**K-nearest neighbor smoothing (KNN)**

In a rectangular window, choose the K pixels which are closest in value to the center pixel, and average those.

3×3 window \( K = 6 \)
7×7 window \( K = 24 \)

*Note: All the spatial avg. filters do badly for salt & pepper noise*

---

**Median Filtering**

The median of a discrete sequence \( a_1, a_2, \ldots, a_N \) for \( N \) odd is that member of the seq. for which \( \frac{N-1}{2} \) elements are smaller or equal in value, and \( \frac{N-1}{2} \) elts are larger or equal in value.

1D MF: sliding window w/ odd number of pixels.

In the output sequence, center pixel in window replaced by the median of the pixels in the window.

\[
\underbrace{a_1 \leq a_2 \leq \ldots \leq a_{\frac{N+1}{2}} \leq \ldots \leq a_{\frac{N-1}{2}}}_{\text{\( \frac{N-1}{2} \) elements}} \quad \uparrow \quad \underbrace{a_{\frac{N-1}{2}} \leq a_N}_{\text{\( \frac{N-1}{2} \) elements}}
\]

If the window size is even, median is the average of the 2 values in the middle. Can analyze only a little:

\[
\text{med}\{Kf(j)\} = K \text{med}\{f(j)\}
\]

\[
\text{med}\{K + f(j)\} = K + \text{med}\{f(j)\}
\]

*But:*

\[
\text{med}\{f(j) + g(j)\} \neq \text{med}\{f(j)\} + \text{med}\{g(j)\}
\]

\( \leftarrow \) not additive
Input

3-pt avg filter

3-pt median filter

\[ \text{med}\{1,1,1\} = 1 \]

single

triple

triangle

3\times3 \text{ MF}

\begin{align*}
\text{unit 1D step } \ u(n) &= \begin{cases} 
1 & n > 0 \\
0 & \text{else}
\end{cases} \\
\text{unit 2D step } &= \text{unit corner} \\
\ u(n,m) &= u(n) u(m) = \begin{cases} 
1 & n > 0, m > 0 \\
0 & \text{else}
\end{cases}
\end{align*}

3\times3 \text{ MF takes out corner}

5\times5 \text{ takes out more}

Far from origin, unchanged

\rightarrow \text{ The median filter does NOT affect steps or ramps}

\rightarrow \text{ Pulses less than one-half the window width are gone}

2D Median Filtering

Extend to 2D by using a 2D window of some desired shape

Name some root signals for the 3\times3 MF:

infinite step edge, checkerboard

line of width 2 in any direction
1. **Filter Shape**: can alter shape – cross, or approx. to octagon

One can think of these as an intermediate size of MF, but more profound effect is that cross preserves horiz/vert lines of width 1. Doesn’t help for diagonally oriented lines.

2. **Separable MF**: Filter 1-d vertically then horizontally. This will preserve 2d steps.

3. **Repeated MF**: What happens if you apply 3×3 averaging filter repeatedly? → const. gray

What happens w/ repeated MF? Textured areas may get leveled to an even gray → image looks patchy. But edges still located correctly. → root signal

4. **Sparse MF**: The size or spatial extent of a MF, and the number of pixels used in the computation, are related but not identical.

Sparse 5×5 completely eliminates 3×3 noise blob, whereas a 3×3 square MF (same #ots) only eats away corners.
5. **Weighted MF**:  

\[
\begin{array}{c|c|c|c|c}
1 & 1 & 1 & 1 & \text{(a)} \\
3 & 3 & 3 & 1 & \text{(b)} \\
1 & 1 & 1 & 1 & \text{(c)} \\
1 & 1 & 3 & 1 & \text{(d)} \\
1 & 3 & 1 & 1 & \text{(e)} \\
\end{array}
\]

\[\Rightarrow \text{Filter mask specifies the \# of times a pixel's graylevel is repeated in the ordering}\]

Consider this image: \[\begin{array}{c}
0 & 0 & 0 & 0 & 0 \\
5 & 5 & 5 & 5 & 5 \\
0 & 0 & 0 & 0 & 0 \\
\end{array}\] 3x3 MF: \[\begin{array}{c}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
5 & 5 & 5 & 5 & 5 \\
\end{array}\]  \[\Rightarrow \text{Line is preserved}\]

(b) preserves vertical edges  
(c), (d) preserve diagonal lines  
(e) preserves both 1-pixel wide vert & horiz lines  
ALL of them eliminate single outlier  

6. **Breakdown Value**: of a filter is defined as the percentage of outlier pixels for which the filter is no longer able to remove the outlier noise.

\[\text{Regular MF: breakdown value } = 50\%\]  
\[\text{Weighted MF: smaller breakdown value } \Rightarrow \text{less noise attenuation}\]

7. **Alpha-trimmed Mean Filter**: Mean Filt does poorly for outlier noise, well for Gaussian. Med Filt is other way around. Some advantages of both: Alpha-trimmed mean: remove p pixels from each end of the ordered pixels, take mean of the rest.

\[a_1 \leq a_2 \leq \ldots \leq a_{n-1} \leq \ldots \leq (a_{n-1} \leq a_n)\]
8. Other Order Statistic Filters:

midpoint filter \( = \frac{1}{2}(\min f_i + \max f_i) \)

avg of max & min graylevels in the ordered set

Good filter to remove uniform noise

maximum filter \( = \max \text{ value } \) (remove pepper noise)

minimum filter \( = \min \text{ value } \) (remove salt noise)

Median, Mean, Weighted Median, Midpoint Filters are all unbiased filters: average brightness of the filtered image remains the same.

Max & Min are biased filters.

9. Implementation:

Filter

Midpoint \( A_0 = \frac{1}{2} \quad A_{N-1} = \frac{1}{2} \quad \text{all others zero} \)

Max \( A_{N-1} = 1 \quad \text{all others zero} \)

Min \( A_0 = 1 \quad \text{"} \quad \text{"} \quad \text{"} \)

Median \( A_{N-1} = 1 \quad \text{"} \quad \text{"} \quad \text{"} \quad \text{for } N \text{ odd} \)

Alpha-trimmed mean \( A_i = \frac{1}{N-2p} \) for \( i = p \) to \( N-p-1 \)

mean \( A_i = \frac{1}{N} \) for all \( i \)

Inefficient way to compute mean \( \rightarrow \) don't need sort

Also faster to find min, max individually \( \rightarrow \) don't need to sort the whole set of numbers