Denoising

$I_{uv} = S_{uv} + N_{uv}$

$N_{uv} \sim N(0, \sigma_n^2)$

$$S = [1 \; 1 \; 10 \; 8 \; 3 \; 7 \; 2]$$
$$N = [0 \; -1 \; -2 \; 1 \; 2 \; 0 \; -1]$$
$$I = [1 \; 0 \; 8 \; 9 \; 5 \; 7 \; 1]$$

$\hat{I} = f(I)$

$RMSE(I, S) > RMSE(\hat{I}, S)$

- What is $f()$?
  - Noise is independent
  - Signal is smoother
    - ↳ not random

$$\hat{I}_{uv} = \frac{1}{|N_{uv}|} \sum_{i,j \in N_{uv}} I_{ij}$$

↑ Size of block to average

- Let's say signal is constant...

$$\hat{I}_{uv} = \frac{1}{|N_{uv}|}\left(\sum_{i,j \in N_{uv}} S_{ij} + \sum_{i,j \in N_{uv}} N_{ij} \right)$$

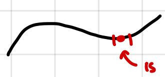
$$\hat{I}_{uv} = S_{uv} + \alpha$$

← new noise w/ smaller distribution

$$\alpha \sim N\left(0, \frac{\sigma_n^2}{|N_{uv}|}\right)$$

$$RMSE(\hat{I}, S) = \sqrt{E(\alpha)} = \frac{\sigma_n}{\sqrt{|N_{uv}|}}$$ which is smaller than σ_n

- We can do this when image is not constant, too.



↖ is approx avg of neighbors, so doesn't get that affected

$$S = [1 \; 2 \; 3 \; 4 \; 5 \; 6 \; 7 \; 8 \; 9 \; 10]$$
$$N = [0 \; 1 \; 1 \; -1 \; 1 \; 0 \; 0 \; 1 \; 1 \; 1]$$
$$I = {}^{(1)}[1 \; 3 \; 4 \; 3 \; 6 \; 6 \; 7 \; 9 \; 10 \; 11]^{(11)}$$ pad

$$N \sim \begin{cases} 1 & p = 1/3 \\ 0 & p = 1/3 \\ -1 & p = 1/3 \end{cases}$$

Filter: $[1/3, 1/3, 1/3]$ (convolution) (box filter)

↑ Linear filter

$$\hat{I} \; [5/3 \; 8/3 \; 10/3 \; 13/3 \; 15/3 \; 19/3 \; 22/3 \; 16/3 \; 10 \; 32/3]$$

$RMSE(S, I) = .84$

$RMSE(S, \hat{I}) = .57$ } yay!

In 2D: $\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$

Classical filter for removing noise: $[1/4, 1/2, 1/4]$

↑ Gaussian shape

filter: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ detects edges (HPF)

Code for linear filtering

1D : $\sum_{n=-\infty}^{\infty} f[k]\,h[n-k]$ but we don't need to flip! (so we are doing correlation)

$$y[x] = \sum_{k=-\infty}^{\infty} f[x]\,h[x+k]$$

```
def filter1d(f_in, h):
    f = np.zeros((len(h)-1)/2).tolist() + x + np.zeros((len(h)-1)/2).tolist()   //zero padding
    y = x  (to initialize)
    for x in range(len(f)):
        y[i] = 0
        for k in range(len(h)):
            y[x] += f[x]·h[x+k]
```