For the convenience of the representation, we redefine the temporal neighborhood of the pixel at (x, y) of the k-th frame as

$$D_k = \{d_{k-1}, d_{k-2}, \ldots, d_{k-N}\} \quad (2)$$

where the data set D_k stores the previous N pixel data (d_{k-1}, $d_{k-2}, \ldots, d_{k-N}$) located at (x,y). Here we assume that the pixel value is in the range of 0 and 255. O_{Mid} means the middle order of the N data, i.e. $O_{\mathrm{Mid}}=(N-1)/2$ (N must be odd). Then, the median selection based on the histogram is described as the following function.

Function: m = medhist(D)
// Input: D stores the previous N pixel data
// Output: m returns the median of D

```
hn = hist(D)  // hist(.) returns the histogram of the data set
csum= 0   // csum means the cumulative function of the histogram
for i = 0 to 255
        if hn[i]>0 then
                csum+= hn[i]
                if csum≥O_Mid then break
        end if
end for
return i
```

The above histogram selection first calculates the histogram of the input data set. Then, the cumulative function of the histogram is evaluated by incrementing index from 0 to 255. When the cumulative function reaches the middle order, the current index is the median of the data set required.

B. *Median Selection Based on Histogram and Repetition Checking*

To develop the fast algorithm of the histogram selection, we first design a lower bound and an upper bound of the cumulative function at the median, denoted by lb and ub. By slightly modifying the above histogram selection scheme, we obtain the following function to evaluate the median as well as the two bounds of a data set.

```
Function: {m, lb, ub} = medhist_bnd(D)
// Input: D stores the previous N pixel data
/* Ouput: m returns the median of D. lb and ub respectively
returns the lower bound and upper bound of the cumulative
function at the median. */

hn = hist(D)  // hist(.) returns the histogram of the data set
csum = 0 // csum is the cumulative function of the histogram
for i = 0 to 255
        if hn[i]>0
                csum += hn[i]
                if csum≥O_Mid then
                lb = csum – hn[i]+1, ub = csum
                        break
                end if
        end if
end for
return i, lb, ub
```

Similar to the original histogram selection, when the cumulative function of the histogram (*csum*) reaches the middle order, *lb* can be obtained by $lb=csum-hn[i]+1$, where $hn[i]$ is the value of the indexed histogram and *ub* is equal to *csum*. Fig. 1 shows an example of cumulative function of histogram, and obviously the middle order satisfies the relation of $lb \leq O_{\mathrm{Mid}} \leq ub$. We apply the relation to develop the repetition checking scheme.

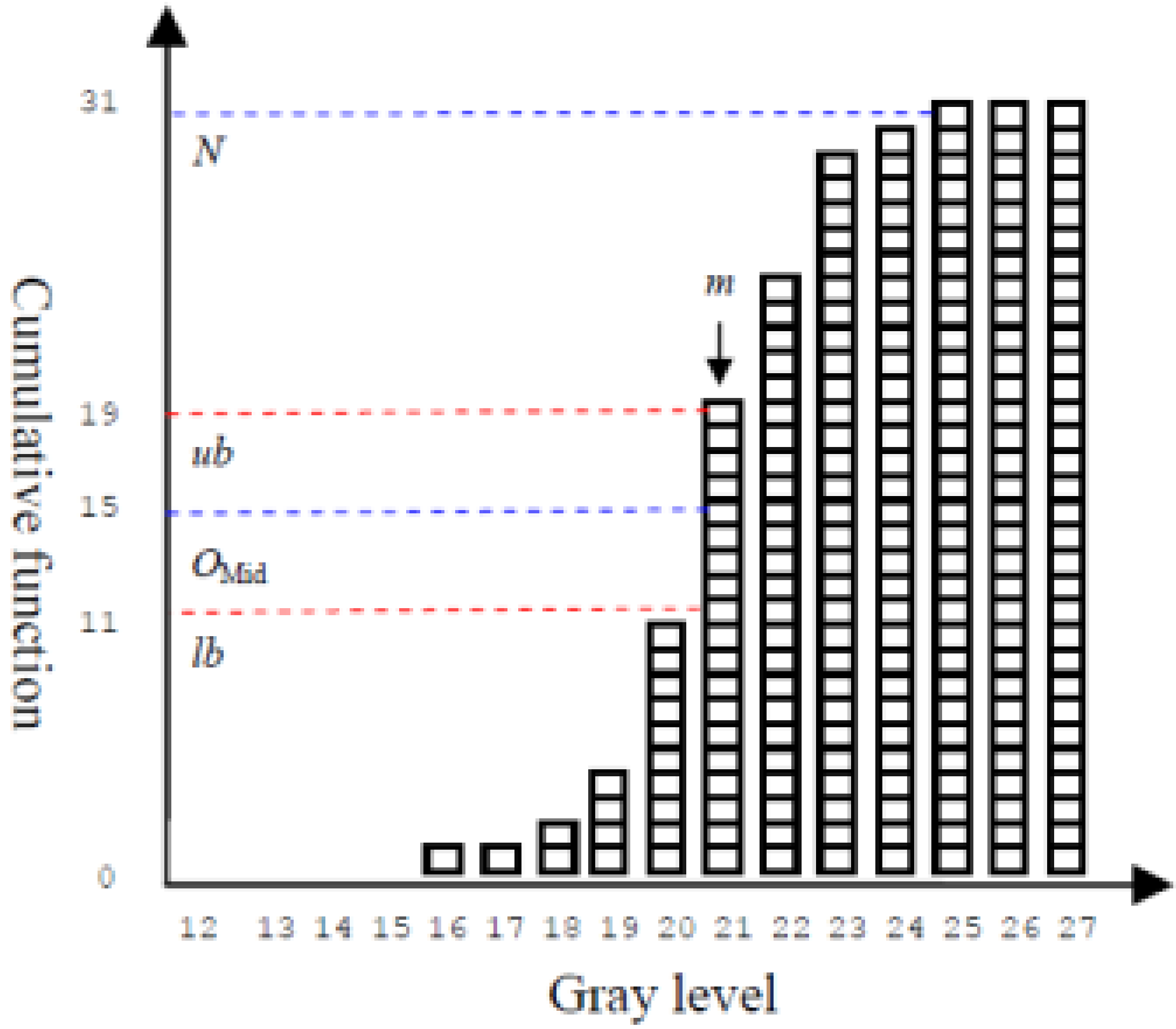

Figure 1. An example of cumulative function of histogram

The data set of D_k contains pixel data within the previous N frames of the k-th frame, as shown in Eq.(2). The next data set of D_{k+1} for (k+1)-th frame is as

$$D_{k+1} = \{d_k, d_{k-1}, \ldots, d_{k-N+1}\} \tag{3}$$

where d_k is the pixel data of the k-th frame. The difference of D_{k+1} and D_k is just d_k and d_{k-N}, which implies D_{k+1} and D_k are highly correlated. Thus, it has high possibility that the medians of the two data sets are equal, which we call median repetition. Consequently, the proposed repetition checking of the median between the two consecutive frames has promise to greatly reduce the median operation in temporal direction.

The temporal median filter with histogram selection and repetition checking is implemented by the following function.

Function: $\{m_{k+1}, lb_{k+1}, ub_{k+1}\}$ = medhist_repchk(D_k, d_k, m_k, lb_k, ub_k)
/* Input: D_k stores the previous N pixel data of the k-th frame. d_k is the pixel data of the k-th frame. m_k, lb_k and ub_k are respectively the median, lb and ub of D_k. */
/* Ouput: m_{k+1} returns the median of D_{k+1}. lb_{k+1} and ub_{k+1} respectively return lb and ub of D_{k+1}. */

insert d_k into D_k and delete d_{k-N} from D_k to obtain D_{k+1}
$\{tf, lb_{k+1}, ub_{k+1}\}$ = repchk(d_{k-N}, d_k, m_k, lb_k, ub_k) //call function repchk(.) for repetition checking
if tf then
    $m_{k+1} = m_k$ // if repetition checking is true
else
    $\{m_{k+1}, lb_{k+1}, ub_{k+1}\}$ = medhist_bnd(D_{k+1}) // if repetition checking is false
end if
return m_{k+1}, lb_{k+1}, ub_{k+1}

Given the data and parameters of the k-th frame, d_{k-N}, d_k, m_k, lb_k and ub_k, the repetition checking algorithm first calculate the parameter of the next frame, lb_{k+1} and ub_{k+1}, according to the relations of the values of d_{k-N}, d_k and m_k. The relation of d_{k-N} and m_k contains three cases: "less than", "equal to" and "greater than". The relation of d_k and m_k also include the same three cases. Thus, the two relations generate nine permutations, which can be utilized to calculate lb_{k+1} and ub_{k+1} from lb_k and ub_k. For examples, when the deleted element d_{k-N} and the inserted element d_k are less than the previous median of m_k, lb and ub are unchanged, i.e. $lb_{k+1}= lb_k$ and $ub_{k+1}= ub_k$. When d_{k-N} is less than m_k and the d_k is equal to m_k, lb is decreased by 1 but ub is unchanged, i.e. $lb_{k+1}= lb_k-1$ and $ub_{k+1}= ub_k$. Similarly, the other seven conditions are used to update the next lower bound and upper bound. Finally, if $lb_{k+1} \le O_{\mathrm{Mid}} \le ub_{k+1}$, the median of the next frame is equal to that of the current frame; i.e., $m_{k+1}= m_k$. The repetition checking is implemented by the following function.

Function: $\{tf, lb_{k+1}, ub_{k+1}\} = \mathrm{repchk}(d_{k-N}, d_k, m_k, lb_k, ub_k)$
/* Input: d_{k-N} and d_k respectively denote the deleted and inserted element for the next data set. m_k means the previous median. lb_k and ub_k are the previous bounds of the cumulative function at the median. */
/* Output: tf returns 1 if the current median is equal to the previous median, otherwise tf returns 0. */

if $d_{k-N} < m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$
if $d_{k-N} < m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k - 1$, $ub_{k+1} = ub_k$
if $d_{k-N} < m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k - 1$, $ub_{k+1} = ub_k - 1$
if $d_{k-N} = m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k + 1$, $ub_{k+1} = ub_k$
if $d_{k-N} = m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$
if $d_{k-N} = m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k - 1$
if $d_{k-N} > m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k + 1$, $ub_{k+1} = ub_k + 1$
if $d_{k-N} > m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k + 1$
if $d_{k-N} > m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$

if $lb_{k+1} \le O_{\mathrm{Mid}}$ and $O_{\mathrm{Mid}} \le ub_{k+1}$, then tf=1 else tf=0
return tf, lb_{k+1}, ub_{k+1}