(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 May 2002 (02.05.2002)

PCT

(10) International Publication Number
WO 02/35471 A1

(51) International Patent Classification[7]: G06T 5/00, 5/20, 5/30, 1/20

(21) International Application Number: PCT/US01/32525

(22) International Filing Date: 16 October 2001 (16.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/693,378 20 October 2000 (20.10.2000) US

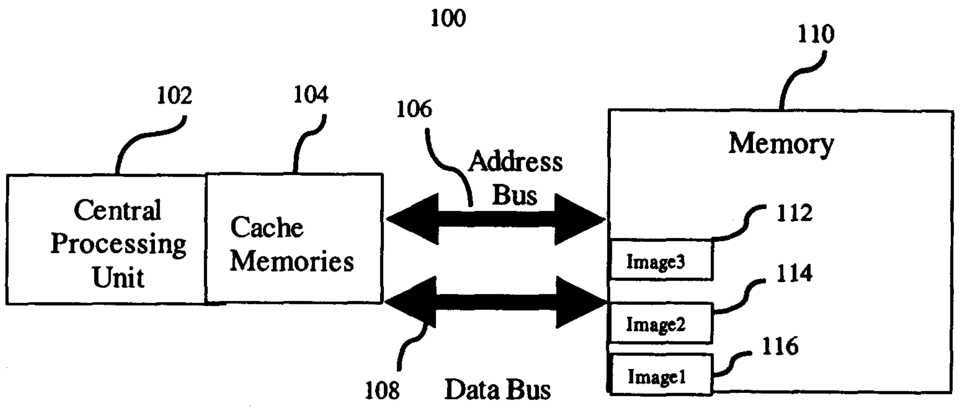
(71) Applicant and
(72) Inventor: **LEE, Shih-Jong** [US/US]; 15418 SE 53rd Place, Bellevue, WA 98006 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(54) Title: IMAGE PROCESSING APPARATUS USING A CASCADE OF POLY-POINT OPERATIONS



(57) Abstract: A high speed image processing apparatus comprising a central processing unit 102 anda memory to store images 110 wherein a program directs the central processing unit to perform a poly-point operation 320 on image data. Poly-point operations filter images without multiplication or division yet achieve versatile filter characteristics. Poly-point operations accomplish linear or non-linear filter operations quickly and allow complex image processing operations on general purpose computing apparatus.

Image Processing Apparatus Using a Cascade of Poly-Point Operations

Technical Field

This invention relates to high-speed image processing operations.

Background Art

Conventional approaches for image processing implement image operations directly from their specifications. These approaches demand significant computational resources. For example, a 3 by 3 image convolution requires 9 multiplications, 8 additions, and one division for each pixel in an image. In the prior art solution, floating point operations may be required in order to express the fine detail of the filter kernel and to normalize the result. The precision required for these computations frequently is not consistent with the quantization and sampling of the input image, which is limited to reduce the image storage and transmission requirement. As the image coding (number of pixels, color quantization and luminance quantization) becomes more complete, the computations required to process even simple algorithms becomes overwhelming because of the increase in volume of data. Prior art image processors have been created to meet the demands of current generation images. Specialized hardware has been created to address the high computational demand. However, the specialized hardware is expensive and difficult to program and the result achieved is not worth the effort expended. In the prior art, little effort has been expended to reduce the complexity of the computation; rather the prior art attempts to meet the complexity with increased facility. Moreover, general large kernel or three-dimensional image processing operations are still prohibitively expensive even with the specialized hardware.

The prior art has expended the resources to implement the computations required for the general solution of the image processing problem, the capacity requirements generally exceed those available on general purpose computers like the Personal Computer (PC). Thus, the technical advances in general purpose PCs and the price advantages have not heretofore replaced the specialized image processing hardware.

The programming tasks for the specialized hardware are complex because computation optimization frequently requires assembly level instructions, management of several different types of computing resources, several different types of development tools, parallel processing with interleaved results, and/or use of highly optimized primitive functions provided by a hardware vendor that are not easily tailored to any particular application.

Disclosure of Invention

The invention discloses methods for cascading simple, easily programmable poly-point operations in general purpose computers. The invention allows image processing programming in high level programming language such as C, C++ or Java that can be compiled into different general purpose computing platforms. The efficiency of the operations allows sufficient throughput improvement to enable real time image processing in resource constrained mobile computing or information/e-appliance platforms or to enable real time high performance image processing for industrial, scientific and medical applications using lower cost computing facilities or image processing units.

Brief Description of Drawings

Figure 1 shows hardware architecture for Poly-Point image processing.

Figure 2 shows a processing flow diagram for the steps in performing a poly-point image operation.

Figure 3A shows a first quad point kernel for use in creating a 32 point linear filter programmed as a cascade of three quad point filters.

Figure 3B shows a second quad point kernel for use in creating a 32 point linear filter programmed as a cascade of three quad point filters

Figure 3C shows a third quad point kernel for use in creating a 32 point linear filter programmed as a cascade of three quad point filters.

Figure 4A shows the intermediate filter result for a cascade of the first and second quad point kernels shown in Figure 3A and 3B

Figure 4B shows the 32 point linear filter equivalent of a cascade of 3 quad point filters shown in Figures 3A, 3B, and 3C.

Figure 5 shows the method for performing a cascade of poly-point operations.

Figure 6A shows part 1 of six dual point additions that cascade to form a 32 point linear filter.

Figure 6B shows part 2 of six dual point additions that cascade to form a 32 point linear filter.

Figure 6C shows part 3 of six dual point additions that cascade to form a 32 point linear filter.

Figure 6D shows part 4 of six dual point additions that cascade to form a 32 point linear filter.

Figure 6E shows part 5 of six dual point additions that cascade to form a 32 point linear filter.

Figure 6F shows part 6 of six dual point additions that cascade to form a 32 point linear filter.

Figure 7A shows the intermediate result of 6A of the 6 dual point additions.

Figure 7B shows the intermediate result of 6A cascaded with 6B of the 6 dual point additions.

Figure 7C shows the intermediate result of 6A cascaded with 6B cascaded with 6C.

Figure 7D shows the intermediate result of 6A cascaded with 6B, 6C, and 6D.

Figure 7E shows the 32-point linear filter result of the cascade of all 6 dual point additions.

Figure 8A shows the first of three quad point filters that when cascaded will form the filter shown in Figure 9B that could be used for vertical edge detection.

Figure 8B shows the second of three quad point filters which when cascaded will form the filter shown in Figure 9B which could be used for vertical edge detection

Figure 8C shows the third of three quad point filters which when cascaded will form the filter shown in Figure 9B that could be used for vertical edge detection.

Figure 9A shows the intermediate filter result from cascading the first and second quad point filters shown in figures 8A, and 8B.

Figure 9B shows the filter formed by cascading the filters of Figure 8A, 8B, and 8C. This filter could be used for vertical edge detection.

Figure 10A shows the first of three quad point maximum kernels used to perform a 32 point grayscale morphological dilation programmed as cascade of three quad-point maximum kernels wherein each darkened element indicates part of the maximum function.

Figure 10B shows the second of three quad point maximum kernels used to perform a 32 point grayscale morphological dilation programmed as cascade of three quad-point maximum kernels wherein each darkened element indicates part of the maximum function.

Figure 10C shows the third of three quad point maximum kernels used to perform a 32 point grayscale morphological dilation programmed as cascade of three quad-point maximum kernels wherein each darkened element indicates part of the maximum function.

Figure 11A shows the first maximum operator for the 32-point grayscale morphological dilation

Figure 11B shows the cascade result of the first and second maximum operators of Figures 10A and 10B.

Figure 11C shows the cascade result of the first, second, and third quad-point maximum operators to form a 32 point grayscale morphological dilation.

Figure 12A shows the first of 6 dual point maximum operators used to form a 32 point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 12B shows the second of 6 dual point maximum operators used to form a 32 point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 12C shows the third of 6 dual point maximum operators used to form a 32-point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 12D shows the fourth of 6 dual point maximum operators used to form a 32-point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 12E shows the fifth of 6 dual point maximum operators used to form a 32-point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 12F shows the sixth of 6 dual point maximum operators used to form a 32-point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 13A shows the intermediate result of the first of 6 dual point maximum operators used to form a 32-point grayscale morphological dilation programmed as cascade of six dual-point maximum

Figure 13B shows the intermediate result of combining the first and second dual point maximum operators shown in Figures 12A, and 12B.

Figure 13C shows the intermediate result of combining the first, second and third dual point maximum operators shown in Figures 12A, 12B, and 12C.

Figure 13D shows the intermediate result of combining the first, second, third and fourth dual point maximum operators shown in Figures 12A, 12B, 12C, and 12D.

Figure 13E shows the result of combining all six dual point maximum operators shown in Figures 12.

Best Mode for Carrying Out the Invention

Referring to Figure 1, a computer **100** has at least one Central Processing Unit (CPU) **102** and one memory module **110**. Simple computers could have the CPU and memory on a single chip. More complicated computers may have multiple CPUs and multiple memory boards. This invention stores an image as a contiguous block in memory or other convenient way for memory addressing. Multiple images **112, 114, 116** can be efficiently stored and accessed. The interface between the CPU and the memory is through an address bus **106** and a data bus **108**. Most CPUs have on-chip or external high speed cache memories **104**. This architecture exists on almost all computers. The memory access can be under the control of the CPU or through a Direct Memory Access (DMA) module.

Images are efficiently accessed by sequentially incrementing the address corresponding to single or multiple memory locations depending upon word length and pixel quantization. Poly-point operations (or simply point operations) are performed in the CPU on data

loaded from the memory addressed. Poly-point operations can be complex and may include any mathematical operation that uses a pixel value as input.
The results of the poly-point operations are stored in either an internal buffer, cache memory or an image in memory. The steps shown in Figure 2 **130** can carry out the poly-point image operations.

As shown in Figure 2, memory addresses are incremented sequentially **132** and data associated with the addressed memories are loaded into the CPU **102**. The desired operation is performed on the data **134** and the result of the operation is saved to an internal buffer or memory **136**. A check is performed to determine whether the whole image is processed **138**. If the whole image is processed, the poly-point image operation is completed **140**. Otherwise, the memory addresses are incremented **132** and the steps are repeated.

Many memory capacity, access and processing speed up features are built into general purpose CPUs. For example, the Intel® Pentium® III processor integrates the P6 Dynamic Execution micro-architecture, Dual Independent Bus (DIB) Architecture, a multi-transaction system bus, Intel® MMX™ media enhancement technology, and Internet Streaming SIMD (Single Instruction Multiple Data) Extensions. It also includes Advanced Transfer Cache and Advanced System Buffering to achieve higher data bandwidth. It has memory cache-ability up to 4 GB of addressable memory space and system memory scalability up to 64 GB of physical memory that allows the storage of a huge number of images (Intel Pentium® III Processor for SC242 at 450 MHz to 1.13 GHz Datasheet). PowerPC 7400 uses AltiVec technology vector processing units to speed processing (see http://www.altivec.org for a complete list of AltiVec related papers and articles, including technology overviews).

The poly-point image operations in this invention are simple and predictable. The simplicity and predictability improves the efficiency of memory caching and operation predictions that are built into many CPUs. This invention uses a cascade of poly-point operations to achieve high speed linear filtering and morphological operations that form the bases of most neighborhood based image processing functions.

Filtering is conventionally achieved by neighborhood convolutions. This invention can efficiently achieve the same or equivalent neighborhood convolutions by a simple program that performs a cascade of poly-point additions/subtractions. The poly-points to be added are from different locations of the same image (an exception would be motion detection). This can be efficiently programmed as additions of different memory locations that increment simultaneously. For example, a quad-point addition can be programmed as the sum of four incremental memory contents as follows:

```
I_out[++i] = I_in[i] + I_in[j++] + I_in[k++] + I_in[l++];
```

The memory pointers i, j, k and l are offset according to the kernel specification.

The simple addressing mode of image memory allows efficient use of prefetch and cacheability instructions provided by the CPU. For example, a substantial portion, the neighborhood portion, of the input image could be loaded into the cache memory **104** to facilitate high speed processing.

Figure 4B shows a 32 point linear filter that can be programmed as cascade of three quad-point kernels shown in Figure 3A, **160,** Figure 3B, **170,** and Figure 3C, **180**. Here the quad-point name stems from the four unit values in each kernel. As will be apparent to those skilled in the art, similar kernels with fewer or more values could also be used. The kernels are selected to ease the processing demand yet achieve a computationally efficient filtering result. Note that all the primitive kernels **160, 170, 180** have four unit coefficients. These coefficients render the multiplication operation in convolution moot, since the result of multiplication by 1 or zero is identical to the value being multiplied or it is zero. Thus, no time needs to be spent performing the multiplication operation. Only addition is required to convolve these kernels with each other or with input image pixels. In addition, because the kernels are small, few pixels need to be retrieved from memory to compute the output result. All operations are integer. Pixels may be typically expressed in integer 8 bit values. To maintain 8 bit values for the output image and to normalize the output result, a right shift of 2 bits (effectively divide by 4) can be applied as part of the operation. Filters involving non-unity values (requiring multiplication) or large size kernels are broken down into a sequence of small, simple and fast addition (and bit shift) operations. The particular small kernels selected are chosen to approximate the characteristics of the filter they replace. The shapes of filters that can result from differing combinations of these basic filters approximates very well behaved traditional filters and thus produces most of their performance while at the same time creating a large decrease in computational load for the CPU. In the conventional approach, a 32 point linear filter requires 32 multiplications, 31 additions, and a divide for each pixel. Depending upon the kernel, floating point operations may also be required. Yet, the result achieved may not be significantly different than what could be achieved with the simpler approach described herein. In the example, the quad-point operations only require 12 additions and three 2 bit shifts per output pixel value.

To illustrate how this approach develops complex filter shape or large size kernels, the example kernels **160, 170, 180** are combined. **160** combined with **170** produces **200** (figure 4A). Therefore, convolving **160** and **170** with an input image to produce a result1 is equivalent to filtering of the same input image with **200**. But the time required is less using the multi-step primitive kernels **160, 170**. Cascading **180** with Result1 to produce Result2 is equivalent to filtering of the input image with **210**. Again, it will be clear that time is saved. As will be apparent, the time saved becomes dramatically large where large kernel size or multi-dimensional filters are involved. Note that the principle of simple kernels with unit values can be used to produce a wide variety of filters. The kernels are not restricted to quad element figures, as in this example, but can be constructed from a variety of primitive kernels.

The steps for performing a cascade of the poly-point operations are shown in Figure 5 **350**. In step **310** memory address pointers are setup according to the size and shape of the kernel for the poly-point operation. Poly-point operation is performed for the entire image, **320**. A check is performed **330** to determine whether all stages in the image filter cascade are completed. If all stages are completed **340**, the image filtering function is completed. Otherwise, the memory address pointers are set up for the next poly-point operation **310** and the steps are repeated.

In one embodiment of the invention, the operations as shown in Figure 5 **350** can be programmed in the pseudo codes as follows:

```
char I[image_size], I_out[image_size], I_2[image_size];
register int i, j, k, l;

For (i = -1, j =0, k = line_length, l = line_length+1; i<image_size;)
    I_out[++i] = (I [i] +  I [j++] + I [k++] + I [l++])>>2;
For (i = -1, j = line_length-2, k = line_length-1, l = j+line_length; i<image_size;)
    I_2[++i] = ( I_out [i] +  I_ out [j++] + I_ out [k++] + I_ out [l++])>>2;
For (i = -1, j = 1, k = 2*line_length-1, l = k+2; i<image_size;)
    I_ out [++i] =( I_2 [i] +  I_2 [j++] + I_2 [k++] + I_2 [l++])>>2;
```

In this implementation, image memories are declared as arrays in a software program. Registers are used for the address pointer to increase the access speed. "line_length" is the length of each line of the image. "image_size" is the size of the image. The program is very simple and straightforward so it can be easily optimized or written in C/C++, Java, assembly language or other computer languages for the best performance. Additional programs may be written to handle image boundary conditions. As an example, each line of an image can be filtered in its entirety by extending image memory to include extra data for boundary condition handling. Alternatively, the boundary condition management can be done one line at a time. The origin of each kernel can affect a shift in the image position that can also be dealt with as part of the boundary condition.

The same 32 point linear filtering can be programmed as a cascade of six dual-point additions as shown in Figure 6. The cascade of kernels of figure 6 are equivalent to a larger kernel which is developed in Figures 7A, 7B, 7C, 7D, 7E wherein poly-point operations are cascaded beginning with **400**, cascaded with **410** to produce **460**, cascaded with **420** to produce **470**, cascaded with **430** to produce **480**, and cascaded with **440** and **450** to produce **490**. Cascading very small kernels may be appropriate for CPU's with slower ALU yet fast memory speed. The dual-point operations only require 6 additions for each pixel. This compares favorably to the prior art approach that requires 32 multiplications and 31 additions per pixel. In one embodiment of the invention, the operations of Figure 6 can be programmed in the pseudo codes as:

```
char I[image_size], I_out[image_size], I_1[image_size];
register int I, j, k, l;

For (i = 0; i<image_size;)
    I_1[i] = (I [i++] + I [i] )>>1 ;
For (i = -1, j = line_length; i<image_size;)
    I_out[++i] = (I_1 [i] + I_1 [j++])>>1 ;
For (i = -1, j = line_length-1; i<image_size;)
    I_1[++i] = (I_out [i] + I_out [j++])>>1 ;
For (i = -1, j = line_length-2; i<image_size;)
    I_out [++i] = (I_1 [i] + I_1 [j++])>>1 ;
For (i = -1, j = 1; i<image_size;)
    I_1[++i] = (I_out [i] + I_out [j++])>>1 ;
For (i = -1, j = 2*line_length-1; i<image_size;)
    I_out[++i] = (I_1 [i] + I_1 [j++])>>1 ;
```

As can be appreciated by those skilled in the art, the poly-point operations do not have to be limited to 2 or 4 points. Poly-point additions can be easily programmed in a general purpose CPU and additions of different points can be cascaded together to form the desired filter kernel in a most efficient fashion for the available computing resource. The number of points per stage of operation can also be flexibly adjusted to match the CPU and memory speed to avoid processing dead time due to CPU or memory bottleneck. Fewer points are used for slower CPU and more points are used for slower memory.

Poly-point filters may operate in conjunction with prior art filters in a system. Certain filter actions may require extreme precision, and others may be capable with a more approximate result. For example, in filtering of color components of an image, lower precision is generally required than is necessary for the luminance information. Poly-point filters may also be cascaded with prior art filters.

This invention is efficient for large kernel filtering. It achieves convolution without multiplication and can be easily used for efficient multi-dimensional processing. For example, tracking of objects in images whose position changes with time may be done using poly-point filtering. Poly-point operations can be created that use subtraction or combination of addition and subtraction to create other linear filters. Figures 8A, 8B, 8C, shows small changes to Figure 3C kernel **180** to create a cascade of filters **160, 170, 520** that could be used for vertical edge detection. In this example, **160** is cascaded with **170** to create **200** (Figure 9A) which is cascaded with **520** to create **620** (Figure 9B).

Morphologic Filtering by Cascade of Poly-Point Maximum/Minimum

Similar to linear filters, morphologic filtering is conventionally achieved by neighborhood operations. This invention can efficiently achieve the same operations by a simple program that performs cascade of poly-point maximum/minimum. The poly-points to be operated are from different locations of the same image. This can be efficiently programmed as maximum/minimum of different memory locations that increment simultaneously. For example, a quad-point maximum can be programmed as the maximum of four incremental memory contents as follows:

```
I_out[++i] = MAX(MAX(I_in[i], I_in[j++]), MAX(I_in[k++], I_in[l++]));
```

Where memory pointers i, j, k and l are offset according to the specification of the kernel.

Figure 10A, 10B, 10C, and Figure11A, 11B, 11C, shows 32 point grayscale morphological dilation **750** programmed as a cascade of three quad-point maximum kernels **700,710,720.** **730** is cascaded with **710** to produce **740** which is cascaded with **720** to produce **750.** Wherein the maximum associated value of each of the 4 darkened elements of the kernels replaces the value of the pixel in the image being filtered. Note that fewer and more closely grouped pixels from the image are required to perform any individual filtering operation. This speeds up memory access. In addition, 12 overall maximum operations are performed per pixel instead of 32. In one embodiment of the invention, the operations can be programmed in pseudo code as:

```
char I[image_size], I_out[image_size], I_2[image_size];
register int i, j, k, l;

For (i = -1, j =0, k = line_length, l = line_length+1; i<image_size;)
    I_ out [++i] = MAX( MAX(I [i],  I [j++]), MAX(I [k++], I [l++]) );
For (i = -1, j = line_length-2, k = line_length-1, l = j+line_length; i<image_size;)
    I_2[++i] = MAX( MAX(I_out [i],  I_out [j++]), MAX( I_out [k++], I_out [l++]) );
For (i = -1, j = 1, k = 2*line_length-1, l = k+2; i<image_size;)
    I_ out [++i] = MAX( MAX(I_2 [i],  I_2 [j++]), MAX(I_2 [k++], I_2 [l++]) );
```

The same 32 point grayscale morphological dilation can be programmed as cascade of six dual-point maximum as shown in Figures 12 and 13 wherein the maximum value of each of two darkened elements of the kernels **800, 810, 820, 830, 840** and **850** replaces the value of the pixel in the image being filtered. In this example, the two element maximum operators **800** (same as **900**) and **810** in combination effect the maximum operator **910**. The combination of **800, 810**, and **820**, effect a maximum operator **920**. The combination of **800, 810, 820** and **830** effect a maximum operator **930**. The combination of **800, 810, 820, 830, 840** and **850** effect a maximum operator **940**. In comparison with the quad-point operators of Figure 10 that produce the same result, this may be appropriate for CPU's with slower ALU yet fast memory assess speed. Alternatively, an intermediate result **910, 920,** or **930** may be desired as an output.

In one embodiment of the invention, the operations of Figure 12 can be programmed in pseudo codes as:

```
char I[image_size], I_out[image_size], I_1[image_size];
Register int I, j, k, l;

For (i = 0; i<image_size;)
```

```
    I_1[i] = MAX (I [i++] , I [i]) ;
For (i = -1, j = line_length; i<image_size;)
    I_out [++i] = MAX (I_1 [i], I_1 [j++]) ;
For (i = -1, j = line_length-1; i<image_size;)
    I_1[++i] = MAX (I_out [I], I_out [j++]) ;
For (i = -1, j = line_length-2; i<image_size;)
    I_out [++i] = MAX (I_1 [i], I_1 [j++]) ;
For (i = -1, j = 1; i<image_size;)
    I_1[++i] = MAX (I_out [i], I_out[j++]) ;
For (i = -1, j = 2*line_length-1; i<image_size;)
    I_out [++i] = MAX (I_1 [i], I_1 [j++]) ;
```

As can be appreciated by those skilled in the art, the poly-point operations do not have to be limited to 2 or 4 points. Poly-point maximum can be easily programmed in a general purpose CPU and the neighborhood maximum of different points can be cascaded together to form the desired structuring element in the most efficient fashion for the available computing resource. The number of points per stage of operation can also be flexibly adjusted to match the CPU and memory speed to avoid processing dead time due to CPU or memory bottleneck. Fewer points are used for slower CPUs and more points are used for slower memory.

This invention is very efficient for large kernel morphological operations. It can be easily applied to highly efficient multi-dimensional processing. The operations can also be expanded. Changing the neighborhood operator from maximum to minimum will change dilation into erosion. Combinations of dilation and erosion will create morphological opening, closing and hit-or-miss transformation. From the hit-or-miss transformation, morphological thinning and thickening operations can be created.

The invention has been described herein in considerable detail in order to comply with the Patent Statutes and to provide those skilled in the art with the information needed to apply the novel principles and to construct and use such specialized components as are required. However, it is to be understood that the inventions can be carried out by specifically different equipment and devices, and that various modifications, both as to the equipment details and operating procedures, can be accomplished without departing from the scope of the invention itself.

The Claims defining the invention are:

1. A high speed image processing apparatus comprising:
    a. a central processing unit having a data bus and an address bus to access and load data;
    b. a memory to store images, wherein the memory is connected to the central processing unit through the data bus and the address bus
    c. a program directs the central processing unit to perform a poly-point operation on image data loaded from the memory through the data bus and the address bus.

2. The apparatus of claim 1 further comprises a means to save the poly-point operation results to memory.

3. The apparatus of claim 1 wherein the poly-point operation comprises the steps of:
    a. incrementing at least one memory addresses sequentially and
    b. Loading data associated with the addressed memory into a CPU;
    c. performing at least one ALU operation on the data and
    d. saving the result of the operation to memory
    e. checking whether the whole image is processed
    f. repeating the poly-point image operation

4. The apparatus of claim 1 further comprising a program to perform cascade of poly-point operations.

5. The apparatus of claim 1 wherein the data from the memory is loaded to the central processing unit by a DMA device.

6. The apparatus of claim 2 wherein the means for saving the poly-operation results to the memory of claim 2 comprises a DMA device.

7. The apparatus of claim 3 wherein the poly-point operation steps include at least one addition operation.

8. The apparatus of claim 3 wherein the poly-point operation steps include at least one subtraction operation.

9. The apparatus of claim 3 wherein the poly-point operation steps include at least one maximum operation.

10. The apparatus of claim 3 wherein the poly-point operation steps include at least one minimum operation.

11. The apparatus of claim 3 wherein the poly-point operation result is saved to memory by a DMA device.

12. The method of claim 4 further comprising the steps of:
    a. setting up memory address pointers according to the size and shape of the kernel for a poly-point operation;
    b. performing the poly-point operation for at least a portion of an image;
    c. performing a check to determine that all stages in the image filtering cascade are completed.
    d. performing a cascaded poly-point operation for at least a portion of an image

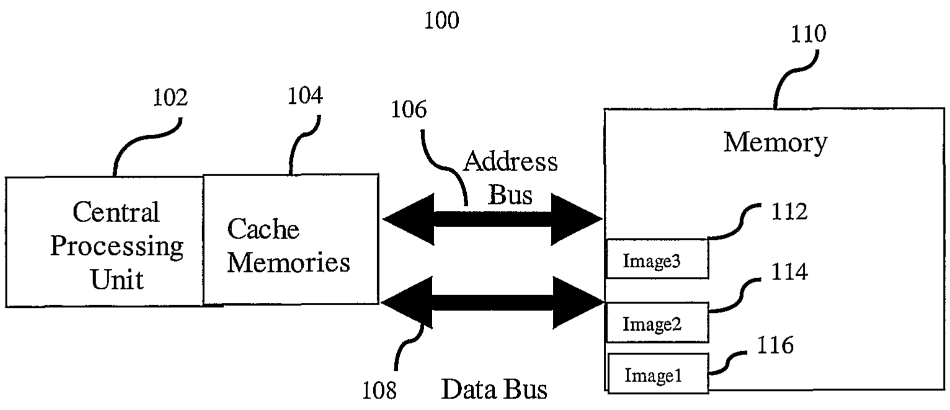


Figure 1

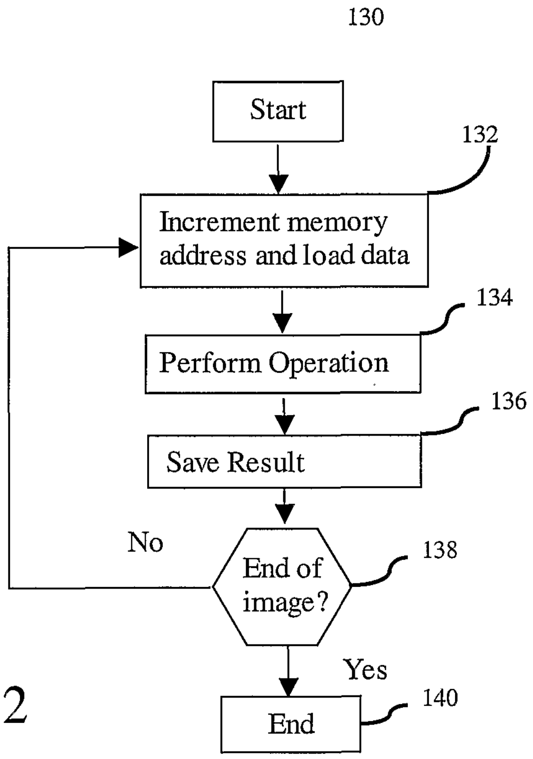


Figure 2

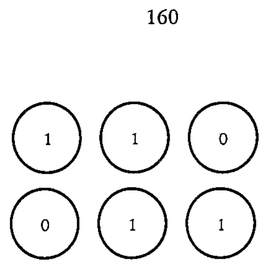


Figure 3A

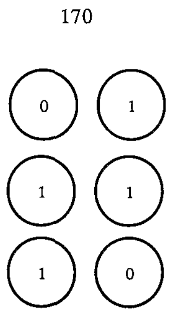


Figure 3B

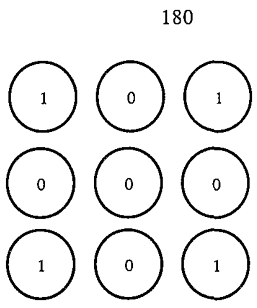


Figure 3C

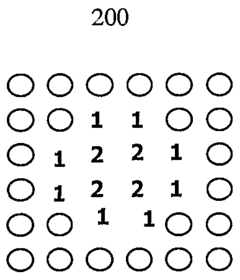


Figure 4A

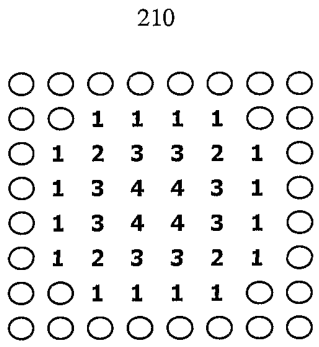


Figure 4B

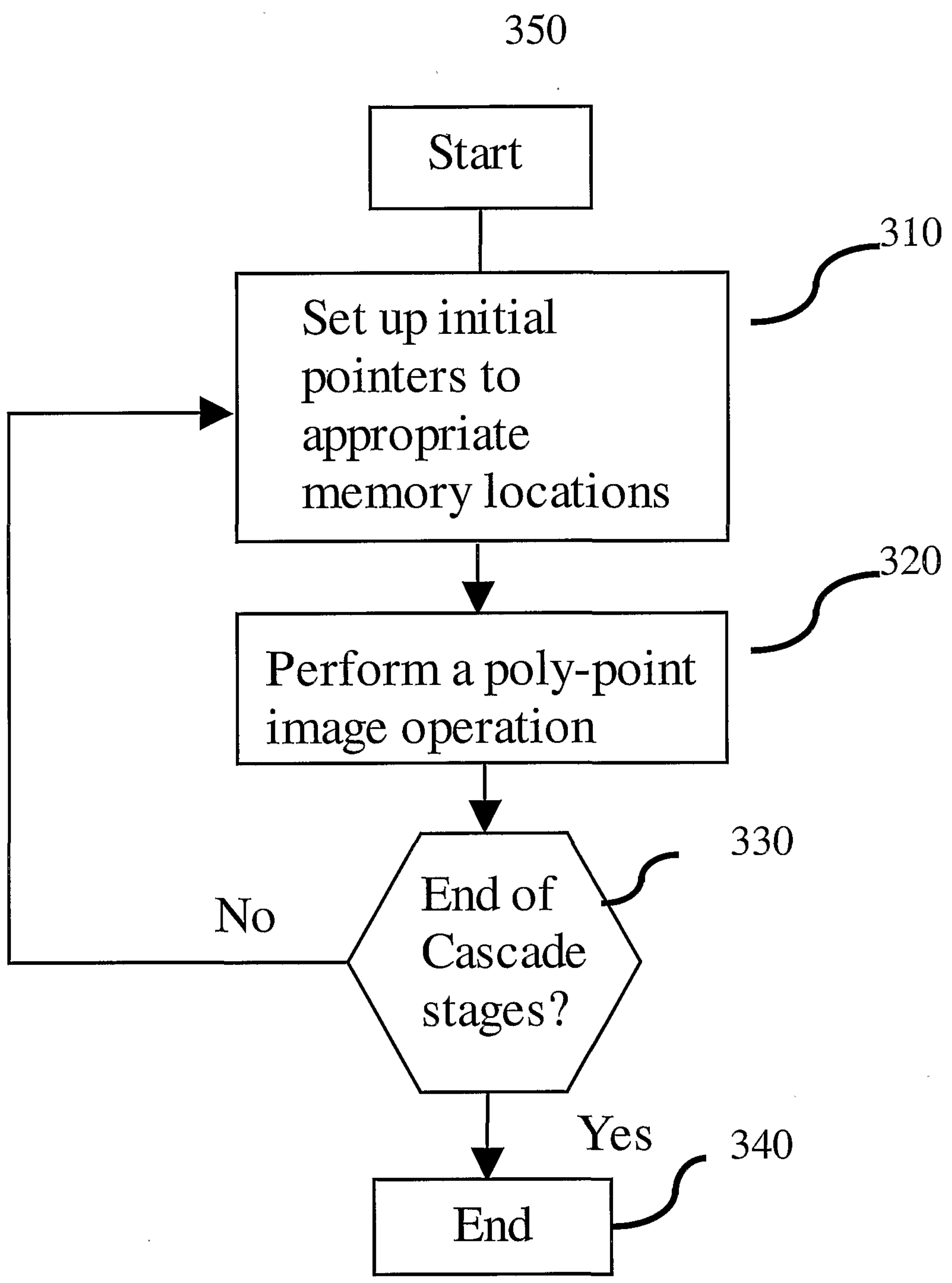

350
Start
Set up initial pointers to appropriate memory locations
310
Perform a poly-point image operation
320
End of Cascade stages?
330
No
Yes
End
340

Figure 5

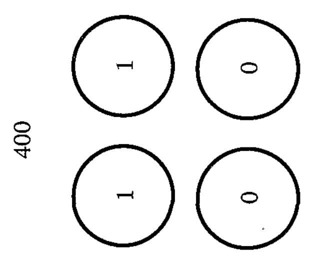


Figure 6A

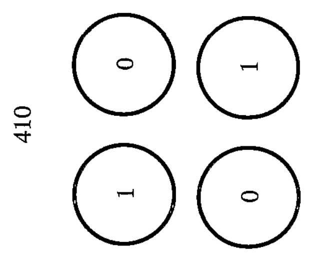


Figure 6B

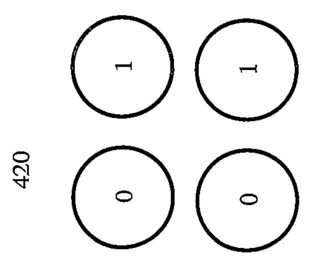


Figure 6C

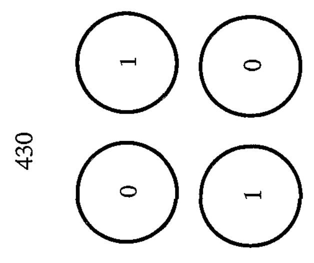


Figure 6D

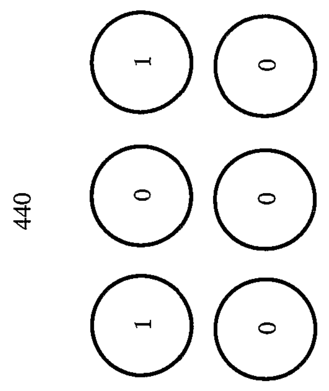


Figure 6E

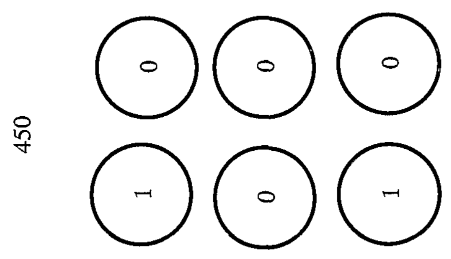


Figure 6F

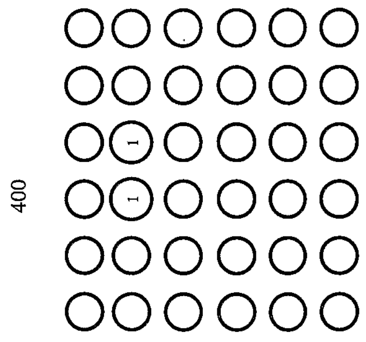


Figure 7A

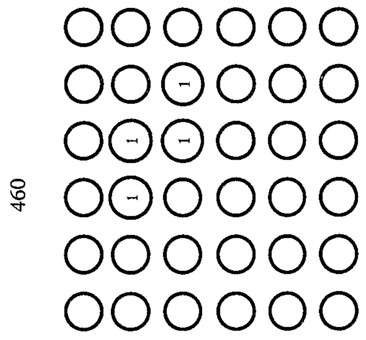


Figure 7B

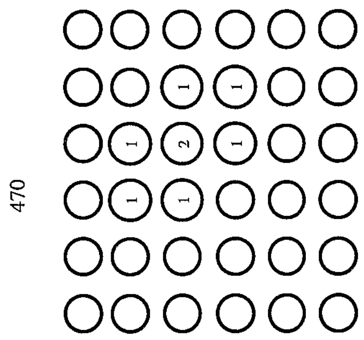


Figure 7C

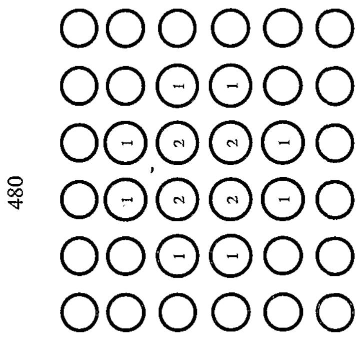


Figure 7D

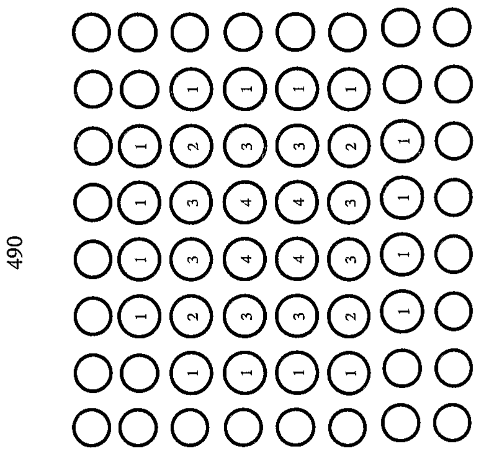


Figure 7E

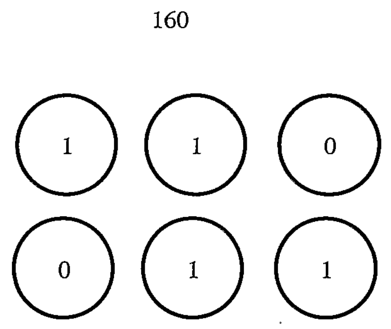


Figure 8A

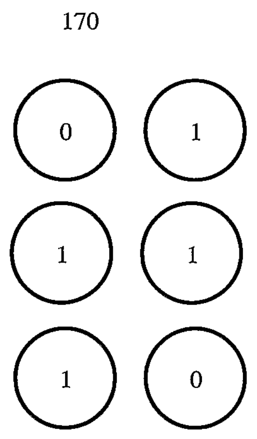


Figure 8B

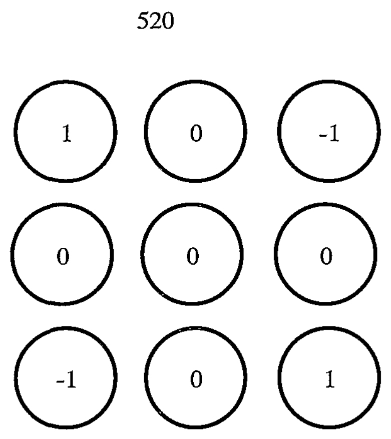


Figure 8C

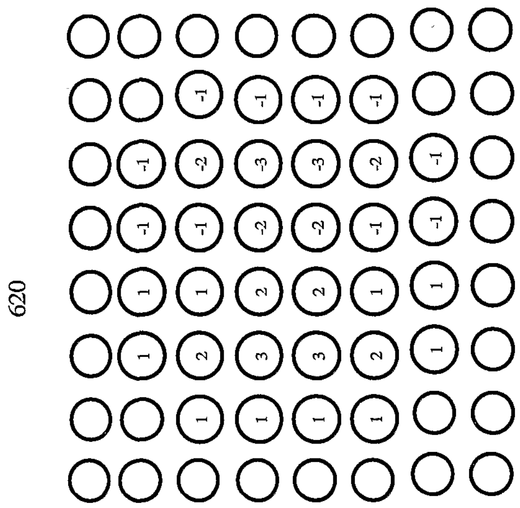


Figure 9B

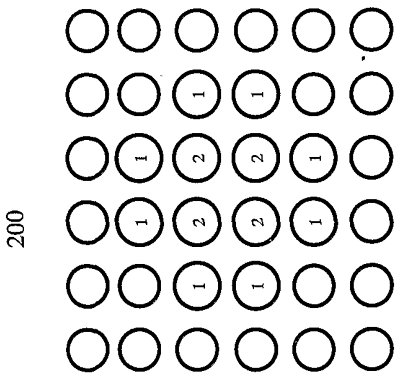


Figure 9A

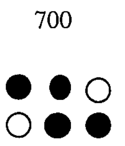


Figure 10A

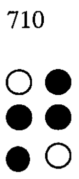


Figure 10B

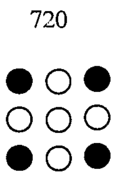


Figure 10C

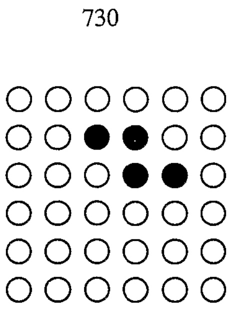


Figure 11A

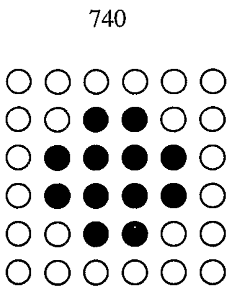


Figure 11B

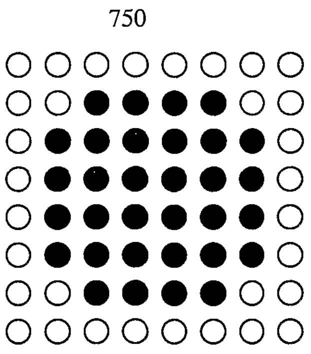


Figure 11C

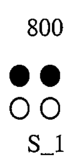


Figure 12A

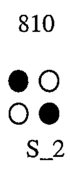


Figure 12B

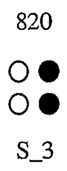


Figure 12C

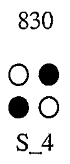


Figure 12D

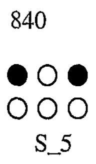


Figure 12E

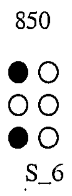


Figure 12F

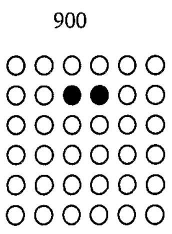


Figure 13A

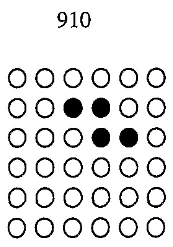


Figure 13B

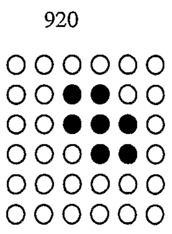


Figure 13C

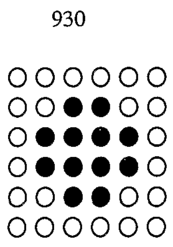


Figure 13D

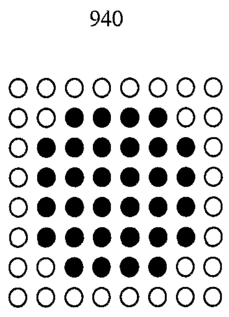


Figure 13E

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/32525

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06T 5/00, 5/20, 5/30, 1/20
US CL : 382/257,260,302,303,307,308

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 382/257,260,302,303,307,308,304

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 5,046,190 A (DANIEL et al) 03 September 1991, col. 3, lines 25-60, col. 4, lines 25-30, col. 5, lines 37-50, and col. 10, lines 37-64.	1-4 and 9-14 ----------- 5-6 and 11
X --- Y	US 4,692,944 A (MASUZAKI et al) 08 September 1987, se abstract, col. 2, lines 11-35, col. 2, line 49 to col. 4, line 19, and claims 6 and 12.	1-3, 7-8 ----------- 5-6, and 11

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier application or patent published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 25 February 2002 (25.02.2002)	Date of mailing of the international search report 26 MAR 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Dave Moore [illegible] Telephone No. 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)