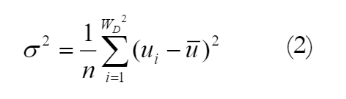
Variance:

In image processing, variance filter is often used for Highlighting edges in the image by replacing each pixel with the neighbourhood variance.

C:\Users\Franck\Desktop\var2.png

Equation\_1: Variance filter is the square of the standard deviation, where u(x) is image intensity at the location x(x1, x2), **σ** represent the standard deviation, W is size of a filtering window, u(x-q) is the set of all pixels within the filtering window and q is an element of the set W.



Equation\_2: Equation used for compute the standard deviation.

Where n is the total of pixels within the window (W), and ū is the mean of all the pixels within the window (W).

This filter is implemented in imageJ through the rankfilters class in java. <img src="https://github.com/imagej/imagej1/blob/ab7633f0f238ba08f65cb1ef5e104dba3d3f68af/ij/plugin/filter/RankFilters.java " alt="code" />.For variance algorithm, according to the input image and the size of the kernel, it will not react in the same way. If the kernel’s radius size is less than 2 (5x5), it will compute the sum over all the pixels, whereas for a kernel’s radius size greater than 2, the sum won’t be calculated. In that case this sum is calculated for the first pixel of every line only. For the following pixels, it’ll add the new values and subtract those that are not in the sum any more. This way, the computational time is then reduced. Once, the kernel reaches the end of the thread, it start over at the next line until the end of the input image. It’s notable that the variance algorithm is closely related to the mean algorithm.

In application, this algorithm works by using one “window” defined here by a circular kernel, which slides, entry by entry until the end of the signal. It can process through rows or columns.

Consider a matrix 

1. Consider a matrix 

Now pad the matrix with extending border values outside. 

1. In this example we’re using a small kernel of size 3 by 3 and we’re not considering boundary issues, starting from matrix M(1,1) place the “Window”, the value to be changed is the middle element 1, 
2. Sort data values from smallest to highest in the “Window”,  , here the middle value remains 1.
3. The procedure is then repeated by sliding the window to the next position M(1,2) pixel by pixel until the end.

 🡪  🡪 

1. The output matrix is 

In the previous example, because there is no entry preceding the first value, the first value is repeated (as is the last value) to obtain enough entries to fill the window. What effect does this have on the boundary values? There are other approaches that have different properties that might be preferred in particular circumstances: – Avoid processing the boundaries, with or without cropping the signal or image boundary afterwards. – Fetching entries from other places in the signal. With images for 4 example, entries from the far horizontal or vertical boundary might be selected. – Shrinking the window near the boundaries, so that every window is full. What effects might these approaches have on the boundary values?

Get median of values within kernel-sized neighborhood. Kernel size kNPoints should be odd.

This method is simple, moreover it’s characterised by low computational complexity compared to other methods (Cany, Sobel).

However it’s not devoid of weakness because of its low resistance to noise. Indeed the impulse and Gaussian noise significantly decreases quality of edge detection [citation].

Example:

Out-of-image pixels are set to the value of the nearest edge pixel.

1. Consider a matrix 

Now pad the matrix with extending border values outside. 

1. In this example we’re using a small kernel of size 3 by 3 and we’re not considering boundary issues, starting from matrix M(1,1) place the “Window”, the value to be changed is the middle element 1, 
2. Use the formula to compute the new value, variance = 0.47( ū = 1.44, **σ** = 0.68 )

Change the value of 1 by 0.47,  🡪 

1. The procedure is then repeated by sliding the window to the next position M(1,2)

 pixel by pixel until the end.

1. The output matrix is 

The kernels used in the different filters are partially out of bound of the image when centered on pixels near the boundaries of an image, and in these case there are less pixels available to compute the central value. There are multiple ways to handle these cases.  The simplest way is to ignore each case where the kernel is out of bound, resulting in an output image that is cropped compared to the input image. Another method consist to attribute values to out of bound pixels, by giving them the value of the nearest in bound pixels, thus creating enough values to realise the convolution. The first examples consist to pad the matrix with extending border values.

1.  🡪
2.  🡪

A third method is used to bypass boundaries issues by shrinking the kernel near the boundaries, in order to fill the kernel.



In image processing, variance filter is often used for highlighting edges in the image by replacing each pixel with the neighbourhood variance.

![GitHub Logo](https://github.com/fsoubes/FilterRank/blob/master/images/var2.png)

### Equation\_1: Variance filter is the square of the standard deviation, where u(x) is image intensity at the location x(x1, x2), σ represent the standard deviation, W is size of a filtering window, u(x-q) is the set of all pixels within the filtering window and q is an element of the set W.

![GitHub Logo2](https://github.com/fsoubes/FilterRank/blob/master/images/var3.png)

### Equation\_2&3: Equation used for compute the standard deviation. Where n is the total of pixels within the window (W), and ū is the mean of all the pixels within the window (W).

This method is simple, moreover it’s characterised by low computational complexity compared to other methods (Cany, Sobel).

However it’s not devoid of weakness because of its low resistance to noise. Indeed the impulse and Gaussian noise significantly decreases quality of edge detection [^Fab2011].

<img src="http://www.monsite.com/image.png" alt="Comment mettre un lien menant à une autre page" />