# Optimal Pivot Selection in Fast Weighted Median Search

André Rauh and Gonzalo R. Arce, *Fellow, IEEE*

***Abstract*—Weighted median filters are increasingly being used in signal processing applications and thus fast implementations are of importance. This paper introduces a fast algorithm to compute the weighted median (WM) of $N$ samples which has linear time and space complexity as opposed to $O(N \log N)$ which is the time complexity of traditional sorting algorithms. A popular selection algorithm often used to find the WM in large data sets is Quickselect whose performance is highly dependent on how the pivots are chosen. We introduce an optimization based pivot selection strategy which results in significantly improved performance as well as a more consistent runtime compared to traditional approaches. The selected pivots are order statistics of subsets. In order to find the optimal order statistics as well as the optimal subset sizes, a set of cost functions are derived, which when minimized lead to optimal design parameters. We compare the complexity to Floyd and Rivest's algorithm SELECT which to date has been the fastest median finding algorithm and we show that the proposed algorithm compared with SELECT requires close to 30% fewer comparisons. It is also shown that the proposed selection algorithm is asymptotically optimal for large $N$.**

***Index Terms*—Algorithms, median filters, nonlinear filters, order statistics, quicksort, sorting.**

## I. Introduction

WEIGHTED MEDIANS (WM), introduced by Edgemore over two hundred years ago in the context of least absolute regression, have been extensively studied in signal processing over the last two decades [1]–[4]. WM filters have been particularly useful in image processing applications as they are effective in preserving edges and signal discontinuities, and are efficient in the attenuation of impulsive noise properties not shared by traditional linear filters [1], [4]. The properties of the WM are inherited from the sample median—a robust estimate of location. An indicator of an estimator's robustness is its breakdown point, defined as the smallest fraction of the observations which when replaced with outliers will corrupt the estimate outside of reasonable bounds. The breakdown of the sample mean, for instance, is $1/N$ indicating that a single outlier present in the data can have a detrimental effect in the estimate. The median, on the other hand, has a breakdown point of 0.5 meaning that half or more of the data needs to be corrupted before the median estimate is deteriorated significantly [5]. This is the main motivation to perform median filtering. Assuming that the data is contaminated with noise and outlying observations, the goal is to remove the noise while retaining most of the behavior present in the original data. The median filter does just that and does not introduce values that are not present in the original data. Significant efforts have been devoted to the understanding of the theoretical properties of WM filters [1], [6]–[14], their applications [13], [15]–[19] and their optimization. A number of generalizations aimed at improving the performance of WM filters have recently been introduced in [20]–[25]. WM filters to date enjoy a rich theory for their design and application.

A limiting factor in the implementation of WM filters, however, is their computational cost. The most naive approach to computing the median, or any $k$th order statistic, is to sort the data and then select the $k$th smallest value. Once the data is sorted finding any order statistic is straightforward. Sorting the data leads to a computation time of $O(N \log N)$ and since the traversing of the sorted array to find the WM is a linear operation, the cost of this approach is the cost of sorting the data. Several approaches to alleviate the computational cost have been proposed. In many signal processing applications the filtering is performed by a running window and the computation of a median filter benefits from the fact that most values in the running window do not change when the window is translated. In such case, a local histogram can be utilized to compute a running median since the median computation takes into account only the element values and not their location in the sliding window. Simply maintaining a running histogram at each location of the sliding windows enables the computation of the median [26], [27]. In a running histogram, the median information is indeed present since pixels are sorted out into buckets of increasing pixel values. Removing pixels from buckets and adding more is a simple operation, making it easier to keep a running histogram and updating it than to go from scratch for every move of the running window. The same idea can be used to build up a tree containing pixel values and the number of occurrences, or intervals and number of pixels. One can thus see the immediate benefit of retaining this information at each step.

Other approaches to reduce the computation of running medians include separable approximations where 2D processing is attained by a 1D median filtering in two stages: the first along the horizontal direction followed by a second in the vertical direction [9], [13], [28]. All of the above mentioned techniques focus on the median computation of small kernels—a set of samples inside running windows. Depending on the signal's sampling resolution, the sample set may range from a small set of four or nine samples to larger windows that span a few hundred samples. However, emerging applications in signal processing are

Manuscript received May 12, 2011; revised September 30, 2011; accepted April 09, 2012. Date of publication May 02, 2012; date of current version July 10, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Arie Yeredor. This work was partially supported by the National Science Foundation (NSF) by Grants EECS-0725422 and CIF-0915800.

The authors are with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19711 USA (e-mail: rauh@ee.udel.edu; arce@ee.udel.edu).

Digital Object Identifier 10.1109/TSP.2012.2197394

beginning to demand WM computation of much larger sample sets. In particular, WM computations are not simply needed to process data in running windows. WM are often needed for the solution of optimization problems where absolute deviations are used as distance metrics. While $L_2$ norms, based on square distances, have been used extensively in signal processing optimization, the $L_1$ norm has attracted considerable attention recently because of its attributes when used in regression. First, the $L_1$ norm is more robust to noise, missing data, and outliers, than the $L_2$ norm [29]–[32]. The $L_2$ norm sums the square of the residuals and thus places small weight on small residuals and strong weight on large residuals. The $L_1$ norm penalty, on the other hand, puts more weight on small residuals and does not weight as heavily large residuals. The end result is that the $L_1$ norm is more robust than the $L_2$ norm in the presence of outliers or large measurement errors.

Second, the $L_1$ norm has also been used as a sparsity-promoting norm in the sparse signal recovery problem, where the goal is to recover a high-dimensional sparse vector from its lower-dimensional sketch [33]. In fact, the use of the $L_1$ norm for solving data fitting problems and sparse recovery problems traces back many years. In 1973, Claerbout *et al.* [34] proposed the use of the $L_1$ norm for robust modeling of Geophysical data. Later, in 1989, Donoho and Stark [35] used $L_1$ minimization for the recovery of a sparse wide-band signal from narrowband measurements. Over the last decade, a wide use of the $L_1$ norm for robust modeling and sparse recovery began to appear. It turns out, that it is often the case that WMs are required to solve optimization problems when $L_1$ norms are used in the data fitting model. For instance, the algorithm in [36] uses weighted medians on randomly projected compressed measurement to reconstruct the original sparse signal. For this application the input sizes on which a WM is performed are the size of the signals which range from several thousands up to several million data points. The computation of WMs for such very large kernels becomes critical and thus fast algorithms are needed. The data structures are no longer running windows and rough approximations are inadequate in optimization algorithms. To this end, fast and accurate WM algorithms are sought.

In this paper we introduce a new algorithm which solves the problem of finding the WM of a set of samples. The algorithm is based on Quickselect which is similar to the Quicksort algorithm. Even though the algorithm is explained and implemented for the WM problem it is straight forward to use similar concepts to construct a novel selection algorithm to find the order statistics of a set. The median is a special case of an order statistic. In many applications of data processing it is crucial to calculate statistics about the data. Popular choices are quantiles such as quartiles or 2-quantile (for instance in finance time series) both of which reduce to a selection problem. Often, these consist of thousands or millions of samples for which a fast algorithm is of importance in order to allow quick data analysis.

The $k$th order statistic of a set of $N$ samples is formally defined as the $k$th smallest element of the set. In this paper we adapt the standard notation of $X_{(k)}$ to refer to the $k$th order statistic. Additionally, the weight which is associated with the sample $X_{(k)}$ is denoted as $W_{[k]}$. Moreover, $W_0$ is needed as a threshold parameter and is formally defined as $W_0 = \frac{1}{2}\sum_{i=1}^{N} W_i$. Without loss of generality it is assumed that all weights are positive. All results can be extended to allow negative weights by coupling the sign of the weight to the corresponding sample and use the absolute value of the weight [7].

The problem of estimating a constant parameter $\beta$ under additive noise given $N$ observations can be solved by minimizing a cost function under different error criteria. The well-known sample average can be derived by using the $L_2$ error norm. Extending the idea by incorporating weights assigned to each sample into the equation results into the familiar weighted mean. The sample median follows from minimizing the error under the $L_1$ error norm. Conversely allowing the input samples to have different weights leads to the cost function of weighted median:

$$T_{WM}(\beta) = \sum_{i=1}^{N} W_i |X_i - \beta| \tag{1}$$

where the weights satisfy $W_i > 0$. The WM $\hat{\beta}$ can be defined as the value of $\beta$ in (1) which minimizes the cost function $T_{WM}(\beta)$ as $\hat{\beta} = \arg\min_{\beta} T_{WM}(\beta)$. Fig. 1 depicts an example cost function $T_{WM}(\beta)$ for $N = 6$. It can be seen in the figure that $T_{WM}(\beta)$ is a piecewise linear continuous function. Furthermore, it is a convex function and attains its minimum at the sample median which is one of the samples $X_i$. Fig. 1 also depicts the semiderivative of the cost function where it is observed as a piecewise constant nondecreasing function with the limits $\pm 2W_0$ as $\beta \to \pm\infty$. Note that the WM is the sample where the derivative crosses the horizontal axis. Therefore the WM can be defined by the following formula:

$$\hat{\beta} = \left\{ X_{(k)} : \min k \quad \text{for which} \sum_{i=0}^{k} W_{[N-i]} \geq W_0 \right\}.$$

Note that finding the $k$th order statistic is a special case of the above definition and can be found by replacing $W_0$ by $N - k$ and set all weights to 1.

Fig. 1 illustrates the algorithm to find the WM which is summarized as follows:

**Step 1:** Sort the samples $X_i$ with their concomitant weights $W_{[i]}$, for $i = 1, \ldots, N$.

**Step 2:** Traverse the sorted samples summing up the weights.

**Step 3:** Stop and return the sample at which the sum is higher or equal to $W_0$.

It is well known that sorting an array of $N$ elements requires $O(N \log N)$ comparisons, both, in the average as well as in the worst case. However, using a similar approach as in Quicksort, Hoare [2] introduced Quickselect which is an average linear time algorithm to find the $k$th order statistic of a set of samples. This algorithm can be extended such that Quickselect solves the WM problem. This is further described in Section III. The runtime of both algorithms greatly depend on the choice of the pivot elements which are used for partitioning the array. In Quicksort, a pivot close to the median is best and in Quickselect a pivot close to the sought order statistic is best. Moreover we extend the concept of Quickselect which seeks the $k$th order statistic to the more general case of WM filters which is needed

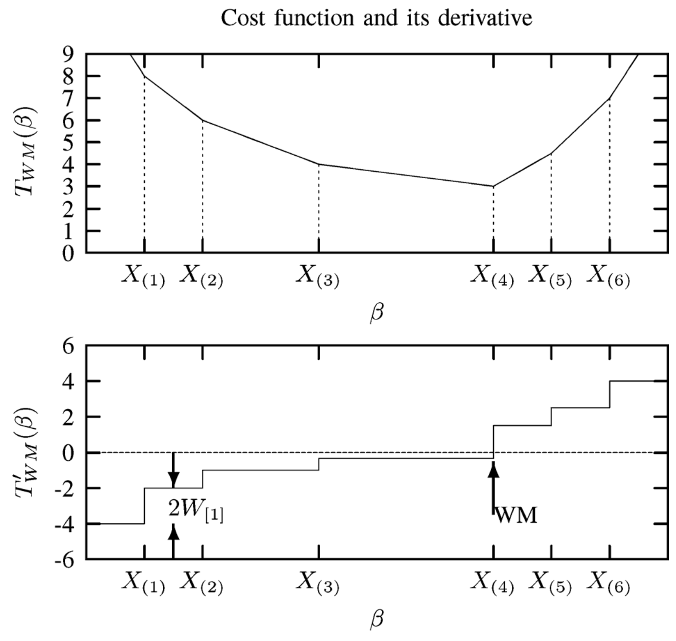

Fig. 1. (Top) An example cost function $T_{WM}(\beta)$ with six input samples $X_1$ to $X_6$ and a minimum at $\beta = X_{(4)}$. (Bottom) The derivative $T'_{WM}(\beta)$ and the zero-crossing point at $\beta = X_{(4)}$ which produces the WM. The weights $W_i$ range from 0 to 1.

for many signal processing applications. This paper introduces a new concept to pivot selection. The idea is based on an optimization framework in order to minimize the expected cost of solving the WM problem. The cost functions are then minimized in order to determine the optimal parameters. In particular, the cost is defined as the number of comparisons needed until the algorithm terminates which is the standard measurement for selection algorithms. Our approach uses order statistics of a subset of samples to select the pivot. The optimization framework finds the optimal value of the parameter $k$, which determines which order statistic to choose, and the optimal value of the subset size $M$. For practical performance comparisons the proposed algorithm was implemented in the C programming language. Numerous simulations validate the theory and show the improvements gained by the proposed algorithm.

## II. Preliminaries

The sample selection problem in essence, is equivalent to finding the $k$th-order statistic of a set of samples. The algorithms introduced in this paper to solve the selection problem can easily be extended to solve the WM problem. To this end, our treatment focuses on the selection problem to simplify analysis but will go into the details of solving WMs when necessary. Before introducing the two major algorithms it is necessary to define what "fast" means in terms of algorithm runtime. In algorithm theory a fast algorithm is one which solves the problem and at the same time has low complexity [37]. Complexity is defined in different ways which depends on the type of algorithm. In sorting and selection it is defined as the number of comparisons until termination. This is a sensible measure as the main cost of these algorithms is the partitioning step which compares all elements with the pivot. Furthermore the computational complexity is differentiated into worst case, average case and best case complexities. The best case complexity is of little interest since it is usually $O(N)$ for selection and $O(N)$ for sorting. The average case complexity is of most interest in practice since it is the runtime which can be expected in a real implementation with well-distributed input data. Of similar importance in theory as well as practice is the worst case complexity as this can be exploited by malicious users to attack the algorithm if the worst case complexity is unexpectedly higher than the average case.

It was shown in [38] that a lower bound on the expected time of the selection problem is in $O(N)$, i.e., linear time. This result is not surprising since given a solution it takes linear time to verify if the solution is correct. Additionally, in [39] it is shown that by choosing the pivots carefully it is possible to avoid the $O(N^2)$ worst case performance of the traditional Quickselect and also achieve a $O(N)$ worst case complexity. Hence it is important to note that there exists a linear time worst case selection algorithm as well as a proof that the lower bound is also linear. For this reason, we do not only compare the computational complexity of the algorithms in terms of their limiting behavior (i.e., asymptotic notation or *Landau* notation) but also analyze the behavior for low to moderate input sizes and obtain accurate numbers for the number of comparisons in order to compare performance. Taking into account the constants involved in the equations is important especially in practice. A popular example is the preferred Quicksort to Heapsort: Despite Heapsort's advantageous behavior of having worst case as well as average case complexity of $O(N \log N)$, Quicksort—with its worst case of $O(N^2)$—is often preferred as the smaller constant term of Quicksort makes it outperform Heapsort on average.

To this date the fastest selection algorithm has been *SELECT* which was introduced by Floyd and Rivest in 1975 [38]. The algorithm is asymptotically optimal for large $N$ and as shown by [38] the number of comparisons for selecting the $k$th-order statistic given $N$ elements is $N + \min(k, N-k) + o(N)$.[1] Note that our proposed algorithm is asymptotically optimal as well. Furthermore, as can be seen by the simulations in Section VI our algorithm outperforms *SELECT* and converges to the optimum more quickly. Quickselect, which will be introduced in Section III is another very popular selection algorithm due to its simplicity and similarity to Quicksort. Even though it is widely used in practice, its performance is always worse than *SELECT* except for very small input sizes. In particular, for a median-of-3 pivot selection approach Quickselect needs on average $2.75N$ comparisons for large $N$ as shown by [40].

## III. The Quickselect Algorithm

Quickselect was first introduced in [2] as an algorithm to find the $k$th-order statistic. Note that Quicksort and Quickselect are similar and the only difference between the two is that the former recurs on both subproblems—the two sets after partitioning—and the latter only on one.

The original algorithm chooses a sample at random from the sample set $\mathbf{X}$ which is called the first pivot $p_1$. Later in this paper, the method of choosing the pivot will be made more accurately, than selecting a random sample, and instead optimal

[1]We use the notation $O(N)$ and $o(N)$ in the following way: $f(N) \leq g(N) + O(N)$ means $(\exists k > 0)(\forall N) f(N) - g(N) \leq kN$ and $f(N) \leq g(N) + o(N)$ means $\lim_{N\to\infty}((f(N) - g(N))/N) = 0$

```
procedure QUICKSELECT($\mathbf{X}, k$)    ▷ Returns the $k^{\text{th}}$ order statistic of the set $\mathbf{X}$
    Select a pivot $p \in \mathbf{X}$ at random
    $\mathbf{X}_{\leq} \leftarrow \{X_i \in \mathbf{X} | X_i \leq p\}$
    $\mathbf{X}_{>} \leftarrow \{X_i \in \mathbf{X} | X_i > p\}$
    if $|\mathbf{X}_{\leq}| > k$ then
        return QUICKSELECT($\mathbf{X}_{\leq}$, k)
    else if $|\mathbf{X}_{>}| > k$ then
        return QUICKSELECT($\mathbf{X}_{>}$, $k - |\mathbf{X}_{\leq}|$)
    else
        return $p$
    end if
end procedure
```

Fig. 2. Standard Quickselect algorithm using random pivots.

order statistics which minimize a set of cost functions are used. By comparing all other elements to the pivot, the rank $r$ of the pivot is determined. The pivot is then put into the $r$th position of the array and all other elements smaller or equal than the pivot are put into positions before the pivot and all elements greater or equal are put after the pivot. This step is called partitioning and can be implemented very efficiently by running two pointers towards each other. One from the beginning of the array and one from the end, swapping elements if necessary until both pointers cross each other. If $r > k$ then the $k$th-order statistic is located in the first part of the array and Quickselect recurses on this part. If $r < k$ then Quickselect recurses on the second part but instead continues to seek the $(k - r)$th-order statistic. If $k = r$ the recursion terminates and the pivot is returned. A pseudocode description of the Quickselect algorithm is depicted in Fig. 2. The case of $k = (N+1)/2$ (N odd) is the well-known median and is considered a special case of the WM with all weights equal to one. Small modifications of Quickselect lead to a WM-finding Quickselect algorithm in the general case with arbitrary weights: Instead of counting the number of elements less than or equal to the pivot, the algorithm sums up the weights of all the samples which are less than or equal to the pivot. We define $W_l$ to be the sum of weights of the partition which contains all elements smaller than or equal to the pivot. Respectively, $W_r$ contains the sum of weights of the other partition. The next step is to compare $W_r$ and $W_l$ to $W_0$ and either recourse on the partition which contains the WM or return the pivot which terminates the algorithm.

## IV. Optimal Order Statistics in Pivot Selection

### A. *First Pivot* $p_1$

The run time of Quickselect is mostly influenced by the pivot choice. A good pivot can significantly improve the performance. Consider an example: If the pivot is small compared to the sought WM then only elements which are less than the pivot are discarded. In the worst case—i.e., if the pivot is the smallest element of the set—no elements are discarded. The main cost of the partitioning step is to compare all $N_0 - 1$ elements to the pivot. Where $N_0$ is the number of elements of the original set before any reductions have been performed. Clearly, a pivot close to the actual WM is desired. Assuming no prior knowledge of the sample distribution or their weights, the only good estimate for a pivot is to choose the median of the samples. The median—by its definition—ensures that half of the samples are removed after partitioning. However, finding the median is itself a selection type problem which would cost too much time to be computed. Instead, an approximation of the median is used as the pivot. A straightforward approach is to take a random subset of $\mathbf{X}$ and find the median of this smaller set. Let $M_0$ be the size of this subset with $M_0 \ll N_0$.

Martínez and Roura [3] studied the optimal subset size as a function of $N_0$ with the objective to minimize the average total cost of Quickselect. We found however that in practice the runtime was improved if $M_0$ was chosen larger. For this reason, we introduce a model to obtain a closed form solution for the near optimal $M_0$. Consider a set of samples $X_1, X_2, \ldots, X_N$. Assume each sample $\{X_i\}_{i=1}^{N_0}$ is independent and identically distributed (i.i.d.). Furthermore consider the random subset $\mathbf{X}' \subset \mathbf{X}$ with $|\mathbf{X}'| = M_0$. We seek the pivot $p_1 = \mathrm{median}(\mathbf{X}')$ as well as the optimal $M_0$ to minimize the expected samples left $(N_1)$ after the partitioning step. The cost function has to differentiate between the three main cases:

1) the pivot is less than the WM of $\mathbf{X}$;
2) the pivot is greater than the WM of $\mathbf{X}$;
3) the pivot is equal to the WM of $\mathbf{X}$.

The problem arises that both—pivot and WM—are not known beforehand and are in fact random variables. In order to obtain a simpler yet accurate enough cost function which can be solved, various assumptions and simplifications are applied to our model:

1) Each sample of the set $\mathbf{X}$ is modeled as uniformly distributed random variables. This approximation is in fact very accurate since the working point of our model is near the median of the true distribution function at which most distribution behave like a uniform distribution.
2) Finding $\mathrm{median}(\mathbf{X}')$ can be done in $cM_0$ comparisons where $c$ is some constant independent of $M_0$. Additionally, solving the remaining WM problem after the partitioning step can be done in $cN_1$ comparisons as well. Since $M_0 \ll N_1$ this does not hold true as finding the constant $c$ decreases with increasing samples. However, the difference is small and can be neglected.
3) The WM coincides with the median of the standard uniform distribution which is at 0.5. As stated earlier the WM is a random variable. However the variability of the WM can be accounted for in the pivot. Increasing the variance of the pivot distribution accordingly allows to perform this simplification.

Let $R$ be the expected number of elements removed after the partitioning step and by using the assumptions 1)-3) above, $R$ is derived as

$$R \approx \int_0^{1/2} x N_0 \frac{x^{M_0/4-1}(1-x)^{M_0/4-1}}{B(M_0/4, M_0/4)} dx \tag{2}$$

$$= \frac{1}{2} N_0 I_{1/2}\left(\frac{M_0}{4}+1, \frac{M_0}{4}\right) \tag{3}$$

where $N_0$ is the size of the original problem set $\mathbf{X}$, where $B$ is the beta function, and $I$ is the regularized incomplete beta function [41]. The first term $(xN_0)$ of (2) is the expected number of elements less than the pivot. The second term of (2) is the probability that the pivot is located at $x$. Assuming $M_0$ is odd then the median of a random subset of size $M_0$ is beta distributed with the parameters $(M_0+1)/2$ and $(M_0+1)/2$ [42]. To account for the third item of our approximation model we further change the variance of the median. Halfing the samples of the beta distribution approximately doubles the variance. Furthermore, we can use the fact

$$B\left(\frac{M_0+1}{4}, \frac{M_0+1}{4}\right) \approx B\left(\frac{M_0}{4}, \frac{M_0}{4}\right)$$

to obtain (2). Solving the integral of (2) cannot be done in a closed form. However since the resulting equation is again the p.d.f. of a beta distribution the result is the c.d.f. of the beta distribution evaluated at 0.5 in (3).

With the derived expression for $R$, the new cost function defined as the expected number of comparisons is given by

$$\begin{aligned} T_{M_0} &= c(N_0 - R) + cM_0 \\ &= cN_0\left(1 - \frac{1}{2}I_{1/2}\left(\frac{M_0}{4}+1, \frac{M_0}{4}\right)\right) + cM_0 \quad (4)\end{aligned}$$

where $c$ is the constant mentioned in the second item of the simplification model. The first summand is the expected number of comparisons necessary to solve the remaining problem after the partitioning step. The second summand accounts for the expected number of comparisons to find the pivot (i.e., the median of the subset). The minimum of $T_{M_0}$ in (4) is defined to be $M_0^*$:

$$M_0^* = \underset{1 \le M_0 \le N}{\arg\min}\, T_{M_0}.$$

Minimizing $T_{M_0}$ cannot be done in an algebraic way hence further approximations are necessary. First note that the division of the two parameters of the beta distribution $\frac{M_0/4+1}{M_0/4}$ is close to one as $M_0$ is large. This fact allows to use the normal distribution to approximate the beta distribution. The variance of the beta distribution is $\frac{M_0}{2(M_0+2)^2}$ which can be approximated as $(2M_0)^{-1}$. The resulting approximate cost function is:

$$\tilde{T}_{M_0} = M_0 + N_0 - \frac{N_0}{4}\operatorname{erfc}\left(\frac{\sqrt{M_0}}{M_0+2}\right) \quad (5)$$

where *erfc* is the complementary error function. It is easy to show that (5) is convex for $M_0 > 5$.

*Theorem 1:* For large $N_0$, the optimal subset size $M_0^*$ for choosing the first pivot is approximately

$$\tilde{M}_0^* \approx \frac{1}{\sqrt[3]{8\pi}} N_0^{2/3}.$$

*Proof:* Differentiating (5) with respect to $M_0$ yields

$$\tilde{T}'_{M_0} = -\frac{(M_0-2)N_0}{4\sqrt{\pi M_0}(M_0+2)^2} e^{-\frac{M_0}{(M_0+2)^2}} + 1.$$

Taking $e^{-M_0/(M_0+2)^2} \approx 1$ and $(M_0-2)/(M_0+2)^2 \approx 1/M_0$ yields the result. ∎

Note that in an implementation $M_0^*$ is rounded to the nearest odd integer. Now we can formally define the first pivot $p_1 = \mathrm{MEDIAN}(X'_1, \ldots, X'_{\tilde{M}_0^*})$.

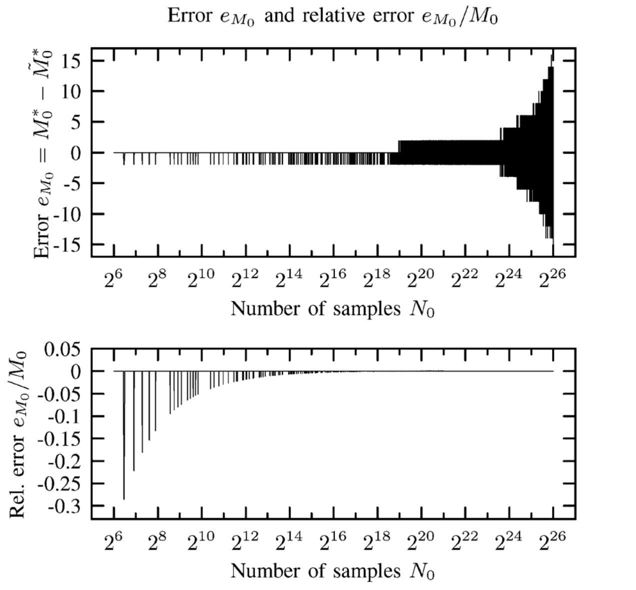

Fig. 3. The error of the optimal $M_0^*$ and the approximated $\tilde{M}_0^*$. The error increases as $N_0$ becomes increasingly large. However the relative error stays close to zero since the error grows slower than $M_0$. This result shows the applicability of our approximations.

The error introduced by the approximation is very small which can be seen by analyzing the error as well as the relative error. Fig. 3 depicts the error $e_{M_0} = M_0^* - \tilde{M}_0^*$ as well as the relative error $e_{M_0}/M_0^*$ for $2^6 \le N_0 \le 2^{26}$. The error is almost always zero except for very few $N_0$ for which the error is a small even number (due to rounding to odd numbers). In fact the number of values of $N_0$ where the error is not zero is 67964 between $2^6$ and $2^{26}$, i.e., approximately 99.999% of all samples between $2^6$ and $2^{26}$ have zero error. For large $N_0 \approx 2^{23}$ however, the error starts to increase which makes it important to analyze the relative error $e_{M_0}/M_0^*$. As can be seen by the lower graph of Fig. 3 the relative error stays close to zero as the error increases. The simulations were run over all integer numbers $N_0$ between $2^6$ and $2^{26}$. As $N_0$ becomes larger it becomes difficult to compute the error. Random $N_0$ of up to $2^{50}$ were picked and the error was bounded by 0.001 for the few chosen numbers which indicates that the error does increase faster than $M_0$ increases.

## B. Second Pivot $p_2$

For a large number of samples it is unlikely to find the exact WM with the first pivot. Thus it is assumed that the first pivot was either larger than or smaller than the WM. The next step is to choose the second pivot $p_2$. Should the median of a subset be used again? Intuitively, if the first pivot was smaller but close to the WM then a good choice for the second pivot is an element close to the WM but slightly larger than it. If we select the median of a subset again, we will most likely be far away from the WM, thus not discarding many elements. This is the result of a skewed sample median as many samples were discarded during the first step of the algorithm. It is natural to use the approach of using the $k$th-order statistic of a subset as the second pivot. The number of samples left after the first iteration of our algorithm is denoted as $N_1$. $M_1$ is the cardinality of the random subset $\mathbf{X}''$

of the remaining $N_1$ samples. Again the formula of Theorem 1 is used to determine the optimal $M_1$ as a function of $N_1$.

To find the optimal $k$, the approach of minimizing the expectation of an approximate cost function is again used. Since the goal is to remove as many elements as possible by choosing the pivot appropriately the cost was defined as the expected number of elements left after the partitioning step. A chosen pivot can be either larger, smaller or equal to the WM. Since the cost is negligible if the pivot $p_2$ happens to be the WM there are only two terms for the other two cases. Only the case in which the pivot was smaller than the WM will be explained now, the other case follows from symmetry.

The cost is defined formally as the expected number of elements left after the partitioning step and is given by

$$T_\beta = (kC-1)P_\beta + (N_1 - kC)(1-P_\beta) \tag{6}$$

where

$$C = \frac{N_1+1}{M_1+1}$$

and where $\beta = k/(M_1+1)$ is introduced to normalize $k$. The minimum of (6) is attained by the $k^*\mathrm{th}$ order statistic

$$k^* = \arg\min_{1\le k\le M_0} T_\beta. \tag{7}$$

$P_\beta$ is the probability that the expected order statistic of the WM is less than or equal to $\beta$ and will be formally defined below in (9). $\beta$ can be interpreted as the mean of the $k$th-order statistic of $M_1$ i.i.d. standard uniform distributed random variables. Equation (6) constitutes of two simple summands, the first of which accounts for the case that the pivot is greater than the WM and the second for the case it is smaller. The first term of the first summand is the expected number of elements which are less than the pivot. Again, we model the WM as a beta distributed random variable with the parameters $\alpha(M_1+1)$ and $M_1 - \alpha(M_1+1)+1$. Where $\alpha$ is the point at which the WM is expected to lie

$$\mathrm{WM} \approx X''_{(\lfloor \alpha(M_1+1)+0.5\rfloor)}$$

with

$$\alpha = \frac{W_0 - W^1_\le}{2W_0 - W^1_\le} \tag{8}$$

where $W^1_\le$ is the sum of all weights which were lower than the first pivot $p_1$ and formally defined as $W^1_\le = \sum_{i=1}^{\mathrm{rank}(p_i)} W_{[i]}$. $W_{[i]}$ are the concomitant weights as defined in the introduction. Note that the mean is $\alpha$ as desired. The terms for the second summand of (6) are similar to the first summand but cover the case when the pivot is smaller than the WM.

For the above model to hold, we assume that the input samples are uniformly distributed at the vicinity of the sample median. This holds true for many distributions. The pivot is modeled as being the $k$th-order statistic drawn from a standard uniform distribution. Hence $P_\beta$ can be expressed as [43]

$$\begin{aligned} P_\beta &= \Pr\{X''_{(\alpha(M_1+1))} \le \beta\} \\ &= I_\beta(\alpha(M_1+1), M - \alpha(M_1+1)+1) \end{aligned} \tag{9}$$

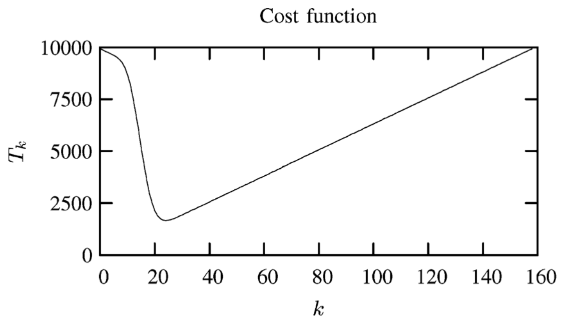

Fig. 4. Expected cost $T_k$ for choosing the $k$th-order statistic as the second pivot. ($N_1 = 10000, M_1 = 159, \alpha = 0.1$).

where $I$ is the incomplete beta function. Note that $X''_{(\alpha(M_1+1))}$ is not an order statistic since $\alpha(M_1+1)$ is most likely not an integer but can still be evaluated correctly since the incomplete beta function allows noninteger arguments.

An example of a cost function is depicted in Fig. 4. This figure shows that the 30th-order statistic ($k^* = 30$) of the set of $M_1 = 159$ samples should be chosen as the pivot in order to minimize the expected cost. This can be explained by looking at the parameters. $\alpha$ is 0.1 which means the WM is expected to lie close to $X''_{(20)}$. However if $X''_{(20)}$ was chosen as the pivot the probability that this pivot is again lower than the actual WM is higher than if $X''_{(30)}$ was chosen.

There is no closed form algebraic solutions to (6) so that further approximations are necessary. First $P_\beta$ is approximated by the normal distribution with mean $\alpha$ and variance $\sigma^2 = \alpha(1-\alpha)/(M_1+2)$. We call this approximation $\tilde{P}_\beta$. Replacing $k$ with $\beta(M_1+1)$, taking the derivative with respect to $\beta$ and division by the constant $N_1+1$ yields

$$\tilde{T}'_\beta = 2\tilde{P}_\beta - 1 + (2\beta-1)\tilde{P}'_\beta. \tag{10}$$

Note that $\tilde{P}_\beta$ is a cumulative density function and $\tilde{P}'_\beta$ is a probability density function.

*Lemma 1:* The function $\tilde{T}_\beta$ is quasi-convex for $\beta \in [0,1]$.

*Proof:* The proof is divided into three parts:

for $0 \le \beta \le \alpha$: Since $\tilde{P}_\beta$ has median $\alpha$ it follows that (10) is strictly negative.

for $\alpha < \beta < 1/2$: Taking the second derivative of $\tilde{P}_\beta$:

$$\tilde{T}''_\beta = 4\tilde{P}'_\beta + (2\beta-1)\tilde{P}''_\beta$$

shows that the function is convex on this interval as both terms are strictly positive.

for $1/2 \le \beta \le 1$: Since all terms are positive, $\tilde{P}_\beta$ is positive as well.

Combining the three intervals proofs the quasi-convexity. ■

Since the cost function is quasi-convex as shown by Lemma 1 the function has only one global minima between [0,1]. Minimizing (6) in order to find the optimal $k^*$ yields the following.

*Theorem 2:* The optimal order statistic which is to be chosen as the second pivot is

$$\tilde{k}^* \approx \left(\alpha + \sqrt{2}\sigma \log\left(\frac{1+2\alpha}{\sqrt{2\pi}\sigma}\right)\right) M_1 \tag{11}$$

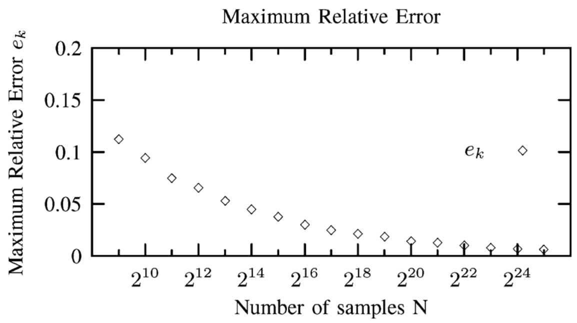

Fig. 5. The maximum relative error of the optimal order statistic $k^*$ and the approximation $\tilde{k}^*$. The error decreases as the input size increases.

where $\sigma = \sqrt{\frac{\alpha(1-\alpha)}{M_1+2}}$, $\alpha < 1/2$ as defined in (8), and $M_1$ is the size of the random subset $\mathbf{X}''$.

*Proof:* Taking the derivative of (6) and setting it to zero yields the following equation:

$$e^{x^2}\operatorname{erf}(x) - \frac{1-2\sqrt{2}\sigma x + 2\alpha}{\sqrt{2\pi}\sigma} = 0 \tag{12}$$

where $x = \frac{\beta-\alpha}{\sqrt{2}\sigma}$. First, the approximation $\operatorname{erf}(x) \approx 1$ is used to simplify (12). As $M_1$ increases so does $x$ and hence the error function is close to 1 which justifies this step. Next, the approximation $2\sqrt{2}\sigma x = 2\sqrt{2}\sigma\frac{\beta-\alpha}{\sqrt{2}\sigma} = 2(\beta-\alpha) \approx 0$ can be used since $\beta \approx \alpha$ for large $M_1$. Solving the resulting equation yields the desired result. ■

Note that in an implementation $\tilde{k}^*$ is rounded up to the nearest integer. The reason is that the cost function is steeper towards smaller number and more flat towards larger number which makes this step plausible.

The error introduced by the approximations is very small as can be seen in Fig. 5. The error was defined as

$$e_k = \max_{0<\alpha<1/2} \frac{|\tilde{k}^* - k^*|}{M_1}.$$

Note that the maximum relative error is declining with increasing $N$. Fig. 6 shows an example of the relative error. Note that the maximum occurs at $\alpha > 0.25$ and the error is close to zero for small $\alpha$. Most likely $\alpha$ will be close to zero. This is true if the first pivot was close to the actual WM. Hence the maximum error has a low impact on the average performance.

Given that $\tilde{k}^*$ is now known, Quickselect is called recursively to compute the pivot. With high probability this pivot will be slightly larger than the WM and hence many samples will be discarded. However, it is not guaranteed that the pivot is smaller than the WM and in the other case very few samples are discarded. The case that the pivot is on the same side of $T'_{WM}(\beta)$ as the first pivot is to be avoided as it would only result in discarding the elements between $p_1$ and $p_2$. Since this approach is based on probabilistic analysis it will fail sometimes which will result in a repetition of this approach until the two pivots are located on different sides such that $T'_{WM}(p_1)T'_{WM}(p_2) < 0$. Note that in the successful case we have a bounded problem, i.e., we know that the WM is between the first and the second pivot.

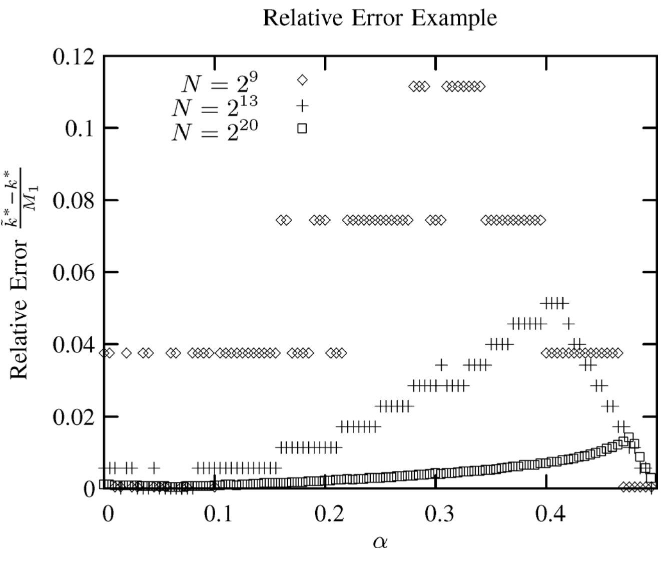

Fig. 6. The relative error of the optimal order statistic $k^*$ and the approximation $\tilde{k}^*$ for $N = 2^9$ ($M = 27$), $N = 2^{13}$ ($M = 175$) and $N = 2^{20}$ ($M = 4439$). It is important to note that the error is small for small $\alpha$. This is crucial for the algorithm to work well as small $\alpha$ are more likely to occur in practice.

## C. Subsequent Pivots

Given the first and second pivots, these can be considered as the lower and upper bounds of the array since all elements outside of these bounds have been discarded. Therefore, at each iteration hereafter the pivot is defined as a convex combination of the maximum and minimum

$$p_i = \beta X_{\min} + (1-\beta)X_{\max}$$

where $p_i$ is the *i*th pivot, $\beta$ is some constant between 0 and 1, and $X_{\min}$ and $X_{\max}$ are the minimum and maximum points left in the array. After each iteration the minimum and maximum will be updated and hence new bounds are established. This strategy is very different to the existing ones since the pivots are not selected but computed which implies that the pivots are most likely not an element of the set. Note however, that the crucial step for the algorithm—partitioning the set—is still possible.

After the partitioning step, the set which does not contain the WM will be discarded and the new pivot takes over the weights of the discarded set. The pivot will act as a proxy of the samples of the discarded set. This will introduce new samples into the set which originally did not exist. However, this will not change the WM or the zero-crossing point and is therefore a valid operation.

An optimal $\beta$ at some iteration can again be found by minimizing a cost function which will be of similar structure to the cost function (6). The result is similar to (11)

$$\beta \approx \alpha + \sqrt{2}\sigma \log\left(\frac{1+2\alpha}{\sqrt{2\pi}\sigma}\right)$$

where $\alpha$ and is defined the same way as in (8), $\sigma^2 = \alpha(1-\alpha)/(N_i+2)$ and $N_i$ being the number of elements left in the array at the *i*th iteration.

Intuitively, if there are many samples between $X_{\min}$ and $X_{\max}$ then this approach works very well. If, however, few samples are within the boundaries then the assumption of the data being uniformly distributed breaks down which results in degraded performance. Therefore, the algorithm stops when the problem size gets below a certain threshold level and solves the remaining problem by the standard median-of-3 Quickselect implementation. Also, in the case that this approach does not remove any elements—this can happen for some inputs—the algorithm falls back to the median-of-3 Quickselect as well to protect from an infinite loop.

## V. Complexity Analysis

It has been shown, e.g., in [44], that the average number of comparisons to find the median using the standard Quickselect is approximately 3.39N and 2.75N for median-of-3 Quickselect. The space complexity of our algorithm is $O(N)$ since the partitioning step can be done in-place and hence does not need additional memory. For the first and second partition step the algorithm needs to choose the pivot from a random subset of $\mathbf{X}$. However, this approach would require additional memory and time to generate the random numbers. In practice it is more efficient to sample uniformly. Selecting an order statistic from this subset can also be done in-place and hence there is no need to copy the elements.

Since the most time is spent in the partitioning step, the time complexity is defined as the number of element-wise comparisons the algorithm makes until termination. In addition the comparisons to select the pivots recursively have to be taken into account. Using a median-of-$M$ samples as the pivot, it was shown in [3] that the number of comparisons is $2N + o(N)$ on average. Simulations in the next section show that the proposed algorithm beats this result. This is mainly due to the strategy used for selecting the second pivot. As explained above, selecting just the median is not optimal in general.

The worst case runtime of the proposed algorithm is $O(N^2)$ trivially since the algorithm will always do better or equal than the median-of-3 Quickselect which has the same worst case runtime. It is shown in [3] that the worst case runtime for a median-of-$M$ approach is $O(N^{3/2})$ for $M = \Theta(\sqrt{N})$[2] which is a similar approach used for the proposed algorithm. This is an indication that the proposed algorithm's worst case complexity is also lower than $O(N^2)$. Performing theoretical average complexity analysis is not possible due to the complex approach of choosing the subset size and the pivots. However, given the simulations from Section VI the complexity can be approximated by fitting a function. An approximation is useful to give the reader or programmer an intuitive measure of runtime behavior of the proposed algorithm. Using the tool *Eureqa*, which was introduced in [45], we obtain the following formula for the complexity $C(N) = 1.5N + 6.435N^{0.629}$. We ran Eureqa with the basic building plots and the block "power". We did not smooth the data prior and choose the most compact formula over the most accurate one.

## VI. Simulation Results

To compare the simulation results among different sample set sizes the number $\bar{C} = C/N$ is used. $C$ is the sample mean of the number of comparisons and $N$ the input data set size. The

[2]We use the notation $\Theta(N)$ in the following way: $f(N) \in \Theta(g(N))$ means $\exists k_1, k_2 > 0, N_0 \forall N > N_0 : |g(N)|k_1 \leq |f(N)| \leq |g(N)|k_2$

TABLE I
Number of Comparisons Are Not Affected by Heavy Tailed Input Distributions

| alpha | 2 | 1.8 | 1.4 | 1.0 | 0.8 | 0.6 | 0.4 |
|---|---|---|---|---|---|---|---|
| $\bar{C}$ | 1.72 | 1.72 | 1.72 | 1.72 | 1.72 | 1.72 | 1.72 |

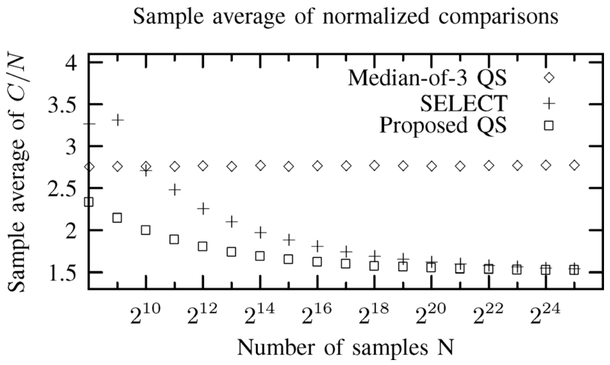

Fig. 7. The sample average of the normalized number of comparisons $(C/N)$ for the different algorithms.

proposed algorithm was implemented in MATLAB and simulated with different input distributions of $\mathbf{X}$. Since WMs are often used with heavy tailed input distributions, the algorithm was run on $\alpha$-stable distributions where $\alpha \in [0, 2]$ [1]. As alpha gets smaller the tails of the distribution get heavier which results in more outliers present in the input data. For $\alpha = 2$ the data is Gaussian distributed. The results for $N = 8193$ are depicted in Table I. Note that the mean does not change and is robust to heavy tailed inputs. This can be easily explained: After the first two steps the algorithm makes the problem bounded above and below and thus all outliers are already discarded. Furthermore due to the approach to pick the first two pivots it is highly unlikely to select an outlier as the pivot. Uniformly distributed input data does not change the complexity which is expected since the design of the proposed algorithm is based on the assumption that the input is uniformly distributed around the WM.

We chose to compare the performance to two other algorithms. The standard median-of-3 Quickselect and Floyd and Rivest's SELECT algorithm [38]. Since both algorithms only solve the more specific problem of finding the order statistic of a set of samples, a version of the proposed algorithm which finds the median was implemented in the C programming language. The results are depicted in Fig. 7. The proposed method clearly outperforms SELECT for smaller and medium input sizes and both methods converge to $\bar{C} = 1.5$ optimum for very large input sizes. The $\bar{C} = 1.5$ optimum can be explained easily: For very large input sizes the number of samples removed is almost $N_0/2$ after the first partitioning step which does $N_0$ comparisons. The second pivot will also remove almost $N_0/2$ samples which cost $N_0/2$ comparison to partition. All what is left is a fraction of the input size and therefore the number of comparisons converge to $1.5N_0$. The asymptotic optimality is proved in Appendix B.

To explain why the proposed algorithm performs better we have to look at how SELECT works: SELECT first chooses two pivots which lie with high probability just to the left and just to the right of the sought median. This however is suboptimal unless the input size is very large since the two pivots have to be

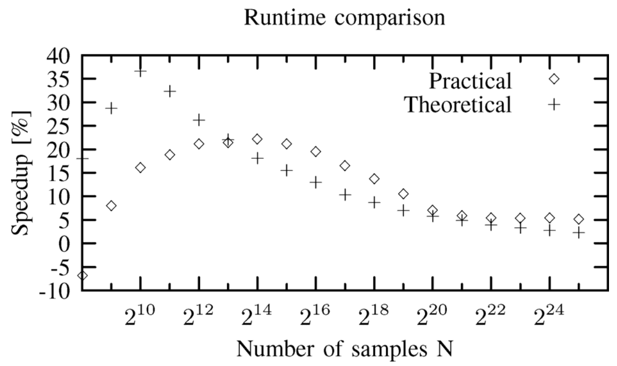

Fig. 8. Speedup against Floyd and Rivest's SELECT.

chosen to be relatively far away from the median and therefore many unnecessary comparisons are done. The proposed method on the other hand tries to get the first pivot as close as possible to the median and then chooses the second pivot such that the median is with high probability between the first and the second pivot. This explains Fig. 7.

However since the proposed algorithm requires more code than the other two algorithms, it is of interest to show how the runtime of the proposed algorithm compares. For these simulations, a C-implementation was run and timed and the sample averages are compared. The speedup is measured as $\frac{T_{\text{slow}}}{T_{\text{new}}}$ where $T_{\text{slow}}$ is the runtime of the faster of the two algorithms SELECT or Quickselect. The tests were carried out on a Linux host on an *AMD Opteron 2376* processor and *GCC 4.6* compiler with all optimizations enabled. Fig. 8 shows in addition to the speedup as defined above also the theoretical speedup which is defined as $\frac{C_{\text{slow}}}{C_{\text{new}}}$. Similar as above, $C_{\text{slow}}$ is the number of comparisons for the faster of the two algorithms and $C_{\text{new}}$ the number of comparisons for the proposed algorithm. The proposed method is up to 23% faster than SELECT for a wide range of input sizes and converges to $\approx 5\%$ for very large input sizes. There are subtle differences in the theoretical and practical speedup which can be explained. First, the speedup for the input sizes 513, 1025 is smaller due to the involved overhead of the more complex implementation. This increased complexity however pays off nicely for larger input sizes. For this reason the proposed method is slower than SELECT for input sizes smaller than 513 samples. Second, even though the theoretical speedup approaches zero as $N$ becomes large the practical speedup approaches 5%. It seems that the compiler can optimize our algorithm slightly better than SELECT. For all simulations the input samples were i.i.d. normal distributed.

## VII. Conclusion

A fast new WM algorithm is introduced which is based on Quickselect. The algorithm is based on the optimality of several parameters used to design a set of pivots. In particular, the set of order statistics and the sample set size are the crucial parameters for optimal performance. Minimizing cost functions in combination with a well-approximated model for the cost were used to develop this novel algorithm. Theory explains that the proposed method should be faster than Floyd and Rivest's SELECT unless input sizes are very small. This was backed up by experiments which showed a speedup of up to 23%. Furthermore the proposed algorithm can compute the median as well as the WM but the same ideas can be applied to finding the $k$th-order statistic. The extension is straightforward. In addition the proposed algorithm can be used to solve multidimensional selection problems which require medians of large data sets.

A C-implementation can be downloaded from the homepage of one of the authors at: http://www.eecis.udel.edu/~arce/fqs/

## Appendix A

The standard Quickselect algorithm was stated in Fig. 2. A simple overview of the modifications to the standard Quickselect is given:

**Step 1:** First pivot is chosen as the median of a subset of size $M_0$.

**Step 2:** Second pivot is chosen as the $k$th-order statistic of a subset of size $M_1$. Repeat this approach until problem is bounded.

**Step 3:** All following pivots are convex combinations of the maximum and minimum sample in the remaining set. This is repeated until the size is less than a computer dependent threshold after which the problem is solved with a median-of-3 Quickselect.

## Appendix B

This proofs that $C(N_0 \to \infty) = 1.5N_0$ for $k = (N_0+1)/2$ (median):

*Proof:* As $N_0 \to \infty$ then $M_0 \to \infty$ and hence $p_1 \to \text{MEDIAN}(\mathbf{X})$. This removes $N_0/2$ elements with cost $N_0$. As $M_0 \to \infty$ then $M_1 \to \infty$. As $p_1 \to \text{MEDIAN}(\mathbf{X})$ then either $\alpha \to 0$ or $\alpha \to 1$. Hence either $k^* \to 1$ (since $\alpha \to 0$) or $k^* \to M_1$ (since $\alpha \to 1$). Hence $p_2 \to \text{MEDIAN}(\mathbf{X})$. This removes $N_0/2$ elements with cost $N_0/2$.

Since the first and second pivot each removed $N_0/2$ elements, the number of remaining samples $N_2 \to 0$. Hence the total cost $\to 1.5N_0$. ■

## References

[1] G. R. Arce, *Nonlinear Signal Processing: A Statistical Approach*. New York: Wiley, 2005.
[2] C. Hoare, "Find (algorithm 65)," *Commun. ACM*, pp. 4:321–4:322, 1961.
[3] C. Martínez and S. Roura, "Optimal sampling strategies in quicksort and quickselect," *SIAM J. Computing*, vol. 31, no. 3, pp. 683–705, 2002.
[4] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*. Boca Raton, FL: CRC Press, 1997.
[5] C. L. Mallows, "Some theory of nonlinear smoothers," *Ann. Stat.*, vol. 8, no. 4, pp. 695–715, 1980.
[6] N. C. Gallagher and G. L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 29, no. 6, pp. 1136–1141, Dec. 1981.
[7] G. R. Arce, "A general weighted median filter structure admitting negative weights," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3195–3205, 2002.
[8] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: A tutorial," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 3, pp. 157–192, 1996.
[9] G. R. Arce and M. P. McLoughlin, "Theoretical analysis of the max/median filter," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 1, pp. 60–69, Jan. 1987.
[10] G. R. Arce and N. C. Gallagher, "State description for the root-signal set of median filters," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 30, no. 6, pp. 894–902, 1982.

[11] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation," *IEEE Trans. Signal Process.*, vol. 39, no. 2, pp. 395–410, 1991.

[12] G. R. Arce and J. L. Paredes, "Recursive weighted median filters admitting negative weights and their optimization," *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 768–779, 2000.

[13] M. P. McLoughlin and G. R. Arce, "Deterministic properties of the recursive separable median filter," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 1, pp. 98–106, 1987.

[14] A. Flaig, G. R. Arce, and K. E. Barner, "Affine order-statistic filters: Medianization of linear FIR filters," *IEEE Trans. Signal Process.*, vol. 46, no. 8, pp. 2101–2112, 1998.

[15] G. R. Arce and R. E. Foster, "Detail-preserving ranked-order based filters for image processing," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 37, no. 1, pp. 83–98, Jan. 1989.

[16] G. R. Arce, "Multistage order statistic filters for image sequence processing," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1146–1163, 1991.

[17] G. R. Arce and N. C. Gallagher, "BTC image coding using median filter roots," *IEEE Trans. Commun.* , vol. 31, no. 6, pp. 784–793, 1983.

[18] K. E. Barner and G. R. Arce, "Permutation filters: A class of nonlinear filters based on set permutations," *IEEE Trans. Signal Process.*, vol. 42, no. 4, pp. 782–798, 1994.

[19] M. Fischer, J. L. Paredes, and G. R. Arce, "Weighted median image sharpeners for the world wide web," *IEEE Trans. Image Process.*, vol. 11, no. 7, pp. 717–727, 2002.

[20] T. C. Aysal and K. E. Barner, "Meridian filtering for robust signal processing," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 3949–3962, 2007.

[21] S. Kalluri and G. R. Arce, "Adaptive weighted myriad filter algorithms for robust signal processing in $\alpha$-stable noise environments," *IEEE Trans. Signal Process.*, vol. 46, no. 2, pp. 322–334, 1998.

[22] J. G. Gonzalez and G. R. Arce, "Optimality of the myriad filter in practical impulsive-noise environments," *IEEE Trans. Signal Process.*, vol. 49, no. 2, pp. 438–441, 2001.

[23] S. Kalluri and G. R. Arce, "Fast algorithms for weighted myriad computation by fixed-point search," *IEEE Trans. Signal Process.*, vol. 48, no. 1, pp. 159–171, 2000.

[24] J. L. Paredes and G. R. Arce, "Stack filters, stack smoothers, and mirrored threshold decomposition," *IEEE Trans. Signal Process.*, vol. 47, no. 10, pp. 2757–2767, 1999.

[25] S. Kalluri and G. R. Arce, "A general class of nonlinear normalized adaptive filtering algorithms," *IEEE Trans. Signal Process.*, vol. 47, no. 8, pp. 2262–2272, 1999.

[26] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 27, no. 1, pp. 13–18, Feb. 1979.

[27] S. Perreault and P. Hébert, "Median filtering in constant time," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2389–2394, Sep. 2007.

[28] P. M. Narendra, "A separable median filter for image noise smoothing," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 1, pp. 20–29, 2009.

[29] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. Hoboken, NJ: Wiley, 1987.

[30] V. Barnett, T. Lewis, and F. Abeles, "Outliers in statistical data," *Phys. Today*, vol. 32, p. 73, 1979.

[31] Y. Li and G. R. Arce, "A maximum likelihood approach to least absolute deviation regression," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 1762–1769, 2004.

[32] P. Bloomfield and W. Steiger, "Least absolute deviations curve-fitting," *SIAM J. Scientif. Statist. Comput.*, vol. 1, no. 2, pp. 290–301, 1980.

[33] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 877–905, 2008.

[34] J. F. Claerbout and F. Muir, "Robust modeling with erratic data," *Geophys.*, vol. 38, p. 826, 1973.

[35] D. L. Donoho and P. B. Stark, "Uncertainty principles and signal recovery," *SIAM J. Appl. Math.*, vol. 49, no. 3, pp. 906–931, 1989.

[36] J. L. Paredes and G. R. Arce, "Compressive sensing signal reconstruction by weighted median regression estimates," *IEEE Trans. Signal Process.*, vol. 59, no. 6, Jun. 2011.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein*, Introduction to Algorithms*. Cambridge, MA: The MIT Press, 2001.

[38] R. W. Floyd and R. L. Rivest, "Expected time bounds for selection," *Commun. ACM*, vol. 18, no. 3, pp. 165–172, 1975.

[39] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection*," *J. Comp. Syst. Sci.*, vol. 7, no. 4, pp. 448–461, 1973.

[40] C. Martínez, D. Panario, and A. Viola, "Adaptive sampling for quickselect," in *Proc. 15th Ann. ACM-SIAM Symp. Discr. Algorithms (SIAM)*, Philadelphia, PA, 2004, pp. 447–455.

[41] F. Olver, D. Lozier, R. Boisvert, and C. Clark*, NIST Handbook of Mathematical Functions*. New York: Cambridge Univ. Press, 2010.

[42] H. A. David and H. N. Nagaraja*, Order Statistics*, ser. Wiley Ser. Probabil. Math. Statist. New York: Wiley, 2003.

[43] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja*, A First Course in Order Statistics (Classics in Applied Mathematics)*. Philadelphia, PA: SIAM, 2008.

[44] D. E. Knuth*, Mathematical Analysis of Algorithms*. New York: Storming Media, 1971.

[45] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, p. 81, 2009.

**André Rauh** received the Dipl. Ing. degree from the University of Applied Sciences Esslingen, Germany, in 2011.

He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Delaware, Newark. His research interest include signal and image processing, compressive sensing and robust, nonlinear, and statistical signal processing.

**Gonzalo R. Arce** (F'00) received the Ph.D. degree from Purdue University, West Lafayette, IN.

He joined the University of Delaware, Newark, where he is the Charles Black Evans Distinguished Professor of Electrical and Computer Engineering. He served as Department Chairman from 1999 to 2009. He held the 2010 Fulbright-Nokia Distinguished Chair in Information and Communications Technologies at the Aalto University, Helsinki, Finland. He has held Visiting Professor appointments with the Tampere University of Technology and the Unisys Corporate Technology Center. His research interests include statistical signal processing and computational imaging. He is coauthor of the texts *Nonlinear Signal Processing* (Wiley, 2004), *Modern Digital Halftoning* (CRC Press, 2008), and *Computational Lithography* (Wiley, 2010).

Dr. Arce has served as Associate Editor of several journals of the IEEE, OSA, and SPIE.