Efficient Algorithms for Optimization-Based Image Segmentation

Tetsuo Asano
School of Information Science,
JAIST,
Asahidai, Tatsunokuchi, Ishikawa, 923-12 JAPAN.
email:t-asano@jaist.ac.jp

Danny Z. Chen
Dept. of Computer Science and Engr.,
University of Notre Dame,
Notre Dame, IN 46556, USA.
email:chen@cse.nd.edu

Naoki Katoh
Department of Architecture,
Kyoto University,
Yoshida-Honmachi, Sakyou-ku, Kyoto, 606-01 JAPAN.
email:naoki@is-mj.archi.kyoto-u.ac.jp

Takeshi Tokuyama
IBM Tokyo Research Laboratory,
Yamato 242, JAPAN.
email:ttoku@trl.ibm.co.jp

Corresponding address:
Tetsuo Asano
School of Information Science,
JAIST,
Asahidai, Tatsunokuchi, Ishikawa, 923-12 JAPAN.
FAX: +81-761-51-1149
email:t-asano@jaist.ac.jp

Abstract

Separating an object in an image from its background is a central problem (called segmentation) in pattern recognition and computer vision. In this paper, we study the complexity of the segmentation problem, assuming that the object forms a connected region in an intensity image. We show that the optimization problem of separating a connected region in an n-pixel grid is NP-hard under the interclass variance, a criterion that is used in discriminant analysis. More importantly, we consider the basic case in which the object is separated by two x-monotone curves (i.e., the object itself is x-monotone), and present polynomial-time algorithms for computing exact and approximate optimal segmentation. Our main algorithm for exact optimal segmentation by two x-monotone curves runs in $O(n^2)$ time; this algorithm is based on several techniques such as a parametric optimization formulation, a hand-probing algorithm for the convex hull of an unknown point set, and dynamic programming using fast matrix searching. Our efficient approximation scheme obtains an ϵ-approximate solution in $O(\epsilon^{-1} n \log L)$ time, where ϵ is any fixed constant with $1 > \epsilon > 0$, and L is the total sum of the absolute values of brightness levels of the image.

1 Introduction

One of the most important operations that a computer vision system is expected to perform is the separation of an object from the background. This operation is commonly called "segmentation". In this paper, we address the segmentation problem that detects or extracts regions corresponding to certain meaningful objects in a given n-pixel intensity image.

Considerable work has been done on the segmentation of intensity images, and five different approaches [5] have mainly been used: threshold techniques, edge-based methods, region-based methods, hybrid techniques, and connectivity-preserving relaxation methods.

The threshold techniques [24] are effective only if all pixels that belong to the objects have brightness levels within a certain range which can be distinguished from those of the background. Since all spatial information is neglected, threshold techniques do not cope well with blurring around region boundaries. The edge-based methods [10] heavily hinge on edge-detection. A difficulty for edge-based methods is how to connect disconnected edges to form a closed curve, especially in a blurred portion. A typical region-based method [3, 26, 16] consists of the following steps: Partition an image into connected regions by grouping together neighboring pixels that have similar brightness levels, and merge two adjacent regions under some criterion such as homogeneity [23, 8] or weakness of region boundaries. A strict criterion generally leads to creation of many small regions, while a loose one may easily merge two regions which should be separated but are adjacent due to blurred boundaries. There are also several hybrid methods [21, 5] in which the above criteria are combined. A typical connectivity-preserving relaxation method such as the "active contour models" [18] starts with some initial shape of the boundary represented by spline curves and changes the shape according to some energy function. One of the disadvantages of connectivity-preserving relaxation methods is that their search for a global optimum can be trapped into a local optimum.

Our approach for solving the segmentation problem is different from those mentioned above. Our basic standpoint is to formulate the segmentation problem as an optimization problem under certain geometric constraints, and study its computational complexity.

The objective function that we seek to optimize is the *interclass variance* (to be defined in Section 2) that is used in discriminant analysis [15]. Although one could argue that the quality of segmentation results may be evaluated eventually only by human eyes, empirical study based on experimental work suggests that segmentation that is optimal in accordance with discriminant analysis produces reasonably satisfactory outcome in most cases.

One important geometric constraint on the output object that we desire is connectivity. Let G be a $\sqrt{n} \times \sqrt{n}$ grid of n pixels. We are able to show that, however, the problem of finding a connected object in G which maximizes the interclass variance is NP-hard. The NP-hardness of this segmentation problem is proven by reducing to it the connected vertex covering problem of a planar graph with maximum degree 4. Similar results are seen in [14, 25]. We outline the NP-hard proof in the Appendix.

In addition to connectivity, we further consider another important geometric constraint called x-monotonicity. A region S in the grid is said to be *x-monotone* if it is bounded by two x-monotone curves. Note that the intersection of an x-monotone region and any pixel column is a (possibly empty) connected region. We call an object *admissible* if it is x-monotone and connected (Figure 1).

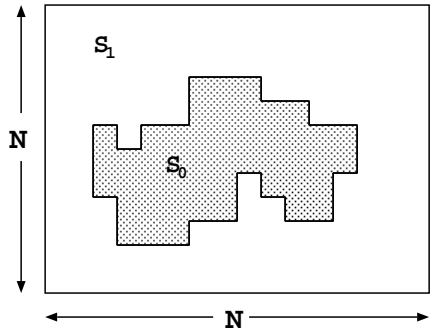

Figure 1: An admissible object bounded by two x-monotone curves.

Hence, we mainly study the associated optimization problem of computing an admissible segmentation in polynomial time, which is stated as follows:

> **Admissible image-segmentation problem:** Partition an intensity image into an admissible region S_0 and its complement S_1 so as to maximize the interclass variance $V(S_0, S_1)$.

Our results on the admissible image-segmentation problem are as follows.

First, we present a dynamic programming formulation of the admissible image-segmentation problem (Section 3). Although the dynamic programming approach in Section 3 is relatively simple, it does provide a starting point for the more sophisticated approaches in subsequent sections. In particular, the dynamic programming formulation enables us to solve in $O(n^2)$ time an easier case in which the boundary of the optimal admissible region S_0 uses only one x-monotone chain, with the other x-monotone chain of S_0 being completely on the grid boundary (Section 3.2). We have implemented this algorithm, and the experimental results in Section 8 show that our method can obtain good image segmentation. We also give an efficient parallel algorithm for this case. The parallel algorithm runs in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ EREW PRAM processors (Section 4), and is based on a divide-and-conquer strategy rather than the seemingly inherently sequential dynamic programming technique.

Second, we consider the case in which an optimal admissible region is obtained by using two x-monotone chains in the partition (Section 5). Our algorithm is based on a parametric method [12, 2], which is called (in computational geometry) *hand probing*, to compute the convex hull of an unknown point set by using $O(n)$ *touching oracles* [11]. We compute a touching oracle in $O(n)$ time by using a dynamic programming with the Monge property [1, 2]. The proposed algorithm thus runs in $O(n^2)$ time and $O(\sqrt{n})$ working space, and practically runs much faster.

Third, we give an efficient algorithm to compute an ϵ-approximate solution in $O(\epsilon^{-1} n \log L)$ time, where ϵ is any fixed constant with $1 > \epsilon > 0$, and L is the total sum of the absolute values of brightness levels of the pixels (Section 6). We also generalize our $O(\epsilon^{-1} n \log L)$ time approximation algorithm to the case in which a *weighted* interclass variance is used (Section 7).

2 Segmentation Problem

Let G be an $N \times N$ grid plane, i.e., $G = \{(i,j) \mid i = 1,2,\ldots,N,\ j = 1,2,\ldots,N\}$, and g_{ij} be the brightness level of a pixel (lattice point) (i,j) of a given image on G. Let n denote the number of pixels of G. Throughout the paper, we assume that we are concerned with all pixels in G. Thus $n = N \times N$. The output object is denoted by S_0, and $S_1 = G - S_0$ is the background. We are interested in image segmentation in which the object S_0 is separated from the background S_1 by one or two x-monotone chains so that a given objective function is optimized.

Let $\mu = \frac{1}{n}\sum_{(i,j)\in G} g_{ij}$ be the average brightness level over the entire image. Let n_i and μ_i $(i = 0, 1)$ be the cardinality of the set S_i and the average brightness level of S_i, respectively. Formally, $n_0 = |S_0|$, $n_1 = |S_1|$, $\mu_0 = \frac{1}{n_0}\sum_{(i,j)\in S_0} g_{ij}$, and $\mu_1 = \frac{1}{n_1}\sum_{(i,j)\in S_1} g_{ij}$.

The objective function that we use is

$$V(S_0, S_1) = n_0(\mu - \mu_0)^2 + n_1(\mu - \mu_1)^2,$$

which is called the *interclass variance* in *discriminant analysis* [15]. The interclass variance is proportional to the sum of squares of *standardized means* [20].

It is known [15] that the maximization of $V(S_0, S_1)$ is equivalent to the minimization of the *intraclass variance* $W(S_0, S_1)$ defined below, which is another typical objective function used in clustering (e.g., [17]):

$$W(S_0, S_1) = \sum_{(i,j)\in S_0}(g_{ij} - \mu_0)^2 + \sum_{(i,j)\in S_1}(g_{ij} - \mu_1)^2.$$

In computer vision, discriminant analysis has already been used to solve the problem of finding an optimal threshold by which an intensity image is transformed into a black/white one (i.e., each pixel has a value of 0 or 1). In the method due to Ohtsu [22], the histogram of intensity is partitioned into two classes based on discriminant analysis and good results are obtained for most cases. However, to the authors' knowledge, this paper gives the first attempt to apply discriminant analysis to image segmentation.

3 Dynamic Programming Approach

We give in this section relatively naive dynamic programming algorithms. These algorithms serve as starting points for the more sophisticated and efficient algorithms in later sections.

3.1 Rewriting the Objective Function

The interclass variance is invariant if we replace g_{ij} by $g_{ij} - \mu$ for all (i,j). Thus, we can assume without loss of generality (WLOG) that $\mu = 0$ and, accordingly, $\mu_1 = -\mu_0 n_0/n_1$.

If $\mu_0 < 0$ in the optimal solution, then we can define another image such that the brightness level of each pixel (i,j) in it is $-g_{ij}$. The image segmentation of this new image has exactly the same solution as the original problem, and $\mu_0 > 0$ holds for the new problem. Hence, from now on, we assume WLOG that $\mu_0 \geq \mu = 0 \geq \mu_1$. We define

$$U(S_0) = \sum_{(i,j)\in S_0} g_{ij} = n_0\mu_0.$$

Lemma 1 *Under our assumption that* $\mu = 0$, $V(S_0, S_1) = n^2(n_0 n_1)^{-1}(U(S_0))^2$.

Proof: Since $\mu = 0$ and $\mu_1 = -\mu_0 n_0/n_1$, $V(S_0, S_1) = n_0\mu_0^2 + n_1\mu_1^2 = (n_0^{-1} + n_1^{-1})(n_0\mu_0)^2$ $= n(n_0 n_1)^{-1}(n_0\mu_0)^2$. ∎

Because $U(S_0) \geq 0$ and n is a fixed number for the grid G, our problem can be written as

$$\text{maximize } D(S_0, S_1) \equiv (n_0 n_1)^{-1/2} U(S_0).$$

Let $P(k)$ denote the subproblem of maximizing the above objective function $D(S_0, S_1)$ under the constraint that n_0 is fixed to k. Since we have fixed $n_0 = k$ and n is given, the multiplicative term $(n_0 n_1)^{-1/2} = (k(n-k))^{-1/2}$ in $D(S_0, S_1)$ is ineffective. Thus, $P(k)$ can be formulated as:

> Find an admissible object S_0 maximizing $U(S_0)$ under the condition that $|S_0| = k$.
> We define $F(k)$ to be the maximum value of $U(S_0)$ with $|S_0| = k$.

Hence, the dynamic programming algorithms first compute $F(k)$ for all $k = 1, 2, \ldots, n$, then compute $\max_k (k(n-k))^{-1/2} F(k)$, and finally find the object S_0 attaining the maximum value of $D(S_0, S_1)$.

3.2 Optimal Separation with a Single Monotone Chain

As a warm-up, we first consider the segmentation by using one x-monotone chain into a connected region S_0 and its complement S_1. WLOG, we assume that S_0 is below the separating chain.

We define G_m to be the m-th column of the grid G, and $G_{\leq m} = \cup_{i \leq m} G_i$. Let $F(k, m)$ be the maximum of $U(S_0)$ under the conditions that $|S_0| = k$, $S_0 \subseteq G_{\leq m}$, and $S_0 \cap G_m \neq \emptyset$. Naturally, $F(k) = \max_{1 \leq m \leq N} F(k, m)$. Thus, it suffices to compute $F(k, m)$ for all k and m.

For each $m = 1, 2, \ldots, N$ and for any interval I of integers in $[1, N]$, we define $f_m(I) = \sum_{i \in I} g_{im}$. Then, we have the following recursion formula of $F(k, m)$.

$$F(k, m) = \max_{1 \leq t \leq \max\{N, k\}} \{F(k - t, m - 1) + f_m([1, t])\}$$

This formula naturally leads us to a simple dynamic programming algorithm. Since $m \leq N$, $t \leq N$, and $k \leq n$, the time complexity of the dynamic programming is $O(N^2 n)$, which is $O(n^2)$. Thus, we have the following lemma:

Lemma 2 *$O(n^2)$ time and $O(n)$ space are sufficient to compute* $\max_{m=1,\ldots,N} \{F(k, m)\}$ *for all* $k = 1, 2, \ldots, n$.

In this manner we can obtain the optimal value of the interclass variance. In order to obtain an actual partition optimizing the interclass variance, we could keep track of the dynamic programming table. Naively, this would use $O(n^2)$ working space, but we can reduce it to $O(n)$. We omit the details of the analysis of the space complexity, since we will show later another algorithm, in which the space complexity is further improved to $O(\sqrt{n})$. We have obtained the following theorem:

Theorem 1 *Given an image with n pixels, an optimal partition with one x-monotone chain can be computed in $O(n^2)$ time and $O(n)$ space.*

The above algorithm has been implemented, and the segmentation results on real image data were satisfactory.

3.3 Segmentation with Two Monotone Chains

We now consider the segmentation by using two x-monotone chains. We use the same notation as the previous subsection.

For $0 < k \leq n$, we define $F(k,t,m)$ to be the maximum of $U(S_0)$ under the conditions that $|S_0| = k$, $S_0 \subseteq G_{\leq m}$, S_0 contains the pixel (t,m) of G, and S_0 is admissible. We define $F(0,t,m) = 0$ for every (t,m).

Since $F(k) = \max_{m=1,\ldots,N}\{\max_{t=1,\ldots,N} F(k,t,m)\}$, it suffices to compute $F(k,t,m)$ for all k, t, and m. For $k > 0$, $F(k,t,m)$ can be computed using the following recursive formula:

$$F(k,t,m) = \max_{l}\{\max_{I \ni t,l}\{F(k-|I|,l,m-1) + f_m(I)\}\}$$

where the maximum is taken over all intervals I containing both t and l. Note that the above recursive formula ensures the constraints on admissible regions. This recursion leads us to a dynamic programming algorithm.

There are $O(N^3)$ possible choices of l and I for computing $F(k,t,m)$ for each triple of k, t, and m. Since the number of such triples of k, t, and m is $O(N^4)$, a naive implementation of this dynamic programming formulation takes $O(N^7) = O(n^{3.5})$ time. This is too expensive.

Instead of this, we will give an $O(n^2)$ time (and practically even faster) solution by using the ideas of *focused images* and *hand probing* in the next section.

4 Parallel Algorithms

Although the dynamic programming algorithms in previous sections give us several advantages, they seem to be quite difficult to parallelize. In this section, we present a divide-and-conquer scheme for finding the optimal partition of an image. This scheme leads to two sequential algorithms: (1) An $O(n^2)$-time and $O(n)$-space algorithm (called **Algorithm A**) for computing the optimal partition of an image by using an *admissible* sequence based on one $x-$monotone chain, (2) an $O(n^2)$-time and $O(n)$-space algorithm (called **Algorithm B**) for computing the optimal partition by using a *not necessarily admissible* sequence based on two $x-$monotone chains (that is, S_0 and S_1 may each consist of several disconnected regions). Based on the divide-and-conquer method, we then show how to obtain efficient parallel algorithms for these partitions that run in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors in the EREW PRAM model. A trade-off between the parallel time and space complexities is also demonstrated.

Given m consecutive columns of N pixels each, the divide-and-conquer scheme is outlined below: In the case of $m > 1$, divide the m columns into two groups of roughly the same number of consecutive columns, then recursively solve the two subproblems, and finally combine the solutions to the two subproblems, in $O(m^2N^2)$ time, to obtain the solution to the original

problem on the m columns; in the case of $m = 1$, compute in $O(N^2)$ time the maximum sum of brightness levels of k consecutive pixels in that column, for every $k = 1, 2, \ldots, N$. Hence the recurrence relation $T(m)$ for the time complexity of a procedure with the above outline is

$T(m) = 2T(m/2) + a * m^2N^2$,

$T(1) = b * N^2$

for some positive constants a and b. It is easy to see that $T(N) = O(N^4)$. We need to show how to achieve the claimed time bounds in the two cases of the above outline and show that $O(N^2)$ space is sufficient for the computation throughout.

Algorithm A can handle the case of $m = 1$ trivially in $O(N)$ time, since based on one $x-$monotone chain, the choice in a column for k consecutive pixels, for every $k = 1, 2, \ldots, N$, is unique. **Algorithm B** can easily handle the case of $m = 1$ in $O(N^2)$ because for every k, there are $N - k + 1$ choices of k consecutive pixels in a column defined by two $x-$monotone chains. The task of dealing with the case of $m > 1$ for **Algorithm B** is much easier than that for **Algorithm A**; this is because no constraint on connectivity is imposed when computing not necessarily admissible partition sequences. We therefore discuss only the computation of **Algorithm A** for the case of $m > 1$.

Algorithm A computes for the m columns nine arrays S_I, S_{EI}, S_{FI}, S_{IE}, S_{IF}, S_{EIE}, S_{EIF}, S_{FIE}, and S_{FIF}, called *sum arrays*, each corresponding to exactly one of the nine patterns discussed in Section 5. Each sum array is of size $m * N$ and contains, for every $k = 1, 2, \ldots, m * N$, the maximum sum of brightness levels of k pixels that is well-defined based on an admissible sequence of a particular pattern (when not well-defined, let the sum be $-\infty$). For example, the pattern of S_I is that each of the m columns contributes at least one but less than n pixels to every maximum sum (if the sum is well-defined), and $S_I(i)$ is not well-defined for every $i < m$. We also denote the sum of brightness levels of all the pixels in the m columns by $full$. Without loss of generality, we assume that m is an even number. The divide-and-conquer scheme divides the m columns into two groups, which we call the *left* and *right* groups. We denote the sum arrays and the total sum of brightness levels for the left (resp., right) group by, for example, S_I^L and $full^L$ (resp., S_I^R and $full^R$).

The problem in the "combine" stage of the divide-and-conquer scheme for **Algorithm A** then becomes that given the nine sum arrays for each of the left and right groups, compute the nine sum arrays for the m columns in (m^2N^2) time (the value $full$ is equal to $full^L + full^R$). Note that each sum array for the left or right group contains the maximum sums for k pixels, $k \leq (m * N)/2$. The following "array operations" are used to generate the nine sum arrays for the original m columns: Given an appropriate sum array A^L for the left group and a sum array A^R for the right group, of size $(m * N)/2$ each, compute the sum array A of size $m * N$, where

$A(k) = \max\{A^L(i) + A^R(j) \mid i + j = k, i \geq 0, j \geq 0$, and k is well-defined for the array $A\}$. We denote such an operation by A^LA^R, and the k-th element in the array A^LA^R by $A^LA^R(k)$. Then, the nine sum arrays for the original m columns are computed as follows:

$$
\begin{aligned}
S_I(k) &= S_I^L S_I^R(k) \\
S_{EI}(k) &= \max\{S_{EI}^L S_I^R(k), S_I^R(k), S_{EI}^R(k)\} \\
S_{FI}(k) &= \max\{S_{FI}^L S_I^R(k), full^L + S_I^R(k), full^L + S_{FI}^R(k)\} \\
S_{IE}(k) &= \max\{S_I^L S_{IE}^R(k), S_I^L(k), S_{IE}^L(k)\} \\
S_{IF}(k) &= \max\{S_{IF}^L(k) + full^R, S_I^L(k) + full^R, S_I^L S_{IF}^R(k)\} \\
S_{EIE}(k) &= \max\{S_{EI}^L S_{IE}^R(k), S_{EI}^L(k), S_{IE}^R(k)\} \\
S_{EIF}(k) &= \max\{S_{EI}^L S_{IF}^R(k), S_{EI}^L(k) + full^R, S_{EIF}^L(k) + full^R\} \\
S_{FIE}(k) &= \max\{S_{FI}^L S_{IE}^R(k), full^L + S_{IE}^R(k), full^L + S_{FIE}^R(k)\} \\
S_{FIF}(k) &= \max\{S_{FI}^L S_{IF}^R(k), S_{FI}^L(k) + full^R, full^L + S_{IF}^R(k), S_{FIF}^L(k) + full^R, full^L + S_{FIF}^R(k)\}
\end{aligned}
$$

It is clear that each of the nine sum arrays for the m columns can be computed in $O(m^2N^2)$ time; this is because for each such sum array, we compute its k-th element in $O(k)$ time, for every $k \leq m * N$. The space for each sum array is $O(m * N)$, and hence the total space used by the algorithm is also $O(m * N)$. Therefore, the time and space for computing the nine sum arrays for the N columns of the image plane are $O(N^4)$ and $O(N^2)$ respectively.

After obtaining the nine sum arrays for the N columns of the image plane, the value of the optimal partition $P(k)$, for every $k = 1, 2, \ldots, N^2$, is chosen from the nine sum arrays. Then we can report the actual optimal partition of the image plane by using the divide-and-conquer algorithm in Section 4, again in $O(N^4)$ time and $O(N^2)$ space.

A straightforward parallel implementation of the above divide-and-conquer algorithms would not give the claimed $O(\log^2 n)$ time and $O(n^2/\log^2 n)$ processor bounds. The reason is that if we used a parallel version of the algorithm in Section 4 to report the actual optimal partition, then the time of such a parallel algorithm would become $O(\log^3 n)$ (with $O(n)$ space and $O(n^2/\log^3 n)$ processors, by Brent's theorem [6]). An approach that avoids using the algorithm in Section 4 is to store all the sum arrays computed at every level of the recursion, together with some pointer information on how the sum arrays at that recursion level were obtained from those computed at the previous recursion level. Then by using these sum arrays and pointers, the actual optimal partition can be reported in $O(\log^2 n)$ time and $O(n^2/\log^2 n)$ processors by Brent's theorem [6]. The space required by this approach is $O(n \log n)$. Hence we have the following theorem.

Theorem 2 *An optimal partition with one x-monotone chain can be computed in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on an EREW PRAM.*

By using a combination of the two parallel approaches described above, we can achieve the following parallel bounds: $O(\log^{2.5} n)$ time, $O(n^2/\log^{2.5} n)$ processors, and $O(n)$ space. Let L be an integer (L will be chosen to be $O(\sqrt{\log n})$). Divide the $O(\log n)$ recursion levels of the algorithms into $O((\log n)/L)$ segments of L levels each. The parallel procedure for reporting the actual optimal partition repeats the following iterations $O((\log n)/L)$ times: For iteration i, recompute and store all the sum arrays for recursion level $i * L$ (with the top level being level 1). The additional amount of space so required is $O(n)$. We then apply a parallel version of the algorithm in Section 4 to compute the actual optimal partition, but now within each iteration, we only need to compute repeatedly the sum arrays at the levels j with $(i-1) * L \leq j < i * L$ (since all the sum arrays at recursion level $i * L$ are already available). Hence each iteration requires $O(L^2 \log n)$ time, and there are totally $O((\log n)/L)$ iterations. The

recomputation of the sum arrays at levels $i * L$, for every $i = 1, 2, \ldots, O((\log n)/L)$, takes altogether $O((\log^3 n)/L)$ time. Thus the best choice for L is $O(\sqrt{\log n})$, and the parallel bounds follow.

The correctness of all the algorithms discussed in this section can be easily proved by induction.

5 An Efficient Algorithm Using Focused Images

Recall that our objective is to maximize $D(S_0, S_1) = (|S_0||S_1|)^{-1/2}U(S_0)$, where $U(S_0)$ is the summation of intensity levels of pixels in the region S_0. Thus, if we have two different ways of partition (S_0, S_1) and (S_0', S_1') such that $|S_0| = |S_0'|$ (and thus $|S_1| = |S_1'|$) and $U(S_0) > U(S_0')$, then we can discard the partition (S_0', S_1'). This is the relationship between two different partitions in the direction perpendicular to the axis corresponding to the cardinality of the region. The same relationship holds for any direction. Let θ be any direction from the positive $x-$axis. Then, in the $|S_0| - U(S_0)$ plane, a maximal region S_0 in the direction perpendicular to the line of angle θ maximizes the quantity $U(S_0) - \theta|S_0|$. We call a region S_0 a *focused region* if it is a maximal region for some θ value.

This property comes from the concavity of the objective function. A partition defined by any connected region S_0 corresponds to a point characterized by $(|S_0|, U(S_0)$ in the $|S_0| - U(S_0)$ plane. Then, if we draw the curve of the form $U(S_0) = D\sqrt{|S_0|(n - |S_0|)}$ so that it passes through the points $(0, 0), (n, 0)$ and $(|S_0|, U(S_0))$, the D value gives us the value of the objective function associated with the partition $(S_0, S - S_0)$. Obviously, the larger the D value the higher such a curve associated with a partition. Because of the concavity of the curve, the higher the curve the higher the tangent line in any fixed angle. This implies that we can find a maximal region by maximizing the constant factor of the tangent line, which is given by $U(S_0) - \theta|S_0|$, instead by maximizing the D value. This is an approach called parametric optimization.

Since this is a linear combination of two values $U(S_0)$ and $|S_0|$, all we have to do to find an optimal partition is to enumerate all maximal regions, which correspond to construction of the convex hull of a point set defined by pairs of regions and their corresponding U values.

We compute the convex hull of the point set without most knowledge of the coordinate values of the points. Specifically, we use the *hand probing* method [12, 11], which computes a convex polygon based on *touching oracles*. We give a segmentation criterion for which the optimal solution (called the *focused image*) can be computed in $O(n)$ time. This focused image computation is used as a touching oracle of the hand probing, and is based on a dynamic programming with the Monge property [1, 2].

5.1 Hand Probing Algorithm

Let $\mathcal{P}$ be the upper chain of the convex hull of an unknown point set P in the plane. We consider computing $\mathcal{P}$ by using the following *touching oracle*:

> Given a slope θ, report the tangent line with slope θ to $\mathcal{P}$, together with the tangent point (see Figure 2).

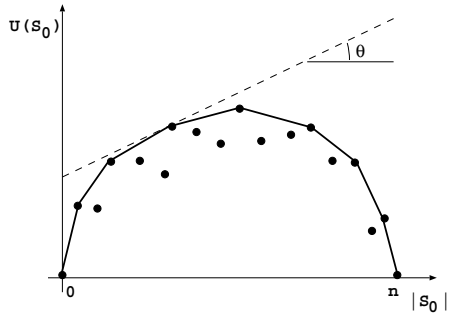

Figure 2: Hand probe.

The above computation model was introduced by Eisner and Severence [12] for solving network optimization problems. It was extended to higher dimensional cases by Dobkin, Edelsbrunner, and Yap [11] in applications to robotics, and was named *hand probing*.

Lemma 3 ([12]) *A convex polygonal chain with K vertices can be computed with $O(K)$ touching oracles.*

Proof: For the sake of completeness, we briefly outline the proof of the lemma. We start with slopes $+\infty$ and $-\infty$, and find the two endpoints of the convex chain. Suppose we have computed a pair of chain vertices u and v such that no vertex of the convex chain between them has been computed so far. Then, we compute the slope θ_{uv} of the line through u and v, and perform a touching oracle with respect to the slope θ_{uv}. In consequence, we either find a new chain vertex w between u and v or know u and v must be adjacent to each other on the convex chain. Thus, we can find either a new vertex or a new edge of the convex chain by performing a touching oracle. ∎

5.2 Focused Images and Convex Hull

For a given real number θ, we define the following cost function

$$U_\theta(S_0) = U(S_0) - \theta|S_0| = (\sum_{(i,j)\in S_0} g_{ij}) - \theta|S_0| = \sum_{(i,j)\in S_0} (g_{ij} - \theta)$$

Note that $U(S_0)$ defined in the previous section coincides with $U_0(S_0)$. We then consider the following problem:

$Q(\theta)$: Compute an admissible object S_0 which maximizes $U_\theta(S_0)$.

We call an admissible object S_0 a *focused image* if it is a solution of $Q(\theta)$ for some θ. The solution of $Q(\theta)$ is called a focused image associated with θ. The following is our key lemma:

Lemma 4 *$Q(\theta)$ can be solved in $O(n)$ time using $O(N) = O(\sqrt{n})$ working space.*

Proof: The proof is given in Section 4.3. ∎

Recall that $F(k)$ was defined to be the maximum value of $U_0(S_0) = \sum_{(i,j)\in S_0} g_{ij}$ under the condition that $|S_0| = k$. Let $S(k)$ be the associated admissible object. We define $F_\theta(k) = F(k) - \theta k$. Then, the following lemma follows immediately from definitions:

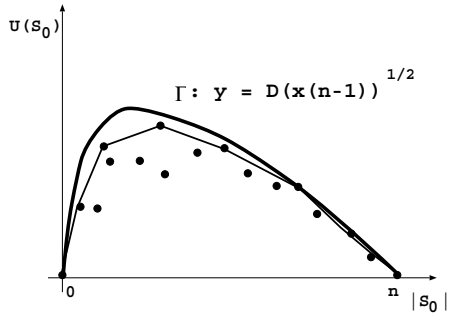

Figure 3: Convex hull and interclass variance.

Lemma 5 $\max_{S_0} U_\theta(S_0) = \max_k F_\theta(k)$.

We define a point set $P = \{(k, F(k)) \mid k = 0, 1, \ldots, n\}$ in the plane. Let $\mathcal{P}$ be the upper chain of the convex hull of P.

Lemma 6 *There exists a tangent line to $\mathcal{P}$ at the point $(j, F(j))$ with a slope θ if and only if $S(j)$ is a focused image associated with θ. Consequently, the point $(j, F(j))$ is a vertex of $\mathcal{P}$ if and only if $S(j)$ is a focused image.*

Proof: Assume that $y = \theta x + b$ is a line tangent to $\mathcal{P}$ at the point $(j, F(j))$. Then it is easy to see that $F_\theta(k) = F(k) - \theta k$ takes the maximum value at $k = j$. Thus $S(j)$ is a focused image associated with θ.

Conversely, if $S(j)$ is a focused image associated with θ, then all points of P lie below or on the line $y - \theta x = F_\theta(j)$. Thus, this line touches $\mathcal{P}$ at the point $(j, F(j))$. ∎

We show a relation between focused images and an optimal object with respect to $D(S_0, S_1)$.

Lemma 7 *The admissible object S_0^* maximizing $D(S_0, S_1)$ is a focused image.*

Proof: Suppose the maximum value D_{opt} of $D(S_0, S_1)$ is taken on S_0^*, and $|S_0^*| = \nu$. We consider the (x, y)-plane in which $P = \{(j, F(j)) \mid j = 1, 2, \ldots, n-1\}$ are plotted. We consider in this plane the curve Γ: $y = D_{opt}(x(n-x))^{1/2}$. Since $D(S_0, S_1)(|S_0|(n - |S_0|))^{1/2} = U_0(S_0)$, the point $(\nu, F(\nu))$ lies on the curve Γ, and all other points $(j, F(j))$ lie below (or on) Γ due to the maximality of D_{opt} (Figure 3).

Since the second derivative of $\sqrt{x(n-x)}$ is negative for $0 < x < n$, the curve Γ is concave in x. Hence, all points $(j, F(j))$ lie below (or on) the tangent line L_1 to Γ at $(\nu, F(\nu))$. Thus, $(\nu, F(\nu))$ is a vertex of $\mathcal{P}$, and S_0^* is a focused image. ∎

Theorem 3 *The image segmentation problem can be solved in $O(n^2)$ time and $O(\sqrt{n})$ working space.*

Proof: It suffices to compute the vertex of $\mathcal{P}$ that maximizes the interclass variance. The time complexity follows straightforwardly from Lemmas 3, 4, and 7.

Although it requires only $O(\sqrt{n})$ working space to compute a focused image, if we naively compute the convex chain $\mathcal{P}$, we need to keep all $O(n)$ vertices of $\mathcal{P}$. In order to avoid keeping many vertices of $\mathcal{P}$, we perform the convex chain computation by using a "process the smallest

interval first" policy, which is explained below. Suppose we have computed a new vertex $w = (s, F(s))$ between a pair of convex chain vertices $u = (i, F(i))$ and $v = (j, F(j))$, and $|s - i| \leq |s - j|$. Then, we first process the convex chain between u and w by finding the vertex in the chain-interval $[u, w]$ maximizing the interclass variance. Next, we process the interval $[w, v]$. In this way, we only need to keep $O(\log n)$ vertices of the convex chain. Thus, the space complexity remains $O(\sqrt{n})$.

Next, we shall show that the working space associated with the dynamic programming algorithm for processing touching oracles or solving a problem $Q(\theta)$ is $O(N) = O(\sqrt{n})$. It is a simple observation that since our dynamic programming to solve $Q(\theta)$ proceeds column by column the space just for a couple of columns suffices for the forward computation to find a maximal value of $U(S_0) - \theta|S_0|$. The succeeding backward computation using only $O(N)$ space to construct a region achieving the maximum value is not trivial. As is stated before, the dynamic programming computes the maximum value of $U(S_0) - \theta|S_0|$ in $O(N^2) = O(n)$ time and $O(N)$ space. However, it is not the case to construct a region that achieves the maximum value in the same computational complexity. If we store a backward pointer at each element of the dynanmic programming table, $O(N^2)$ time suffices. However, if we restrict the space to $O(N)$, we need to sacrifice the time complexity. In fact, a divide-and-conquer approach works. First of all, we implement the dynamic programming to find the leftmost and rightmost pixels of an optimal region. The one pass of the dynamic programming establishes the connection between the leftmost and rightmost columns. Then, the current interval defined by the leftmost and rightmost columns is partitioned into two equal-length interval. The next pass of the dynamic programming establishes the connections between the end columns of the intervals and outputs the pixel positions at the current columns on the optimal paths. Thereafter the same process is iterated until the boundary of the optimal region is reported at every column. The number of iterations is $O(\log n)$. Since each pass is implemented in $O(n)$ time, the overall time needed is $O(n \log n)$. Note that the backward computation is implemented only once although we need to solve the problem $Q(\theta)$ many times and so the time $O(n \log n)$ is not dominant. ∎

Remark. The number K of vertices of $\mathcal{P}$ is practically much smaller than n, for which case the algorithm runs in $O(nK)$ time. Experimentally, K is less than 100 in typical images on a 100×100 grid.

5.3 Proof of Lemma 4

First we shall show the time bound. We define $\tilde{g}_{ij} = g_{ij} - \theta$. We consider the m-th column G_m of the grid G. For a closed interval $I = [u, v]$ of row indices, we define $\tilde{f}_m(I) = \sum_{s \in I} \tilde{g}_{sm}$.

We compute and store both $\tilde{f}_m([1, v])$ and $\tilde{f}_m([u, N])$ for all $1 \leq v, u \leq N$ in $O(N)$ time using $O(N)$ space. Since $\tilde{f}_m([u, v]) = \tilde{f}_m([1, N]) - \tilde{f}_m([1, u-1]) - \tilde{f}_m([v+1, N])$ for $1 < u \leq v < N$, we can query $\tilde{f}_m(I)$ in $O(1)$ time for each interval $I = [u, v]$.

Let $s(i) = \max_{u \leq i} \tilde{f}_m([u, i])$ and let $t(j) = \max_{v \geq j} \tilde{f}_m([j, v])$. Computing $s(i)$ (resp., $t(i)$) is equivalent to computing the positions of the maximum entry in the i-th column (resp., i-th row) of the upper triangle matrix $\mathcal{F}_m = (\tilde{f}_m([u, v]))_{1 \leq u \leq v \leq N}$.

The matrix $\mathcal{F}_m$ is a *Monge* matrix, that is, for $i < i' < j' < j$, $\tilde{f}_m([i, j]) + \tilde{f}_m([i', j']) \geq$

$\tilde{f}_m([i,j']) + \tilde{f}_m([i',j])$. (Moreover, equality holds, and hence the matrix $\mathcal{F}_m$ is a *strong Monge* matrix.) We compute and store $s(i)$ and $t(i)$ for all $i = 1, 2, \ldots, N$ in $O(N)$ time and space by using a fast Matrix Searching algorithm on a Monge matrix [1].

For each pair (i,j) of row indices, we define $cover_m(i,j) = \max_{J \ni i,j} \tilde{f}_m(J)$. Then, it is easy to see that $cover_m(i,j) = \tilde{f}_m([s(i),t(j)])$ if $i \leq j$. Thus, we can query $cover_m(i,j)$ for each pair (i,j) in $O(1)$ time.

We define $U_\theta(m,i)$ as the maximum value of $U_\theta(S')$ among all admissible objects $S' \subseteq G_{\leq m}$ that contain the pixel (i,m). We set $U_\theta(0,i) = 0$ for all i. Obviously, $\max_{S_0} U_\theta(S_0)$ $= \max_{0 \leq m \leq N} \{\max_{1 \leq i \leq N} U_\theta(m,i)\}$.

Below, we compute $U_\theta(m,i)$ for all $0 \leq m \leq N$ and $1 \leq i \leq N$ in $O(N^2)$ time. From the definition of admissible objects, the following recursion formula holds:

$$U_\theta(m,i) = \max_{1 \leq j \leq N} \{\max\{0, U_\theta(m-1,j)\} + cover_m(i,j)\}$$

We compute $U_\theta(m,i)$ for all i by using the above formula. It is not difficult to see that the matrix $(cover_m(i,j))_{1 \leq i \leq j \leq N}$ is a strong Monge matrix. Accordingly, the matrix $\mathcal{D}_m$ defined by $\mathcal{D}_m(i,j) = U_\theta(m-1,j) + cover_m(i,j)$ is also a strong Monge matrix. (To say precisely, its upper triangular part is a strong Monge matrix, and also its lower triangular part is the transposition of a strong Monge matrix.) The computation of the above formula for a fixed m is equivalent to computing all row maxima of these two matrices, which can be done with $O(N)$ queries of $cover_m(i,j)$ by using a fast Matrix Searching algorithm [1].

Therefore, the computation of $\max_{S_0} U_\theta(S_0)$ altogether needs $O(N^2) = O(n)$ queries each of which takes $O(1)$ time, and hence the total time complexity is $O(n)$. The space complexity is obviously $O(N) = O(\sqrt{n})$.

5.3.1 One-chain case

Next, we consider the case where the (connected) region is separated by an x-monotone chain. We denote $S_0(\theta)$ for the solution of the touching oracle for a slope θ. Without loss of generality, we assume that $S_0(\theta)$ is a region below an x-monotone chain. Hence, the m-th column of $S_0(\theta)$ is either empty or the half-column below a row.

We define $L(m,j,\theta) = \sum_{i=1}^{j} (g_{im} - \theta)$. Moreover, for each column index m, we define the row index $j(m,\theta)$ such that $L(m,j,\theta)$ is maximized at $j = j(m,\theta)$ (we give a symbolic perturbation so that $j(m)$ is unique). We denote $Y(m,\theta)$ for $L(m,j(m,\theta),\theta)$.

Then, the m-th column of $S_0(\theta)$ is either empty or the half colum $C(m,\theta)$, which is the part below $j(m,\theta)$-th row (including the pixel $(j(m,\theta),m)$). Indeed, if we compute the subinterval I of $[1,N]$ maximizing $\sum_{m \in I} Y(m,\theta)$, $S_0(\theta) = \cup_{m \in I} C(m,\theta)$. Once we know $Y(m,\theta)$ for $m = 1, 2, .., N$, it is known that I can be computed in $O(N)$ time [4].

Naively, it needs $O(N)$ time to compute $Y(m,\theta)$ for each m, hence the total time complexity becomes $O(N^2)$. However, we can give preprocessing so that we can compute $S_0(\theta)$ for each θ in $O(N)$ time.

We consider $L(m,j,\theta)$ as a linear function on θ, whose slope is j. Then, $Y(m,x) = \max_{1 \leq j \leq N} L(m,j,x)$ is a convex function. We can compute this convex function in $O(N)$ time using computational geometric algorithms, since it can be considered a dual problem to construction of convex hull of points in a plane with integral x-coordinate values.

Once we have this convex function, we can retrieve the value of $Y(m,\theta)$ for given θ in $O(\log N)$ time using binary search.

Moreover, we can apply similar list searching technique [9, 7] so that we can retrieve the values $Y(1,\theta), Y(2,\theta), ..., Y(N,\theta)$ in $O(N)$ time.

Hence, if we give $O(N^2) = O(n)$ preprocessing time, we can give a touching oracle in $O(N) = O(\sqrt{n})$ time for the single-chain case. The data structure needs $O(N)$ space.

6 Faster ϵ-Approximation Algorithm

Although we have given an $O(n^2)$ time algorithm, there may still be situations in which $O(n^2)$ time is too expensive. We propose an efficient ϵ-approximation algorithm which computes an admissible object S_0 such that $D(S_0, G - S_0) \geq (1-\epsilon)D_{opt}$ (for any fixed ϵ, with $0 < \epsilon < 1$) in $O(\epsilon^{-1} n \log L)$ time. We assume that all brightness levels are integers, and L is the total sum of the absolute values of the brightness levels. The idea of this algorithm is similar to the one given in [19], and is based on the following lemma.

Lemma 8 *Let θ^* denote the optimal parameter value with which an optimal solution S_0^* of $Q(\theta^*)$ maximizes $D(S_0, S_1)$. If $\theta^* \neq 0$, then an optimal solution of $Q((1+\epsilon)\theta^*)$ produces an ϵ-approximate solution to the problem of maximizing $D(S_0, S_1)$.*

Proof:

Let $\nu = |S_0^*|$ and $D_{opt} = D(S_0^*, G - S_0^*)$. Then, $F(\nu) = D_{opt}\sqrt{\nu(n-\nu)}$, and $(\nu, F(\nu))$ is on the curve $C_0 : y = D_{opt}\sqrt{x(n-x)}$. The line L_1

$$L_1 : \quad y = \theta^*(x - \nu) + F(\nu)$$

is the tangent line of the curve C_0 at $(\nu, F(\nu))$, with $\theta^* = (n-2\nu)D_{opt}/2\sqrt{\nu(n-\nu)}$.

As shown in the proof of Lemma 7, S_0^* maximizes U_{θ^*}. Now consider the admissible object S_0^ϵ that maximizes $U_{(1+\epsilon)\theta^*}$. We claim that S_0^ϵ is an ϵ-approximate solution to the problem of maximizing $D(S_0, S_1)$.

In order to prove this, we consider the lines in the (x,y)-plane whose slopes are $(1+\epsilon)\theta^*$. Among all points $(x, F(x))$, $x = 1, 2, \ldots, n-1$, consider the one that maximizes $-(1+\epsilon)\theta^* x + F(x)$ (i.e., the one corresponding to an optimal solution of $Q((1+\epsilon)\theta^*)$). Let this point be $(\nu', F(\nu'))$ (see Figure 4).

We shall show that $D' = F(\nu')/\sqrt{\nu'(n-\nu')}$ is at least $(1-\epsilon)D_{opt}$ (which proves our claim). Due to the maximality of $(\nu', F(\nu'))$, the point $(\nu, F(\nu))$ is below or on the line

$$L_2 : \quad y = (1+\epsilon)\theta^*(x - \nu') + F(\nu').$$

Hence, the point $(\nu', F(\nu'))$ is above (or on) the line

$$L_3 : \quad y = (1+\epsilon)\theta^*(x - \nu) + F(\nu).$$

On the other hand, the point $(\nu, F(\nu))$ is above the curve $C_1 : y = D'\sqrt{x(n-x)}$, since $F(\nu) = D_{opt}\sqrt{\nu(n-\nu)} \geq D'\sqrt{\nu(n-\nu)}$.

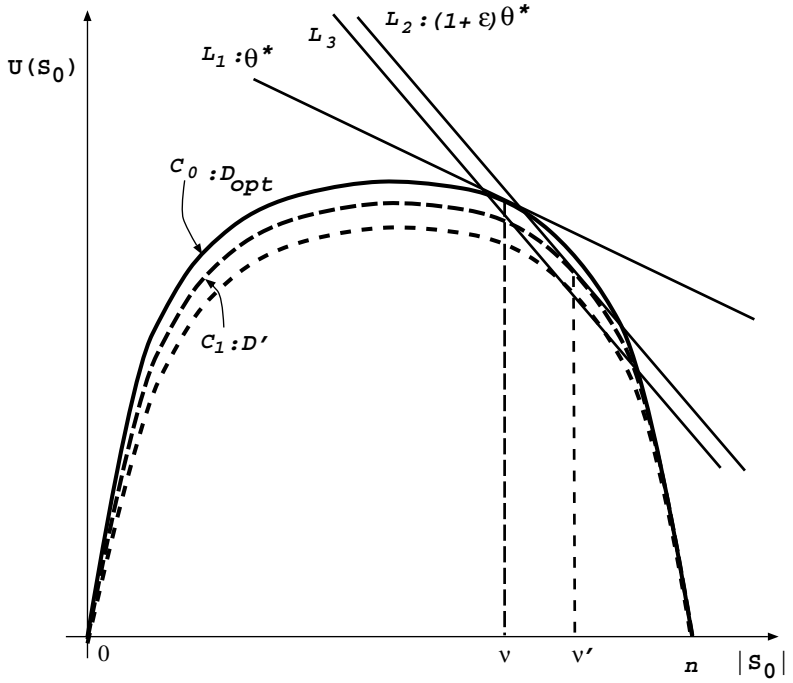

Figure 4: Definition of three lines and three curves used in the proof of Lemma 8.

The point $(\nu', F(\nu'))$ on C_1 is above L_3, and the point $(\nu, F(\nu))$ on L_3 is above C_1; hence the curve C_1 must intersect L_3. At an intersection point of C_1 and L_3, $y/\sqrt{x(n-x)} = D'$. Thus, the minimum value r_{min} of $y/D_{opt}\sqrt{x(n-x)}$ on the line L_3 is at most D'/D_{opt}. We shall show that r_{min} is at least $1-\epsilon$. By definition,

$$r_{min} = \min\{\frac{1}{\sqrt{x(n-x)}}\{\frac{(1+\epsilon)(n-2\nu)}{2\sqrt{\nu(n-\nu)}}(x-\nu) + \sqrt{\nu(n-\nu)}\} \mid 0 < x < n\}.$$

The term to be minimized in the above equation is rewritten into

$$\frac{1}{2\sqrt{\nu(n-\nu)}}(\frac{x(n-2\nu)(1+\epsilon)}{\sqrt{x(n-x)}} + \frac{2\epsilon\nu^2 + n\nu(1-\epsilon)}{\sqrt{x(n-x)}}) = \frac{1}{2\sqrt{\nu(n-\nu)}}(at + bt^{-1}).$$

Here, $a = (n-\nu)(n+\epsilon(n-2\nu))/n$ and $b = \nu(n-\epsilon(n-2\nu))/n$ are constants, and $t = \sqrt{x/(n-x)}$.

The minimum of $at + bt - 1$ is attained at $t = \sqrt{b/a}$. Thus, r_{min} is

$$\frac{2\sqrt{ab}}{2\sqrt{\nu(n-\nu)}} = \frac{\sqrt{n^2 - \epsilon^2(n-2\nu)^2}}{n} \geq \sqrt{1-\epsilon^2} \geq 1-\epsilon.$$

This proves the lemma. ∎

Theorem 4 *An ϵ-approximate solution can be computed in $O(\epsilon^{-1} n \log L)$ time.*

Proof: From the assumption of the integrality of g_{ij}, $|\theta^*|$ satisfies $1/2 \leq |\theta^*| \leq L$, provided that $\theta^* \neq 0$. We define a sequence $\{\theta_l, -\theta_l\}$ of parameters θ by $\theta_l = (1+\epsilon)^{l-1}/2$ for $0 \leq l \leq \lfloor \log L / \log(1+\epsilon) \rfloor$. The approximation algorithm solves the parametric problem $Q(\theta_l)$ for all values of the above defined parameters and chooses the one that maximizes $D(S_0, S_1)$. It is clear from Lemma 8 that such a solution is an ϵ-approximation of our segmentation problem. The number of parameters so generated is $O(\log L / \log(1+\epsilon)) = O(\epsilon^{-1} \log L)$. Hence, the running time of the algorithm is $O(\epsilon^{-1} n \log L)$. ∎

6.1 Other Approximate Solutions

If the size of the output object is approximately given, then we can solve the approximate segmentation problem faster.

Lemma 9 *Given an integer k_1 in $[1, N]$, we can compute the focused image whose size is nearest to k_1 in $O(n \log L)$ time.*

Proof: The binary search for θ gives an $O(n \log L)$ time solution straightforwardly. ∎

Corollary 1 *Suppose we are given a subinterval $I = [k_1, k_2]$ of $[1, N]$, such that there exists a focused image whose size is in I. Then, we can compute the admissible object maximizing the interclass variance among all the objects whose sizes are in I in $O(n(k_2 - k_1 + \log L))$ time.*

7 Generalization to Weighted Interclass Variance

Our techniques are also applicable to a maximization problem with a function $U_c(S_0)/p(|S_0|)$ for any constant c and any concave function $p(x)$ in the interval $0 < x < n$. For example, we consider the problem of maximizing the weighted interclass variance $V^\alpha(S_0, S_1) = n_0^\alpha(\mu - \mu_0)^2 + n_1^\alpha(\mu - \mu_1)^2$.

Theorem 5 *If $0 \leq \alpha \leq 2$, then an admissible object S_0 maximizing $V^\alpha(S_0, S_1)$ can be computed in $O(n^2)$ time. Its ϵ-approximate solution can be computed in $O(\epsilon^{-1} n \log L)$ time.*

Proof: We assume WLOG that $\mu_0 \geq \mu_1$ and $\mu = 0$. Accordingly, $\mu_1 = -n_0\mu_0/n_1$. Then, $V^\alpha(S_0, S_1) = (n_0^{\alpha-2} + (n-n_0)^{\alpha-2})(U_0(S_0))^2$, and $\sqrt{V^\alpha(S_0, S_1)} = (\sqrt{n_0^{\alpha-2} + (n-n_0)^{\alpha-2}})U_0(S_0)$. It is not difficult to verify that $1/\sqrt{x^{\alpha-2} + (n-x)^{\alpha-2}}$ is concave if $0 \leq \alpha \leq 2$. ∎

8 Experimental Results

We have implemented our algorithms in cooperation with Dainippon Screen MFG, Kyoto , Japan, which manufactures intelligent printing machines. Figure 4 shows one of the examples, where an original input image is laid above and the resulting segmented image is below. The program accepts color images. To apply our algorithms we need to convert color informations into monocrome intensity information. Our method for this purpose is just to add intensity levels in three colors (red, green, and blue). Of course, this may be too simple, but it seems to

be the best way from our experience of experiments. The results are quite satisfactory. Since an entire image is decomposed into a sequence of small subimages which should be partitioned into two parts, x-monotonicity is not too restricted from a practical point of view. We plan to implement the algorithm for the segmentation using two monotone chains in the near future. The focused image computation algorithm of Lemma 4 has already been implemented [13], and its linear-time performance has been confirmed.

9 Discussions

In this paper, we have studied the problem of detecting an optimal connected object region in an intensity image based on discriminant analysis. We have shown that this problem in general is NP-hard, and presented polynomial-time algorithms for the basic case of separating an optimal region with two x-monotone chains. To the authors' knowledge, this is the first computational-geometric attempt to the image segmentation problem.

An issue that needs to be pointed out is that the monotonicity of object boundaries is a constraint that may be oversimplified to some practical applications. However, if we imagine a small floating window over an image in which optimal monotone boundaries of objects are computed, we may obtain a new image-filtering or edge-detection scheme with confidence weighted by inter-cluster distance.

Acknowledgement

The authors would like to thank Shigeaki Shimazu of Dainippon Screen MFG, Kyoto, Japan, for his efforts on the experiments and valuable suggestions from a practical point of view.

References

[1] A. Aggarwal, M.M. Klawe, S. Moran, P.W. Shor, and R. Wilber: "Geometric applications of a matrix-searching algorithm," *Algorithmica,* Vol.2, pp.195-208, 1987.

[2] A. Aggarwal, B. Schieber, and T. Tokuyama: "Finding a minimum-weight k-link path in graphs with the concave Monge property and applications," *Discrete & Comput. Geom.*, Vol.12, pp.263-280, 1994.

[3] T. Asano and N. Yokoya: "Image segmentation schema for low-level computer vision," *Pattern Recognition*, Vol.14, nos.1-6, pp.267-173, 1981.

[4] J. L. Bentley, "Programming Pearls – Algorithm Design Techniques," *CACM* 27-9 (1984), 865-871.

[5] P.J. Besl and R.C. Jain: "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern and Machine Intell.*, Vol.10, pp.167-192, 1988.

[6] R.P. Brent: "The Parallel Evaluation of General Arithmetic Expressions," *J. of the ACM*, Vol.21, pp.201-206, 1974,

[7] B. Chazelle and L. Guibas, "Fractional Cascading: I. A Data Structure Technique," Algorithmica 1 (1986), 133-162

[8] F. Cheevasuvit, H. Maitre, and D. Vidal-Madjar: "A robust method for picture segmentation based on split-and-merge procedure," *Comput. Vis., Graph. Image Process.,* Vol.34, pp.268-281, 1986.

[9] R. Cole, "Searching and Storing Similar Lists," J. of Algorithms 7 (1986), 202-220.

[10] L.S. Davis: "A survey of edge detection techniques," *Comput. Graph. Image Process.,* Vol.4, pp.248-270, 1975.

[11] D. Dobkin, H. Edelsbrunner, and C. K. Yap: "Probing convex polytopes," *Proc. 18th ACM STOC*, pp.387-392, 1986.

[12] M. J. Eisner and D. G. Severence: "Mathematical techniques for efficient record segmentation in large shared databases," *J. ACM*, 23, pp.619-635, 1976.

[13] T. Fukuda, S. Morishita, Y. Morimoto, and T. Tokuyama: "Data Mining Using Two-Dimensional Optimized Association Rules – Scheme, Algorithms, and Visualization," Proc. SIGMOD Int. Conf. on Management of Data, pp.13-23, 1996.

[14] M. R. Garey and D. S. Johnson: "The rectilinear Steiner tree problem is NP complete," *SIAM J. Appl. Math.*, **32**, pp.826-834, 1977.

[15] D.J. Hand, *Discrimination and Classification,* John Wiley & Sons, 1981.

[16] S. L. Horowitz and T. Pavlidis: "Picture segmentation by a directed split-and-merge procedure," *Proc. 2nd Int. Joint Conf. Pattern Recognition,* pp.424-433, 1974.

[17] M. Inaba, N. Katoh, and H. Imai: "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering," *Proc. 12th Symp. on Computational Geometry,* pp.332-339, 1994.

[18] M. Kass, A. Witkin, and D. Terzopoulos: "Snakes: Active contour models," *Int. J. of Comput. Vision,* Vol.1, pp.321-331, 1988.

[19] N. Katoh and T. Ibaraki: "A parametric characterization and an ϵ-approximation scheme for the minimization of a quasiconcave program," *Discrete Applied Mathematics*, Vol.17 (1987), pp.39–66.

[20] L. Maisel, *Probability, Statistics, and Random Processes*, Simon & Schuster, NY, (1971).

[21] F. Meyer and S. Buecher: "Morphological segmentation," *J. Vis. Commun. Image Represent.,* Vol.1, pp.21-46, 1990.

[22] N. Ohtsu: "Discriminant and least squares threshold selection," *Proc. 4th IJCPR*, pp.592-596, 1978.

[23] T. Pavlidis and Y.-T. Liow: "Integrating region growing and edge detection," *IEEE Trans. Pattern and Machine Intell.*, Vol.12, pp.225-233, 1990.