International Journal of Research Studies in Science, Engineering and Technology
Volume 1, Issue 8, November 2014, PP 85-97
ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

An Efficient Semi Supervised Segmentation Model for Collection of Images

***T Srimallika, ** B. Sivanageswara rao**

*PG Student, Eswar College of Engineering, Kesanupalli, Narasaraopet, Guntur(Dt)
** Assistant Professor, Eswar College of Engineering, Kesanupalli, Narasaraopet, Guntur(Dt)

Abstract: *In this paper, we consider the problem of segmentation of large collections of images. We propose a semi supervised optimization model that determines an efficient segmentation of many input images. The advantages of the model are twofold. First, the segmentation is highly controllable by the user so that the user can easily specify what he/she wants. This is done by allowing the user to provide, either offline or interactively, some (fully or partially) labeled pixels in images as strong priors for the model. Second, the model requires only minimal tuning of model parameters during the initial stage. Once initial tuning is done, the setup can be used to automatically segment a large collection of images that are distinct but share similar features.*

1. Background

In this project we are going to segment the large collection of data by two fold First, the segmentation is highly controllable by the user so that the user can easily specify what he/she wants. The model requires only minimal tuning of model parameters during the initial stage. Once initial tuning is done, the setup can be used to automatically segment a large collection of images.

1.1 Segmentation

All image processing operations generally aim at a better recognition of objects of interest, i. e., at finding suitable local features that can be distinguished from other objects and from the background. The next step is to check each individual pixel to see whether it belongs to an object of interest or not. This operation is called *segmentation* and produces a *binary image*.

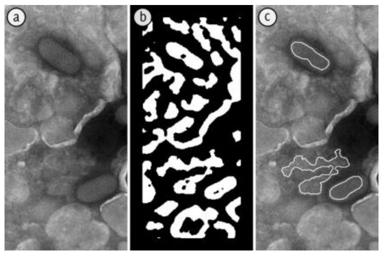


Fig 2.1. *Image Segmentation*

A pixel has the value one if it belongs to the object; otherwise it is zero. Segmentation is the operation at the threshold between *low-level image processing* and *image analysis*. After segmentation, it is known that which pixel belongs to which object. The image is parted into regions and we know the discontinuities as the boundaries between the regions. The different types of segmentations are

1.1.1 THRESHOLDING

The simplest method of image segmentation is called the thresholding method. This method is based on a clip-level (or a threshold value) to turn a gray-scale image into a binary image.

The key of this method is to select the threshold value (or values when multiple-levels are selected). Several popular methods are used in industry including the maximum entropy method, Otsu's method (maximum variance), and k-means clustering.

Recently, methods have been developed for thresholding computed tomography (CT) images. The key idea is that unlike Otsu's method, the thresholds are derived from the radiographs instead of the (reconstructed) image.

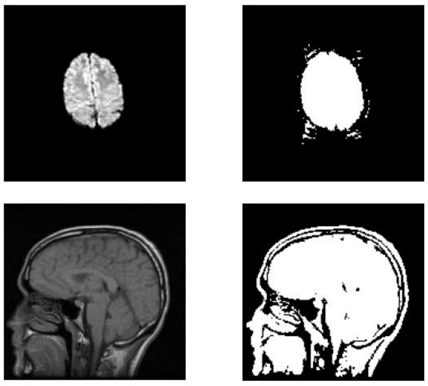


Fig 2.2. *Thresholding Based Image Segmentation*

1.1.2 REGION-BASED METHODS

It focus attention on an important aspect of the segmentation process missed with point-based techniques. There a pixel is classified as an object pixel judging solely on its gray value independently of the context. This meant that isolated points or small areas could be classified as object pixels, disregarding the fact that an important characteristic of an object is its *connectivity*.If we use not the original image but a feature image for the segmentation process, the features represent not a single pixel but a small neighborhood, depending on the mask sizes of the operators used.

At the edges of the objects, however, where the mask includes pixels from both the object and the background, any feature that could be useful cannot be computed. The correct procedure would be to limit the mask size at the edge to points of either the object or the background. But how can this be achieved if we can only distinguish the object and the background after computation of the feature? Obviously, this problem cannot be solved in one step, but only iteratively using a procedure in which feature computation and segmentation are performed alternately.

In the first step, the features are computed disregarding any object boundaries. Then a preliminary segmentation is performed and the features are computed again, now using the segmentation results to limit the masks of the neighborhood operations at the object edges to either the object or the background pixels, depending on the location of the center pixel. To improve the results, feature computation and segmentation can be repeated until the procedure converges into a stable result.

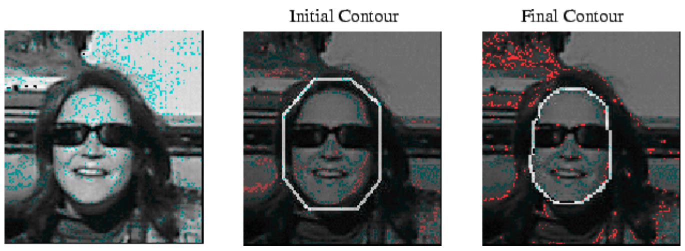


Fig 2.4. *Region Based Image Segmentation*

1.1.3 MODEL BASED SEGMENTATION

The central assumption of such an approach is that structures of interest/organs have a repetitive form of geometry.

Therefore, one can seek for a probabilistic model towards explaining the variation of the shape of the organ and then when segmenting an image impose constraints using this model as prior. Such a task involves

(i) registration of the training examples to a common pose,

(ii) probabilistic representation of the variation of the registered samples, and

(iii) Statistical inference between the model and the image.

State of the art methods in the literature for knowledge-based segmentation involve active shape and appearance models, active contours and deformable templates and level-set based methods.

Module1.

Separate labeled and unlabeled pixels

Step 1. RGB to Gray Conversion

Take the input image. And it converts into grayscale image.

RGB Images

- An RGB image represents each pixel color as a set of three values, representing the red, green, and blue intensities that make up the color.
- In MATLAB, the red, green, and blue components of an RGB image reside in a single m-by-n-by-3 array.
- m and n are the numbers of rows and columns of pixels in the image, and the third dimension consists of three planes, containing red, green, and blue intensity values.
- For each pixel in the image, the red, green, and blue elements combine to create the pixel's actual color.
- An RGB array can be of
  - Class double, in which case it contains values in the range [0, 1].
  - Class uint8, in which case the data range is [0,255].
  - For uint16, values range from [0, 65535].

Grayscale Images

- Also referred to as monochrome or one-color images.
- Contain only brightness information. No color information.
- Typically contain 8 bits/pixel data, which corresponds to 256 (0 to 255) different brightness (gray) levels
- Why 8 bits/pixel?
  - Provides more than adequate brightness resolution.
  - Provides a "noise margin" by allowing approximately twice gray levels as required.
  - Byte (8-bits) is the standard small unit in computers
- However, there are applications such as medical imaging or astronomy that requires 12 or 16 bits/pixel.
  - Useful when a small section of the image is enlarged.
  - Allows the user to repeatedly zoom a specific area in the image.

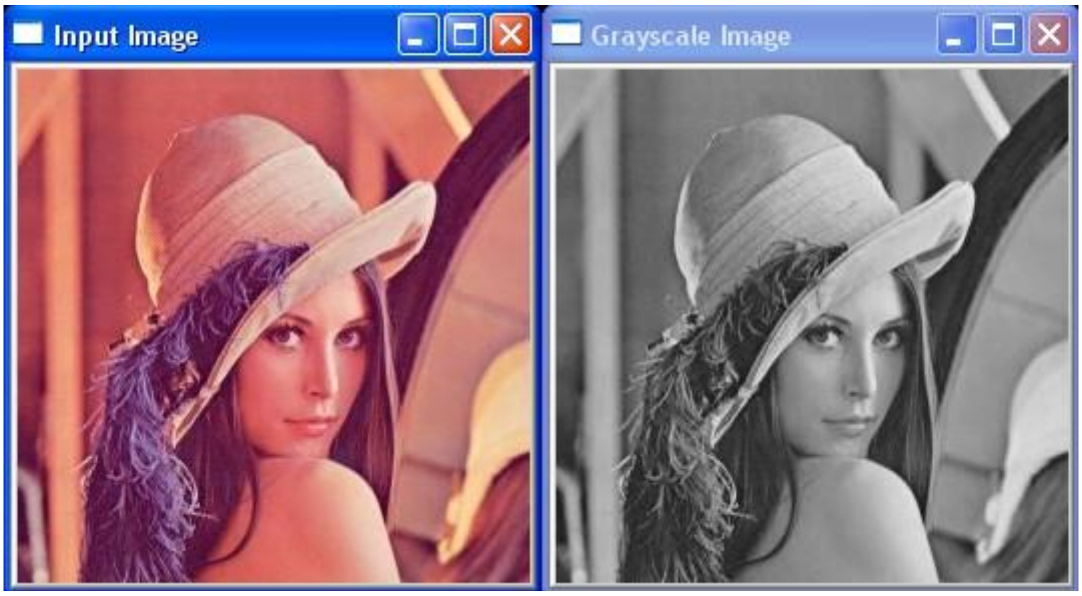


Fig 2.7. *RGB 2 Gray Scale Conversion*

Step 2. Normalization and Enhancement

Give the enhancing factor between (0 - 3) in point variables e.g. (0.5, 1.5). Depends upon enhancing factor enhance of the image.

Normalization

In image processing, normalization is a process that changes the range of pixel intensity values. Applications include photographs with poor contrast due to glare, for example. Normalization is sometimes called contrast stretching. In more general fields of data processing, such as digital signal processing, it is referred to as dynamic range expansion.

The purpose of dynamic range expansion in the various applications is usually to bring the image, or other type of signal, into a range that is more familiar or normal to the senses, hence the term normalization. Often, the motivation is to achieve consistency in dynamic range for a set of data, signals, or images to avoid mental distraction or fatigue.

Image Enhancement

Image enhancement is the improvement of digital image quality (wanted e.g. for visual inspection or for machine analysis), without knowledge about the source of degradation. If the source of degradation is known, one calls the process image restoration. Both are *iconical* processes, viz. input and outputs are images.

Many different, often elementary and heuristic methods are used to improve images in some sense. The problem is, of course, not well defined, as there is no objective measure for image quality. Here, we discuss a few recipes that have shown to be useful both for the human observer and/or for machine recognition. These methods are very problem-oriented: a method that works fine in one case may be completely inadequate for another problem.

Apart from geometrical transformations some preliminary grey level adjustments may be indicated, to take into account imperfections in the acquisition system. This can be done pixel by pixel, calibrating with the output of an image with constant brightness. Frequently space-invariant grey value transformations are also done for contrast stretching, range compression, etc. The critical distribution is the relative frequency of each grey value, the *grey value histogram*. Examples of simple grey level transformations in this domain are:

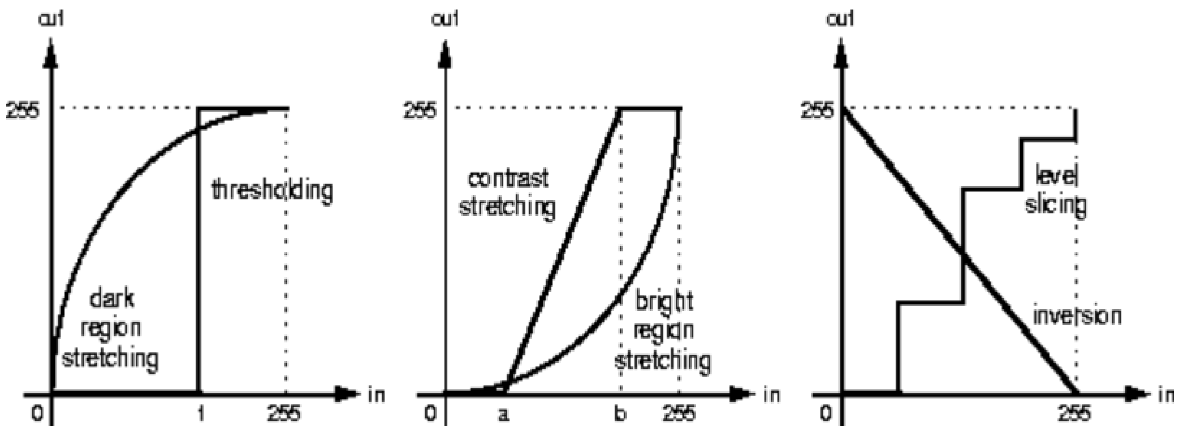


Fig 2.8. *Gray Level Transformations*

Grey values can also be modified such that their histogram has any desired shape, e.g flat (every grey value has the same probability). All examples assume *point processing*, viz. each output pixel is the function of one input pixel; usually, the transformation is implemented with a look-up table:

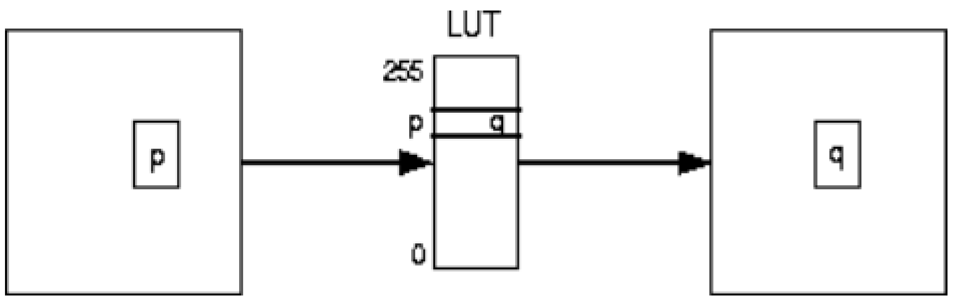


Fig 2.9. *Look Up Table*

Physiological experiments have shown that very small changes in luminance are recognized by the human visual system in regions of continuous grey value, and not at all seen in regions of some discontinuities. Therefore, a design goal for image enhancement often is to smooth images in more uniform regions, but to preserve edges. On the other hand, it has also been shown that somehow degraded images with enhancement of certain features, e.g. edges, can simplify image interpretation both for a human observer and for machine recognition. A second design goal, therefore, is image sharpening. All these operations need *neighbourhood processing*, viz. the output pixel is a function of some neighbourhood of the input pixels:

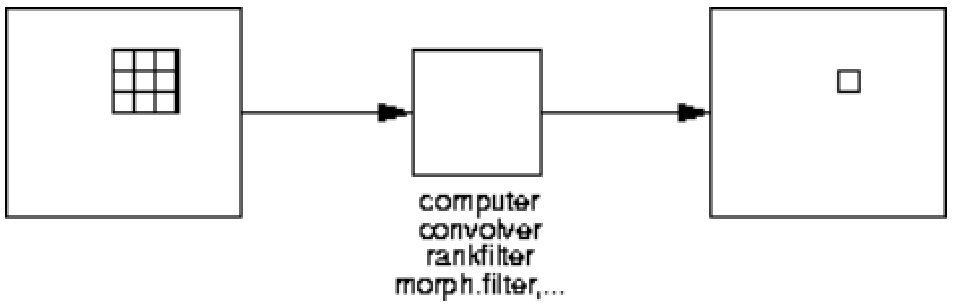
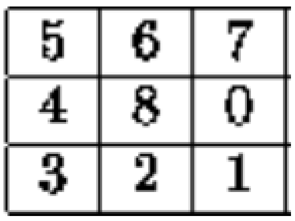


Fig 2.10. *Neighbourhood Operation*

These operations could be performed using linear operations in either the frequency or the spatial domain. We could, e.g. design, in the frequency domain, one-dimensional low or high pass filters (→ Filtering), and transform them according to McClellan's algorithm to the two-dimensional case.

Unfortunately, linear filter operations do not really satisfy the above two design goals; in this book, we limit ourselves to discussing separately only (and superficially) Smoothing and Sharpening.

Here is a trick that can speed up operations substantially, and serves as an example for both point and neighbourhood processing in a binary image: we number the pixels in a 3 X 3 neighbourhood like:



5	6	7
4	8	0
3	2	1

Fig 2.11. *3 X 3 Neighbourhood Examble*

and denote the binary values (0,1) by b_i (i = 0,8); we then concatenate the bits into a 9-bit word, like $b_8b_7b_6b_5b_4b_3b_2b_1b_0$. This leaves us with a 9-bit grey value for each pixel, hence a new image (an 8-bit image with b_8 taken from the original binary image will also do). The new image corresponds to the result of a convolution of the binary image, with a 3 X 3 matrix containing as coefficients the powers of two. This *neighbor image* can then be passed through a look-up table to perform erosions, dilations, noise cleaning, skeletonization, etc.

Apart from point and neighbourhood processing, there are also *global processing techniques*, i.e. methods where every pixel depends on all pixels of the whole image. Histogram methods are usually global, but they can also be used in a neighbourhood.

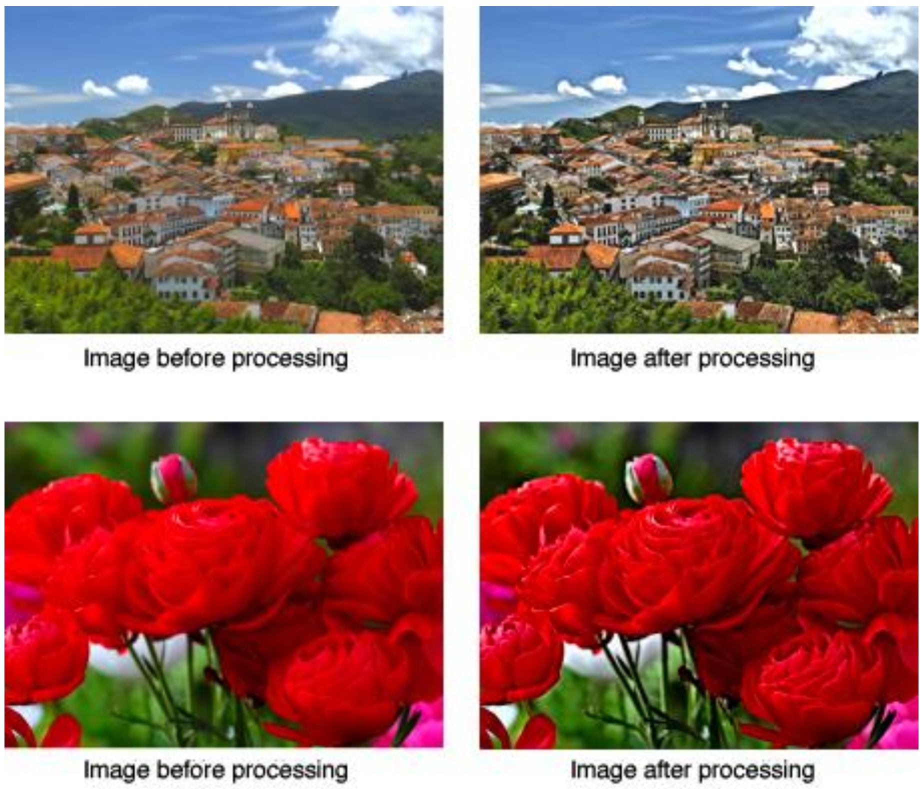


Fig 2.12. *Image Normalization & Enhancement*

Step 3. Binary Image Conversion

Obtain threshold value by taking mean of original image. Depends upon the threshold value binarized the original image. Generate the white image. Check which are the locations are equal to 255 in threshold image, that same location pixels are in enhance image are above the threshold value means that location values are replaced by 0.

Binary Image

- In a binary image, each pixel assumes one of only two discrete values.
- Essentially, these two values correspond to on and off.
- A binary image is stored as a two-dimensional matrix of 0’s (off pixels) and 1’s (on pixels).
- Each pixel is just **black** or **white**. Since there are only two possible values for each pixel (0,1), we only need **one bit** per pixel.
- Binary images are often created from gray-scale images via a threshold operation.
    - White (‘1’) if pixel value is larger than threshold.
    - Black (‘0’) if it is less.
- A binary image is a digital image that has only two possible values for each pixel
- Binary images are also called bi-level or two-level
- A binary image is usually stored in memory as a bitmap, a packed array of bits

- Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding.

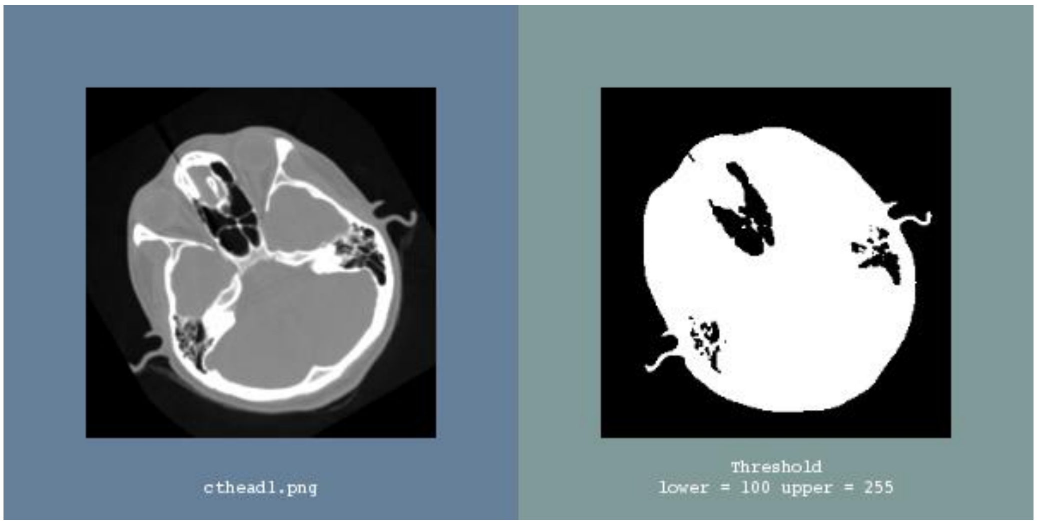


Fig 2.13. *Gray Scale to Binary Image Convertion*

Step 4. Labeled Objects

Give the required size of labeled image. Require size is compared with labeled objects, if the given labeled value is greater than the each cell it will be replaced with 1.else it will be replaced with 0.

Labeled images are integer images where the values correspond to different regions. I.e., region 1 is all of the pixels which have value *1*, region two is the pixels with value 2, and so on. By convention, region 0 is the background and processed differently.

Connected-Component Labeling

The bwlabel and the bwlabeln functions perform *connected-component labeling*, which is a method for identifying each object in a binary image. The bwlabel function supports 2-D inputs only; the bwlabeln function supports inputs of any dimension.

These functions return a matrix, called a *label matrix*. A label matrix is an image, the same size as the input image, in which the objects in the input image are distinguished by different integer values in the output matrix.

For example, bwlabel can identify the objects in this binary image.

```
BW = [00     0     0     0     0     0     0;
        0     1     1     0     0     1     1     1;
        0     1     1     0     0     0     1     1;
        0     1     1     0     0     0     0     0;
        0     0     0     1     1     0     0     0;
        0     0     0     1     1     0     0     0;
        0     0     0     1     1     0     0     0;
        0     0     0     0     0     0     0     0];
```

X = bwlabel(BW,4)

```
X =
0     0     0     0     0     0     0     0
0     1     1     0     0     3     3     3
0     1     1     0     0     0     3     3
0     1     1     0     0     0     0     0
0     0     0     2     2     0     0     0
0     0     0     2     2     0     0     0
0     0     0     2     2     0     0     0
0     0     0     0     0     0     0     0
```

In the output matrix, the 1's represent one object, the 2's a second object, and the 3's a third. (If you had used 8-connected neighborhoods (the default), there would be only two objects, because the first and second objects would be a single object, connected along the diagonal.)

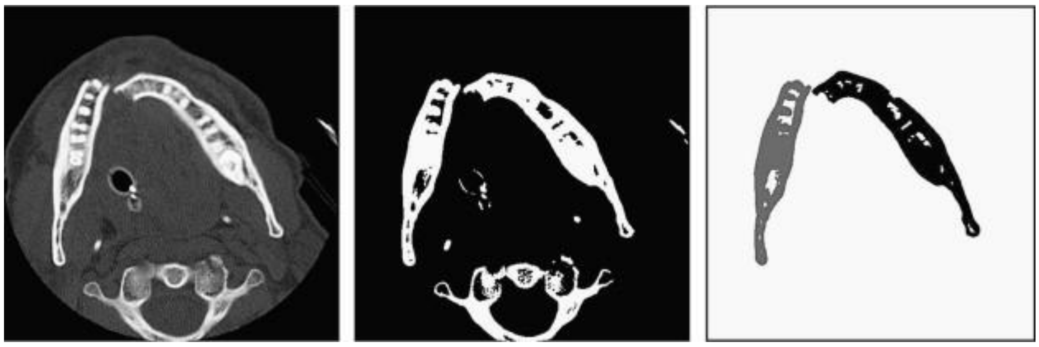


Fig 2.14. Labeled Components

Step 5. Labeled and Unlabeled Image

Give the required size of labeled image. Require size is compared with labeled objects , what are the objects are above the required size of labeled image that objects are replaced by 1 otherwise 0.now we get the labeled and unlabeled image.

Module 2. Similarity Measure

Two kinds of similarity measures, namely, geometric and photometric, are considered. The former is based on pixel locations, whereas the latter is based on color features.

For each pixel $i \in \gamma^s$, its geometric neighbor $G_i^{s,s} \subset \gamma^s$ is defined as,

$$G_i^{s,s} := \{ j \in \gamma^s : 0 < \|i - j\|_\infty \leq r_g \}$$

where $r_g > 0$ is a constant controlling the size of the window, and $\|.\|_\infty$ is the vector maximum norm. We often set $r_g = 1$ so that a 3×3 window around pixel i is used. Note that $i \notin G_i^{s,s}$ and the geometric neighbor is not defined across two images. The geometric similarity $g_{i,j}^{s,s}$ is defined as

$$g_{i,j}^{s,s} : = \begin{cases} ce^{-\frac{\|i-j\|_2^2}{\sigma_i^2}}, & if\ j \in g_{i,j}^{s,s} \\ 0: & otherwise \end{cases}$$

where c is a normalization constant such that $\sum j \in \gamma^s g_{i,j}^{s,s} \equiv 1$, and σ_i^2 is computed as the sample variance of the geometric locations within $G_i^{s,s}$.

For each pixel $\in \gamma^s$, let F_i be its feature vector. We use the RGB values over a 3×3 window around pixel to construct a feature vector of dimension 27. Then, the *within-image photometric neighbor* $p_i^{s,s} \subset \gamma^s$ is defined to be the top 4 pixels within the 17×17 window around pixel (excluding pixel itself), whose feature vectors are nearest to F_i (in Euclidean norm). Using a larger window size allows us to connect photo metrically similar pixels that are further apart. However, doing so will increase the computational cost. The choice of the size 17×17 is a balance between both extremes. The *within-image photometric similarity* is defined as

$$p_{i,j}^{s,s} : = \begin{cases} ce^{-\frac{\|F_i - F_2\|_2^2}{\sigma_i^2}}, & if\ j \in P_{i,j}^{s,s} \\ 0 & , otherwise \end{cases}$$

where c is a normalization constant such that $\sum j \in \gamma^s p_{i,j}^{s,s} \equiv 1$, and $\rho_i{}^2$is computed as the sample variance of the photometric features within $P_i^{s,s}$.

Module 3. Optimization

Each unlabeled pixel can be connected to a labeled pixel through a sequence of directed edges, each of which connects a pixel to one of its neighbors in the same image or a different image.

Let u_s for $s = 1{,}2$ be two given multichannel images. Their sizes are not necessarily the same. Let γ^s be the set of all pixels in image u^s. Let Ω^s be the set of all unlabeled pixels in image u^s. Let Γ^s be the set of pixels in image u^s labeled to one of the classes M by the user. Thus $\gamma^s = \Omega^s \cup \Gamma^s$ Here, we allow both images to contain labeled and unlabeled pixels for the sake of generality. The set of labeled pixels Γ^s is divided into $\Gamma^{s/1}, \dots, \Gamma^{s/m}$, where is the set of pixels that are labeled with class m, for $m = 1, \dots, M$.

Let $s^{'} = 2$ if $s = 1$, and let $s^{'} = 1$ if $s = 2$, so that $s^{'}$ is an index referring to an image different from the image indexed by s. For each pixel $i \in \gamma^s$ and each pixel $j \in \gamma^t$, let $\omega_{i,j}^{s,t} \geq 0$ be a similarity score between the pair of pixels, for $s, t = 1{,}2$. When $t = s$, the similarity $\omega_{i,j}^{s,t}$ is computed within image u^s; when $t = s^{'}$, the similarity is computed across two images. For each $i \in \gamma^s$, it is assumed that the similarity scores are normalized such that

$$\sum_{j \in \gamma^s} \omega_{i,j}^{s,s'} + \sum_{j \in \gamma^{s'}} \omega_{i,j}^{s,s'} \equiv 1$$

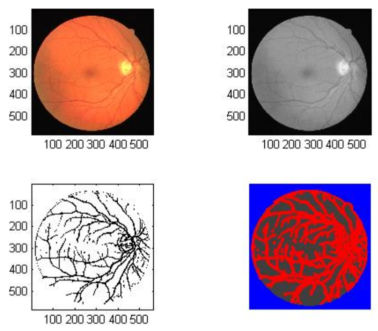
For each pixel $i \in \gamma^s$, let $N_i^{s,t} \subset \gamma^t$ be a set of pixels in image u^t, which is called the *neighbor* of i in u^t. The *within-image neighbor* and the *across-image neighbor* are defined respectively by $N_i^{s,s} := G_i^{s,s} \cup P_i^{s,s}$ and $N_i^{s,s'} := P_i^{s,s'}$. Presumably, these pixels have high similarity scores with i. For each $i \in \gamma^s$, let $\alpha_i^{s/m} \in [0{,}1]$ be the degree of membership of pixel $i \in \gamma^s$ to class m. It is required that $\sum_{m=1}^{M} \alpha_i^{s/m} = 1$. we also denote by $\alpha^{s/m}$ the vector $\left(\alpha_i^{s/m}\right) i \in \gamma^s$.

The basic idea of the model is that the memberships of similar pixels should be similar. For each unlabeled pixel $i \in \Omega^s$, the membership to class m inferred from its neighbors is the weighted average, i.e.

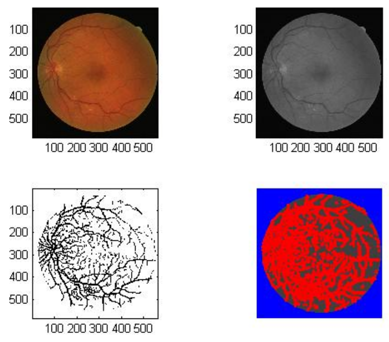
$$\sum_{j \in N_i^{s,s}} \omega_{i,j}^{s,s} \alpha_j^{s/m} + \sum_{j \in N_i^{s,s'}} \omega_{i,j}^{s,s'} \alpha_j^{s'/m} \equiv 1$$

4.3 Screenshots

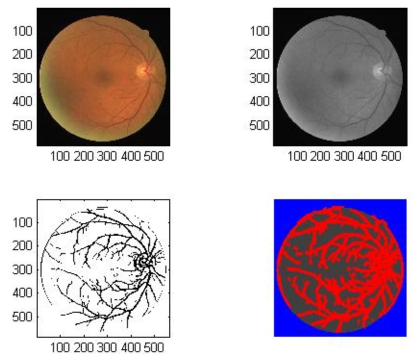
1. First Image



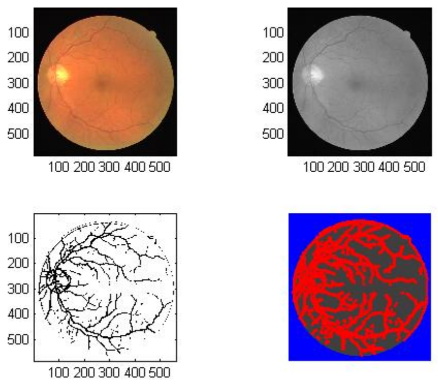
2. Second Image



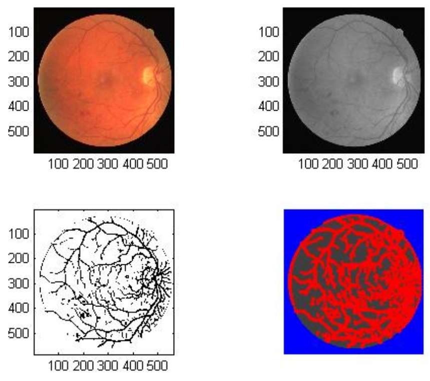
3. Third Image



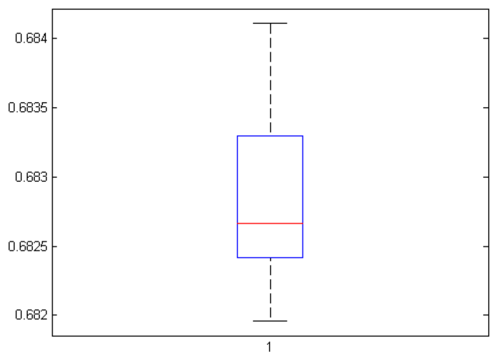
4. Fourth Image



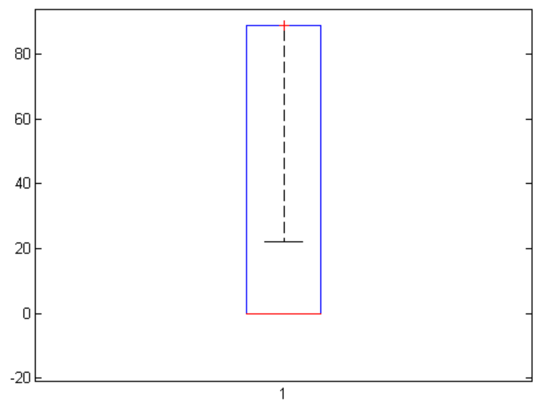
5. Fifth Image



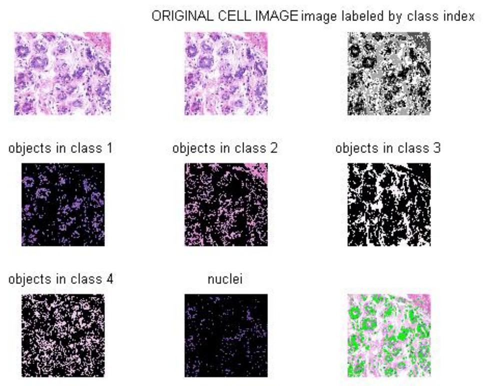
6. Performance of Probability Error



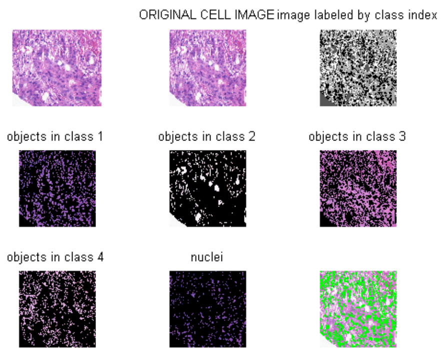
7. Performance of Accuracy



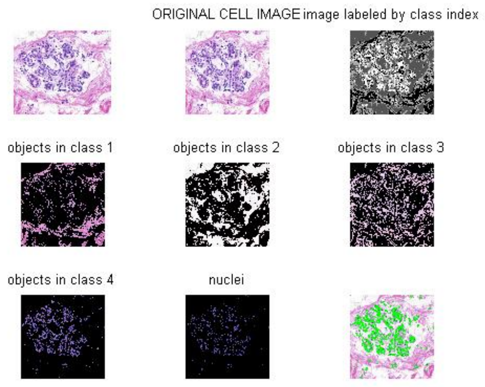
8. First Image



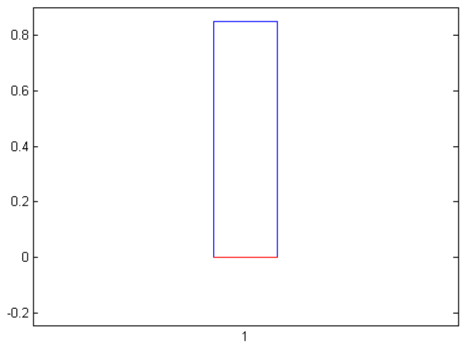
9. Second Image



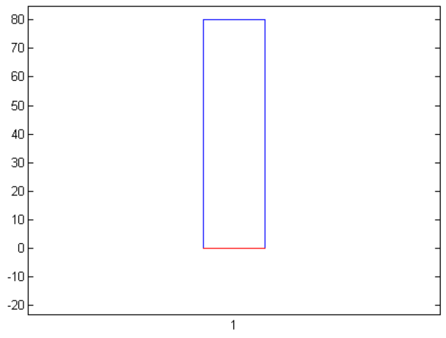
10 . Third Image



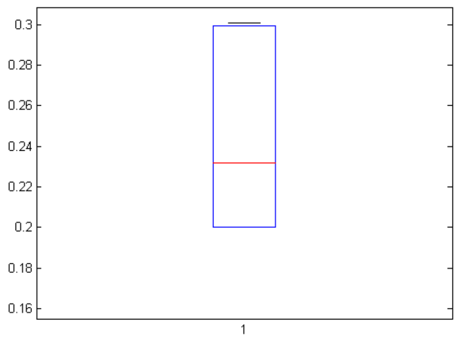
11. Performance of Probability Error



12. Performance of Accuracy



13. Performance of Hausdrouf Distance



CONCLUSION

In this paper, we have proposed and developed a semiautomatic optimization model for segmentation of multiple images. The model has a quadratic objective function and linear constraints. Due to the discrete maximum/minimum principles, the optimality conditions simply boil down to solving linear systems (as opposed to the nonlinear Karush–Kuhn–Tucker systems

for general quadratic programming problems). In our applications, the two parameters can be easily tuned. Once initial tuning is done, the setup can be used to segment all other images within the collection automatically. The quality of the results is also high. However, it relies on the logical assumption that the different classes can be separated in the feature space and that the user-supplied samples can represent each class well.

REFERENCES

[1] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[2] J. Guan and G. Qiu, “Interactive image matting using optimization,” Sch. Comput. Sci. Inf. Technol., Univ. Nottingham, University Park, U.K., Tech. Rep. 01-2006, 2006.

[3] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004.

[4] J.Wang andM. Cohen, “An iterative optimization approach for unified image segmentation and matting,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, pp. 936–943.

[5] S. Yu and J. Shi, “Segmentation given partial grouping constraints,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 173–183, Feb. 2004.

[6] G. Gilboa and S. Osher, “Nonlocal linear image regularization and supervised segmentation,” *Multiscale Model. Simul.*, vol. 6, no. 2, pp. 595–630, 2007.

[7] Y. Boykov and M. P. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in -d images,” in *Proc. IEEE Int Conf. Comput. Vis.*, 2001, pp. 105–112.

[8] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum, “Lazy snapping,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 303–308, Aug. 2004.

[9] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, “Interactive image segmentation using an adaptive GMMRF model,” in *Proc. ECCV*, 2004, pp. 428–441.

[10] A. Protiere and G. Sapiro, “Interactive image segmentation via adaptive weighted distances,” *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1046–1057, Apr. 2007.

[11] N. Houhou, X. Bresson, A. Szlam, T. F. Chan, and J.-P. Thiran, “Semisupervised segmentation based on non-local continuous min-cut,” in *Proc. SSVM*, 2009, pp. 112–123.

[12] J. Guan and G. Qiu, “Interactive image segmentation using optimization with statistical priors,” in *Proc. ECCV*, Graz, Austria, 2006.

[13] M. K. Ng, G. Qiu, and A. M. Yip, “Numerical methods for interactive multiple-class image segmentation problems,” *Int. J. Imag. Syst. Technol.*, vol. 20, no. 3, pp. 191–201, Sep. 2010.