Advanced Neighborhood Operation Based Image Zooming In

Dr.Mithaq N. Raheema

Received on: 16/9/2010
Accepted on: 3/3/2011

Abstract

Perhaps the most interesting feature of digitally encoded image data is that the image can be analytically manipulated. An entire field of computer graphics, named image processing, has emerged as a result of this capability. Once the data is numerically represented and stored in a manner that corresponds to the image itself, the information can be manipulated by a computer in order to compress the image, correct defects, enhance its qualities, geometrically transform it, perform measurements, detect patterns and objects, and manipulate it in many other forms. This paper provides an overview of the types of image processing operations and introduces an advanced algorithm to enlarge an image to get a closer look. This algorithm changes the magnification of the image and displays the new view in a new figure. It determines the pixel's 3-by-3 neighborhood at the output image, distributes the value of the center pixel in the input matrix to the entire pixel's neighborhood corresponding to it, and filtering the enlarged image by median. This proposed algorithm is tested with many image classes and the results show its very good ability to perform the image zooming in. Some preprocessing and post processing is done automatically which improve this algorithm and give its advantages over the traditional zooming operations. The programming language MATLAB is used to realize the proposed algorithm.

Keywords: Image zooming in, pixel's neighborhood, median filter.

تكبير الصور بالاعتماد على عملية التجاور المتقدمة

الخلاصة

لعل الميزة الأكثر أهمية في الترميز الرقمي للبيانات الصورية هي أن الصورة يمكن معالجتها تحليلياً . ونتيجةً لهذه القابلية فقد نشأ حقل تام لصور الحاسوب يدعى بمعالجة الصور. بعد تمثيل البيانات عددياً وخزنها بطريقة مطابقة للصورة نفسها، فأن هذه المعلومات يمكن أن تعالج بواسطة الحاسوب وذلك من أجل ضغط الصورة، أصلاح العيوب، تحسين الصفات، تحويلها على نحو هندسي، إجراء القياسات، كشف الأنماط والمواضع، ومعالجتها في أشكال أخرى عديدة. يعطي هذا البحث نظرة عامة على أنواع عمليات معالجة الصور ويقدم مخطط حسابي متقدم لتكبير الصورة للحصول على نظرة أقرب وعرضها بمربع صورة جديد. هذا الإجراء ينشر قيمة كل نقطة ضوئية في الصورة الداخلة إلى كل النقاط الضوئية المجاورة له بمربع 3 في 3 في مواقع الصورة الخارجة، مع عمل ترشيح للصورة المكبرة باستخدام طريقة المتوسط. تم اختبار هذا المخطط المقترح بأصناف صور متعددة وبينت النتائج قابليته الجيدة في تكبير الصور. بعض المعالجات السابقة و اللاحقة يتم القيام فيها تلقائياً في المخطط لتحسين الإجراء وتميزه على عمليات التكبير التقليدية. تم استخدام لغة MATLAB البرمجية لتحقيق المخطط المقترح.

* Electrical & Electronic Engineering Department, University of Technology/Baghdad

https://doi.org/10.30684/ etj.29.4.13
University of Technology-Iraq, Baghdad, Iraq/2412-0758

1-Introduction

In last few years, the development and commercial availability of increasingly powerful and affordable digital computers has been accompanied by the development of advanced digital signal and image processing algorithms for a wide variety of applications such as noise reduction, telecommunications, radar, sonar, video and audio signal processing, pattern recognition, geophysics explorations, data forecasting, and the processing of large databases for the identification, extraction and organization of unknown underlying structures and patterns [1]. Thus, digital signal processing techniques have played a key role in the expansion of electronic products for everyday use, especially in the field of audio, image and video processing. Nowadays, digital signal is used in MP3 and DVD players, digital cameras, mobile phones, and also in radar processing, biomedical applications, seismic data processing, etc [2]. A digital image is an image that is recorded in precise numbers (binary codes as 0 or 1) in a digitized code. It is usually saved as a 640x480 pixel resolution using the JPEG (Joint Photographic Expert Group) compression format by most digital devices [3]. A digital image consists of a set of numeric values representing image brightness. The resulting data array can be transmitted as data, and manipulated numerically in order to analyze and enhance the information contained in the data [4]. This section discusses the generic image processing functions.

Pixels: These are the small units or blocks that comprise a digital image, which may or may not be individually visible, depending on the size of the image [3].

Zoom Function: The zoom function (optical or digital) generally refers to magnification of an image [3].

The Overall Magnification is the product of the lenses (Objective*Ocular) and the distance over which the image is projected. Human eye is only capable to discriminate ¼ mm (M= distance* Objective*Ocular/250 mm). According to the Abbe rules the magnification capable to enlarge an object from ¼ μm to ¼ mm to be seen by human vision is 1000x [5].

2. Characteristics and Types of Image Operation

There is a variety of ways to classify and characterize image operations. The types of operations that can be applied to digital images to transform an input image **a[m,n]** into an output image **b[m,n]** (or another representation) can be classified into three categories as shown in Table 1 [6]; and this is shown graphically in figure 1.

2.1. Pixel Point Operations

Each pixel in the 2-dimensional plane is located at coordinates x, y, where x is the pixel's position along the horizontal pixel line and y is its vertical position. The letter "**a**" represents input and the letter "**b**" is used for output. Subscripts are used to denote several input or output images if more than one is present. The letter "**M**" represents an operation on the input image or data set, which generates an

output image or data set. The general equation for a pixel point process is:

b(x,y) = M[a(x,y)] …… (1)

M, often called the mapping function, converts brightness values in one or more input data sets into brightness values in the output data set. **M** can be any mathematical or logical operation. The following restrictions are placed on the mapping function **M** for pixel point operations:

1. The spatial location of the output pixel must be the same as that of the input pixel.
2. If more than one input pixel is considered in the mapping function (as is the case in processing multiple images), then the images must be symmetrical and the pixels must be at same spatial location [4]. Figure 1.a shows the notation used do designate the elements of a pixel point processing operation.

2.2. Pixel Group (Neighborhoods) Operations

A second type of image operations takes into account neighborhood conditions. In pixel point operations, only one input value is used to determine each output value, and both input and output must be at the same coordinates. The pixel-to-pixel mode of operation, and the constraint that both pixels must be at the same spatial location, simplify the problem. However, pixel point operations cannot be used to alter spatial details within the image [4]. There are several types of template operations that are more easily implemented in terms of neighborhood operations [7].

Pixel group operations take into consideration the pixel values in an area adjacent to the point under consideration. Usually, assume that the target pixel is at the center and that the adjoining pixels, the neighborhood, form some kind of pattern around the target. The target and its associated neighborhood pixels are sometimes called a ***kernel*** [4].

Types of Neighborhoods

Neighborhood operations play a key role in modern digital image processing. It is therefore important to understand how images can be sampled and how that relates to the various neighborhoods that can be used to process an image. In most cases, images are sampled by laying a rectangular grid over an image [6].

Local operations produce an output pixel value **b[m=mo,n=no]** based upon the pixel values in the neighborhood of **a[m=mo,n=no]**. Some of the most common neighborhoods are the 4-connected neighborhood, the 8-connected neighborhood, and the 24-connected neighborhood, which are called as 1 by 1, 3 by 3 and 5 by 5 kernels, respectively. Figure 2 shows some typical neighborhood patterns used in pixel group operations.

Neighborhood or Block Processing:

Certain image processing operations involve processing an image in sections, called blocks or neighborhoods, rather than processing the entire image at once. There are several functions can be used to implement image processing algorithms as a block or neighborhood operation.

Sliding Neighborhood Operations:

A sliding neighborhood operation is an operation that is performed a pixel at a time, with the value of any given pixel in the output image being determined by the application of an algorithm to the values of the corresponding input pixel's neighborhood. A pixel's

neighborhood is some set of pixels, defined by their locations relative to that pixel, which is called the center pixel. The neighborhood is a rectangular block, and as one move from one element to the next in an image matrix, the neighborhood block slides in the same direction.

Figure (3) shows the neighborhood blocks for some of the elements in a 6-by-5 matrix with 2-by-3 sliding blocks. The center pixel for each neighborhood is marked with a dot.

Determining the Center Pixel: The center pixel is the actual pixel in the input image being processed by the operation. If the neighborhood has an odd number of rows and columns, the center pixel is actually in the center of the neighborhood. If one of the dimensions has even length, the center pixel is just to the left of center or just above center. For example, in a 2-by-2 neighborhood, the center pixel is the upper left one. For any m-by-n neighborhood, the center pixel is:

floor(([m n]+1)/2) …. (2)

In the 2-by-3 block shown in figure (3), the center pixel is (1,2), or the pixel in the second column of the top row of the neighborhood.

Padding Borders in Sliding Neighborhood Operations: As the neighborhood block slides over the image, some of the pixels in a neighborhood might be missing, especially if the center pixel is on the border of the image. For example, if the center pixel is the pixel in the upper left corner of the image as shown in figure (3), the neighborhoods include pixels that are not part of the image. To process these neighborhoods, sliding neighborhood operations pad the borders of the image, usually with 0's. In other words, these functions process the border pixels by assuming that the image is surrounded by additional rows and columns of 0's. These rows and columns do not become part of the output image and are used only as parts of the neighborhoods of the actual pixels in the image.

Distinct Block Operations: In distinct block processing, one divides a matrix into m-by-n sections. These sections, or distinct blocks, overlay the image matrix starting in the upper left corner, with no overlap. If the blocks don't fit exactly over the image, zero padding can be added to the matrix so that they do. Figure (4) shows a 6-by-16 matrix divided into 4-by-6 blocks. Note how the zero padding process adds 0's to the bottom and right of the image matrix, as needed. After zero padding, the matrix in the figure is size 8-by-18.

3. Smoothing Operations

These algorithms are applied in order to reduce noise and/or to prepare images for further processing such as segmentation [6].

3.1. Linear Filters: Several filtering algorithms can be presented together with the most useful supports, such as Uniform, Triangular, Gaussian, and other filters [6].

3.2. Non-Linear Filters: A variety of smoothing filters have been developed that are not linear. While they cannot, in general, be submitted to Fourier analysis, their properties and domains of application have been studied extensively.

• **Median filter**: The median filter is based upon moving a window over an image (as in a convolution) and computing the output pixel as the median value of the brightnesses within the input window. If the

window is **J × K** in size we can order the **J•K** pixels in brightness value from smallest to largest. If **J•K** is odd then the median will be the **(J•K+1)/2** entry in the list of ordered brightnesses. Note that the value selected will be exactly equal to one of the existing brightnesses so that no roundoff error will be involved if we want to work exclusively with integer brightness values. The algorithm has a generic complexity per pixel of **O(J•K•log(J•K))** [6].

The median filter is a smoothing technique that causes minimal edge blurring. However, it will remove isolated spikes and may destroy fine lines. The technique involves replacing the pixel value at each point in an image by the median of the pixel values in a neighborhood about the point [7].

The choice of neighborhood and median selection method distinguish the various median filter algorithms. Neighborhood selection is dependent on the source image. The machine architecture will determine the best way to select the median from the neighborhood [7].

Examples of the effect of various smoothing algorithms are shown in Figure (5) [6].

The classical approach to removal of impulsive noise is the median filter. The median of a set of samples is obtained by sorting the samples in ascending or descending order, and then selecting the mid-value [1].

In contrast, the mean, and, in particular, the variance of a set of numbers are sensitive to the presence of impulsive-type noise. An important property of median filters, particularly useful in image processing, is that they preserve edges or stepwise discontinuities in the signal. Median filters can be used for removing impulses in an image without smearing the edge information; this is of significant importance in image processing [1].

4. The Proposed Algorithm

The proposed zooming in algorithm returns output image that is scale times the size of input image. The input image can be a grayscale, RGB, or binary image.

The previous algorithms resize the image using one of the following interpolation methods:

- Nearest-neighbor interpolation: the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
- Bilinear interpolation: the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood.
- Bicubic interpolation: the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood.

So the enlarged output image may be distorted. In this work the proposed advanced neighborhood algorithm uses a different process. For image zooming in, the advanced neighborhood algorithm computes each output pixel by taking the value of each input pixel and distributes it to the corresponding output pixel's 3-by-3 neighborhood (that is, the pixel itself and its eight contiguous neighbors). Figure (6) illustrates this distribution.

The proposed advanced image zooming in algorithm performs median filtering for the image matrix using the 3-by-3 neighborhood. It controls how the matrix boundaries

are padded; 'zeros' or 'symmetric'. The input image matrix may padded with zeros at the boundaries, or it is symmetrically extended at the boundaries.

In median filtering, a window of predetermined length slides sequentially over the signal, and the mid-sample within the window is replaced by the median of all the samples that are inside the window

4.1. The proposed Algorithm of Advanced Neighborhood Operation

To perform the image zooming in by using the advanced neighborhood operation,

1. Select a single pixel from the input image.
2. Find the block of pixels in the output image matrix whose positions correspond to the input pixel. This block represents the input pixel as a center with the pixel's 3-by-3 neighborhood.
3. Set these output pixels to the value of the input pixel.
4. Repeat steps 1 through 3 for each pixel in the input image.
5. Apply the median filter to the enlarged output image.
6. Set the values of the corner pixels without filtering to the output image.

Figure (7) shows the flow chart of this algorithm.

5. Results

The proposed zooming in algorithm is tested with different image types. The programming language MATLAB is used to realize the proposed algorithm. Figure (8) shows an example, in this figure the input image is enlarged to 81 times; where each pixel in the original image matrix is distributed into 3-by-3 neighborhood twice time, as shown in figure (8 b and c). At each enlarge operation median filter is used as shown in figure (8 d and e).

In this work columnwise processing is used to speed up sliding neighborhood or distinct block operations. Performing sliding neighborhood and distinct block operations columnwise can reduce the execution time required to process an image.

For example, suppose an operation involves computing the mean of each block. This computation is much faster if one first rearranges the blocks into columns, because one can compute the mean of every column with a single call to the mean function, rather than calling mean for each block individually.

In the finite dimension of image, the problem arises for pixels on the border of the image, when the neighborhood block is not totally included in that of the image, as shown in figure (9).

Applying median filter and relations means one must prolong the image outside its natural support. There are two main techniques for processing side effects, according to the hypothesis made for this extension. First technique is prolonging the image by pixels of null intensity. In the absence of information about the image, one can assume that the unobserved pixels have a null intensity (black pixels). Second technique is prolonging by duplicating the border pixels. It consists of artificially extending the image support by attaching to it supplementary rows and columns that are identical to the external rows and columns of the image. This is shown in Figure (10).

There are other techniques for processing border effects with

finite impulse response filters. In this work we will decide not to calculate the value of the output's filter for the pixels situated on the border of the image, as shown in figure (11).

This makes it possible to avoid the arbitrary choice of the prolongation mode of support of the image, but it also reduces the usable size of the filtered image. When several filters are cascaded, the usable image rapidly becomes smaller. However, we should remember that this effect is lessened if we use separable filers.

6. Conclusions

1. The proposed algorithm calculates the value of each pixel's 3-by-3 neighborhood in the output image by passing the value of the input pixel corresponding to it.
2. Many functions need to convert the image to class double because the specific operation function is not defined for some image data type such as uint8. This algorithm doesn't need this conversion because of its function just to repeats each input pixel value to its eight contiguous neighbors.
3. The proposed algorithm does not require that the image be with same dimensions; and does not need to pad the borders of the image with additional rows and columns of 0's. This will avoid the problem in many image processing functions in which one must make sure that the specify function returns blocks of the appropriate size.
4. In this advanced method the zooming in take place only in the first two dimensions. For example, if the input image is **100-by-100-by-3**, then the output image is **900-by-900-by-3**.
5. It computes the number of rows and columns automatically to preserve the image aspect ratio, returns an output image with a colormap that is the same as the original colormap in the input image.
6. The input image can be logical (uint8, uint16, or double) or numeric, and the output image is of the same class as the input image.
7. If the input image A is of integer class, all of the output values are returned as integers. If the number of pixels in the neighborhood (i.e., **M*N**) is even, some of the median values may not be integers. In these cases, the fractional parts are discarded. Logical input is treated similarly.
8. In this algorithm the columnwise processing is used to speed up neighborhood operation. Performing sliding neighborhood and distinct block operations columnwise can reduce the execution time required to process an image.
9. Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.
10. In median filtering of the input image matrix in two dimensions, each output pixel contains the median value in the **M-by-N** neighborhood around the corresponding pixel in the input image. It pads the image with zeros on the edges, so the median values for the points within **[M N]/2** of the edges may appear distorted.

References

[1] Saeed V. Vaseghi, "Advanced Digital Signal Processing and Noise Reduction", Third Edition, John Wiley & Sons Ltd, England, 2006.

[2] Mohamed Najim, "Digital Filters Design for Signal and Image Processing", ISTE Ltd, USA, 2006.
[3] Singh I, "Digital Camera–A Review of its Applications in Urology and Genitourinary Surgery", Journal of Clinical and Diagnostic Research, Feb. 2009, Vol. 3, issue 1, pp.1341-1347.
[4] Julio Sanchez; Maria P. Canton, "Space Image Processing", CRC Press LLC 1998.
[5] O. Ferrer Roca and F. Marcano, "Computer Assisted Microscopy: The Era Small Size Slides & 4m Microscopes", Proceedings of the Third International Conference on Bio-inspired Systems and Signal Processing, Valencia, Spain, January 20 - 23, , Biosignals 2010, pp.517-522.
[6] Ian T. Young, Jan J. Gerbrands, Lucas J. van Vliet, "Fundamentals of Image Processing", The Netherlands at the Delft University of Technology., 1998.
[7] Gerhard X. Ritter; Joseph N. Wilson, "Handbook of Computer Vision Algorithms in Image Algebra", CRC Press LLC, 1996.

Table (1) Types of image operations: Image size = N*N; neighborhood size = P*P. Note that the complexity is specified in operations *per pixel* [6].

Operation	Characterization	Generic Complexity/Pixel
Point	The output value at a specific coordinate is dependent *only* on the input value at that same coordinate.	constant
Local	The output value at a specific coordinate is dependent on the input values in the *neighborhood* of that same coordinate.	P^2
Global	The output value at a specific coordinate is dependent on *all* the values in the input image.	N^2

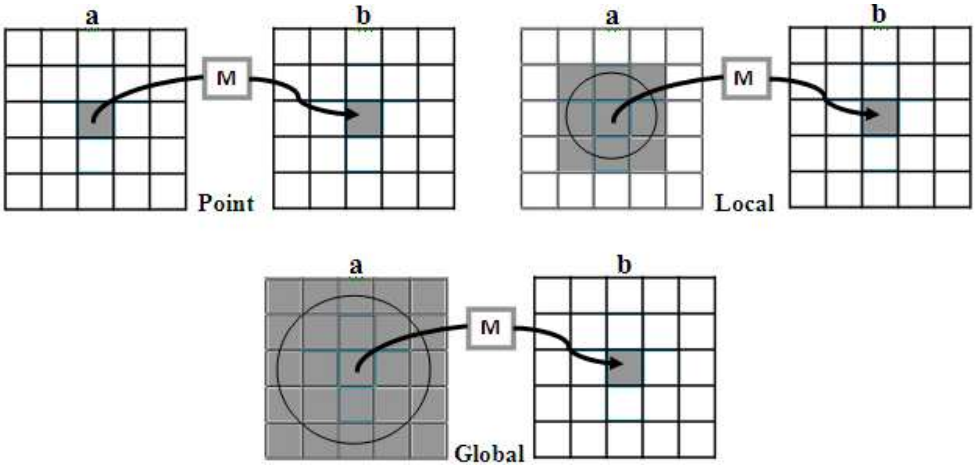


Figure (1) Illustration of Various Types of Image Operations [6]

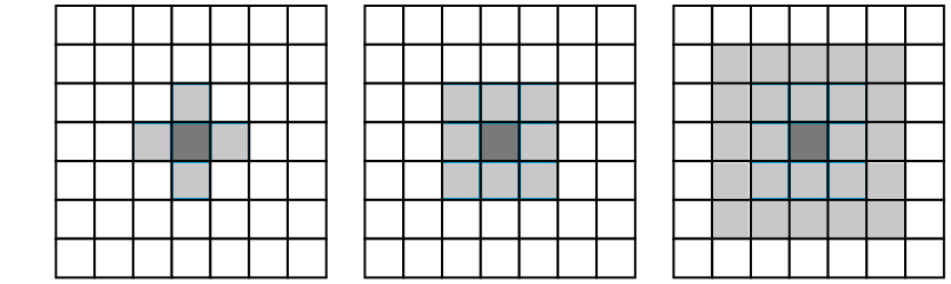


Figure (2) Pixel Neighborhood Patterns

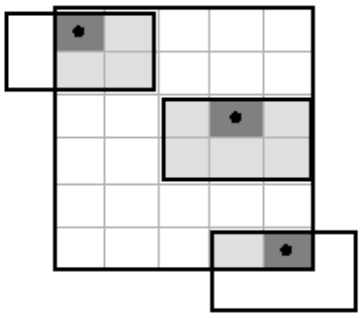


Figure (3) Neighborhood Blocks in a 6-by-5 Matrix

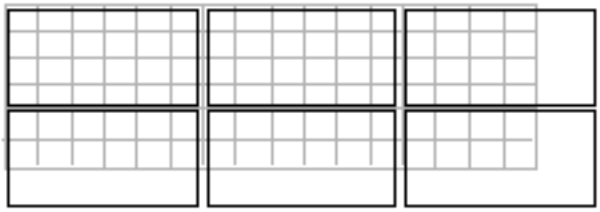
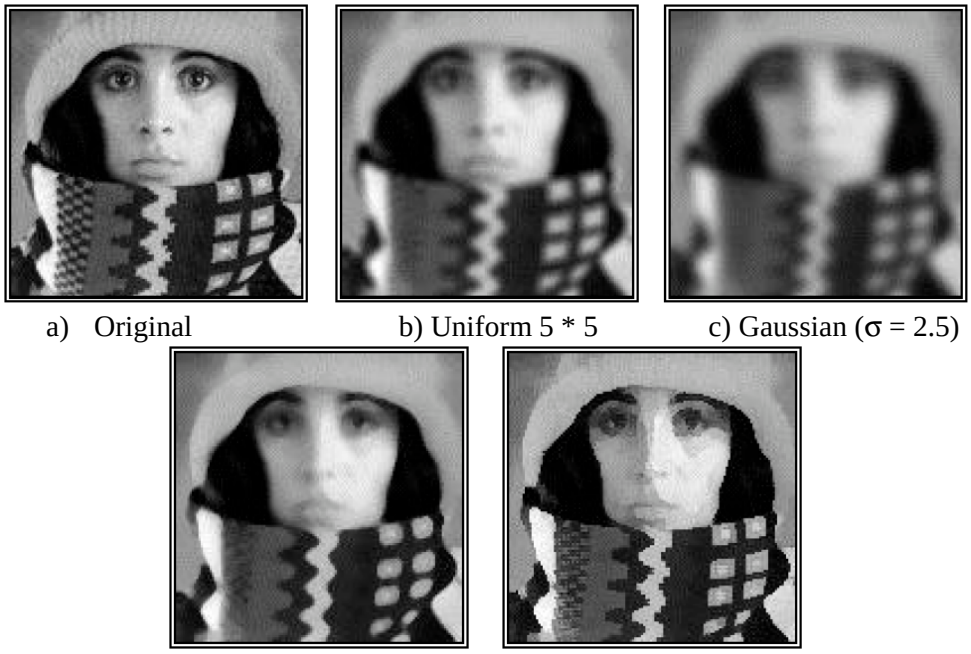


Figure (4) Image Divided into Distinct Blocks



a) Original b) Uniform 5 * 5 c) Gaussian (σ = 2.5)

d) Median 5 * 5 e) Kuwahara 5 * 5

Figure (5) Illustration of Various Linear and Non-Linear Smoothing Filters [6]

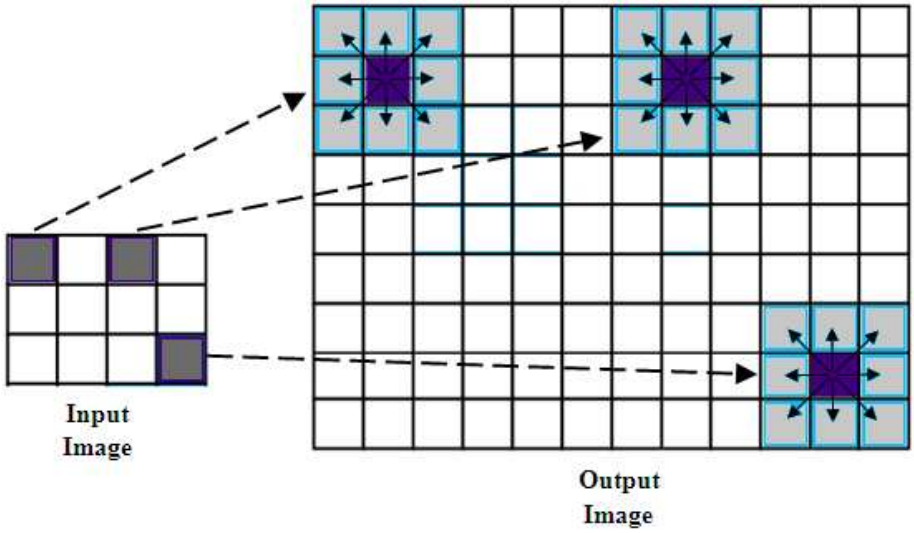


Figure (6) Distribution of the value of each input pixel to the corresponding output pixel's 3-by-3 neighborhood

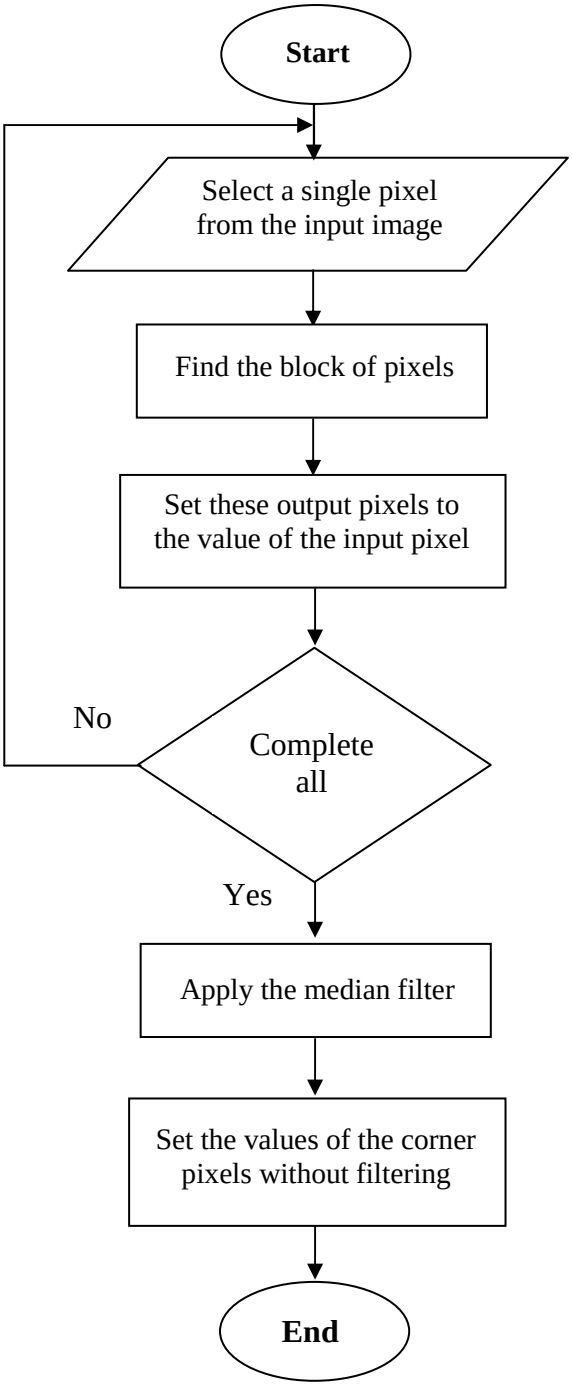


Figure (7) The Flow Chart of the Proposed Algorithm.

(a) (b) (c)

(d) (e)

Figure (8) Example of Test the Proposed Advanced Neighborhood Image Zooming In Algorithm: (a) Original Image, (b) Enlarged Image of (a), (c) Enlarged Image of (b), (d) Filtered Image of (b), (e) Filtered Image of (c)

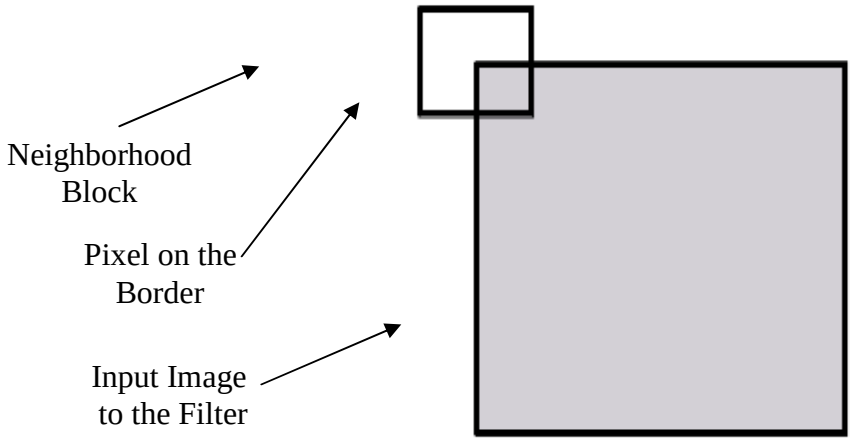


Figure (9) Pixel Situated at Image Border

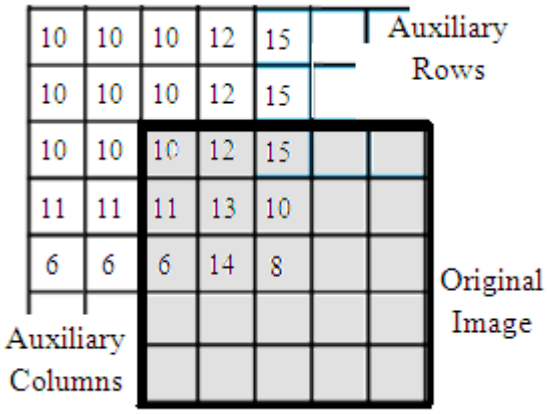


Figure (10) Duplication of Block pixels

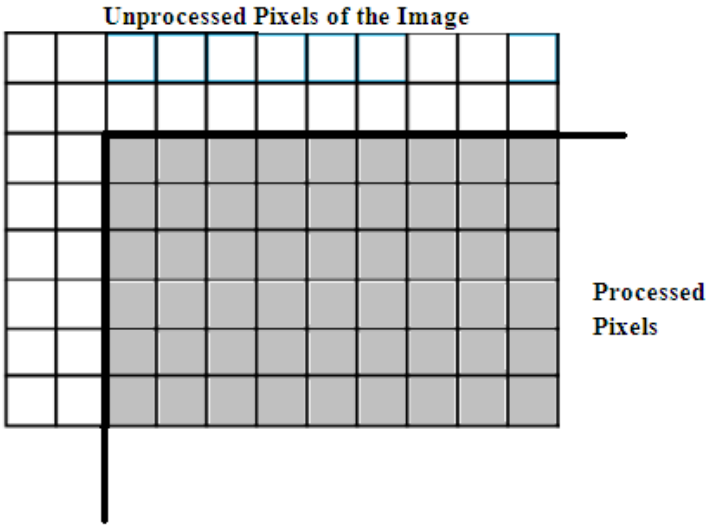


Figure (11) Unprocessed Pixels