ln M^\wedge(\omega)$, i.e. $\omega^* = \omega^\wedge = \omega^\bullet$. We therefore set out to maximize the function $\frac{1}{\beta} \ln M^\wedge(\omega)$, which does not depend on the value of β. The segmentation algorithm is initialized with the image ω_W, for which $\frac{1}{\beta} \ln M^\wedge(\omega_W) = 0$. By maximizing $\frac{1}{\beta} \ln M^\wedge(\omega)$ instead of $M(\omega)$, we do not have to calculate the initial value of $M(\omega_W)$.

3 Implementation

Assuming that we are given a training set, the first step of the implementation is to calculate the local potentials as described in Section 1. The second step is to find the appropriate value of γ for the given training set. Let f be the percent of foreground pixels (pixels with value 1) in the binary training images. (In trabecular bone images, f is the average bone volume fraction in the given training images.) We run a variant of the Metropolis Algorithm described above to produce random binary images from the distribution $Q(\omega)$. We keep track of the percent of foreground pixels in these random images. We search for the value of γ for which these percentages are close to f.

The third step is to calculate the values of μ_t and σ_t for $0 \leq t < 6$. This is done as follows: For a given clique configuration type t_0, find all the pixels in all the binary training images whose clique configurations are internal and of type t_0, calculate the corresponding values of $g(h)$ from the gray-scale training images, and calculate the mean and standard deviation of the values of $g(h)$.

Finally, we discuss how an iterative step of the segmentation algorithm can be carried out using the tables described in Section 2 and a 4-byte random integer of the form $\left\lceil \frac{\alpha}{\beta} \ln x \right\rceil$. For any image ω and pixel h', we define two integers

$$S_{h'}(\omega) = \sum_{h \in C(h')} \{[g(h)]\, T_h^1(\omega) + [g(h)]^2\, T_h^2(\omega)\},$$

$$v_{h'}(\omega) = \begin{cases} T_{h'}^0(\omega) - S_{h'}(\omega), \text{ if } \omega(h') = 0, \\ -T_{h'}^0(\omega) - S_{h'}(\omega), \text{ if } \omega(h') = 1. \end{cases}$$

Using previously given definitions, it is easy to check that in the iterative step of the Metropolis Algorithm $v_{h'}(\omega_1) \cong \frac{\alpha}{\beta} \ln \frac{M(\omega_2)}{M(\omega_1)} = \frac{\alpha}{\beta} \ln \frac{M^\wedge(\omega_2)}{M^\wedge(\omega_1)}$ (equal up to some rounding error), irrespective of the color of h'. It follows that, whenever it is decided to change the color of h', the resulting value of $\frac{\alpha}{\beta} \ln M^\wedge(\omega_2)$ can be estimated by adding the integer $v_{h'}(\omega_1)$ to the known value $\frac{\alpha}{\beta} \ln M^\wedge(\omega_1)$. (For a large value of α, $v_{h'}(\omega)$ is a good approximation of the difference in the value of $\frac{\alpha}{\beta} \ln M^\wedge(\omega)$. After many iterations, we cannot be sure how close the accumulated value is to the actual value of $\frac{\alpha}{\beta} \ln M^\wedge(\omega)$, which mandates experimental verification of the usefulness of the process that is being described here.)

Also, it is the case that a correct way to decide whether or not ω_1 should be replaced by ω_2 in the Metropolis Algorithm is to check whether or not $\left\lceil \frac{\alpha}{\beta} \ln x \right\rceil \leq v_{h'}(\omega_1)$. Indeed, since $v_{h'}(\omega)$ is an integer, this is equivalent to $\ln x \leq \frac{\beta}{\alpha} v_{h'}(\omega_1)$, which is equivalent to

$$x \leq \exp(\frac{\beta}{\alpha} v_{h'}(\omega)) \cong \frac{M(\omega_2)}{M(\omega_1)} = p,$$

and the probability of this is p since x is uniformly distributed over the interval

$(0, 1]$.

We create the random number files and look-up tables described in Section 2 before running the segmentation program. At the beginning of the segmentation program, we calculate and store the values of $[g(h)]$ and $[g(h)]^2$. We start the algorithm with the white image ω_W. In each iteration, we

(i) read a random row and a random column, which corresponds to randomly choosing a pixel h',

(ii) look-up the corresponding table value $T^0_{h'}(\omega_1)$ and calculate the value of $S_{h'}(\omega_1)$,

(iii) calculate the value of $v_{h'}(\omega_1)$,

(iv) read a random integer $\left\lceil \frac{\alpha}{\beta} \ln x \right\rceil$, and

(v) if $\left\lceil \frac{\alpha}{\beta} \ln x \right\rceil \leq v_{h'}(\omega_1)$, then change the color of h' in ω_1 and add $v_{h'}(\omega_1)$ to $\frac{\alpha}{\beta} \ln M^{\wedge}(\omega_1)$ in order to obtain the value of $\frac{\alpha}{\beta} \ln M^{\wedge}(\omega_2)$.

The algorithm performs $120,000$ cycles, where each cycle is H iterations as described above. We start each of the $10,000$ cycles (except the first $10,000$ cycles which starts from a white image) with the image with the highest value of $\frac{\alpha}{\beta} \ln M^{\wedge}(\omega)$ so far. The output is the image for which the highest value of $\frac{\alpha}{\beta} \ln M^{\wedge}(\omega)$ was obtained (which is the image for which the highest value of $\frac{1}{\beta} \ln M^{\wedge}(\omega)$ was obtained since $\alpha > 0$).

4 Experiment

As a preliminary experiment, we scanned with high resolution one 128×128 nuclear magnetic resonance (NMR) gray-scale image of trabecular bone (taken from [2]); see the left image in Figure 1. Using tools from Adobe Photoshop version 5.5 (Adobe System Inc., San Jose, CA) and The Image Processing Tool Kit version 3.0 (Reindeer Games Inc., Raleigh, NC) we created the "gold-standard" segmentation (this segmentation is obviously not perfect.); see the right image in Figure 1. In this experiment, the training set consists of only the one gray-scale image and its corresponding binary segmented image introduced here.

From our training set, we first calculate the values of the local potentials (9.37, 7.32, 4.12, 3.26, 3.31, 0.11), for $0 \leq t < 6$ respectively. The percent of foreground pixels (pixels with value 1) in the binary training image is $f = 17.6\%$. We search for the value of γ (to two significant digits) for which the average number of foreground pixels in random images from the distribution $Q(\omega)$ is as close as possible to f. In this experiment, this was fulfilled by the value $\gamma = 0.083$.

We need approximately 2 billion rows/columns/integers each in $120,000$ cycles of one segmentation of a 128×128 trabecular bone image. We claim that a file of 50 million random rows/columns/integers each is large enough if

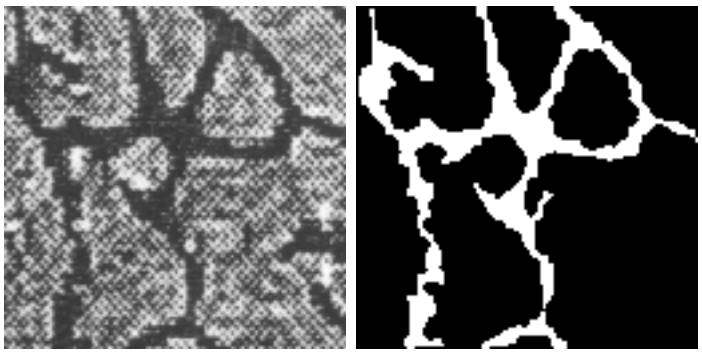

Fig. 1. The image on the left has been obtained by scanning from a printed article a NMR gray-scale image of trabecular bone (128×128 pixels). (In NMR images, bone gives off no signal and is therefore black.) The image on the right is the "gold-standard" segmentation.

a random starting position for reading 16.4 million numbers from each file is chosen every $1,000$ cycles.

We create the following random number tables of 50 million random numbers each: one table of 1-byte unsigned characters for row picking, one table of 1-byte unsigned characters for column picking, and one table of 4-byte integers of type $\left\lfloor \frac{\alpha}{\beta} \ln x \right\rfloor$ for each value of β in the annealing schedule, as described above. In this experiment, we choose the value $\alpha = 4,500,000$. We use the following simple annealing schedule: start at $\beta = 0.1$, increase the value of β by 0.1 every $10,000$ cycles for a total of $120,000$ cycles. (The smallest value that x can be is $\frac{1}{\text{RAND\_MAX}}$. In our case $\text{RAND\_MAX} = 2^{31} - 1$ and for the value of $\beta = 0.1$, $\left\lfloor \frac{\alpha}{\beta} \ln x \right\rfloor$ is approximately $-1,000,000,000$, which can be stored as a 4-byte integer.)

We found the values of μ_t (112.27, 187.45, 164.64, 145.07, 174.93, 166.80) and σ_t (32.07, 5.60, 16.06, 20.68, 12.85, 17.60), for $0 \leq t < 6$ respectively, from the gray-scale image and its corresponding binary image. We create the three look-up tables described in Section 2.

See the left image in Figure 2 for an example of the output of our segmentation algorithm. There are 8.6 percent pixels different between the segmented image shown here and the binary training image. The value of $\left\lfloor \frac{1}{\beta} \ln M(\omega) \right\rfloor$ (which does not depend on the value of β) for the binary training image is $-75,737$, while it is $-73,124$ for the segmented binary image. Notice that the segmentation method presented here produces an image for which the value of $\left\lfloor \frac{1}{\beta} \ln M(\omega) \right\rfloor$ is higher than for the binary training image. This is not surprising since the binary training image doesn't pick up all the characteristics of the gray-scale image.

There are 23.6 percent foreground pixels in the binary training image, compared to 17.6 percent foreground pixels in the output of our segmentation. It is clear that the output of our algorithm contains many small (connected)

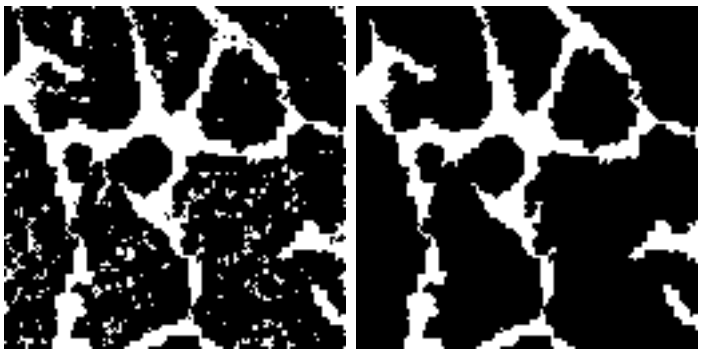

Fig. 2. The image on the left is an example of an output image of our segmentation algorithm. The image on the right is obtained by post-processing the image on the left using threshold values: threshold-black=30 and threshold-white=60 (see text).

components that are not part of the trabecular bone (trabecular bone is mostly a connected structure). These components seem to arise from the noisy gray-scale image which we tried to segment. Some post-processing is necessary in order to remove these components.

5 Post-Processing

The images that are output from our segmentation algorithm contain both small white and black connected-components that need to be removed (the color needs to be switched). Let *threshold-white* be the number of pixels in the largest 8-connected-component of white pixels that will be removed (color will be turned to black). Let *threshold-black* be the number of pixels in the largest 4-connected-component of black pixels that will be removed (color will be turned to white). Let the *size* of a connected-component be the number of pixels in the component.

We did the following experiment. We find and remove all the 8-connected components of white pixels whose size is less than or equal to threshold-white. Next, we find and remove all the 4-connected components of black pixels whose size is less than or equal to threshold-black. We came to the conclusion that removing the components in the manner described above during the segmentation algorithm was not necessary; it is sufficient to remove the small components from the output segmentation.

Having come to this conclusion, we had the task of finding good values for threshold-white and threshold-black. We found that in this particular application, using threshold-white=60 and threshold-black=30 gave good results. Using larger threshold values resulted in losing parts of the bone, while using smaller threshold values resulted in some unwanted small connected-components in the binary image. For these threshold values, the percent of different pixels between the post-processed segmented image and the gold-standard segmentation is 4.6, compared to 8.6 percent different pixels between the not post-processed segmented image and the gold-standard image,

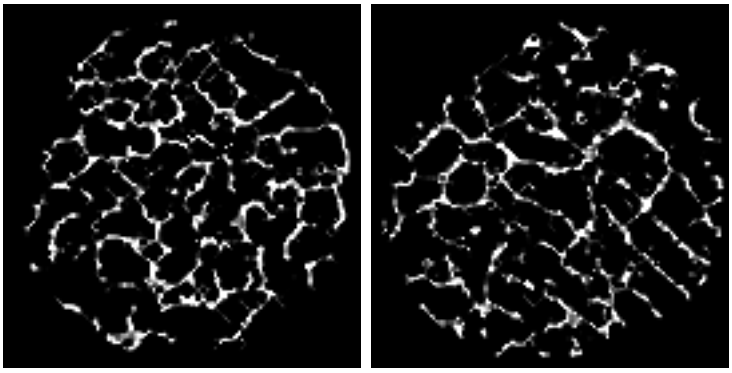

Fig. 3. These are two-dimensional images of trabecular bone taken *ex-vivo*, which were processed to resemble *in-vivo* images.

see Figure 2. There are 19.7% foreground pixels in the post-processed segmented image (compared to 17.6% in the training image and 23.6% in the not post-processed segmented image).

6 Discussion

Encouraged by the preliminary results reported here, we plan to segment *in-vivo* NMR images of trabecular bone. In Figure 3 we show a couple of two-dimensional images of trabecular bone taken *ex-vivo*, which were processed to resemble *in-vivo* images. For the segmentation of such images to be successful, it is necessary to choose appropriate clique configuration types. It is clear the the configuration types used in the experiment described in Section 4 would not be a good choice for these images which contain much smaller detail. A suitable model for the posterior information must be found. After some preliminary experiment, it seems that a different type of posterior information than that proposed here would prove more useful in this case. Finally, some experimentation is necessary in order to determine what is a good annealing schedule for these segmentations.

7 Conclusion

Gibbs priors appear to be a potentially useful tool for segmentation of complex objects in noisy images.

References

[1] Carvalho, B. M., G. T. Herman, S. Matej, C. Salzberg and E. Vardi, *Binary Tomography for Triplane Cardiography*, in Information Processing in Medical Imaging, A. Kuba, M. Samal, A. Todd-Pokropek (eds.), Springer, Berlin. (1999), 29–41.

[2] Hwang, S. N., F. W. Wehrli, and J. L. Williams, *Probability-based structural parameters from 3D NMR images as predictors of trabecular bone strength*, Med. Phys. **24** (1997), 1255–1261.

[3] Liao, H. Y., and G. T. Herman, *Automated estimation of the parameters of Gibbs priors to be used in binary tomography*, Electronic Notes in Theoretical Computer Science **46** (2001),
URL: `http://www.elsevier.nl/locate/entcs/volume46.html`.

[4] Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. **21** (1953), 1087–1092.

[5] Vardi, E., G. T. Herman and T. Y. Kong, *Speeding up stochastic reconstruction of binary images from limited projection directions*, Linear Algebra and its Applications: Special Issue on Discrete Tomography (to appear).