Streaming max-min Filter using no more than 3 comparisons per element

Preface

在图像去雾算法中，Dr. Kaiming He 给出了 dark channel 的方法，其中第一步是做暗通道提取。本文介绍了一种取最大值最小值的算法。

Preface

- **N个数据需要n-1次比较得到最大或最小值。**
- **需要1.5*n-2次比较才能得到最大和最小值。**

Running max-min filter:

given an array $a1, \ldots, an$, find the maximum and the minimum over all windows of size w, that is max /min $_{i \in [j,\ j+w)}$ ai for all j.

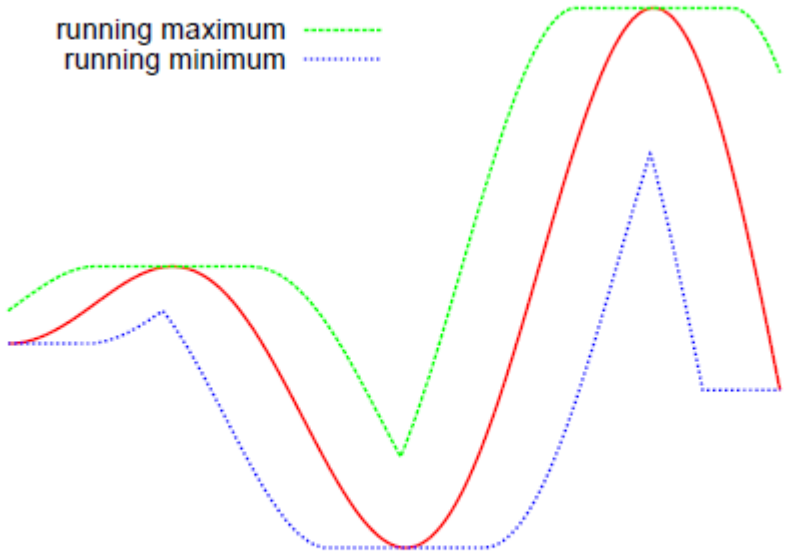


Fig. 1.1: Example of a running MAX-MIN filter.

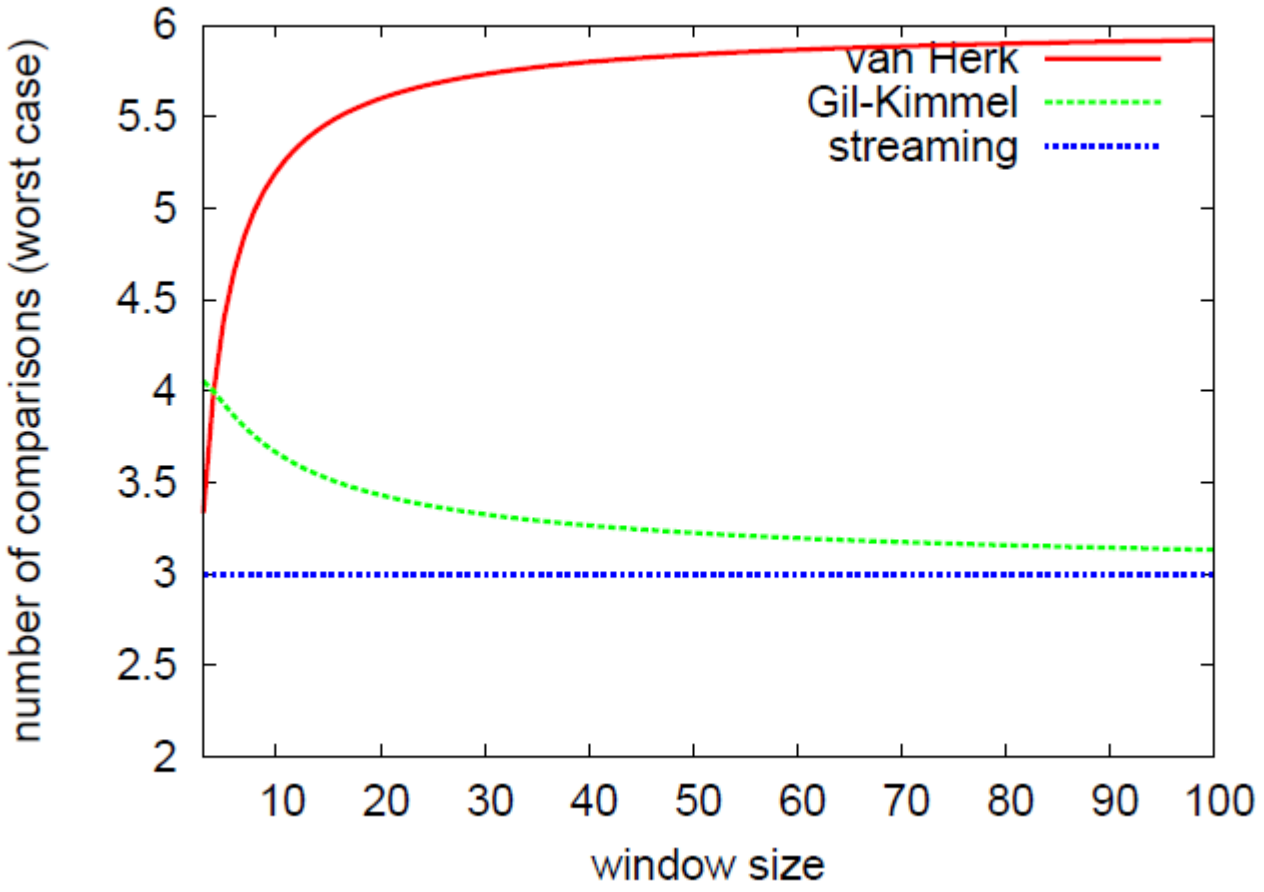
Running Max-Min Filter

TABLE I: Worst-case number of comparisons and stream latency for competitive MAX-MIN filter algorithms. Stream latency and memory usage (buffer) are given in number of elements.

algorithm	comparisons per element (worst case)	stream latency	buffer
naive	$2w - 2$	**0**	$O(1)$
van Herk [1992], Gil and Werman [1993]	$6 - 8/w$	w	$4w + O(1)$
Gil and Kimmel [2002]	$3 + 2 \log w/w + O(1/w)$	w	$6w + O(1)$
New algorithm	**3**	**0**	$2w + O(1)$

*当数据流是分段单调的时候，该算法的CPE(worst)可以降到2。
*stream latency是指延时流。

Running Max-Min Filter



Running Max-Min Filter

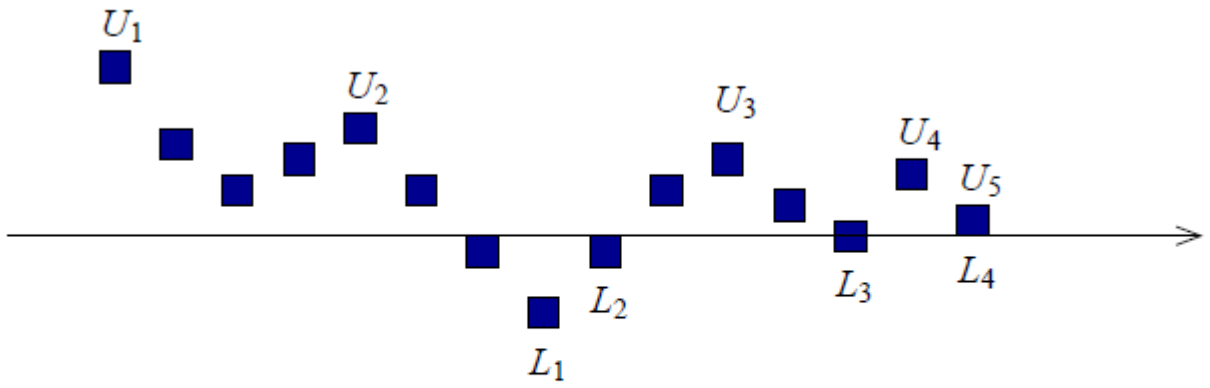
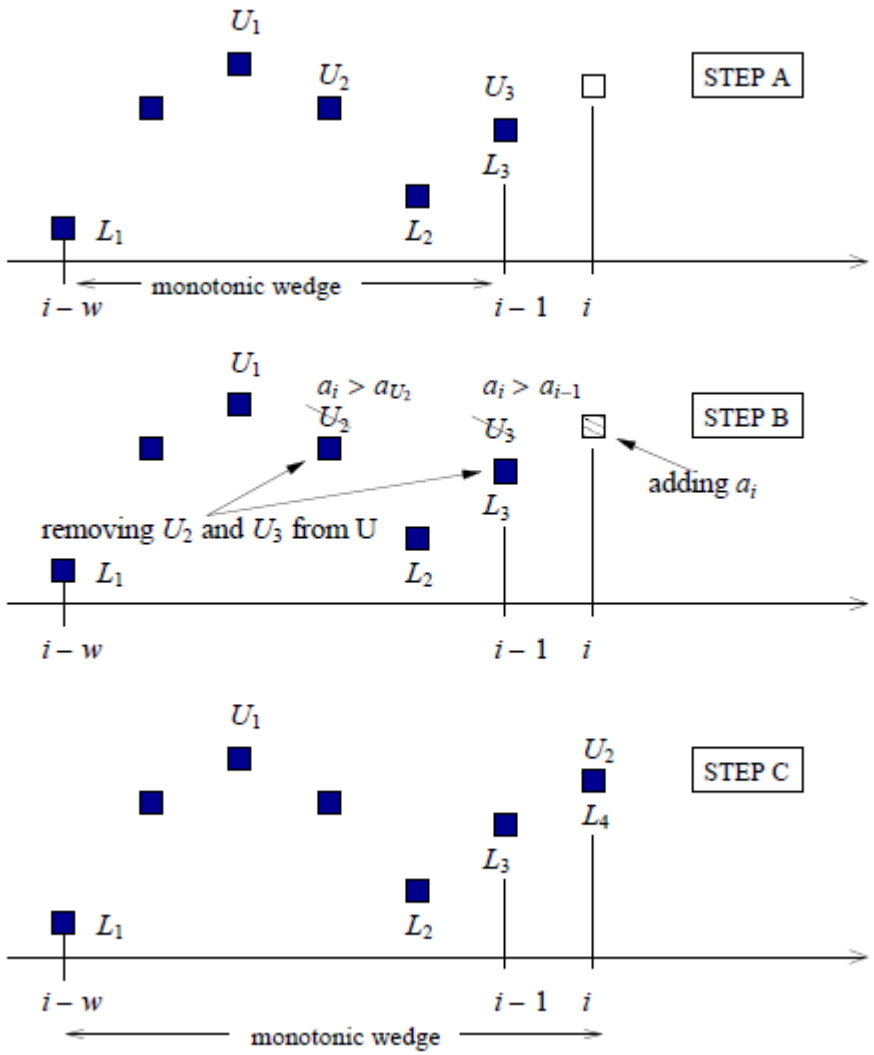


Fig. 4.3: Example of a monotonic wedge: data points run from left to right.

Running Max-Min Filter

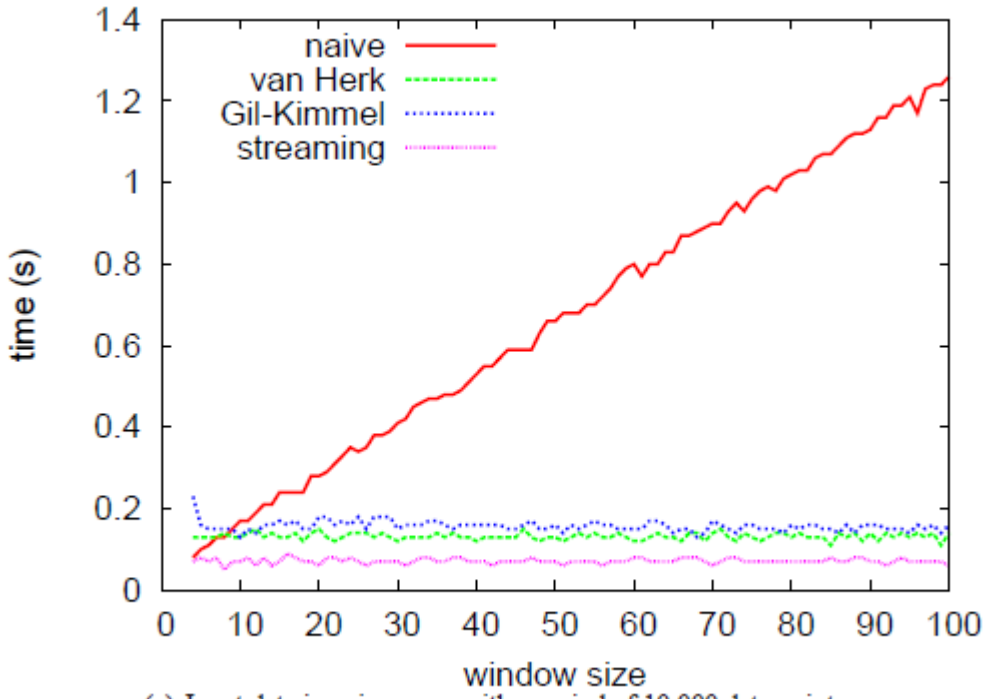


Running Max-Min Filter

Algorithm 1 Streaming algorithm to compute the MAX-MIN filter using no more than 3 comparisons per element.

```
1:  INPUT: an array a indexed from 1 to n
2:  INPUT: window width w > 2
3:  U, L ← empty double-ended queues, we append to "back"
4:  append 1 to U and L
5:  for i in {2, ..., n} do
6:      if i ≥ w + 1 then
7:          OUTPUT: a_front(U) as maximum of range [i − w, i)
8:          OUTPUT: a_front(L) as minimum of range [i − w, i)
9:      if a_i > a_{i−1} then
10:         pop U from back
11:         while a_i > a_back(U) do
12:             pop U from back
13:     else
14:         pop L from back
15:         while a_i < a_back(L) do
16:             pop L from back
17:     append i to U and L
18:     if i = w + front(U) then
19:         pop U from front
20:     else if i = w + front(L) then
21:         pop L from front
```

Running Max-Min Filter



(a) Input data is a sine wave with a period of 10 000 data points.

Appendix

```
// input: array a, integer window width w
// output: arrays maxval and minval
// buffer: lists U and L
// requires: STL for deque support
deque<int> U, L;
for(uint i = 1; i < a.size(); ++i) {
 if(i>=w) {
   maxval[i-w] = a[U.size()>0 ? U.front() : i-1];
   minval[i-w] = a[L.size()>0 ? L.front() : i-1];
 }// end if
 if(a[i] > a[i-1]) {
  L.push_back(i-1);
  if(i == w+L.front()) L.pop_front();
  while(U.size()>0) {
   if(a[i]<=a[U.back()]) {
    if (i == w+U.front()) U.pop_front();
    break;
   }// end if
   U.pop_back();
  }// end while
```

```
  } else {
   U.push_back(i-1);
   if (i == w+U.front()) U.pop_front();
   while(L.size()>0) {
    if(a[i]>=a[L.back()]) {
     if(i == w+L.front()) L.pop_front();
     break;
    }//end if
    L.pop_back();
   }//end while
  }// end if else
 }// end for
 maxval[a.size()-w] = a[U.size()>0 ? U.front() : a.size()-1];
 minval[a.size()-w] = a[L.size()>0 ? L.front() : a.size()-1];
```