Computing rank-convolutions with a mask

László Babai and Pedro Felzenszwalb
University of Chicago
Email:{laci, pff}@cs.uchicago.edu

Abstract

Rank-convolutions have important applications in a variety of areas such as signal processing and computer vision. We define a "mask" as a function taking only values zero and infinity. Rank-convolutions with masks are of special interest to image processing.

We show how to compute the rank-k convolution of a function over an interval of length n with an arbitrary mask of length m in $O(n\sqrt{m}\log m)$ time. The result generalizes to the d-dimensional case. Previously no algorithm performing significantly better than the brute force $O(nm)$ bound was known.

Our algorithm seems to perform well in practice. We describe an implementation, illustrating its application to a problem in image processing. Already on relatively small images, our experiments show a signficant speedup compared to brute force.

1 Introduction

1.1 Motivation

Min-convolution and more generally, rank-convolutions (a. k. a. rank filters) have important applications in a variety of areas, including signal processing, pattern recognition, computer vision, and mathematical programming. For example, in computer vision they have been used for object recognition, depth estimation, and image restoration. Rank-convolutions such as the median-convolution have the advantage of being more robust against certain types of noise.

Fast algorithms for computing ordinary convolutions are at the heart of a wide range of applications. We expect that efficient algorithms for computing the min-convolution and more generally, rank-convolutions, would also have an impact in a variety of areas.

Unfortunately, however, no significantly subquadratic algorithm appears to be known to compute the min-convolution. In addition to its theoretical interest, this question has practical significance because in many signal processing applications the size of the input is on the order of many thousands or millions.

Our main result is an efficient algorithm for computing rank-convolutions of a function with a *mask* (a function that takes values 0 and ∞ only); we save essentially a $\sqrt{m}$ factor compared to brute force, where m is the size of the mask. The result is based on a new reduction of rank-k convolution to ordinary convolutions of Boolean functions.

This appears to be the first algorithm to obtain a substantial improvement over the brute force method for this class of functions, even in the special case of min-convolution. An efficient method for computing the min-convolution of an arbitrary function with a function that has a small range follows immediately.

Rank-convolutions with masks are interesting in the context of image processing, because they generalize classical non-linear filters and are not affected by low-level noise. Previous image processing systems have mainly used small masks. Our result opens up the possibility of building efficient systems that use larger masks.

A prototype implementation shows that our algorithm can easily be implemented using standard tools and seems to perform well in a realistic setting.

Below we give a somewhat detailed introduction to the min-convolution and rank-convolution problems, their applications, and history, as many readers may not be familiar with these problems and their significance.

1.2 Definitions

Let G be an additive abelian group. The case of importance for us will be $G = \mathbb{Z}^d$ under componentwise addition ($d \geq 1$); in fact the case $d = 1$ already illustrates the idea behind our algorithm and the resulting speedup. On the other hand, both the definitions and our algorithm become more transparent when stated on the level of a general abelian group G.

The sum of subsets $A, B \subseteq G$ is defined as $A + B = \{a + b : a \in A, b \in B\}$.

Let $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$.

Definition 1.1. *For a function $f : G \to \overline{\mathbb{R}}$ we define the* ∞-support $\operatorname{supp}_\infty(f)$ *as the set*

$$\operatorname{supp}_\infty(f) = \{a \in G : f(a) < \infty\}; \tag{1}$$

and the 0-support $\operatorname{supp}_0(f)$ *as the set*

$$\operatorname{supp}_0(f) = \{a \in G : f(a) \neq 0\}. \tag{2}$$

We shall use the 0-support in the context of the ordinary convolution and the ∞-support in the context of rank-convolutions (including the min-convolution). To facilitate parallel treatment we shall often refer to these two concepts as the "support" and use the notation $\operatorname{supp}(f)$. This will not lead to ambiguity, the context being either specified or implied in each case.

In each context, we shall only be interested in functions with a finite support. If S is a finite subset of G and $f : S \to \overline{\mathbb{R}}$ is a function, we shall view f as a function $f : G \to \overline{\mathbb{R}}$ by extending f to $G \setminus S$ so that $\operatorname{supp}(f)$ remains a subset of S (i. e., for $a \in G \setminus S$ we set $f(a) = \infty$ in the context of rank-convolutions and $f(a) = 0$ in the context of the ordinary convolution).

Recall that the *ordinary convolution* of functions $f, g : G \to \mathbb{R}$ with finite support is the function $h = f * g : G \to \mathbb{R}$ defined by

$$h(i) = \sum_{j \in G} f(j) g(i - j). \tag{3}$$

Observe that the sum (3) has only a finite number of nonzero terms and is therefore well defined.

Replacing addition by min, multiplication by addition, and 0 by ∞ in (3), we obtain the definition of min-convolution:

Definition 1.2. *Let $f, g : G \to \overline{\mathbb{R}}$ be functions with finite support. The* min-convolution *of f and g is the function $h = f \otimes_{\min} g : G \to \overline{\mathbb{R}}$ defined by*

$$h(i) = \min_{j \in G} (f(j) + g(i - j)). \tag{4}$$

In other words, min-convolution is the convolution operation in the $(\min, +)$ semiring of functions $G \to \overline{\mathbb{R}}$. Observe that in (4), the minimum is taken over a finite number of values and is therefore well defined.

Like the ordinary convolution, min-convolution is commutative and associative. Moreover, in further analogy with ordinary convolution which is distributive over addition, min-convolution is distributive over "min," i. e., if $g_1, \dots g_k : G \to \overline{\mathbb{R}}$, then

$$f \otimes_{\min} \min\{g_1, \dots, g_k\} = \min\{f \otimes_{\min} g_1, \dots, f \otimes_{\min} g_k\}, \tag{5}$$

and vertical shift (addition of a constant) has the expected effect, i. e., for any constant $c \in \overline{\mathbb{R}}$,

$$f \otimes_{\min} (g + c) = (f \otimes_{\min} g) + c. \tag{6}$$

An important generalization of min-convolution concerns order statistic. For a function $f : G \to \overline{\mathbb{R}}$ with finite ∞-support S of size $m = |S|$, for $k \geq 1$ we define the rank-k order-statistic of f as $R_k(f) := f(b_k)$ where the support $S = \{b_1, \dots, b_m\}$ is sorted by the f-values: $f(b_1) \leq f(b_2) \leq \dots \leq f(b_m)$. For $k > m$ we stipulate $R_k(f) = \infty$.

Definition 1.3. *Let $f, g : G \to \overline{\mathbb{R}}$ be functions with finite support and let $k \geq 1$. The* rank-k convolution *of f and g is the function $h = f \otimes_k g : G \to \overline{\mathbb{R}}$ defined by*

$$h(i) = R_k\{f(j) + g(i - j) : j \in G\}. \tag{7}$$

Note that $\otimes_{\min} = \otimes_1$.

Observe that for every $i \in G$, the function $r_i(j) := f(j) + g(i - j)$ $(j \in G)$ has finite support and therefore the quantity $h(i)$ in (7) is well defined.

Rank-k convolution is commutative and satisfies (6), but for $k \geq 2$ it is not associative and does not satisfy any distributive law analogous to (5).

A *mask* is a function with finite ∞-support that takes the values 0 and ∞ only. We shall use masks in the context of rank-convolutions only. The term "mask" comes from the way such functions are used in image processing. A mask is uniquely determined by its ∞-support and therefore we can identify masks with finite subsets of G. The effect of rank-k convolution with a mask is that for every $i \in G$, we replace $f(i)$ by the rank-k order statistic of the values within a shifted copy of the mirror reflection of the mask "centered" at i.

Let $\circ$ denote either of the $*$ and $\otimes_k$ operations. We conclude this section with the observation that for functions f, g with finite supports we have

$$\operatorname{supp}(f \circ g) \subseteq \operatorname{supp}(f) + \operatorname{supp}(g). \tag{8}$$

1.3 The main results

Let $G = \mathbb{Z}^d$; we refer to d as the *dimension* of the convolution problems over G. We use the notation $[n] = \{0, 1, \dots, n - 1\}$. We shall assume the supports of our functions f, g satisfy

$$\operatorname{supp}(f) \subseteq [n]^d; \qquad \operatorname{supp}(g) \subseteq [m]^d. \tag{9}$$

It follows by (8) that

$$\operatorname{supp}(f \circ g) \subseteq [n + m - 1]^d, \tag{10}$$

where, as before, $\circ$ denotes either of our convolution operations.

The input functions f, g and the output function $f \circ g$ will be given as d-dimensional arrays of sizes n^d, m^d, and $(n+m-1)^d$, respectively (recall our convention that we regard a function defined on a finite subset of G as a function defined on all of G by extending it so as not to change its support). Throughout the paper we assume both m and n are powers of 2.

Our computational model is a RAM capable of performing arithmetic operations on numbers $\le (n+m)^d$ (the addresses to the output array) and comparisons of real numbers (the data in the f array) at unit cost.

One-dimensional rank-convolutions can be computed by brute force in $O(nm)$ operations (in the RAM model with real arithmetic), using linear time median selection [3, 19]. One of our main goals is to understand if (or when) the computation of min-convolution and rank-convolutions can be done in substantially subquadratic time ($O(n^{2-c})$ for some constant $c > 0$) for the case $m = n$.

Our main result is a new upper bound on the complexity of computing the rank-k convolution of any function $f : [n]^d \to \overline{\mathbb{R}}$ with a mask $g : [m]^d \to \{0, \infty\}$.

Theorem 1.4. *(a) Let f be a function $[n]^d \to \overline{\mathbb{R}}$ and g a mask $[m]^d \to \{0,\infty\}$ where $m \le n$. Then the rank-k convolution $f \otimes_k g$ can be computed in $O\left(8^d n^d m^{d/2}\sqrt{d\log m}\right)$ time, with $O(dn^d \log n)$ comparisons of reals. (b) If $m = n$, we obtain the time bound $O\left(2^d n^{3d/2}\sqrt{d\log n}\right)$.*

Note that for bounded d, our time bound simplifies to

$$O\left(n^d m^{d/2}\sqrt{\log m}\right), \tag{11}$$

saving a factor of $m^{d/2}/\sqrt{\log m}$ compared to the brute force bound $O\left((nm)^d\right)$.

Our algorithm works via a generic reduction to ordinary convolution of binary arrays, which in turn can be performed via the Fast Fourier Transform (FFT) [8].

Even in dimension $d = 1$ and in the special case of min-convolution, this appears to be the first result that beats the brute force bound $O(nm)$ by more than a logarithmic factor for a significant class of pairs of functions with no convexity assumptions.

For the case of min-convolutions, the main result extends to the case when g is not a mask but its range is small.

Corollary 1.5. *Let f be a function $[n]^d \to \overline{\mathbb{R}}$ and g a function $[m]^d \to \overline{\mathbb{R}}$ where $m \le n$. Assume g takes s distinct values. Then $f \otimes_{\min} g$ can be computed in $O\left(s8^d n^d m^{d/2}\sqrt{d\log m}\right)$ time.*

In Section 4 we state our main result in the general context of rank-convolutions over abelian groups (Theorem 4.1).

2 Prior work

In this account, we assume the dimension is $d = 1$. This is the case that received the most attention; and the improvement produced by our algorithm in one dimension is essentially the same as in higher dimensions, saving a factor of $\widetilde{\Theta}(\sqrt{K})$ where K is the size of the mask.

For simplicity, we shall also assume $m = n$. In this case the brute force computation of the one-dimensional min-convolution takes $O(n^2)$ operations (in the RAM model with real arithmetic). We discuss previous attempts at improving this quadratic behavior.

2.1 Early work: the slope transform

Bellman and Karush [2] introduced a transform (the “maximum transform”) which is an extension of the Legendre transform to not necessarily convex functions. Bellman and Karush showed how this transform can be used to compute the min-convolution of a pair of convex functions. Maragos [17] rediscovered the Bellman–Karush transform and called it the *slope transform*; this term has gained acceptance in the mathematical morphology community. We note that the concept was defined for continuous signals and some ad-hoc discretization was used for its approximate calculation. Maragos saw that the slope transform plays a similar role for morphological operators as the Fourier transform does for linear operators.

In analogy with the convolution theorem for the Fourier transform, the slope transform of $f \otimes_{\min} g$ is the sum of the slope transforms of f and g. While the slope transforms of sampled functions $f, g : [n] \to \overline{\mathbb{R}}$ can be sensibly defined, and computed in linear time, this does not lead to a fast min-convolution algorithm because the slope transform is not invertible. The slope transform of h is identical to the slope transform of the lower hull of h (the maximal convex function dominated by h). The slope transform can be used to compute $f \otimes_{\min} g$ in $O(n)$ time when both f and g are convex, in which case the output is also convex.

2.2 Complexity of the general case

It seems that the best existing bound for the complexity of computing the min-convolution of two arbitrary functions is $O(n^2/\log n)$. This result follows from a technique used in Chan's $O(n^3/\log n)$ algorithm for the all-pairs shortest path problem [6]. Bremner et al. [4] describe the method and also present an $O(n^2(\log\log n)^2/\log n)$ algorithm for computing arbitrary rank-convolutions. This appears to be the best bound to-date. Bremner et al. [4] also describe faster algorithms in the non-uniform linear decision tree model.

Bussieck et al. [5] considered the average-case complexity of min-convolution and developed an algorithm that runs in expected $O(n \log n)$ time for random inputs where every permutation of the values of f and g is equally likely to occur. Unfortunately this is not a reasonable assumption to make in most applications.

2.3 Convexity assumptions

Under convexity assumptions, faster algorithms are known for min-convolution, but not for the potentially more important case of rank-convolutions.

The only special case we are aware of where a fast algorithm is known for rank-convolutions is the case when the mask is an axis-parallel box. This case is common in computer vision and image processing. Rank-convolutions in this case can be computed in $O(n \log n)$ time; and min-convolution in $O(n)$ (Gil and Werman [15]).

Computing min-convolutions when f is arbitrary but g is convex has received special attention because of its applications to sequence alignment [9] and to computing distance transforms of images [10]. As pointed out by Eppstein [9], the problem can be solved in $O(n)$ time by using the totally monotone matrix search algorithm of Aggarwal et al. [1].

When g is concave, the min-convolution can be computed in $O(n\alpha(n))$ time using the matrix search algorithm of Klawe and Kleitman [16]. Here α is the extremely slowly growing inverse Ackermann function. Eppstein [9] combined the convex and the concave cases and generalized them to an algorithm that computes the min-convolution in $O(nk\alpha(n/k))$ time when g can be decomposed into a sequence of k convex or concave segments.

Felzenszwalb and Huttenlocher [10] developed a different algorithm for the case when g is convex by noting the relationship between min-convolutions and lower-envelopes (minimum of a family of functions). The same approach can be used when g is concave. This algorithm runs in $O(n)$ time if an intersection point between shifted copies of g can be computed in constant time. In the worst case, binary search will find an intersection in $O(\log n)$ time, yielding an $O(n \log n)$ algorithm.

3 Applications

3.1 Applications of min-convolution

Bellman and Karush considered min-convolution in the context of optimization problems in economics and operations research such as optimal distribution of effort and allocation processes [2]. Min-convolution plays an important role in signal processing because it is closely related to the dilation and erosion of real valued signals [20, 17]. Recently one and two-dimensional min-convolutions have been crucial in developing a variety of fast algorithms in computer vision [10, 12, 11, 7] and sequential data analysis [13]. The operation arises naturally in the solution of sequence alignment problems [9]. Sequence alignment is widely used in computational biology to find similarities between DNA/RNA/protein sequences, as well as for time-warping in speech/sound recognition and other pattern matching situations [18].

3.2 Significance of masks

Masks appear in many applications. In signal processing, the min-convolution of a discrete signal f with a mask corresponds to a type of min-filtering. In this case g represents a binary mask M that is "on" at indices where g equals zero and "off" at indices where g equals infinity. Now the min-convolution corresponds to placing a mirror reflected version of M on top of f at all horizontal shifts and finding the minimum entry in f that is under an "on" position of M.

The min-filtering operation can be seen as the erosion of f by a binary structuring element. This is one of the basic operations in *mathematical morphology* where it is used to analyze images; in particular, erosions are commonly used to detect structures (such as lines, "blob" shapes like cells under a microscope, wood grain, etc.) in grayscale images [20, 21].

In image analysis the masks used for min-filtering are usually convex (balls, squares), but the masks are sometimes tuned to the shape of a particular object. We believe that an $O(N \log N)$ algorithm for the general two-dimensional min-filtering problem (where $N = n^2$ is the size of the image) would inspire a host of new applications; even the more modest progress made in this paper might help expand the application horizon. In the case of images, N tends to be so large that a quadratic algorithm is not practical.

While min-convolutions with masks are quite common in image processing, rank-convolutions seem to be more important because they can filter out noise.

3.3 Applications of rank convolution

Rank-convolutions are important because they are more robust than min-convolutions. The min operation is fragile because a single low value in a set of values makes the minimum low, even if most of the remaining values are high. In pattern recognition we often have noisy inputs and we do not want low-level noise to have a drastic effect on the output.

Rank-convolutions have been used for a variety of image processing tasks [21]. Images often suffer from what is called "salt-and-pepper" artifacts, where a few pixels get arbitrary values due to noise in the sensing. A classical approach for removing such noise is to compute a median-filter, where the value of a pixel is replaced by the median value in a small region around it.

In Section 5 we describe an application of rank-convolutions to object detection, where a large mask is used to detect translated copies of an object with a particular shape in an image that contains salt-and-pepper noise.

4 Algorithms

4.1 Notation, terminology

A *Boolean function* is a function which takes only two values, 1 ("TRUE") and 0 ("FALSE"). A *log–Boolean function* (or a "mask") is a function which takes only two values, 0 ("TRUE") and ∞ ("FALSE"). Note that f is Boolean if and only if $-\log(f)$ is log-Boolean.

If f is Boolean or log-Boolean we define the *support* of f, $\mathrm{supp}(f)$, as $f^{-1}(\mathrm{TRUE})$. This terminology is consistent with our previous usage since Boolean functions will only occur in the context of ordinary convolutions and log-Boolean functions only in the context of rank-convolutions.

For $S \subseteq G$, the *characteristic function* X_S is the Boolean function $G \to \{0,1\}$ with support S; and the *log-characteristic function* $L_S = -\log X_S$ is the analogously defined log-Boolean function.

4.2 Main algorithmic result

In this section we reduce the problem of computing the rank-k convolution of an arbitrary function and a log-Boolean function to ordinary convolution of Boolean functions. (All functions are assumed to have finite support.)

Our input will be two functions represented by arrays over finite subsets $A, B \subseteq G$. The output will be represented by an array over $A + B$. Our computational model is a RAM capable of performing arithmetic operations on integers up to $|A + B|$ and comparisons of pairs of real numbers at unit cost.

Assume we have an algorithm $\mathcal{A}$ to compute the ordinary convolution of Boolean functions with finite supports over G. For finite subsets $A, B \subset G$, let $T(A, B)$ denote the maximum cost incurred by $\mathcal{A}$ in computing $u * v$ where $u : A \to \{0,1\}$ and $v : B \to \{0,1\}$ are Boolean. Note that $T(A, B) \geq |A + B|$ since $|A + B|$ is the size of the output.

Theorem 4.1. *Let $k \geq 1$. Let A, B be finite subsets of the abelian group G. Given a function $f : A \to \overline{\mathbb{R}}$ and a log-Boolean function $g : B \to \overline{\mathbb{R}}$ as arrays over their respective domains, the rank-k convolution $f \otimes_k g$ can be computed, for any integer $q > 0$, in $O\left(qT(A,B) + (1/q)|A+B||A|\right)$ time, with $O(|A|\log|A|)$ comparisons of reals.*

In particular, if $T(A, B) = O(|A + B| \log |A + B|)$ then the total cost of the computation is $O(|A + B|\sqrt{|A| \log |A + B|})$.

The assumption $T(A, B) = O(|A+B| \log |A+B|)$ is justified when $G = \mathbb{Z}^d$ and $A = B = [n]^d$; in this case we can use the d-dimensional FFT [8]. Using FFT, ordinary one-dimensional convolution can be computed in $O(n \log n)$ operations over complex numbers. The "row-column algorithm," deriving its name from the two-dimensional case, yields d-dimensional FFT and ordinary convolution using $O((2n)^d \log(n^d))$ arithmetic operations over complex numbers [8].

Our calculation of ordinary convolutions needs to be accurate up to an additive error of 1/3, which we follow by rounding to the nearest integer. This overall accuracy can be achieved by performing arithmetic over complex numbers to $O(\log |A + B|)$-digit accuracy. Arithmetic operations of this accuracy cost $O(1)$ in our model.

The $n = m$ case of our main result, part (b) of Theorem 1.4, follows, noting that in that result we have $|A| = |B| = n^d$ and $|A + B| = (2n - 1)^d$. □

4.3 The main algorithm

We now describe the algorithm that will prove Theorem 4.1.
Let q be a positive integer parameter which we will set later around $\sqrt{|A|}$.

Phase 0. Preprocessing f

1. Sort A by the f-values: $A = \{a_1, \dots, a_s\}$ where $f(a_1) \leq f(a_2) \leq \cdots \leq f(a_s)$ and $s = |A|$.
2. Divide up $\{a_1, \dots, a_s\}$ into q consecutive subintervals $A_1, \dots, A_q$ such that $|A_t| \leq \lceil s/q \rceil$.
   (: Note that if $t_1 < t_2$ then for every $x_i \in A_{t_i}$ $(i = 1, 2)$ we have $f(x_1) \leq f(x_2)$. :)
3. Let $e_t : G \to \{0, 1\}$ be the characteristic function of A_t.
   (: Note that $\operatorname{supp}(e_t) \subseteq \operatorname{supp}(f) \subseteq A$. :)

Phase 1. Counting via ordinary convolution of Boolean functions

For $t = 1, \dots, q$, let $u_t = e_t * \mathrm{e}^{-g}$.
(: Note that both e_t and e^{-g} are Boolean functions; $\operatorname{supp}(\mathrm{e}^{-g}) = \operatorname{supp}(g) \subseteq B$. :)

Phase 2. Final selection

For $i \in A + B$,

if $k > \sum_{t=1}^{q} u_t(i)$ then let $h(i) = \infty$
else let

$$\ell_i = \min\{\ell : \sum_{t=1}^{\ell} u_t(i) \geq k\} \quad \text{and} \quad k_i = k - \sum_{t=1}^{\ell_i - 1} u_t(i); \tag{12}$$

$$h(i) = R_{k_i}\{f(j) + g(i - j) : j \in A_{\ell_i}\}. \tag{13}$$

This concludes the description of the algorithm. The key observation to justify correctness is that ordinary convolution can be used for counting; specifically, $u_t(i)$ counts the entries $j \in A_t$ which contribute a finite value $f(j) + g(i - j)$ in the definition of $h(i)$ (equation (7)).

4.4 Timing analysis

The preprocessing phase takes $O(|A| \log |A|)$ time with $O(|A| \log |A|)$ comparisons of reals. The time is negligible compared to our final estimate; and no more comparisons of reals will be made.

In Phase 1 we compute q ordinary convolutions of Boolean functions over A and B. This takes at most $qT(A, B)$ time.

In Phase 2, the computation of $h(i)$ for each $i \in A + B$ takes $O(q + |A|/q)$ time. In computing $h(i)$ we can consider the values $j \in A_{\ell_i}$ in sorted order (it has already been sorted in Phase 0) until we find k_i values with $g(i - j) < \infty$.

The total cost is as stated in the Theorem (using that $T(A, B) \geq |A + B|$). Under the stated assumption on $T(A, B)$ we choose $q = \lceil \sqrt{|A| \log |A + B|} \rceil$ to obtain the more specific bound. □

4.5 Min-convolution

We can go somewhat further for $k = 1$ (min-convolution) in view of equations (5) and (6). In this case, we do not need to assume that g is log-Boolean; it suffices to assume that it has small range. We obtain the following result.

Proposition 4.2. *Let A, B be finite subsets of G. Assume we can compute the min-convolution of a function $f : A \to \overline{\mathbb{R}}$ with any log-Boolean function (mask) $g : B \to \{0, \infty\}$ in $O(t(A, B))$ time. Then we can compute the min-convolution of f with any $g : B \to \overline{\mathbb{R}}$ in $O(|R(g)|t(A, B))$ time, where $R(g)$ denotes the range of g.*

The result follows from equations (5) and (6) and the following observation.

Observation 4.3. *Assume the number of distinct finite values in the range of $g : G \to \overline{\mathbb{R}}$ is s. Then g is the lower-envelope (minimum) of vertical shifts of s log-Boolean functions.*

Proof. For $r \in \mathbb{R}$, let L_r denote the log-characteristic function of the set $g^{-1}(r)$. Let R be the set of finite values in the range of g. Then $g = \min_{r \in R}\{L_r + r\}$. $\square$

All the sets $g^{-1}(r)$ can be found in $O(|B|)$ time and the decomposition $g = \min_{r \in R}\{L_r + r\}$ can be described explicitly in $O(s|B|)$ time. Therefore, by using identities (5) and (6), it suffices to compute each min-convolution $f \otimes_{\min} L_r$ in $O(t(A, B))$ time. $\square$

4.6 Small masks

In this section we complete the proof of our main result (Theorem 1.4) by considering the case $m < n$. We reduce this case to the case $m = n$ (part (b) of Theorem 1.4) by a standard tiling argument.

The main idea is to tile the output array, and compute h through many smaller rank-k convolutions. Each of these rank-k convolutions is between functions defined over similar sized domains. This reduction makes no assumptions about the values of g and how the smaller rank-k convolutions are computed. The following lemma describes the reduction.

Lemma 4.4. *Let $f : [n]^d \to \overline{\mathbb{R}}$ and $g : [m]^d \to \overline{\mathbb{R}}$ be functions with $m < n$. Suppose we can compute $f' \otimes_k g$ in $O(t(m, d))$ time for $f' : [2m-1]^d \to \overline{\mathbb{R}}$. Then we can compute $f \otimes_k g$ in $O\left(((n/m) + 2)^d t(m, d)\right)$ time.*

Proof. As before, we extend f and g to $\mathbb{Z}^d$ by assigning them the value ∞ outside their original domains. Let $h = f \otimes_k g$; so $\operatorname{supp}(h) \subseteq [n + m - 1]^d$. For $t \in \mathbb{Z}^d$, let f_t denote the restriction of f to the translate $[2m-1]^d + mt + \underline{1}$ of $[2m-1]^d$, where $\underline{1} = (1, \ldots, 1) \in \mathbb{Z}^d$. (Let $f_t(j) = \infty$ for $j \notin [2m-1]^d + mt + \underline{1}$.)

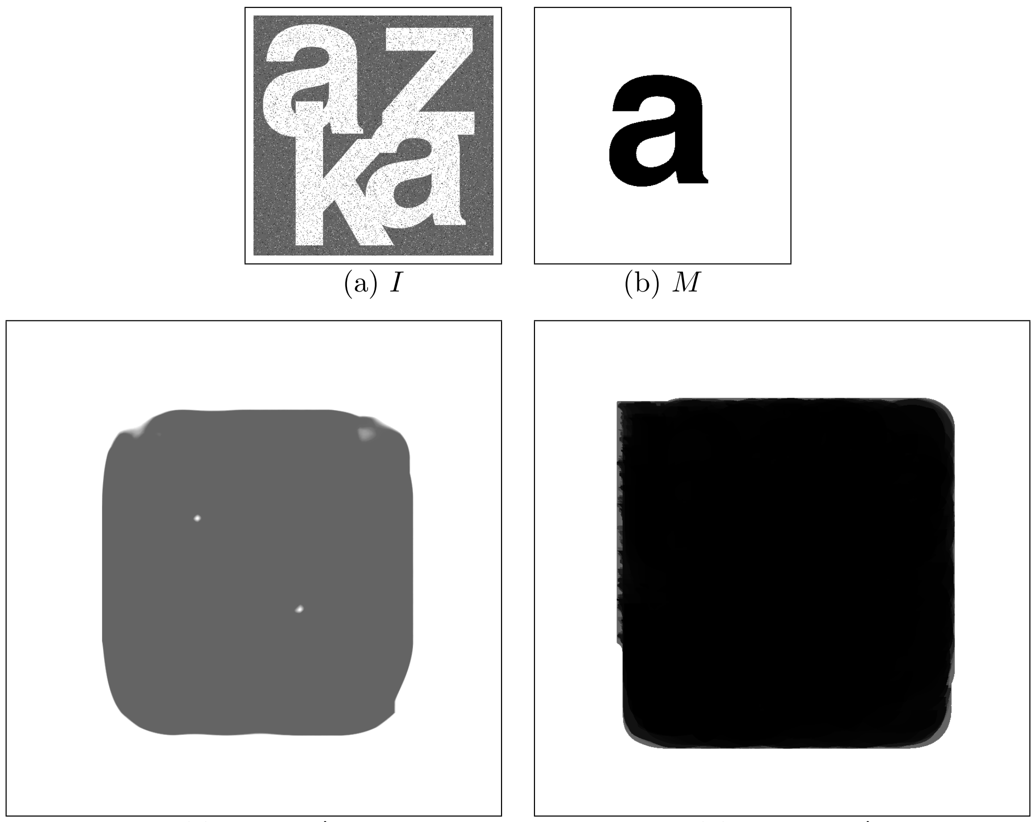
Let $h_t = f_t \otimes_k g$. For $r = (r_1, \ldots, r_d) \in \mathbb{R}^d$ let $\lfloor r \rfloor = (\lfloor r_1 \rfloor, \ldots, \lfloor r_d \rfloor)$.

We claim that for every $i \in \mathbb{Z}^d$,

$$h(i) = h_t(i) \quad \text{where} \quad t = \lfloor i/m \rfloor - \underline{1}. \tag{14}$$

Indeed, the only $j \in \mathbb{Z}^d$ for which the value $f(j)$ matters when evaluating $h(i) = R_k\{f(j) + g(i-j) : j \in G\}$ are those for which $i - j \in [m]^d$, i. e., $j \in i - [m]^d$ (otherwise $g(i-j) = \infty$). So we need to show that $i - [m]^d \subseteq [2m-1]^d + mt + \underline{1}$, i. e., $i - m\lfloor i/m \rfloor + (m-1)\underline{1} - [m]^d \subseteq [2m-1]^d$. Now we note that $i - m\lfloor i/m \rfloor \in [m]^d$ and $(m-1)\underline{1} - [m]^d = [m]^d$, so the claim follows from the fact that $[m]^d + [m]^d = [2m-1]^d$.

Now if $i \in [n + m - 1]^d$ then $\lfloor i/m \rfloor \in [\lfloor (n-2)/m \rfloor + 2]^d$ and therefore the number of relevant values of t is less than $((n/m) + 2)^d$, completing the proof. $\square$


(a) I (b) M

(c) $I \otimes_k M'$ (d) $I \otimes_{\min} M'$

Figure 1: The rank-2500 convolution of (a) and a mirror reflection of (b) is shown in (c). The image in (b) represents a mask, with black indicating a value of zero and white indicating a value of infinity. In the other images bright pixels represent high values, while darker pixels represent lower values. The two peaks in (c) indicate good shifts for the mask, corresponding to locations where the target object may be present. The min-convolution of (a) and (b) is shown in (d). In this case the result is meaningless due to the noise in (a).

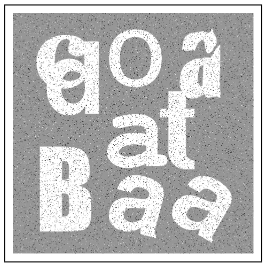
When $m < n/2$ we can combine this lemma with the case $m = n$ (part (b) of Theorem 1.4) and observe that $((n/m) + 2)^d < (2n/m)^d$ to obtain part (a) of the Theorem. The case $m \geq n/2$ follows from part (b) by simply extending g to $[n]^d$. □

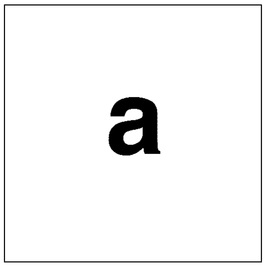
For min-convolutions with functions with small range we can use Theorem 1.4 and Proposition 4.2 to obtain Corollary 1.5.

5 Image processing application and implementation

5.1 The application

Figure 1 illustrates an application of our algorithm for locating translated copies of an object in an image. Here we have an image I with target objects that are brighter than the background. This image is corrupted with salt-and-pepper noise (each pixel is changed to a random value with probability 0.1). We also have a mask M that has approximately the same shape as a target object (it fits inside the object). Good locations for the object show up as peaks in the rank-k convolution $I \otimes_k M'$, where M' is a mirror reflection of M.


(a) J



(b) N

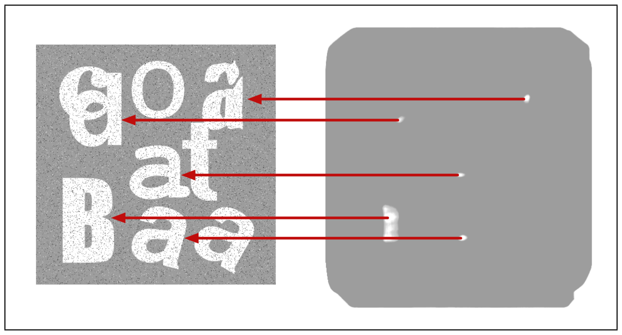

(c) $J \otimes_k N'$

Figure 2: The rank-600 convolution of (a) and a mirror reflection of (b) is shown in (c). Each peak in the rank-convolution gives a location in J with a bright object that approximately contains N. The arrows point to the objects "found" by each peak. Note the two rotated copies of (b) in the image, one by 15° (detected) and one by 30° (missed). Note also the "false positives:" the mask almost fits in the union of '**c**' and '**d**,' and the large '**B**' under several shifts.

This example demonstrates how rank-convolutions filter out salt-and-pepper noise. Figure 1 also shows the result of the min-convolution, $I \otimes_{\min} M'$. In this case the target locations do not show up in the output due to the noise in I.

The image I has 500×500 pixels and the support of the mask M fits inside this domain. The ideal value for k in the rank-convolution depends on the amount of noise in the input image, and the size of the support of the mask. In this example we used $k = 2500$ because the mask M has approximately 25000 "on" pixels. This allows for up to 10% of the pixels in the image to have arbitrary values, without affecting the result of the computation by much.

Figure 2 shows another example. Here the image J contains several letters from different fonts, and the mask N is the letter '**a**' from a particular font. As in the previous example, 10% of the pixels in J are corrupted by salt-and-pepper noise. The mask N has approximately 3500 "on" pixels and we used $k = 600$. This example illustrates how rank-convolutions are not only robust against noise; they can also be used to find objects that approximately contain a particular shape (the mask). Note that J contains two rotated copies of the letter '**a**.' The rank-600 convolution was able to detect an '**a**' rotated by 15° but not an '**a**' rotated by 30°. Note that 12 repetitions of the algorithm could detect all rotations of the mask (we need to rotate the mask by 30 degrees each time). The output also has false detections because the mask almost fits inside the union of the '**c**' and the '**d**,' and the '**B**' makes a large area of J bright. Larger values of k would allow the detection of objects that are less similar to the mask but would also lead to more false positives.

5.2 The implementation

We implemented our algorithm for rank-convolutions in C, using the FFTW library for computing FFTs [14]. Our implementation stipulates $m = n$, thus it does not take advantage of the fact that the mask can be defined over a smaller domain.

For comparison, we also implemented the $O(n^2K)$ brute force algorithm, where $K \leq n^2$ is the size of the support of the mask. So this implementation takes advantage of the smaller size of the masks. If the size of the smallest square containing the support of the mask is $s \times s$ then $K \leq s^2$. For the examples shown here, K is much less than n^2 (10% in Fig. 1 and 1.4% in Fig. 2).

We measured the running times on a 2.8GHz PowerMac computer with 8GB of RAM running the Mac OS X 10.5 operating system. The programs were compiled using gcc 4.0.1 (the GNU C compiler). In the images in Figure 1 we have $n = 500$, $d = 2$, and $K \approx 25000$. Our algorithm runs in about 8 seconds, while the brute-force implementation takes approximately 180 seconds. Thus we observe a more than 20-fold speedup for this relatively small image, even though our implementation favored the brute-force method in the sense described above.

This speedup is related to the size of the mask. In Figure 2 the image has the same size as before but the mask has smaller support ($K \approx 3500$). In this case the timing of our algorithm barely changes (7 seconds) while the brute-force implementation runs faster in proportion to the size of the support of the mask (25 seconds).

References

[1] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber, *Geometric Applications of a Matrix Searching Algorithm*, Algorithmica **2** (1987), no. 1, 195–208.

[2] R. Bellman and W. Karush, *Mathematical programming and the maximum transform*, J. Soc. Indust. and Appl. Math. **10** (1962), no. 3, 550–567.

[3] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. E. Tarjan, *Time bounds for selection*, J. Comp. Sys. Sci. **7** (1972), no. 4, 448–461.

[4] D. Bremner, T. Chan, E. Demaine, J. Erickson, F. Hurtado, J. Iacono, S. Langerman, and P. Taslakian, *Necklaces, convolutions and X+Y*, 14th European Symp. on Algorithms, LNCS, vol. 4168, Springer, 2006, pp. 160–171.

[5] M. Bussieck, H. Hassler, G. J. Woeginger, and U. T. Zimmermann, *Fast algorithms for the maximum convolution problem*, Operations Research Letters **15** (1994), no. 3, 133–141.

[6] T. M. Chan, *All-Pairs Shortest Paths with Real Weights in $O(n^3/\log n)$ Time*, 9th Workshop on Algorithms and Data Structures, LNCS, vol. 3608, Springer, 2005, pp. 318–324.

[7] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, *Spatial Priors for Part-Based Recognition Using Statistical Models*, IEEE CVPR 2005, vol. 1, IEEE Comp. Soc., 2005, pp. 10–17.

[8] P. Duhamel and M. Vetterli, *Fast Fourier transforms: A tutorial review and a state of the art*, Signal Processing **19** (1990), no. 4, 259–299.

[9] D. Eppstein, *Efficient algorithms for sequence analysis with concave and convex gap costs*, Ph.D. thesis, Columbia University, 1989.

[10] P. F. Felzenszwalb and D. P. Huttenlocher, *Distance Transforms of Sampled Functions*, Tech. Report 2004-1963, Cornell University Computing and Information Science, September 2004.

[11] ———, *Pictorial Structures for Object Recognition*, Int. J. of Computer Vision **61** (2005), no. 1, 55–79.

[12] ———, *Efficient belief propagation for early vision*, Int. J. of Computer Vision **70** (2006), no. 1, 41–54.

[13] P. F. Felzenszwalb, D. P. Huttenlocher, and J. M. Kleinberg, *Fast Algorithms for Large-state-space HMMs with applications to web usage analysis*, Advances in NIPS 16, MIT Press, 2004, pp. 409–416.

[14] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, Proceedings of the IEEE **93** (2005), no. 2, 216–231.

[15] J. Gil and M. Werman, *Computing 2-D Min, Median, and Max Filters*, IEEE Trans. on Pattern Analysis and Machine Intelligence **15** (1993), no. 5, 504–507.

[16] M. Klawe and D. Kleitman, *An Almost Linear Time Algorithm for Generalized Matrix Searching*, SIAM J. on Discrete Math. **3** (1990), no. 1, 91–97.

[17] P. Maragos, *Slope Transforms: Theory and Application to Nonlinear Signal Processing*, IEEE Trans. on Signal Processing **43** (1995), no. 4, 864–877.

[18] D. Sankoff and J. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983.

[19] A. Schönhage, M. Paterson, and N. Pippenger, *Finding the median*, J. Comp. Sys. Sci. **13** (1976), no. 2, 184–199.

[20] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

[21] P. Soille, *On morphological operators based on rank filters*, Pattern Recognition **35** (2002), no. 2, 527–535.