Segmenting Homogeneous Regions in Images using Variance Wells

Satyaj Bhargava[2], John Lorence[2], Ben Cohen[2], Minjie Wu[2,3], Howard Aizenstein[2,3], George Stetten[1,2].

[1] Robotics Institute, Carnegie Mellon University
[2] Department of Bioengineering, University of Pittsburgh
[3] Department of Psychiatry, University of Pittsburgh

Carnegie Mellon University Robotics Institute
Technical Report - CMU-RI-TR-24-33

Corresponding author: George Stetten, stetten@andrew.cmu.edu

Abstract. We present a method to identify small regions of relative homogeneity in an *N*-dimensional image based on local variance. We call these regions "variance wells" or "vWells," because they surround and separate local minima in the variance of image intensity. VWells fall into a class of entities in computer vision known as "super-pixels," generally irregularly shaped small homogenous regions that can be combined to segment objects in the image while preserving sharp boundaries. Computing vWells requires no parameters and is not iterative, having a computation time proportional to the total number of pixels. The vWells in an image can form the nodes of a graph whose edges connect adjacent pairs of vWells. When using such a graph to cluster vWells for segmentation, similarity between adjacent vWells can be computed using well established statistical methods based on the mean and variance of the pixels within each vWell. We use such a graph of vWells to segment arteries in 2D MRI images of the brain by finding the optimal path between two manually placed starting points, and then performing a region-growing algorithm to include adjacent vWells up to the vessel boundary. In this way, we demonstrate that vWells may form a useful preprocessing step to simplify data for further analysis by reducing noise and reducing the number of primitives compared to individual pixels, while preserving boundaries. Our software implementation is available (see Appendix 1).

Keywords: variance wells, vWells, super-pixels, image analysis, segmentation, graph theory.

1 Formation of vWells.

Variance wells, or *vWells*, are an adaptation of Descending Variance Graphs (DVGs), which we previously developed to segment objects in medical images. The initial method for finding vWells is the same as for forming DVGs, which we briefly describe here. We refer the reader to a more detailed description in [1].

The procedure for finding vWells begins as follows: Given an *N*-dimensional input image, we first calculate a *variance image*, whose pixels indicate local variance of the input image intensity within a small kernel centered on every pixel, (3x3… in *N* dimensions, though this could be enlarged). The variance image has low pixel values within regions of relative homogeneity in the input image and high values where the kernel spans regions that are heterogenous, including boundaries between regions of differing intensity. To avoid any loss of precision in computing variance (and mean) we use integer mathematics, as described in Appendix 2.

We then find vWells in the variance image by building a graph whose nodes are the pixels in the variance image (see Fig. 1). Specifically, we create directed edges between every pair of orthogonally connected pixels pointing to that pixel in the pair with the lower variance value. (In the case of pixels with identical values, we point arbitrarily and consistently in a particular direction). As can be seen in a sample image in Fig. 1, three distinct relatively homogenous regions are shown in different

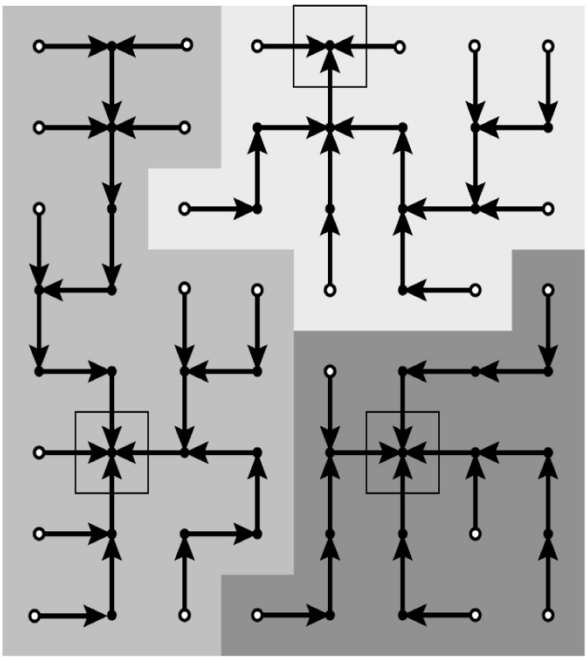


Fig. 1 Directed edges (arrows) between pixels (circles) representing decreasing intensity variance of spheres centered on those pixels. Three disjoint trees form 3 patches (different level of gray) each with its root (square) and terminal nodes (open circles).

shades of gray and result in the formation of three disjoint trees, each of whose root is a local minimum in variance. Each tree can be searched up from the root to find the set of pixels in the corresponding vWell. Each vWell is, in a sense, a local "well" of variance, which tends to be internally relatively homogenous and preserve boundaries in the image between adjoining vWells. VWells do not generally represent entire anatomical structures, since they tend to form distinct subregions even with relatively homogenous regions due to minor intensity variations, but they can be joined with similar adjacent vWells to segment entire structures. We do this by forming a second level graph, this time between adjacent vWells. In this graph, vWells are the nodes and the edges are a statistical metric of the difference, or *distance*, between adjacent vWells. We have chosen the Welch's t-test, based on the mean and variance of the pixel intensity in each vWell, as this metric, since it allows for differing numbers of samples (pixels) in adjoining vWells. More specifically, we use the absolute value of the Welch's t-test as a metric of the distance between nodes along a given edge, since these edges are undirected. Lower t-test values indicate a statistical likelihood that two vWells belong to the same population, while higher t-test values suggest they belong to different populations.

2 Finding the shortest path through the graph between manually selected vWells

Our present application is to segment arteries magnetic resonance images (MRI) of the brain. Our ultimate goal is to operate in 3D on such structures, but in the present report we demonstrate our methods on 2D slices derived from those 3D MR images. In such slices, a longitudinal cross-section of an artery will typically produce a stripe-shaped object that terminates at both ends where the vessel enters and leaves the slice (see Fig. 4). We identify the desired artery to segment by manually placing markers within the artery near either end of the visible section of the vessel as well as along it. These markers identify particular vWells within the object. We then find the shortest path between these vWells, connecting adjacent nodes (vWells) in the graph using Dijkstra's algorithm [2], where *distance* between pairs of nodes is defined above. The vWells along this path tend to stay within the artery even for a curved vessel, because similar neighbors along the vessel present the shortest cumulative distance. This process requires no explicit thresholds, but rather depends on the statistical similarity of the vWells belonging to the artery.

Dijkstra's algorithm is "greedy" in the sense that it begins at a starting node and makes choices to minimize the cumulative distance as it explores outward along edges from node to node. These choices may not all be globally optimal in terms of finding the path with the least cumulative distance to the end point. In fact, the algorithm may head off initially in the wrong direction completely and thereby miss the shortest path. To avoid such problems, we add a bias along the geometric direction between the starting and ending node, using a vWell's root voxel to define its geometric location. This heuristic is commonly used in algorithms to optimize routes along roads in maps [3] and assumes that the optimal path initially heads out from its starting point in a direction towards the ending point, not away from it. We adjust the weight of this bias empirically.

3 Region Growing around the shortest path.

Once a connected set of vWells is found within the artery along the shortest path through the graph, we apply a region-growing algorithm to fill the region of the artery surrounding the initial path. At this point, it is useful to adopt the terminology of *background* and *blob* (sometimes also called *foreground*). In the ideal solution for a segmentation, the *blob* corresponds exactly to the object being segmented and the *background* corresponds to a region immediately surrounding the object. One may want to limit that background as much as possible to the region immediately surrounding the blob, since there may be other objects in the image that should not be included in the background. VWells offer a convenient way to provide such an appropriately restricted background, by simply having the background consist of a layer one vWell thick, i.e., the vWells that are immediate neighbors to those in the blob (see Fig. 2a).

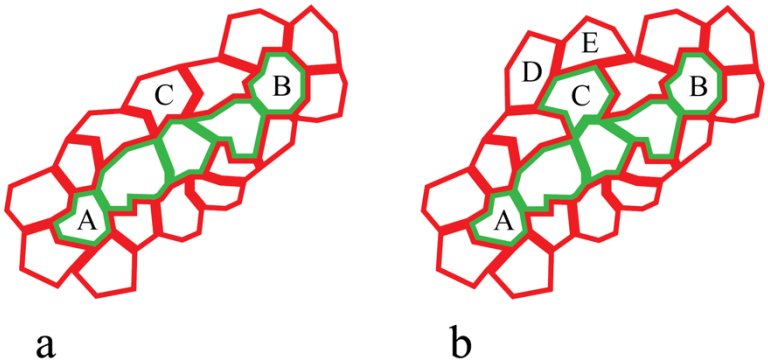


Fig. 2 (a) Original blob of vWells (green) forming the shortest path between manually placed landmarks identifying vWells A and B. Background vWells (red) consist of the neighbors to the vWells in the blob. **(b)** Background vWell C is identified as the best candidate to be transferred to the blob, and two new neighboring vWells, D and E, added to the background to surround the newly enlarged blob.

The region growing algorithm operates by iteratively transferring vWells from the background to the blob. At each iteration, we choose the vWell that increases the difference metric between the blob and the background the most, using the same distance metric as we previously used between individual vWells, but now between sets of vWells. We then add vWells to the background to surround the new vWell in the blob (see Fig 2b). As we saw before with Dijkstra's algorithm, this constitutes a *greedy* algorithm, i.e., one that is likely, though not guaranteed, to approach an optimal solution by choosing what appears at each step to be the

optimal choice. When a final segmentation has been achieved, we would expect the difference metric to be at a maximum, after which adding incorrect background vWells to the blob would start to decrease the difference metric. Whether this is a global maximum is, unfortunately, not assured.

4 Finding objects 1 pixel wide

A problem arises with vWells for structures that are only 1-pixel wide. This is demonstrated in 2D in Fig. 3a, which shows a stripe of pixels with value 2 on a background of 0's. As a 3x3 kernel used to find sample variance passes over the stripe, the resulting variance shows no local minimum where the kernel is centered on the stripe (red). This makes sense, since the set of numbers in the kernel are the same for all 3 cases shown. Without a local minimum in variance, vWells will not form for this 1-pixel wide stipe. The problem can be addressed by double sampling the image using linear interpolation, as shown in Fig. 3b. Now the vertical stripe of value 2 is bordered by two linearly interpolated stripes of value 1. As the kernel passes over the stripe, the variance now has a local minimum where the kernel is centered directly on the stripe (red). Thus, a vWell can form within the stripe. This concept generalizes to any objects whose width is on the order of 1 pixel, in any number of dimensions.

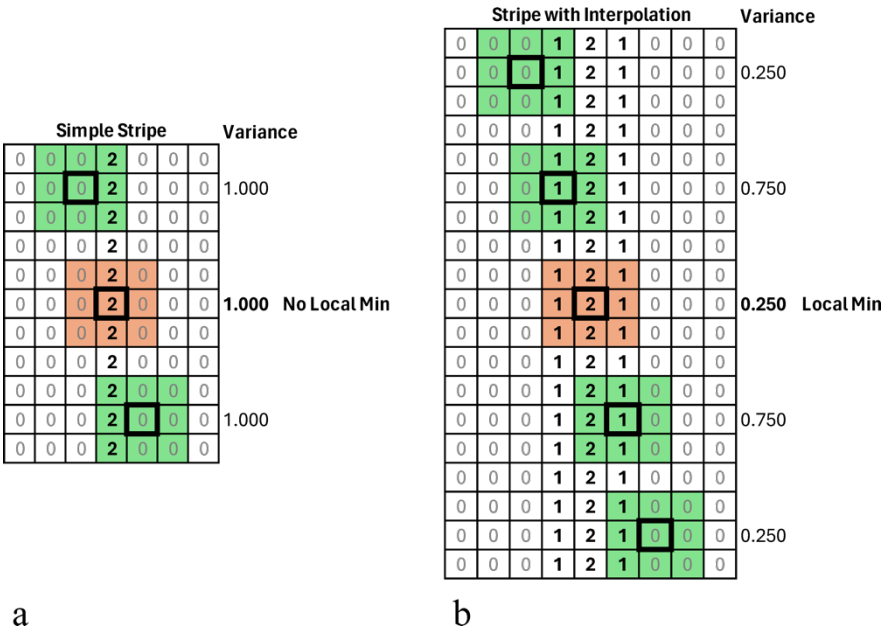


Fig. 3 (a) 2D image containing a 1-pixel wide vertical stripe of value 2, showing that a 3x3 kernel for computing variance yields no local minimum (bold). **(b)** Linear interpolation of stripe yields a local minimum for variance.

5 Demonstration of vWells on a 2D MR Image of a Blood Vessel

In this section we demonstrate the formation of vWells and their use in segmenting arteries using the methods described above, in 2D slices from 3D time-of-flight (TOF) MRI of the brain. In these images, arteries appear bright because the TOF pulse sequence energizes blood flowing into the 3D brain region being imaged. From that 3D region, we have chosen a 2D slice containing a section of artery that includes at least one branch point (Fig. 4a). Since parts of the artery approach a width of 1 pixel, we started by double-sampling the image using linear interpolation (Fig. 4b). We used this double-sampled image for all subsequent processing described below.

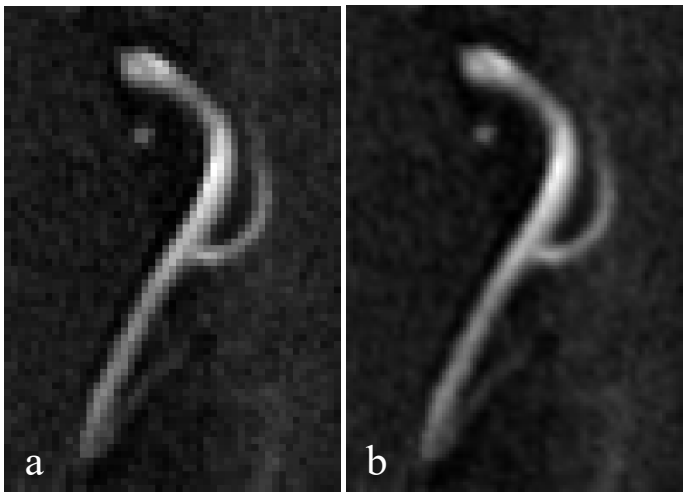


Fig. 4 (a) 2D slice from an MRI image showing a section of an arterial tree. **(b)** Double-sampled version of same image.

We computed the *variance image* using a 3x3 kernel. As can be seen in Fig. 5a, the variance image is bright where the kernel spans the boundaries of the artery, since variance is high for heterogeneous samples of pixels. Along the centers of the artery, the variance image is dark, since here the 3x3 kernel occupies areas of relative homogeneity, even where the width of vessel in the original image approaches a width of just 1 pixel, thanks to our double sampling. These dark areas provided the local minima which formed the roots of vWells (see Fig. 1). We applied the algorithm described in Section 1 to find the set of vWells corresponding to the variance image.

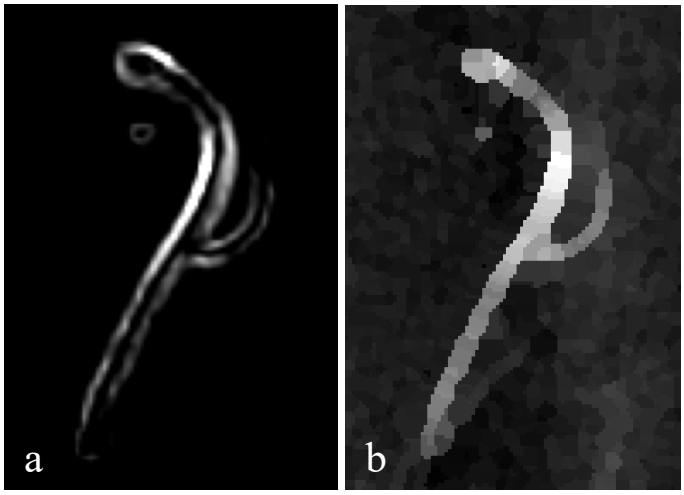


Fig. 5 (a) Variance image corresponding to the double-sampled input image (Fig. 4a). **(b)** Image of vWells, with each vWell's pixels set to the mean of all the pixels in that vWell.

To display vWells as an image, we set the value of all pixels in a given vWell to the mean value of those pixels. The result is shown in Fig. 5b. The vWells appear as small irregularly shaped patches of homogeneous intensity, each patch displaying the mean of that vWell's constituent pixels. Compared to the source image (Fig. 4b), we see that the image of vWell means preserves the structure of the vessel, with sharpened boundaries and lowered noise. In the process, vWells lower the degrees of freedom in the image from the number of pixels to the number of vWells. Finally, vWells preserve proximity information in terms of which vWells are adjacent, allowing us to build graph structures whose nodes are the vWells and whose edges are the distance metric between adjacent vWells described in Section 1. Thus, vWells can be seen as a form of lossy compression, preserving the important information in the image, while reducing the dimensionality of the data.

A human operator identified which portions of the arterial tree to segment by manually placing markers at the two ends of the portion of the main artery visible in the image. As described in Section 2, the artery enters and leaves the 2D slice at these points. A third marker was also entered manually along the artery. We found the shortest path along the markers through nodes in the graph of vWells as described in Section 2. The result is shown Fig. 6 as a line drawn through the set of connected vWells. The location of each vWell along the line was taken to be its root pixel. It should be noted that the root pixel is not necessarily the central pixel. However, in the example shown here, the shortest path does take a central route along the vessel, staying within it around the curve, rather than taking the shortest path geometrically. This is because "distance" is computed using the statistical similarity of the vWells within the artery. Notice that the branch to the right is ignored, since those vWells do not lie on the shortest path along the markers.

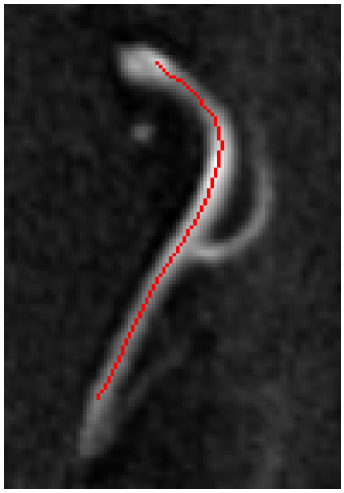
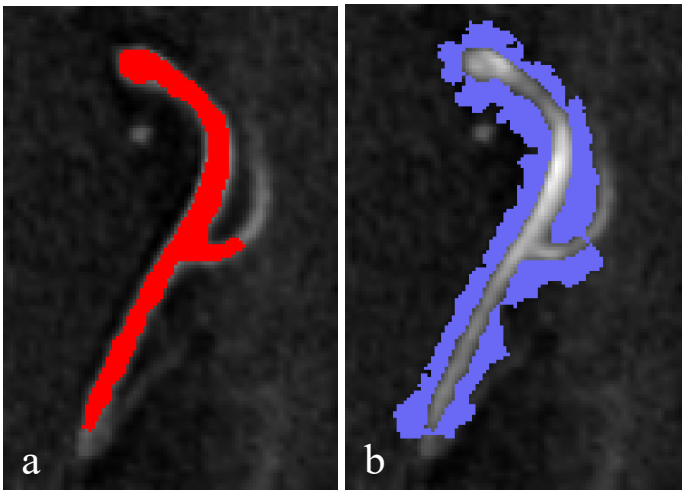


Fig. 6 Shortest path along manually placed markers connecting vWells.



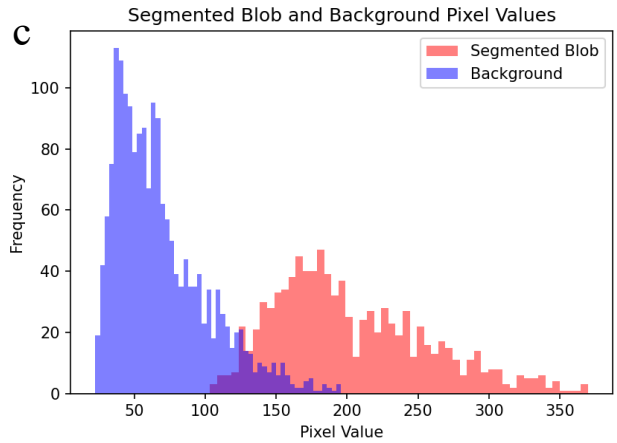


Fig. 7 Shown after application of the region-filling algorithm are **(a)** the set of vWells in the blob, **(b)** the set of vWells in the background, and **(c)** a histogram of constituent voxel intensities in each.

We next proceeded with the region-growing algorithm described above in Section 3. The set of vWells along the path were considered the initial *blob* and their immediate neighbors constituted the initial *background*. The algorithm iterated by adding vWells from the background to the blob, each time choosing the vWell that increased the difference between the blob and background the most. At each step we updated the background by adding new vWells to surround the latest vWell transferred to the blob (see Fig. 2). The algorithm proceeded until the next vWell to add to the blob decreased the difference metric between the blob and the background.

The resulting blob and background are shown in Figs. 7a and 7b, respectively. The blob is seen to have filled the artery initially identified by the markers, with a small amount of spreading into the branch to the right. This is discussed in the next section. Fig. 7c shows a histogram of the final segmentation, whose t-test had reached a maximum during the region growing.

6 Discussion

We have reported here on a new method of segmentation based on graph structures using vWells as the nodes and relations between neighboring vWells as the edges, demonstrating the method in 2D on arteries in MRI images of the brain. We have shown how vWells serve as a useful preprocessing step for graph-based analysis of images by reducing the number of variables and removing noise while preserving sharp boundaries. The graph-based methods we have introduced permit creation of an initial path through an artery as well as region growing around that path to optimize a segmentation of the artery relative to a boundary region immediately surrounding it. By using statistical differences within the desired object and an evolving local background, we avoid the use of intensity thresholds common in segmentation. Nothing in our method limits it to 2D, and we are presently extending it to 3D and validating it on vascular structures in a set of MRI scans of the brain.

One potential complication involves deciding when to terminate the flood-fill operation. This is seen in Fig. 7a, where the right-hand branch begins to be filled. In 3D, this will become more troublesome as arteries no longer are limited to their intersection with a given slice. Clearly, more high-level information about expected shape and topology will be needed to specify a desired segmentation. Furthermore, the initialization of the method could be further automated by using an atlas of expected anatomical structures.

The code used to produce our demonstration is available on Google Colab as well as GitHub (see Appendix 1). By making the code accessible through both platforms, we aim to support reproducibility and encourage further exploration.

References

[1] G. Stetten, C. Wong, V. Shivaprabhu, A. Zhang, S. Horvath, J. Wang, J. Galeotti, V. Gorantla, and H. Aizenstein, “Descending Variance Graphs for Segmenting Neurological Structures,” *3rd International IEEE Workshop on Pattern Recognition in Neuroimaging*, Philadelphia, PA, June 22-24, 2013.

[2] E.W Dijkstra. A note on two problems in connexion with graphs. *Numer. Math*. 1, 269–271 (1959). https://doi.org/10.1007/BF01386390

[3] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136. https://ieeexplore.ieee.org/document/4082128

Appendix 1 – Demonstration in Python

The code used to produce our demonstration is available on Google Colab as well as GitHub

Google Colab Access:

To access and run our algorithm via Google Colab, follow these steps:

1. Open Colab Notebook using the following link: https://drive.google.com/drive/folders/1f6Rcwyrfxy9aYtfnbPg5SOgBCDtaJe59?usp=drive_link.
    a. To view the code only, simply click on the. ipynb file to see the code and pre-loaded output.
    b. To run the code or make changes, proceed with the following steps.

2. After opening the link:
    a. Click on the header labeled "VIA_vWellAlgorithm".
    b. Then, navigate to "Organize > Add Shortcut > My Drive > Add" to save a shortcut in your Google Drive.
    c. Now, simply open the Colab notebook (.ipynb) from your Google Drive and run the code. The settings can be modified as needed, and the algorithm will execute accordingly.

GitHub Repository Access:

The code is also available through our GitHub repository: https://github.com/SatyajBhargava/2024-2D-vWell-Algorithm-.git.

This repository contains the full source code and additional documentation for running the algorithm in different environments. You can clone the repository or download as a ZIP and follow the instructions provided in the README file for setup and usage.

Appendix 2 – Integer Mathematics to Compute Mean and Variance

In computing local variance with the kernel described above (3x3… in N dimensions), we need to be careful in defining our numerical technique. The mean $\bar{x}$ of a set of n numbers x_i is defined as

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Since n is constant in our kernel for a given number of dimensions N ($n = 9$ for $N = 2$, $n = 27$ for $N = 2$, or generally $n = 3^N$) we can simply use the sum of the numbers instead of the mean, and it will be proportional to the mean, as denoted in the following equation

$$\sum_{i=1}^{n} x_i = n\bar{x}$$

Assuming x_i is in an integer (pixel intensity in our case), $n\bar{x}$ has the added advantage that it is an integer as well.

The process of computing the variance of n numbers x_i in a population is generally described as first computing the mean and then summing the squared difference between each number and that mean and dividing by n, or

$$variance = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

In practice, variance is often computed by keeping a running computation two numbers: the sum of the squares of x_i and the square of the sum of x_i. Variance can then be computed from these two numbers by

$$variance = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \frac{\sum_{i=1}^{n}(x_i)^2}{n} - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n^2}$$

The computation can cause problems for floating point representations, because it depends on a relatively small difference between potentially quite large numbers. Therefore, by multiplying the above equation by n^2, we can avoid division and use integer math to avoid any loss of precision. We use $n^2 \times variance$ instead of $variance$, and again, since *n* is fixed, we can compare magnitudes effectively.

$$n^2 \times variance = n \sum_{i=1}^{n} (x_i)^2 - \left(\sum_{i=1}^{n} x_i \right)^2$$

Thus, we simply need to keep running sums of x_i an $(x_i)^2$, from which $n\bar{x}$ and $n^2 \times variance$ can be computed at any time, with all operation avoiding division and preserving full precision as integers.

Our number format must accommodate the largest possible expected values. For 3x3… kernels of up to 4 dimensions (81 pixels) with up to 12-bit pixel intensity resolution, the maximum *n* times the sum of $(x_i)^2$ would be 81 x $(2^{12})^2$ = 15,752,961. The maximum sum of the x_i quantity squared would be quite a bit larger (81 x 2^{12})^2 = 1,275,989,841. These numbers are always positive, assuming the original image pixels are unsigned integers. Thus, we can use unsigned long integers consisting of 4 bytes (2^{32} = 4,294,967,296), we can handle any of these possible numbers, though of course we would expect real images to produce lower numbers. If the image pixels are signed integers, we can use signed long integers and still have room to spare (2^{31} = 2,147,483,648),