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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 January 2015 (29.01.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/013525 A2

(51) **International Patent Classification**:
G06T 5/00 (2006.01) ***G06T 7/40*** (2006.01)
G06T 7/00 (2006.01)

(21) **International Application Number**:
PCT/US2014/048042

(22) **International Filing Date**:
24 July 2014 (24.07.2014)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
13/949,799 24 July 2013 (24.07.2013) US

(71) **Applicant**: **GEORGETOWN UNIVERSITY** [US/US]; 37th and O Street, N.W., Washington, DC 20057 (US).

(72) **Inventors**: **FRIEDER, Ophir**; 37th And O Street, NW, Washington, DC 20057 (US). **FRIEDER, Gideon**; 37th And O Street, NW, Washington, DC 20057 (US). **PARKER, Jon**; 37th And O Street, NW, Washington, DC 20057 (US).

(74) **Agents**: **RYAN, Matthew, K.** et al.; Frommer Lawrence & Haug LLP, 745 Fifth Ave., New York, NY 10151 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

WO 2015/013525 A2

(54) **Title**: SYSTEM AND METHOD FOR ENHANCING THE LEGIBILITY OF IMAGES

(57) **Abstract**: Disclosed ate embodiments for a system, method, and computer program product for performing a process on an original image, the process being implemented by a computer system comprising at least one computer: performing a process on at least one image that renders the processed image more legible than then the original image.

SYSTEM AND METHOD FOR ENHANCING THE LEGIBILITY OF IMAGES

RELATED APPLICATIONS

[0001] The present application is a continuation in part of U.S. Application No. 13/949,799, filed on July 24, 2013, titled "System and method for enhancing the legibility of images," the entirety of which is incorporated herein by reference.

BACKGROUND

[0002] All documents that are not properly preserved suffer degradation over time, and even the most rigorous preservation techniques can only slow degradation of physical documents. Degradation can occur for a variety of reasons, such as time, improper storage, poor environmental conditions, damage, and so on. Documents vulnerable to such stresses and degradation can contain valuable information, as in the case of found military/terrorism-related documents, historical documents, scanned legal documents, etc. Computerized scanning and imaging of such documents can "freeze" the state of the document at the time of imaging.

[0003] Additionally, some documents are not highly legible at creation. Such conditions can occur due to improper typing, improper writing, or improper media. The legibility of these documents can likewise be enhanced.

SUMMARY

[0004] Embodiments of the invention may be implemented by systems using one or more programmable digital computers and computer readable storage media. Disclosed are systems, methods, and computer readable storage products for performing a process on an image, the process being implemented by a computer system comprising at least one data storage device in which is stored image data, at least one computer and at least one computer readable medium

storing thereon computer code which when executed by the at least one computer performs a method, the method comprising the at least one computer performing a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state. In an embodiment, the method comprises the at least one computer: converting the image into a grayscale image; isolating a plurality of pixels near an edge; isolating locally dark or light pixels; computing an intersection between the edge detected image output and the locally dark or light image output and outputting a combined black and white image; and cleaning the outputted combined image.

[0005] The isolating the plurality of pixels near an edge can further comprise: performing edge detection on the image; applying a smoothing operation to the image; computing the local color contrast for each pixel in the smoothed image; and isolating the pixels with high local color contrast.

[0006] The edge detection can be performed by any edge detection known in the art, including for example a Sobel operation, Canny, Canny–Deriche, Differential, or Prewitt Roberts Cross. The edge detection smoothing operation can be performed by any smoothing operation known in the art, including for example applying a Bilateral Filter.

[0007] The computation of the pixel's local color contrast from the smoothed image can comprise: performing a local standard deviation (StdDev) operation on the smoothed image. The local StdDev operation can comprise, for each pixel of an image: identifying a square window around a home pixel; computing the standard deviation (StdDev) of the color values within the window of pixels; storing the StdDev color value of the pixel, and normalizing the computed StdDev values such that the values range from 0 to 255. As will be appreciated, other statistical measures known in the art can be applied.

[0008] The isolating pixels with a high local color contrast can comprise: performing clustering on the locally contrasted image. The clustering operation can be any known clustering method in the art, for example such as Otsu's Method or K-means clustering.

[0009] The isolating pixels with locally dark or light pixels can comprise: for each pixel of the original grayscale image: identifying a square window of the pixels centered around a home pixel, wherein the window is larger than a square window used when isolating the plurality of pixels near an edge; perform a clustering operation to the defined square window of pixels; and save the black or white color value of the home pixel. For example, the square window used for isolating locally dark or light pixels can be about 25% larger than the square window used to isolate pixels that are near an edge. The clustering operation can be known clustering methods such as, for example, Otsu's Method or K-means clustering.

[0010] The cleaning of the outputted combined image can comprise: removing stray pixels from a black and white image; and correcting any erroneous plateaus.

[0011] The removing of stray pixels from the black and white image can comprise, for each window of at least 9 pixels: counting the number of black pixels within the window; counting the number of white pixels within the window; detecting if a threshold number of pixels within the window are the opposite color of the home pixel; and if so, changing the color value of the home pixel to the opposite color value. The color value threshold can be at least 7 pixels.

[0012] Correcting erroneous plateaus can comprise, for identified plateaus: computing the mean and standard deviation of the original pixel colors in an island portion of the identified plateau; computing the mean and standard deviation of the original pixel colors in a border portion of the identified plateau; performing a statistical test on the plateau to determine if the island

portion of the plateau is part of the border portion of the plateau; and if the island portion of the plateau is a part of the border portion of the plateau, correcting the identified erroneous plateau so the island portion and the border portion of the plateau are the same color value.

[0013] According to an embodiment, the present application is directed to a system and method for improving the image processing performance of a computing system. The method comprises dividing an image processing algorithm into a plurality of processes; allocating the plurality of processes among computing modules of the computing system; dividing an input image into a plurality of slices that are to be processed in parallel; and recombining the processed slices to form an output. The image processing algorithm is configured to perform a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state.

[0014] According to an embodiment, the method further determines computer-intensive processes based on a consumed computer resource. The consumed computer resource can be a computing time of each process.

[0015] According to an embodiment, each slice includes additional portions that include pixels of an adjacent slice.

[0016] According to an embodiment, the processing system includes a graphic processing unit, and computer-intensive processes are allocated to the graphic processing unit.

[0017] According an embodiment, the pipelining step dynamically allocates processes among the computing modules of the processing system.

[0018] According to an embodiment, the processing system includes a central processing unit, and computer-intensive processes are allocated to the central processing unit.

[0019] According to an embodiment, the image processing algorithm includes at least one of the processes comprising: converting the input image into a grayscale image; performing edge detection on the image; isolating a plurality of pixels near an edge; isolating locally dark or light pixels; computing an intersection between the edge detected image output and the locally dark or light image output and outputting a combined black and white image; applying a smoothing operation to the image, a local standard deviation (StdDev) operation; and an Otsu's method operation; computing the local color contrast for each pixel in the smoothed image; isolating the pixels with high local color contrast; and cleaning an outputted combined image.

[0020] According an embodiment, the converting process is assigned to the central processing unit, and is not subject to a parallel processing operation.

[0021] According an embodiment, the image processing algorithm includes a process sharpening the grayscale image.

[0022] According to an embodiment, the processing system includes a plurality of central processing units, and the sharpening process is subject to a parallel processing by the plurality of central processing units.

[0023] According to an embodiment, the processing algorithm includes a process for computing a local standard deviation, and the allocating step allocates the computation of the local standard deviation process to the graphical processing unit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] Figs. 1A-1G are flow charts presenting a computerized image processing method of enhancing the legibility and clarity of image data for documents.

[0025] Fig. 2 shows an example of an original color image and a corresponding converted grayscale image.

[0026] Figs. 3A-3D shows exemplary views of intermediate image inputs and outputs.

[0027] Figs. 4 shows a grayscale image and the set of locally dark pixels identified from that grayscale image.

[0028] Figs. 5 shows image outputs and the combination of the output images.

[0029] Fig. 6 shows a plateau and its surrounding boarder.

[0030] Figs. 7A-7B show comparisons of an original image and processed enhanced images with a manually created goal solution image.

[0031] Figs. 8A-8D show original images compared against final enhanced images.

[0032] Fig. 9 shows an exemplary computer system and architecture for carrying out the method for image enhancement.

[0033] Figure 10 shows an exemplary network environment for carrying out the method for image enhancement.

[0034] Fig. 11 shows an exemplary a parallelization process according to an embodiment.

[0035] Fig. 12 illustrates an exemplary embodiment of a pipelining system and method.

[0036] Fig. 13 illustrates an exemplary embodiment of a dynamically adjusted pipeline of the system and method of Fig. 12.

[0037] Fig. 14 illustrates an embodiment of a pipelining system and method.

[0038] Fig. 15 illustrates an exemplary embodiment of a dynamically adjusted pipeline of the system and method of Fig 14.

[0039] Fig. 16 illustrates an exemplary embodiment of a dynamically adjusted pipeline of the system and method of Fig 14.

DETAILED DESCRIPTION OF EMBODIMENTS

[0040] Embodiments of a system and method for enhancing readability of scanned document images are described herein. Embodiments as described herein operate on each document image individually, hence are completely data parallel. In certain embodiments of the disclosed invention no training data or document model is required unlike other approaches that require training data or a document model. Via principle component analysis, edge detection and a local color contrast computation, an automated (namely, user independent) method removes image background noise and improves sharpness of the scripts and characters.

[0041] In embodiments disclosed herein, document image enhancement is training set independent, document model independent, and document language agnostic. Embodiments are applicable to any application that processes scanned documents. These include the processing of found military/terrorism-related documents, historical documents, scanned legal documents, etc. In short, this method can be applied to any corpus of documents that are degraded. In

various embodiments, at least one degraded image (e.g., due to degradation over time and/or due to improper storage) is provided as input and at least one black and white image clearly showing the content of the pre-degraded input image, including content intended to be readable or legible is derived as output.

[0042] It is to be understood that the figures and descriptions of the present invention are simplified to illustrate elements that are relevant for a clear understanding of the present invention, while eliminating, for purposes of clarity, many other elements which are conventional in this art. Those of ordinary skill in the art will recognize that other elements are desirable for implementing the present invention. Since such elements are well known in the art and do not facilitate a better understanding of the present invention, a discussion of such elements is not provided herein.

[0043] The present invention will now be described in detail on the basis of exemplary embodiments.

[0044] One embodiment discloses a method comprising: performing an automated (user independent) process on an image that renders the image from a first state to a second state, wherein the second state is more legible than the first state, wherein the process segregates dark pixels from light pixels. The analysis includes methodology and system configuration that embodies the assumed "truths" that (1) "writing"or "script" will be darker than the local pixels (i.e., background), and (2) "writing" or "script" should generate a detectable edge. Therefore, the logic takes advantage of these truths to find pixels within the scanned images that are both (1) darker than their neighbors and (2) near an edge. As will be understood, the first truth can be inverted for "negative" images, such as images of lithographs or negatives, in which case script is lighter than local pixels and the script will generate a detectable edge. As used herein, script or writing is to be construed broadly as any kind of symbol, figure, script, icon, drawing, and so on intended to be legible.

[0045] Figures 1A-1G are flow charts presenting a method of enhancing the legibility and clarity of image data for documents according to an embodiment of the invention. The method is configured to work on an image, as for example a degraded image, that is input into an image database as an image file (e.g., by scanning, digital photograph, etc.). An image as used herein is discussed on an image by image basis, (e.g., page by page), where each image is a "page" of a document in an image database; however, an original image should be broadly understood as a conventional image consistent with that as understood in the art. At Figure 1A is shown a high level flow for embodiments whereby the system and method is configured to isolate script pixels based that are (a) proximate to an edge and (b) are either locally dark or locally light. In the embodiments described herein, locally dark pixels are isolated, however, as will be appreciated, the methods and systems can be readily configured to isolate and filter locally light pixels (e.g., for a negative).

[0046] In an embodiment, the method for performing a process on an image, is implemented by a computer system performing the method, the method comprising the at least one computer performing a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state, the operations comprising:

converting the image into a grayscale image 20;

isolating a plurality of pixels near an edge 21;

isolating locally dark (or light) pixels 33;

computing an intersection between the edge detected image output and the locally dark (or light) image output and outputting a combined image 34; and

cleaning the outputted combined image 35.

[0047] With reference to Figure 1A, at block 20, if the original image is in color, the image is first prepared by rendering into a grayscale image using methods and techniques known to ordinarily skilled artisans. For example, a conventional technique for converting a color image to a grayscale image is using the standard

Linear Algebra technique known as Principle Component Analysis (PCA). Note that other techniques known in the art can be used to produce viable grayscale images. This step need not be implemented if the original image is in grayscale. An original image has Width w and Height h, resulting in n total pixels ($n = w * h$) for the image. For a color image, each pixel is a 3-dimentional (3-d) vector: $Pixel_i = (red_i, green_i, blue_i)$. The system is configured to flatten or compress the n 3-d vectors into n 1-d vectors, that is, into numerical values. The result is n numbers, one for each pixel. If PCA is used to generate the corresponding grayscale image, namely to compute n numbers, PCA is performed on the n 3-d vectors of the image retaining only the first component (i.e., retaining only the most significant dimension). The resulting numbers are normalized to range from 0 to 1. These normalized numbers are then scaled from "black" to "white," resulting in a grayscale image of the original color image. An example of an original color image 200 being converted to a grayscale image 202 by PCA is shown at Figure 2. As noted before, other transformations known in the art can be used.

[0048] Referring to Figure 1A, at block 21 a pixel area for each pixel of the image is processed to isolate a plurality of pixels near an edge. In an embodiment shown in Figure 1B, the method for isolating the plurality of pixels near an edge further comprises:

performing edge detection on the image 22;

applying a smoothing operation to the image 23;

computing the local color contrast for each pixel from the smoothed image 24; and

isolating the pixels with high local color contrast 29.

[0049] Turning to Figure 1B, taking the grayscale image, at block 22, the method comprises determining if a pixel is proximate to an edge, an embodiment of which is shown at Figure 1C. As shown at block 22 of Figure 1B, the determination comprises performing edge detection on the grayscale image 202. Edge detection can be carried out using edge detection techniques known to

ordinarily skilled artisans. As will be appreciated, image edge detection is a well-documented image processing topic, and includes techniques such as Sobel Operators, Canny Edge Detection, including Canny, Canny–Deriche, Differential, Sobel, Prewitt Roberts Cross, and others. In one embodiment, a Sobel Operation is performed. Edge detection works by computing or estimating a local color gradient for pixel areas. If the gradient is steep, edge detection proceeds on the basis that as colors or shades are changing quickly, there is an "edge." Figure 3A shows a magnified view of a portion of the image input of grayscale image 202a, and the image output 206 after the determination edge detection is run using Sobel Edge detection. The resulting image lightens the areas where the shades are not changing quickly; it retains "edge" areas where shades are changing quickly. Consequently, in the output image 206, the pixels for script take an outlined appearance and the rest of the image having a "washed out" appearance. The result is that dark pixels typically form the boundary between text and non-text.

[0050] Next, as shown at block 23 of Figure 1B, a smoothing operation is applied to the image. As shown in Figure 3B in an exemplary embodiment of a smoothing function, a BiLateral Filter (BLF) is applied to the image 206 on which the Sobel Edge Detection was performed resulting in a smoothed image 208.

[0051] At block 24 of Figure 1B, once the smoothing operation is performed, the system computes the local color contrast for each pixel. In an embodiment, the local color contrast for each pixel from the smoothed image is computed by performing a local standard deviation (StdDev) operation on the smoothed image. As shown in Figure 1C, the local StdDev operation comprises for each pixel of the smoothed image: identifying a square window around a home pixel 25; computing the StdDev of the color values within the window of pixels 26; and storing the StdDev color value of the pixel 27.

[0052] Accordingly, starting at a block 25 of Figure 1C, the local color contrast of the smoothed image 208 is computed. At block 25 of Figure 1C, for each pixel of the smoothed image a square window of pixels centered around a home pixel is identified. In one embodiment, a window of neighboring pixels is a square of pixels centered around a home pixel such as a "center" pixel. The number of pixels in the window is quadratic. That is to say a window size of 9 corresponds to a pixel area containing 81 pixels. This pixel area contains the home pixel *h* and 80 neighboring pixels). A window size of 7 corresponds to a pixel area containing 49 pixels. A window size can take on the values of 3, 5, 7, 9, and so on. Each pixel is identified as a "home" pixel for an area including a plurality of pixels. For example, in an embodiment a 15 by 15 square pixel window is centered around a home pixel (i.e., 7 pixels on each of the sides of the home pixel). As will be appreciated, the area of the square about the home pixel can be enlarged (e.g., 17, 21) or reduced (e.g., 9, 13). However, as described at the appropriate section below, this window size will be smaller than a window sized established when isolating locally dark pixels during the process. As will be noted, the pixel length for the window is an odd number as the number includes the home pixel and the even number of pixels surrounding the home pixel.

[0053] At block 26, the standard deviation of the colors within the window of pixels is calculated, and at block 27, the value is saved as a standard deviation value (StdDev). Once all the pixels are computed thus, at block 28 the stored StdDev values are normalized such that the values range from 0 to 255. As shown in the example of Figure 3C, the resulting image depicts local color contrast 210 and is computed from the smoothed image 208 using the local StdDev operation.

[0054] At block 29 of Figure 1B, the method comprises isolating pixels that have a high local color contrast. The system can be configured to apply a clustering algorithm such as Otsu's Method, which is a computerized technique known to ordinarily skilled artisans for computer vision and image processing, which automatically performs histogram shape-based image thresholding,

namely, the reduction of a grayscale image to a strictly black and white image (0,1). As shown at Figure 3D, the application of Otsu's method to the Local StdDev output 210 of the smoothed image 208 outputs a high contrast black and white binary image 212. As will be appreciated, clustering methods other than Otsu's Method can be used, as for example K-means clustering.

[0055] Returning to Figure 1A, at block 33 the method comprises isolating script pixels based on a filter criterion that filters pixels that are locally dark (or light). Again, in the embodiments described herein, locally dark pixels are isolated, however, as will be appreciated, the methods and systems can be readily configured to isolate and filter locally light pixels (e.g., for a negative). In an embodiment, Figure 1D shows a flow for computing the locally dark pixels. At block 30, with the original grayscale image 202 as input, for each pixel of the original grayscale image 202, a square window of the pixels centered around a home pixel is identified. The window is larger than a square window identified for a smoothed image during a Local StdDev operation at (as shown at block 25 of Figure 1C). At block 31, the clustering operation such as Otsu's method or K-means clustering is applied to the defined square window of pixels, resulting in a binary values for each pixel, namely either black or white. At block 32, the system saves the black or white color value of the home pixel. The Otsu's method is applied thus to every pixel of the image 202.

[0056] As noted above the square window for the smoothed image as computed at blocks 25 of Figure 1C when isolating pixels near an edge is smaller than the square window for the original grayscale image computed at block 30 of Figure 1D when isolating locally dark or light pixels. For example, the square window computed at block 30 for the grayscaled image 202 can be about 25% larger than that computed at block 25. According to an embodiment, at block 30 the system can be configured to highlight a 21 by 21 square window of pixels centered around each home pixel of the grayscaled image 202, whereas a 15 by 15 window is employed as described above at block 25 of Figure 1C. As will be appreciated, the size of these windows may vary, as the image cleaning process described

below improves image legibility. It is the case, however, that the pixel window for the image processing at block 30 must be larger than that of blocks 25 as described above, for example between 20-40% larger.

[0057] At block 31 of Figure 1D, a clustering method is applied to a window of pixels, and the white/black color of the home pixel is saved. This process is carried out for each pixel of the original image. As noted herein, while the clustering of the exemplary embodiment applies Otsu's method, any clustering method known in the art can be applied. As shown at Figure 4, the application of Otsu's method to the original grayscale image 202 outputs a high contrast black and white binary image 214. In the image 214 is such that the script is black and has the appearance of stark white highlighting, while the remaining "background" is speckled in high contrast black and white depending on the grey shade values of the original grayscale image 202.

[0058] Returning to Figure 1A, at block 34 the system is configured to compute an intersection between the edge detected image output 212 and the locally dark/light image output 214 and outputting a combined image 216. As shown at Figure 5, in one embodiment, this is achieved by combining the image with the isolated edge detected output 212 (as described herein with respect to block 21 of Figure 1A and Figures 1B-1C) with the pixels segregated into light pixels and dark pixels as output 214 (as shown at block 33 of Figure 1A and Figure 1D). Figure 6 shows a magnified view of portions of the image inputs of the segregated image 212 and the image 214 and the resulting enhanced image output 216. The resulting enhanced image 216 cross-references the locally dark pixels segregated in image 214 and the script that is proximate to an edge from the clustered image 212. Only pixels that meet both criteria are shown in black in the enhanced image 216, and the rest of the image is white, resulting in a crisp, highly legible image.

[0059] Returning again to Figure 1A, at block 35 the system is configured to clean the combined image 216. As shown at Figure 1E, the cleaning comprises removing stray pixels 36; and correcting plateaus 41.

[0060] As shown in the embodiment described at Figure 1F, the removal of stray pixels from the black and white image comprises, for each window of pixels: counting the number of black pixels within the window; counting the number of white pixels within the window; detecting if a threshold number of pixels within the window are the opposite color of the home pixel; and if so, changing the color value of the home pixel to the opposite color value for each center or home pixel of a window. In accord with the description herein, a window size can take on the values of 3, 5, 7, 9, and so on. Each pixel is identified as a "home" pixel for an area including a plurality of pixels. In the embodiment described below, the exemplary number of pixels in the window is at least 9 (a window size of 3), however the pixel window can have any odd number of pixels depending on the size of the window around a home or center pixel.

[0061] In an embodiment with a window size of at least 3, the removal of stray pixels comprises detecting color values of the home pixel (1) and a number (at least 8) of nearest surrounding pixels (together equaling at least 9 pixels) 37; and detecting if a threshold number of pixels is an opposite color value than the color value of the home pixel 38 by counting the number of black pixels within the window and counting the number of white pixels within the window. If the detected opposite color values of the pixels meets or exceeds the threshold number 39, the system if configured to change color value of the center pixel to the opposite color value 40. As will be appreciated, as the combined image 216 has only binary colors of black and white (dark and light) pixels, the color value is in every case one or the other, that is, black or white.

[0062] In an embodiment where a home pixel and its nearest surrounding neighbors are be a count of 9 pixels, that is, the center or home pixel *h* and the

eight (8) neighboring pixels that surround it on all sides (a window size of 3), the color value threshold is at least 7 pixels as lower thresholds, while possible, can result in more degraded image results. This is because a window size of 3 having 9 pixels is arranged 3x3 in a grid-like arrangement in the square window, in which case a simple majority of 6 same-color values can leave the 3 opposite color pixels occupying the same row of pixels or one column of pixels, as shown in Table 1. Thus, by setting the threshold to require at least 7 pixels, the risk of degradation due is reduced. For example, in a window of 9 pixels (window size 3) with a color value threshold of 7, if there are 7-8 black pixels, the center pixel *h* is changed to black if the center or home pixel were previously white, as shown in Table 2 Contra, if 7-8 of the pixels are white, then the center pixel *h* is changed to white if the center or home pixel *h* were previously black. If, on the other hand, the color threshold is not met (i.e., 6 pixels or under are an opposite color) then the center pixel is left unchanged, as shown in Table 3.

[0063]

Black	White	Black
Black	White	Black
Black	White	Black

Table 1, showing a simple majority (6) of Black pixels leaves a row of same colored White pixels, including the center pixel, that are not in the majority.

[0064]

Black	Black	Black

Black	White to Black	Black
Black	White	Black

Table 2, showing an example where for a threshold of 7, the system is configured to change the center White pixel to a Black pixel where there are at least 7 Black pixels in the window.

[0065]

Black	White	White
Black	White (no change)	White
Black	White	White

[0066] Table 3, showing an example where for a threshold of 7, the system is configured to not to change a center White pixel where there are at least 6 White pixels in the window.

[0067] As will be appreciated, if the window grows to encompass more than the minimum 8 pixels of pixels around the home pixel than the threshold will also grow beyond the value of 7 or greater. As noted above, while a simple majority can be employed, a threshold should be configured such that every row and column of the pixels includes at least one pixel having the same black or white color-value as the majority number of pixels in the square. That is to say, system should be configured such that no row or column can have all the same black or white colored pixel values if those black or white color values are in the minority. For example for a window size of 5 encompassing a grid of 25 pixels or a window size of 7 encompassing a grid of 49 pixels, the threshold can be configured to be at thresholds of at least 21 and 43 pixels respectively. Hence for every window size n, having a grid of n*n pixels, the threshold can be at least (n * n-1) +1 pixels.

[0068] As shown in the embodiment described at Figure 1G, the correction of plateaus comprises, for the identified plateaus:

computing the mean and standard deviation of the original pixel color in an island portion of the identified plateau 42;

computing the mean and standard deviation of the original pixel color in a border portion of the identified plateau 43;

performing a statistical test, for example a two-sample z-test or other statistical tests as known in the art, on the plateau to determine if the island portion of the plateau is part of the border portion of the plateau 44; and

if the island portion of the plateau is a part of the border portion of the plateau 45, correcting the identified plateau so the island portion and the border portion of the plateau are the same color value 46.

[0069] As shown at Figure 6, a combined image 216 may have "island" portions inside script that are opposite the black or white color value of the border script 215, which are identified as plateaus. Plateaus can be the result of, for example, large fonts in a given image. In such an instance, as shown in Figure 1F, when a font is large the center of a letter may not be "near an edge," thus the processing can leave the island 215. Accordingly, in an embodiment the system is configured to determine if an island of a plateau is anomalous and if so, correct it, or determine if the island is in fact a correct representations of the original image 200. At block 42, the mean and standard deviation of the pixel color of the original pixel color from the original grayscale image 202 is computed. At block 43 a mean and standard deviation of the original pixel color of the border portion of the identified plateau is computed. At block 44, a two-sample z-test (a statistical operation) is performed on the plateau using the computed standard deviations from the original grayscale image 202 to determine if the island portion of the plateau is part of the border portion of the plateau. At block 45 if the island portion of the plateau is determined to be a part of the border

portion of the plateau 45, the identified plateau is corrected so the island portion and the border portion of the plateau are the same color value 46.

[0070] The system and method described herein show exemplary novel advantages. For example, a test set of comparative images can be drawn from the Document Image Binarization Contest (DIBCO). Held in 2009 and 2011 at the International Conference on Document Analysis and Recognition (ICDAR), the contest provided "difficult case" input images to be processed for legibility. A particularly difficult original image 200x is shown at Figure 7A, in which the original image 200x is highly texturized, a dark brown background, and has faded script. The imaged texture makes it very difficult for image processing systems to achieve correct binarization results that can distinguish and make legible script from the texturized and colored background. Figure 7A also shows an exemplary, manually produced "goal solution" 217 provided by DIBCO against which systems for binarizing images for legibility were to have image results judged. As shown in Figure 7B the manually produced exemplary "goal solution" 217 provided by the latest DIBCO is placed side-by-side with an image result 218 for the original document 200x processed by the present system and method. The side-by-side for comparison shows the image result 218 nigh mirrors the target solution 217.

[0071]

	1st Place	2nd Place	3rd Place	Described image processing
Mean F1	80.9	85.2	88.7	88.9
Median F1	92.3	92.1	90.6	89.7

Variance F1	794	302	49.7	19.2

[0072] Table 4

[0073] Table 4 presents a comparison of documents processed by the 1st, 2nd and 3rd place winners of the latest DIBCO competition as compared to the same documents processed using an embodiment of the system and method as described herein. DIBCO competitions score document results via the following criteria: (1) Recall, which measured the percentage of truly black pixels that were predicted to be black and (2) Precision, which measures the percentage of predicted black pixels that are truly black. An F1 Measure is the primary measure for judging, which measures in accord with the equation

[0074] $F1 = 2 \cdot (Precision \cdot Recall)/(Precision+Recall)$

[0075] As will be appreciated, higher Mean F1 values and higher Median F1 values represent better images. As shown in Table 1, the present system processed documents as well as or better than the winning systems with respect to Mean and Median F1 scores. However with a variably F1 of 19.2, the present system did so with significantly less variability, at an order of magnitude less than the 1st and 2nd place winners (792 and 304) and significantly less than the 3rd place winner (49.7). This means the present system produced images of high legibility at equivalent quality of the winners of the competition, and did so while processing documents with greater consistency, and hence, far fewer failures. In contrast, while latest DIBCO competition awards points for each test image, 1st and 2nd place frequently do very well but sometime fail outright (hence their mean performance is significantly lower than their median performance).

[0076] As shown at Figures 8A-8D, another dataset shows original images 200a-200d compared against final enhanced images 218a-218d. A dataset of

processed images is a collection of World War II era document images, the originals residing at the Yad Vashem Holocaust Memorial in Israel. Figures 8A-8D are further examples of original images 200a-200d and enhanced images 218a-218d, which show that a wide variety of script and symbols can be enhanced and made more legible, readable, and clear. For instance, Figures 8C and 8D show original historical documents 200c, 200d with highly degraded script which is handwritten. The output enhanced images 218c and 218d show highly readable and legible document images. As will be noted, degraded script that was faded or obscured in the original image is clearly legible enhanced script, and the enhanced script is even and consistent with the script of the whole image in the enhanced image 218c, 218d. Figures 8B and 8D show original images 200b, 200d and enhanced images 218b, 218d having both handwriting and typeset in the same image, and in Figure 8D, the typeset has handwritten edits. As shown in Figure 8C, the original image 200c and enhanced image 218c includes photos; the presence of the photos does nothing to impede the script enhancement, although the photo is treated by it. Thus in one embodiment the system could be configured to identify photos either before or after the script enhancement method for the purpose of reincorporating the original image or grayscale image back into the enhanced image (not shown).

[0077] Embodiments of the invention may be implemented by systems using one or more programmable digital computers and computer readable storage media. In one embodiment, Figure 9 depicts an example of one such computer system 100, which includes at least one processor 110, such as, e.g., an Intel or Advanced Micro Devices microprocessor, coupled to a communications channel or bus 112. The computer system 100 further includes at least one input device 114 such as, e.g., a keyboard, mouse, touch pad or screen, or other selection or pointing device, at least one output device 116 such as, e.g., an electronic display device, at least one communications interface 118, at least one computer readable medium or data storage device 120 such as a magnetic disk or an optical disk and memory 122 such as Random-Access Memory (RAM), each coupled to the

communications channel 112. The communications interface 118 may be coupled to a network 142.

[0078] One skilled in the art will recognize that many variations of the system 100 are possible, e.g., the system 100 may include multiple channels or buses 112, various arrangements of storage devices 120 and memory 122, as different units or combined units, one or more computer-readable storage medium (CRSM) readers 136, such as, e.g., a magnetic disk drive, magneto-optical drive, optical disk drive, or flash drive, multiple components of a given type, e.g., processors 110, input devices 114, communications interfaces 118, etc.

[0079] In one or more embodiments, computer system 100 communicates over the network 142 with at least one computer 144, which may comprise one or more host computers and/or server computers and/or one or more other computers, e.g. computer system 100, performing host and/or server functions including web server and/or application server functions. In one or more embodiments, a database 146 is accessed by the at least one computer 144. The at least one computer 144 may include components as described for computer system 100, and other components as is well known in the computer arts. Network 142 may comprise one or more LANS, WANS, intranets, the Internet, and other networks known in the art. In one or more embodiments, computer system 100 is configured as a workstation that communicates with the at least one computer 144 over the network 142. In one or more embodiments, computer system 100 is configured as a client in a client-server system in which the at least one other computer comprises one or more servers. Additional computer systems 100, any of which may be configured as a work station and/or client computer, may communicate with the at least one computer 144 and/or another computer system 100 over the network 142.

[0080] For example, one or more databases 146 may store the scanned image data as described herein. In various embodiments, the processing disclosed herein may be performed by computer(s)/processor(s) 144 in a host arrangement

with computer system 100, or in a distributed arrangement in computer system 100 and computer(s)/processor(s) 144, or by computer system 100 in cooperation with data stored in database 146. Computer(s)/Processor(s) 144 may perform the processing disclosed herein based on computer code stored in a storage device or device(s) 120, 136, 138 and/or memory 122.

[0081] Figure 10 shows an exemplary network environment 400 adapted to support embodiments as disclosed herein, as for example for data parallel processing of images. The exemplary environment 400 includes a network 142, and a plurality of computers 100, or computer systems 100 (a)...(k) (where "k" is any suitable number). Computers could include, for example one or more SQL servers. Computers 100 can also include wired and wireless systems as described herein. Data storage, processing, data transfer, and program operation can occur by the inter-operation of the components of network environment 400. For example, a component including a program in server 100(a) can be adapted and arranged to respond to data stored in server 100(b) and data input from server 100(c). This response may occur as a result of preprogrammed instructions and can occur without intervention of an operator. As described herein, in certain embodiments the automated method is configured to process images individually on an image-by-image basis, where each image is a "page" of a document in an image database. Accordingly, the system can be configured for data parallel processing of images and pages. Pages or images from a given document or set of documents can be partitioned and distributed among the computer systems 100 (a)...(k) for parallel processing and the document or document set recombined after processing. Again, this response may occur as a result of preprogrammed instructions and can occur without intervention of an operator.

[0082] The network 142 is, for example, any combination of linked computers, or processing devices, adapted to access, transfer and/or process data. The network 142 may be private Internet Protocol (IP) networks, as well as public IP networks, such as the Internet that can utilize World Wide Web (www) browsing

functionality, or a combination of private networks and public networks of any type.

[0083] A computer 100(a) for the system can be adapted to access data, transmit data to, and receive data from, other computers 100 (b)...(k), via the network or network 142. The computers 100 typically utilize a network service provider, such as an Internet Service Provider (ISP) or Application Service Provider (ASP) (ISP and ASP are not shown) to access resources of the network 142.

[0084] The computers 100 may be operatively connected to a network 142, via bi-directional communication channel, or interconnector, 118, which may be for example a serial bus such as IEEE 1394, or other wire or wireless transmission media. Examples of wireless transmission media include transmission between a modem (not shown), such as a cellular modem, utilizing a wireless communication protocol, or wireless service provider or a device utilizing a wireless application protocol and a wireless transceiver (not shown). The interconnector 118 may be used to feed, or provide data.

[0085] The terms "operatively connected" and "operatively coupled", as used herein, mean that the elements so connected or coupled are adapted to transmit and/or receive data, or otherwise communicate. The transmission, reception or communication is between the particular elements, and may or may not include other intermediary elements. This connection/coupling may or may not involve additional transmission media, or components, and may be within a single module or device or between one or more remote modules or devices.

[0086] The terms "client" and "server" may describe programs and running processes instead of or in addition to their application to computer systems described above. Generally, a (software) client may consume information and/or computational services provided by a (software) server.

[0087] Various embodiments of the invention are described herein with respect to scanned image databases and systems related thereto. However, it is to be understood that the invention has application to other image data where, inter alia, legibility and readability of obscured image files are desired.

[0088] *Optimization of Image Processing By Pipelining and Parallelization*

[0089] When a large collection of images needs to be processed, it is desirable to optimize the performance of the methods and algorithms of the image processing system to shorten the processing time, increase the processing capability, or improve the quality of a current processing system, which may be a plurality of servers distributed on a network, a server, a computer, a laptop, or even a mobile computer device such as a mobile phone. According to some embodiments, application parallelization process can be implemented to divide an image into a plurality of slices or portions such that each computing unit may take one slice and implement predetermined image processing algorithms and processes thereof. In embodiments, a pipelining process can be implemented to divide the image processing algorithm into a plurality of processes and sub-processes and selectively group these processes and sub-processes into various stages, each of which can be handled by a separate computing module of a processing system. Embodiments include optimization systems and methods described below can be applied to the image processing algorithms described herein. Embodiments can also be employed for other image processing, for example, as described in U.S. Patent Application No. 13/301,436, filed on November 21 2011 and entitled SYSTEM AND METHOD FOR ENHANCING THE LEGIBILITY OF DEGRADED IMAGES, the entirety of which is incorporated by reference hereby.

[0090] According to an embodiment, described is an optimization system and method including a profiling operation, in which the image processing algorithm is profiled to reveal the computer resources consumed by each process or task of the algorithm. The consumed computer resource may include any metric or

measure that is used to indicate the expense of a computing process. According to en example, computing time is used as an exemplary consumed computer resource. For example, computing time of the various processes image processing algorithms as described herein can be obtained during processing, for example, when a system processes images like those shown in Figs. 2-8, and are shown in Table 5.

[0091] Table 5. Timings for different steps of the image binarization algorithm when implemented as a serial process

[0092]

Step	Percent of Compute Time
Compute greyscale image	4.4%
Identify locally dark pixels	-
Apply Gaussian blur	0.6%
Compute local Otsu	**36.5%**
Identify pixels near an edge	-
Sharpen greyscale image	4.0%
Apply Sobel edge detection	4.3%
Compute local standard deviations	**48.9%**
Otsu	<0.1%
Compute intersection	0.1%

[0093]

[0094]

[0095]

[0096]

[0097] With the results of a consumed computer resource, the optimization method determines which process or processes are "expensive" or computer-intensive, for example when a process a substantial amount of the computer resource, such as a greater percentage of computing time, than other processes. According to Table 5, it is observed that the two processes "Compute local Otsu" and "Compute local standard deviations," which as described above repeatedly analyze local snippets of pixels, make up the majority of the computation and together represent 85.4% of the overall compute time. From a performance point of view, one advantage is speeding up these two processes that perform a "for each snippet of local pixels" type of operation. As will be appreciated, embodiments are not limited to using a serial process to obtain a

consumed computer resource of each process. A parallel process can also be used to obtain the consumed computer resource of each processing task.

[0098] *Parallelization*

[0099] According to an embodiment, image processing algorithms and methods as described herein can be carried out substantially in parallel among a plurality of computing modules of a processing system. For example, when a plurality of servers or computers work cooperatively with each other for processing images, each computer or server may be regarded as a computing module. A single computer may also include multiple Central Processing Units ("CPU") and/or multiple Graphical Processing Units ("GPU"). Each of the CPU or GPU may be a computing module. The parallelization method is capable of assigning processing tasks to each CPU, GPU, and/or compute based on their processing capabilities.

[00100] In an embodiment, the parallelization as set forth herein is applied based on information about local pixels from the image processing algorithm. For example, a scanned image may be divided into n slices, and each slice is processed as if it were the complete input with slight modifications. The optimization method combines the n partial results into one complete result, which represents an effective way to allow multiple processors or cores to reduce the wall clock time it takes to execute those computationally expensive steps.

[00101] According to an embodiment, a slight modification is made such that each of the n slices will be slightly larger than a strictly divided 1/nth of the scanned image. Slices are enlarged by the radius of the snippet window because processing pixels near the strict 1/nth dividing line requires information about pixels on both sides of that dividing line. Enlarging the slices ensures that pixels inside each slice have access to the pixel values of adjacent slices. It is noted that enlarging the slices does not generate a substantial amount of additional work, if

the array containing the input values can be shared, because output values for the pixels in the "extra" portion of a slice need not be computed.

[00102] Fig. 11 shows an example of such a parallelization process 1100 according to an embodiment. A scanned image 1102, to be processing according to an image processing algorithm as described herein, is divided into a plurality of slices 1104(1) to 1104(n). As shown in Fig. 11, the scanned image 1102 is divided into four slices along its horizontal direction 1110. According to some embodiments, a scanned image may be divided along its vertical direction. According to some embodiments, the number of slices and the manner of division of a scanned image are determined according to the capability of a processing system and the steps to be separately handled by each processing unit. For example, the number of slices is determined based on the capability of GPU and/or the relative capability of GPU over CPU in the same processing system. Each divided slice 1104(1)...1104(n) is processing to produce a second state slices 1106(1)...1106(n), which has improved legibility. The second state 1108 of the scanned image 1102 is produced by combining the plurality of the second state slices 1106(1)...1106(n) from the horizontal direction 1110.

[00103] According to embodiments, the division of a scanned image into n slices is implemented so as to avoid memory contention and data movement, thus reducing the risk of memory contention and unnecessary data movement by duplicating the array storing the image's grayscale values. To avoid a wasteful duplication of data, one array is stored in memory configured to be accessed by multiple cores of the computer system without giving rise to performance penalties caused by memory contention. According to embodiments, an array of data is stored in a shared memory and is accessed in consistent manner (i.e., from top left to bottom right) that minimizes the risk of memory contention because no two or more processors, such as multiple Graphical Processing Unit ("GPU") cores, are likely to access the same portion of the array of grayscale values simultaneously. Memory contention effects in systems with GPUs are described in "Quantifying numa and contention effects in multi-gpu systems," Proceedings

of the Fourth Workshop on General Purpose Processing on Graphics Processing Unit, p. 11, ACM, March 2011, the entirety of which is incorporated herein by reference.

[00104] *Pipelining*

[00105] According to an embodiment, the system can be configured with a workflow that allocates separated tasks to one or more cores of the GPUs. Generally speaking, a GPU represents a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. GPUs are very efficient at manipulating computer graphics by employing a parallel structure that processes large blocks of data in parallel. Modern GPUs may have hundreds or even thousands of cores or processors that can implement computing process in parallel. According to embodiments, the separation of processing tasks and allocation of the separated proceeses of an image processing algorithms may be pre-set before the processing of scanned image or may be dynamically adjusted according to the relative capabilities among computing modules of a processing system. This separation and allocation of processing tasks among GPUs may also be referred to as "pipelining."

[00106] Pipelining increases instruction throughput and enables a form of parallel processing by distributing the execution of program instructions to dedicated single purpose components within the CPU, like the arithmetic logic unit ("ALU") and the registry. One reason pipelining improves throughput is that it ensures as many internal CPU components as possible are doing useful work at any one time. According to some embodiments, the algorithm as set forth in the present application is pipelined in a processing system by allocating its sub-processes to different stages where each stage will be processed by a different piece of hardware. According to some embodiments, a dynamic

pipelining method is implemented such that the allocation of sub-processes to different stages can adapt at runtime to reflect the number of available CPU cores and the capability of the GPU.

[00107] *Pipelining with 1 CPU Core*

[00108] Fig. 12 illustrates a pipelining method 1200 according to an embodiment. As shown, a system employs one CPU core, and the initial pipeline has two stages. The first stage 1204 is executed in the CPU and has three computationally inexpensive operations, including computing the grayscale image 1202, sharpening the grayscale image 1208, and performing Sobel edge detection 1210. The second pipeline stage 1206 is executed in parallel by numerous plurality of GPU cores. This GPU stage 1206 includes several expensive computer-intensive operations, such as blurring images 1212, computing local Otsu 1214, computing local standard deviation 1216 and computing Otsu 1218, as well as computing a final intersection 1220. While the first set of images are pushed through, the average times to execute stage 1 and stage 2 are recorded. These initial timing measurements are used to balance the pipeline stages for future images.

[00109] For example, the initial 2-stage pipeline in Fig. 12 may not be altered if the second stage 1206 of the pipeline takes longer than the first stage 1204 on average. On the other hand, when stage 1 1204 takes longer than stage 2 1206, the pipeline can be adjusted such that future images are processed with a pipeline that allocates more work to the GPU.

[00110] Fig. 13 shows an updated pipeline 1300 according to an embodiment. Comparing with the pipeline in Fig. 12, the pipeline 1300 in Fig. 13 moves the edge detection process 1210 from the first stage 1204 to and implements the edge detection process 1302 in the second stage 1206 to balance the processing time between the first stage and the second stage. Such an alternation of the pipeline is based on the first batch of timing measurements or the first few

batches of timing measurements taken for the first pipeline 1200. After the alternation, the next few images through the pipeline are also timed and once again the pipeline may or may not alter its configuration. If the new timing data supports a change, the image sharpening step will also be moved to the GPU stage 1206.

[00111] According to some embodiments, regardless of whether or not the sharpening process 1208 is incorporated into the GPU stage, moving the "to grayscale" process 1202 may not be considered because it can require parallelizing principal component analysis which is technically possible, but may not be amenable to GPU computing.

[00112] *Pipelining with 2 or More CPU cores*

[00113] Embodiments can employ any number of CPU cores. For example, Fig. 14 illustrates a pipeline 1400 with 3 CPU cores according to an embodiment. As shown in Fig. 14, each CPU is assigned a processing task. For example, CPU1 1402 is assigned the process 1404 of converting a scanned image into a grayscale image. CPU2 1406 is assigned the process of sharpening the image, and CPU3 1408 is assigned the process of detecting edges. The GPU 1410 is taking the same processes as those assigned in Fig. 12.

[00114] Fig. 15 illustrates a pipeline 1500 with 5 CPU cores according to an embodiment. As shown in Fig. 15, CPU1 1502 is assigned the process 1504 of converting a scanned image into a grayscale image. It is noted that the conversion step 1504 is not parallelized. At stage 2 1506, two CPUs, such as CPU2 and CPU3, are assigned the process of sharpening the image. At stage 3 1508, two CPUs, such as CPU4 and CPU5, are assigned the process of detecting edges. The GPU 1510 is taking the same process as those assigned in Fig. 12.

[00115] When a plurality of CPUs are used in a processing system, expanding the responsibilities of the GPU stage employs a change to the "work absorption"

process used in the 1 CPU case as shown in Fig. 12 and Fig 13. In the multi-CPU case, absorbing a process into the GPU stage shortens the pipeline by one stage and frees up the CPUs that were responsible for that stage. Fig. 1600 represents a modified pipeline 1600 produced based on the pipeline 1500 according to an embodiment. If the stage 3 edge detection step 1508 in Fig. 15 is moved or absorbed into the stage 4 GPU stage, then as shown in Fig. 16 an updated pipeline 1600 has only three stages 1602, 1606, and 1610 as the edge detection 1508 is no longer performed by CPU4 and CPU5 as shown in stage 3 of Fig. 15. The stage 1602, which converts the scanned image into a grayscale image, remains the same in Fig. 15. As shown in Fig. 16 the second stage 1606, which has four CPUs for processing, uses all four CPUs to implement the image sharpening process in parallel. The GPU stage 1610 absorbs the edge detection process in addition to all the other process of Fig. 15.

[00116] The pipelining approach is designed to accelerate the processing of large corpuses of images by dynamically balancing the distribution of work between one or more CPU cores and the numerous cores of a GPU. Dynamic pipelining begins by using a simple pipeline that allocates only the computationally expensive portions of the image binarization algorithm to the GPU. After timing measurements are taken, the pipeline may or may not allocate more work to the GPU if this reallocation will improve the pipeline's balance and increase throughput.

[00117] While the invention has been described and illustrated with reference to certain preferred embodiments herein, other embodiments are possible. Additionally, as such, the foregoing illustrative embodiments, examples, features, advantages, and attendant advantages are not meant to be limiting of the present invention, as the invention may be practiced according to various alternative embodiments, as well as without necessarily providing, for example, one or more of the features, advantages, and attendant advantages that may be provided by the foregoing illustrative embodiments.

[00118] Systems and modules described herein may comprise software, firmware, hardware, or any combination(s) of software, firmware, or hardware suitable for the purposes described herein. Software and other modules may reside on servers, workstations, personal computers, computerized tablets, PDAs, scanners (including handheld scanners), digital cameras and camcorders, and other devices suitable for the purposes described herein. Software and other modules may be accessible via local memory, via a network, via a browser or other application in an ASP context, or via other means suitable for the purposes described herein. Data structures described herein may comprise computer files, variables, programming arrays, programming structures, or any electronic information storage schemes or methods, or any combinations thereof, suitable for the purposes described herein. User interface elements described herein may comprise elements from graphical user interfaces, command line interfaces, and other interfaces suitable for the purposes described herein. Except to the extent necessary or inherent in the processes themselves, no particular order to steps or stages of methods or processes described in this disclosure, including the Figures, is implied. In many cases the order of process steps may be varied, and various illustrative steps may be combined, altered, or omitted, without changing the purpose, effect or import of the methods described.

[00119] Accordingly, while the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the scope of the invention, and the invention is thus not to be limited to the precise details of methodology or construction set forth above, as such variations and modification are intended to be included within the scope of the invention. Therefore, the scope of the appended claims should not be limited to the description and illustrations of the embodiments contained herein.

What is claimed is:

1. A method for performing a process on an image, the process being implemented by a computer system comprising at least one data storage device in which is stored image data for images, at least one computer and at least one computer readable medium storing thereon computer code which when executed by the at least one computer performs the method, the method comprising the at least one computer:

performing a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state, the operations comprising:

converting the image into a grayscale image;

isolating a plurality of pixels near an edge;

isolating locally dark or light pixels;

computing an intersection between the edge detected image output and the locally dark or light image output and outputting a combined black and white image; and

cleaning the outputted combined image.

2. The method of claim 1, wherein isolating the plurality of pixels near an edge further comprises:

performing edge detection on the image;

applying a smoothing operation to the image;

computing the local color contrast for each pixel in the smoothed image; and

isolating the pixels with high local color contrast.

3. The method of claim 2, wherein the edge detection is selected from a

Sobel operation, Canny, Canny–Deriche, Differential, or Prewitt Roberts Cross.

4. The method of claim 2, wherein the edge detection smoothing operation comprises applying a Bilateral Filter.

5. The method of claim 2, wherein the computation of a pixel's local color contrast from the smoothed image comprises:

performing a local standard deviation (StdDev) operation on the smoothed image.

6. The method of claim 5, wherein the local StdDev operation comprises, for each pixel of an image:

identifying a square window around a home pixel;

computing the standard deviation (StdDev) of the color values within the window of pixels;

storing the StdDev color value of the pixel, and

normalizing the computed stdDev values such that the values range from 0 to 255.

7. The method of claim 2 wherein isolating pixels with a high local color contrast comprises: performing clustering on the locally contrasted image.

8. The method of claim 5 wherein the isolating pixels with locally dark or light pixels comprises:

for each pixel of the grayscale image:

identify a square window of the pixels centered around a home pixel, wherein the window is larger than a square window used when isolating the plurality of pixels near an edge;

perform an clustering operation to the defined square window of pixels; and

save the black or white color value of the home pixel.

9. The method of claim 8 wherein the square window used for isolating locally dark or light pixels is about 25% larger than the square window used to isolate pixels that are near an edge.

10. The method of claim 1 wherein cleaning the outputted combined image comprises:

removing stray pixels from a black and white image; and

correcting any erroneous plateaus.

11. The method of claim 10 wherein removing stray pixels the black and white image comprises, for each window of at least 9 pixels:

counting the number of black pixels within the window;

counting the number of white pixels within the window;

detecting if a threshold number of pixels within the window are the opposite color of the home pixel; and if so,

changing the color value of the home pixel to the opposite color value.

12. The method of claim 11 wherein the color value threshold is at least 7 pixels.

13. The method of claim 10 wherein correcting erroneous plateaus comprises, for identified plateaus:

computing the mean and standard deviation of the original pixel colors in an island portion of the identified plateau;

computing the mean and standard deviation of the original pixel colors in a border portion of the identified plateau;

performing a statistical test on the plateau to determine if the island portion of the plateau is part of the border portion of the plateau; and

if the island portion of the plateau is a part of the border portion of the plateau, correcting the identified erroneous plateau so the island portion and the border portion of the plateau are the same color value.

14. A system for performing a process on an image, the process being implemented by a computer system comprising at least one data storage device in which is stored image data, at least one computer and at least one computer readable medium storing thereon computer code which when executed by the at least one computer performs a method, the method comprising the at least one computer:

performing a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state, the operations comprising:

converting the image into a grayscale image;

isolating a plurality of pixels near an edge;

isolating locally dark or light pixels;

computing an intersection between the edge detected image output and the locally dark or light image output and outputting a combined black and white image; and

cleaning the outputted combined image.

15. The system of claim 14, wherein isolating the plurality of pixels near an edge further comprises the at least one computer performing the operations comprising:

performing edge detection on the image;

applying a smoothing operation to the image;

computing the local color contrast for each pixel in the smoothed image; and

isolating the pixels with high local color contrast.

16. The system of claim 15, wherein the edge detection is selected from a Sobel operation, Canny, Canny–Deriche, Differential, or Prewitt Roberts Cross.

17. The system of claim 15, wherein the edge detection smoothing operation comprises applying a Bilateral Filter.

18. The system of claim 15, wherein the computation of the pixel's local color contrast from the smoothed image comprises the at least one computer performing the operation comprising:

performing a local standard deviation (StdDev) operation on the smoothed image.

19. The system of claim 18, wherein the local StdDev operation comprises, for each pixel of an image:

identifying a square window around a home pixel;

computing the standard deviation (StdDev) of the color values within the window of pixels;

storing the StdDev color value of the pixel, and

normalizing the computed stdDev values such that the values range from 0 to 255.

20. The system of claim 15 wherein isolating pixels with a high local color contrast comprises the at least one computer performing the operation comprising: performing clustering on the locally contrasted image.

21. The system of claim 18 wherein the isolating pixels with locally dark or light pixels comprises the at least one computer performing the operation comprising:

for each pixel of the grayscale image:

identify a square window of the pixels centered around a home pixel, wherein the window is larger than a square window used when isolating the plurality of pixels near an edge;

perform an Otsu's method operation to the defined square window of pixels; and

save the black or white color value of the home pixel.

22. The system of claim 21 wherein the square window used for isolating locally dark or light pixels is about 25% larger than the square window used to isolate pixels that are near an edge.

23. The system of claim 14 wherein cleaning the outputted combined image comprises the at least one computer performing the operation comprising:

removing stray pixels from a black and white image; and

correcting any erroneous plateaus.

24. The system of claim 23 wherein removing stray pixels the black and white image comprises the at least one computer performing the operation comprising, for each window of at least 9 pixels:

counting the number of black pixels within the window;

counting the number of white pixels within the window;

detecting if a threshold number of pixels within the window are the opposite color of the home pixel; and if so,

changing the color value of the home pixel to the opposite color value.

25. The system of claim 24 wherein the color value threshold is at least 7 pixels.

26. The system of claim 23 wherein correcting erroneous plateaus comprises, for identified plateaus, the at least one computer performing the operations comprising:

computing the mean and standard deviation of the original pixel colors in an island portion of the identified plateau;

computing the mean and standard deviation of the original pixel colors in a border portion of the identified plateau;

performing a statistical test on the plateau to determine if the island portion of the plateau is part of the border portion of the plateau; and

if the island portion of the plateau is a part of the border portion of the plateau, correcting the identified erroneous plateau so the island portion and the border portion of the plateau are the same color value.

27. A method for improving the image processing performance of a computing system, comprising:

dividing an image processing algorithm into a plurality of processes;

allocating the plurality of processes among computing modules of the computing system;

dividing an input image into a plurality of slices that are to be processed in parallel; and

recombining the processed slices to form an output,

wherein the image processing algorithm is configured to perform a set of operations on an image that renders the image from a first state to a second state, wherein the second state is more legible than then the first state.

28. The method of claim 27, further comprising:

determining the processes of the image processing algorithm that are computationally-intensive.

29. The method of claim 28, wherein the determination of the one or more processes of the image processing algorithm that are computationally-intensive is based on a computing time of each process.

30. The method of claim 27, wherein each slice includes additional portions that include pixels of an adjacent slice.

31. The method of claim 27, wherein the processing system includes a graphic processing unit, and the one or more computationally intensive processes are allocated to the graphic processing unit.

32. The method of claim 31, wherein a pipelining process is configured to dynamically allocates processes among the computing modules of the processing

system.

33. The method of claim 31, wherein the processing system includes at least one central processing unit, and the one or more processes of the image processing unit not determined to be computationally intensive are allocated to the central processing unit.

34. The method of claim 27, wherein the image processing algorithm includes at least one of the processes comprising:

converting the input image into a grayscale image;

performing edge detection on the image;

isolating a plurality of pixels near an edge;

isolating locally dark or light pixels;

computing an intersection between the edge detected image output and the locally dark or light image output and outputting a combined black and white image;

applying a smoothing operation to the image, a local standard deviation (StdDev) operation; and an Otsu's method operation;

computing the local color contrast for each pixel in the smoothed image; isolating the pixels with high local color contrast; and

cleaning an outputted combined image.

35. The method of claim 34, wherein the converting process is assigned to the at least one central processing unit, and is not subject to a parallel processing operation.

36. The method of claim 34, wherein the image processing algorithm

includes a process for sharpening the grayscale image.

37. The method of claim 36, wherein the processing system includes a plurality of the central processing units, and the sharpening process is subject to a parallel processing operation by the plurality of the central processing units.

38. The method of claim 34, wherein the image processing algorithm includes a process for computing a local standard deviation, and the computing the local standard deviation is allocated to the graphical processing unit.

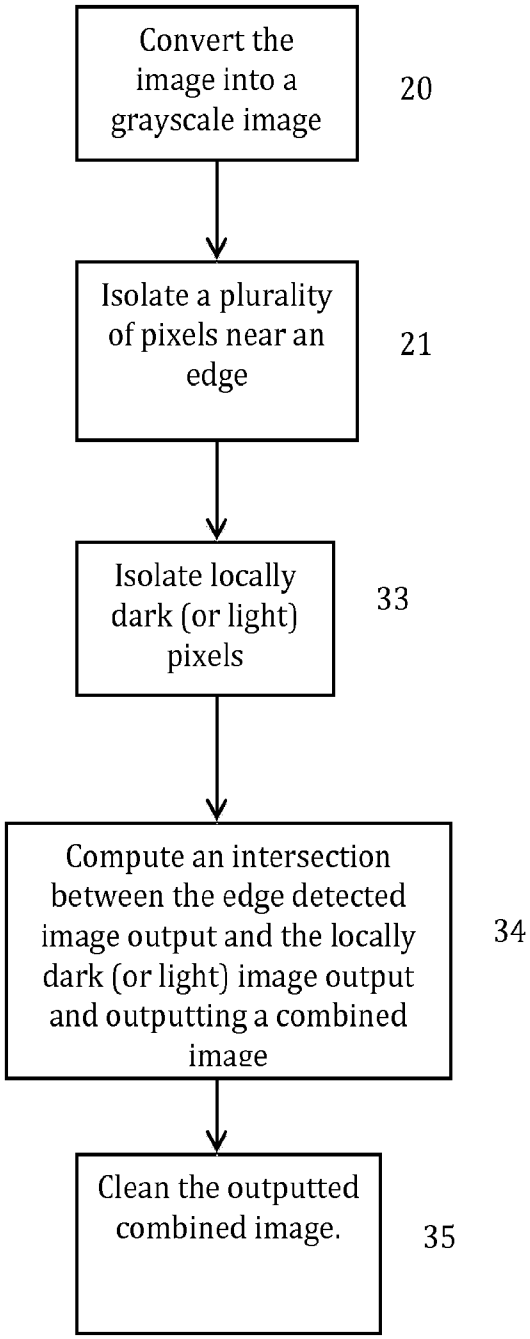

Convert the image into a grayscale image
20
Isolate a plurality of pixels near an edge
21
Isolate locally dark (or light) pixels
33
Compute an intersection between the edge detected image output and the locally dark (or light) image output and outputting a combined image
34
Clean the outputted combined image.
35

FIG. 1A

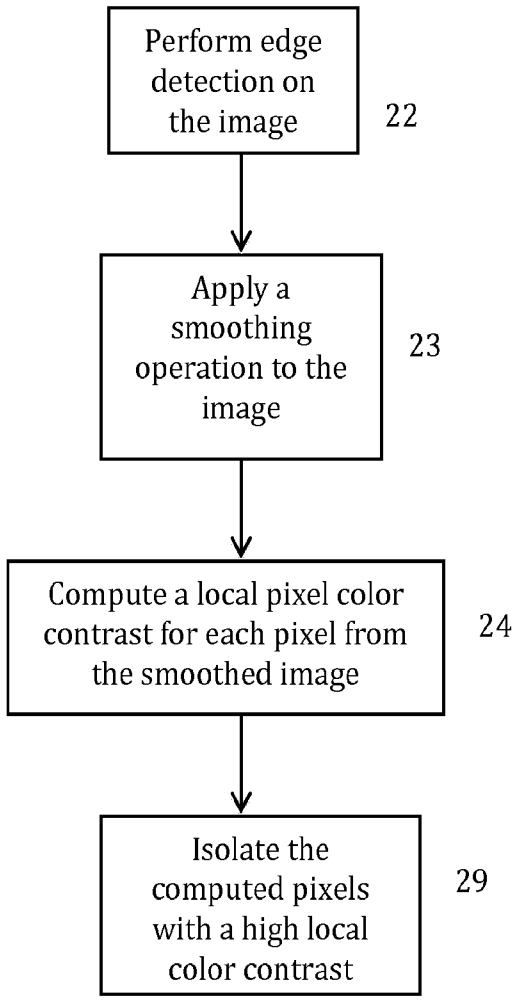

Perform edge detection on the image
22
Apply a smoothing operation to the image
23
Compute a local pixel color contrast for each pixel from the smoothed image
24
Isolate the computed pixels with a high local color contrast
29

FIG. 1B

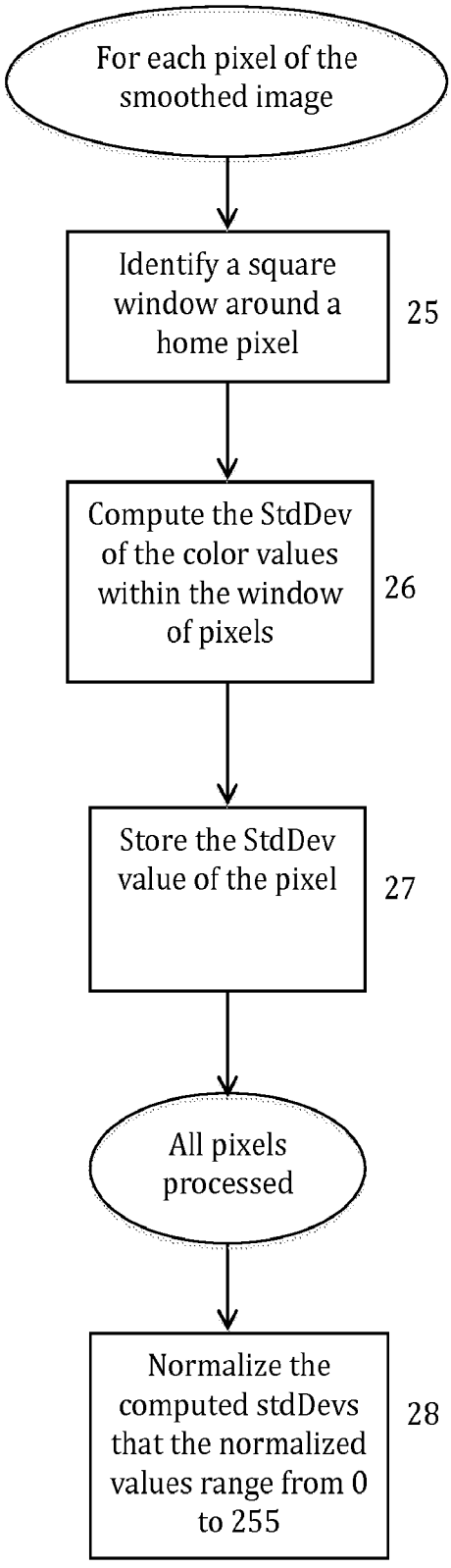


FIG. 1C

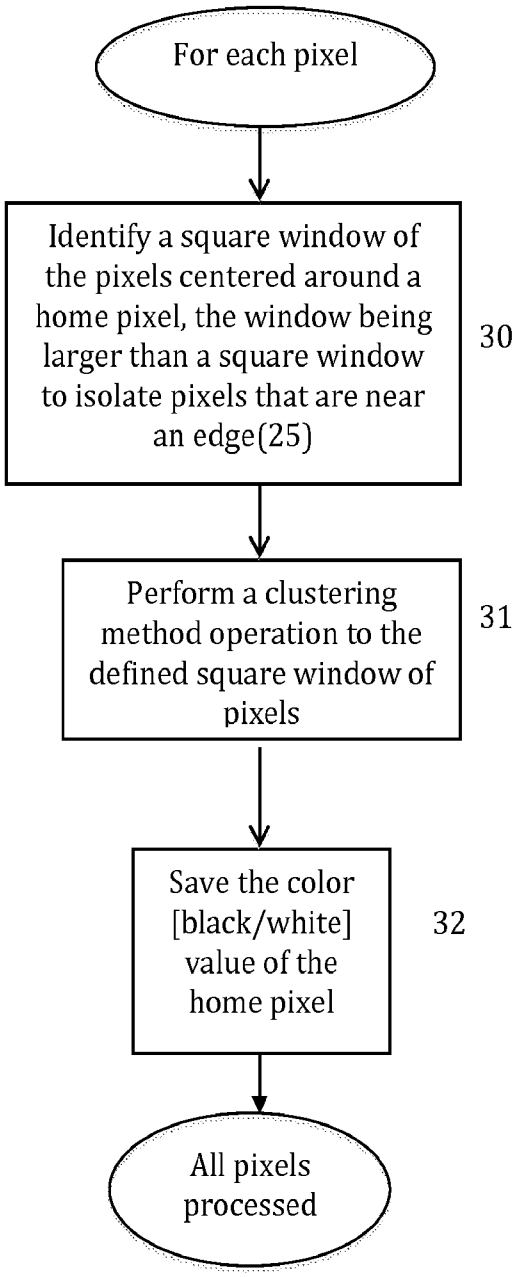


FIG. 1D

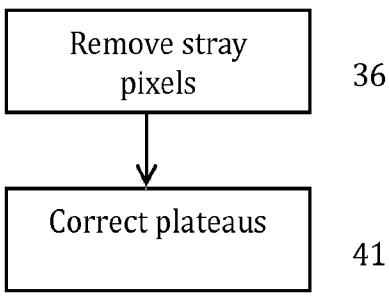


FIG. 1E

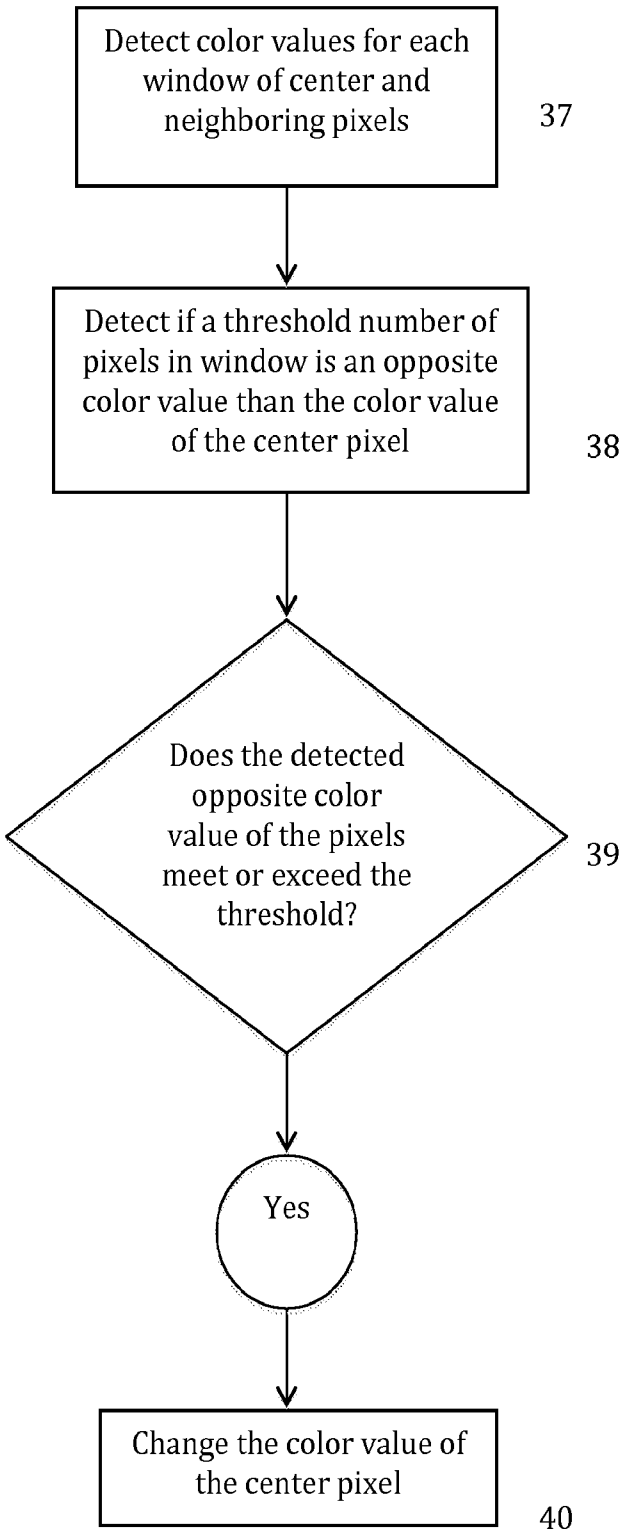

Detect color values for each window of center and neighboring pixels
37
Detect if a threshold number of pixels in window is an opposite color value than the color value of the center pixel
38
Does the detected opposite color value of the pixels meet or exceed the threshold?
39
Yes
Change the color value of the center pixel
40

FIG. 1F

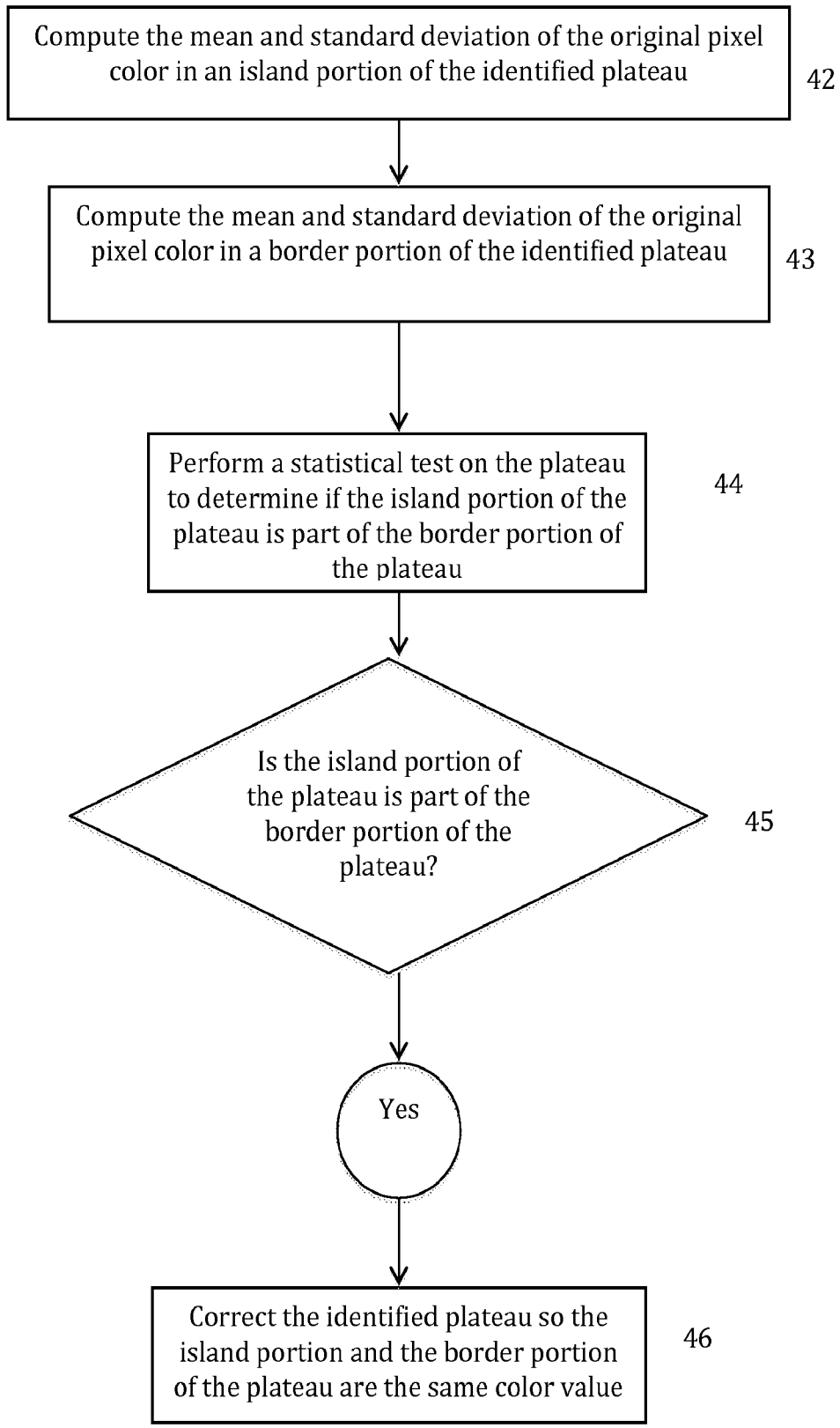
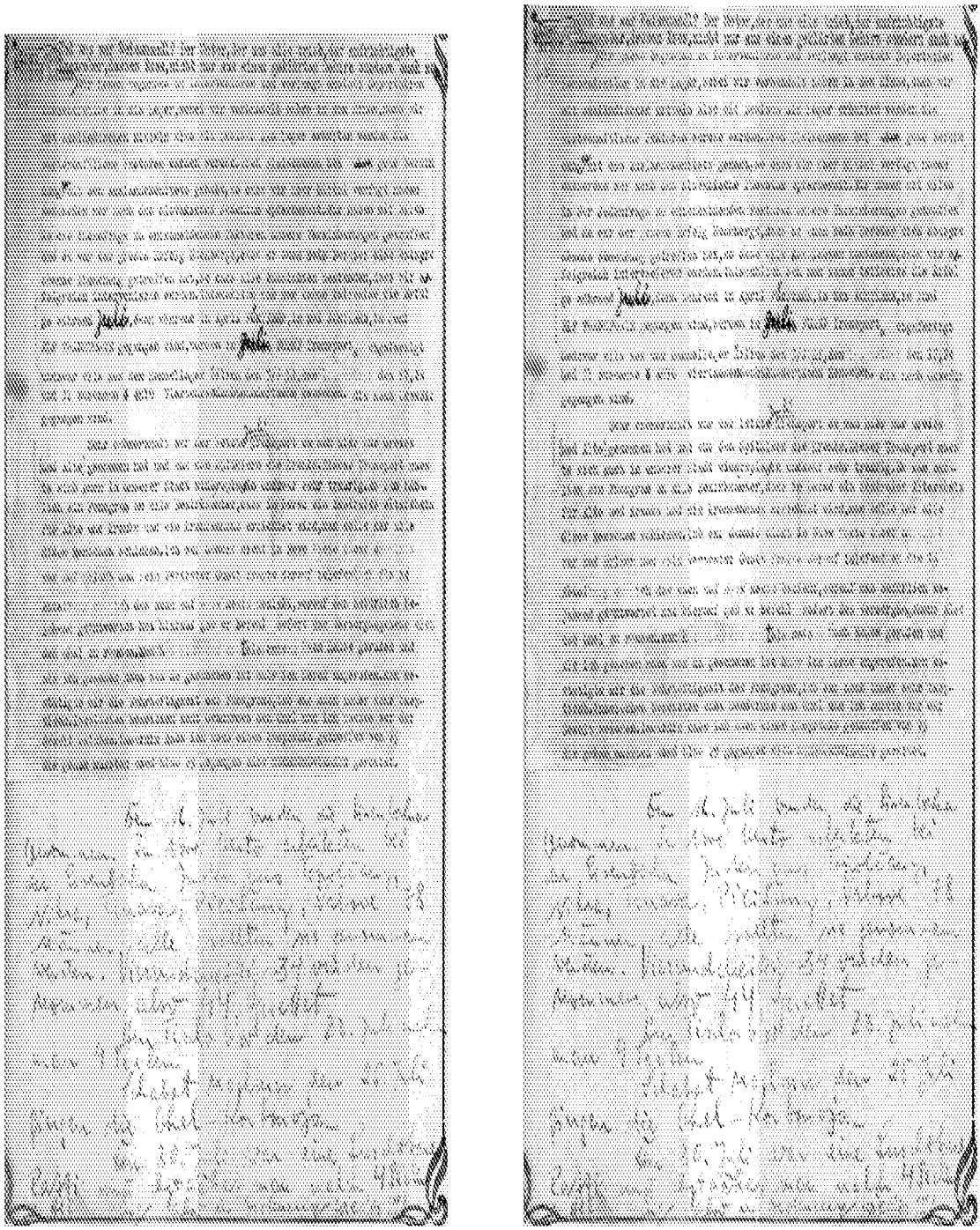
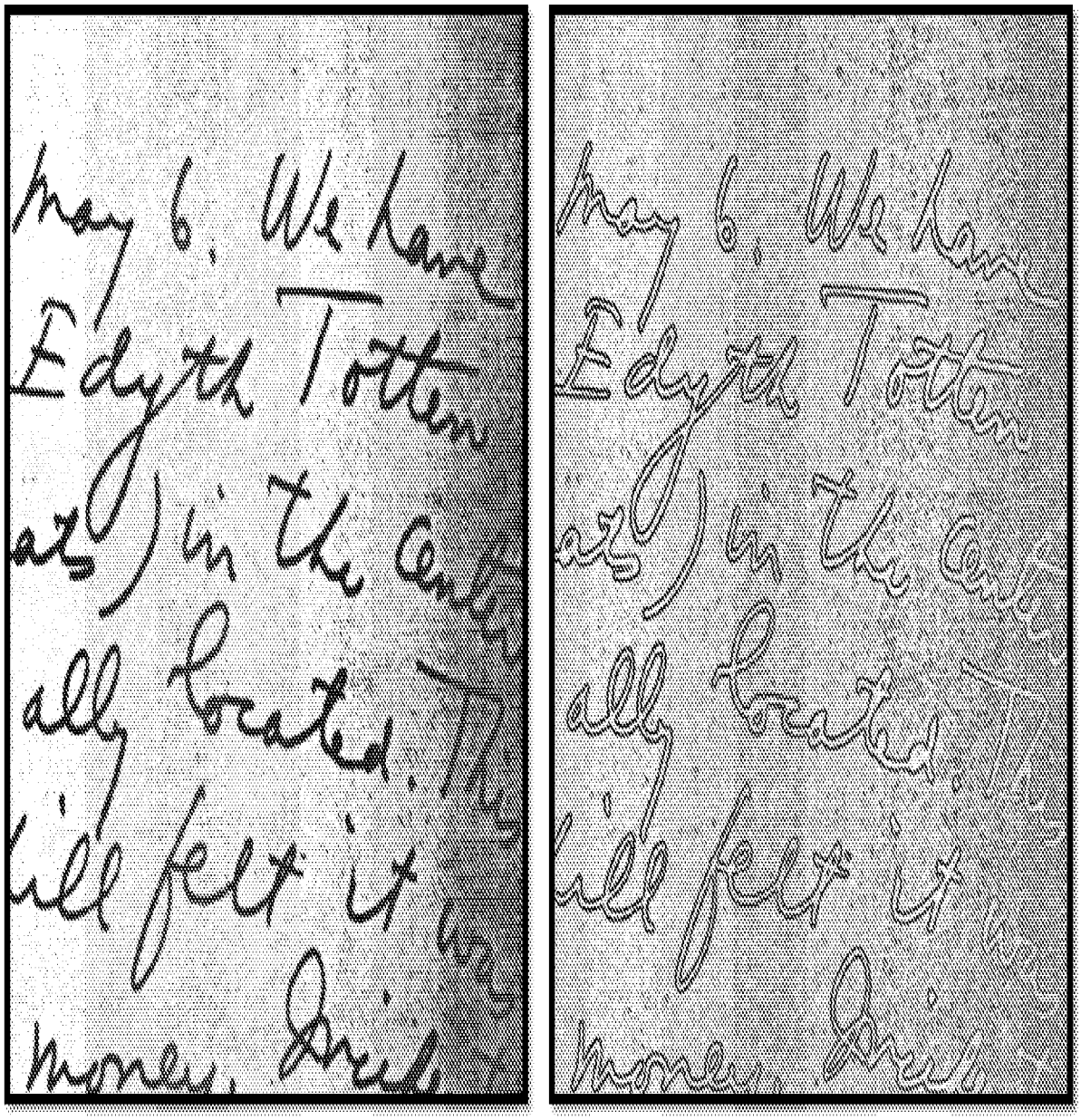

Compute the mean and standard deviation of the original pixel color in an island portion of the identified plateau
42
Compute the mean and standard deviation of the original pixel color in a border portion of the identified plateau
43
Perform a statistical test on the plateau to determine if the island portion of the plateau is part of the border portion of the plateau
44
Is the island portion of the plateau is part of the border portion of the plateau?
45
Yes
Correct the identified plateau so the island portion and the border portion of the plateau are the same color value
46

FIG. 1G



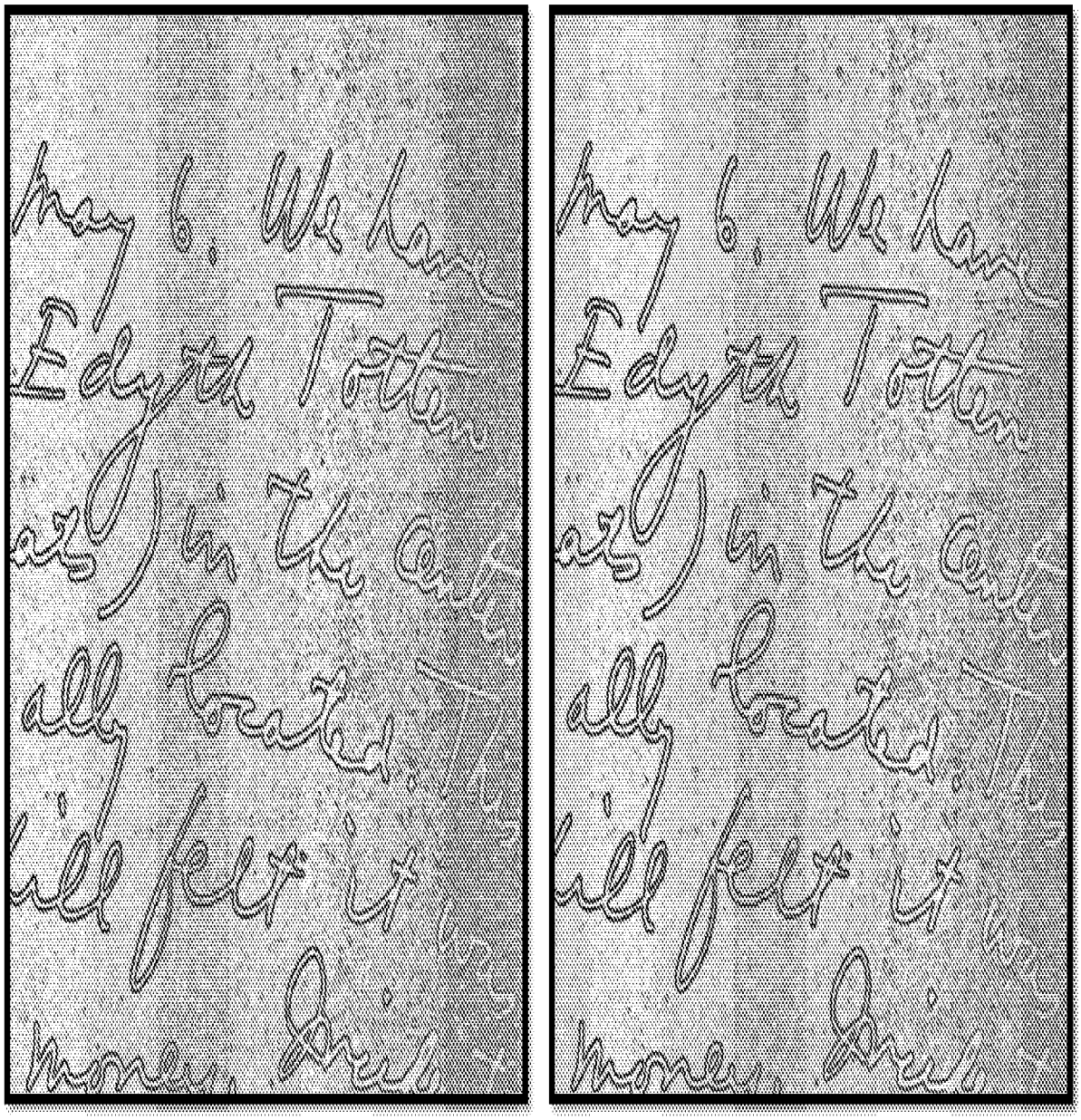
200 202

FIG. 2



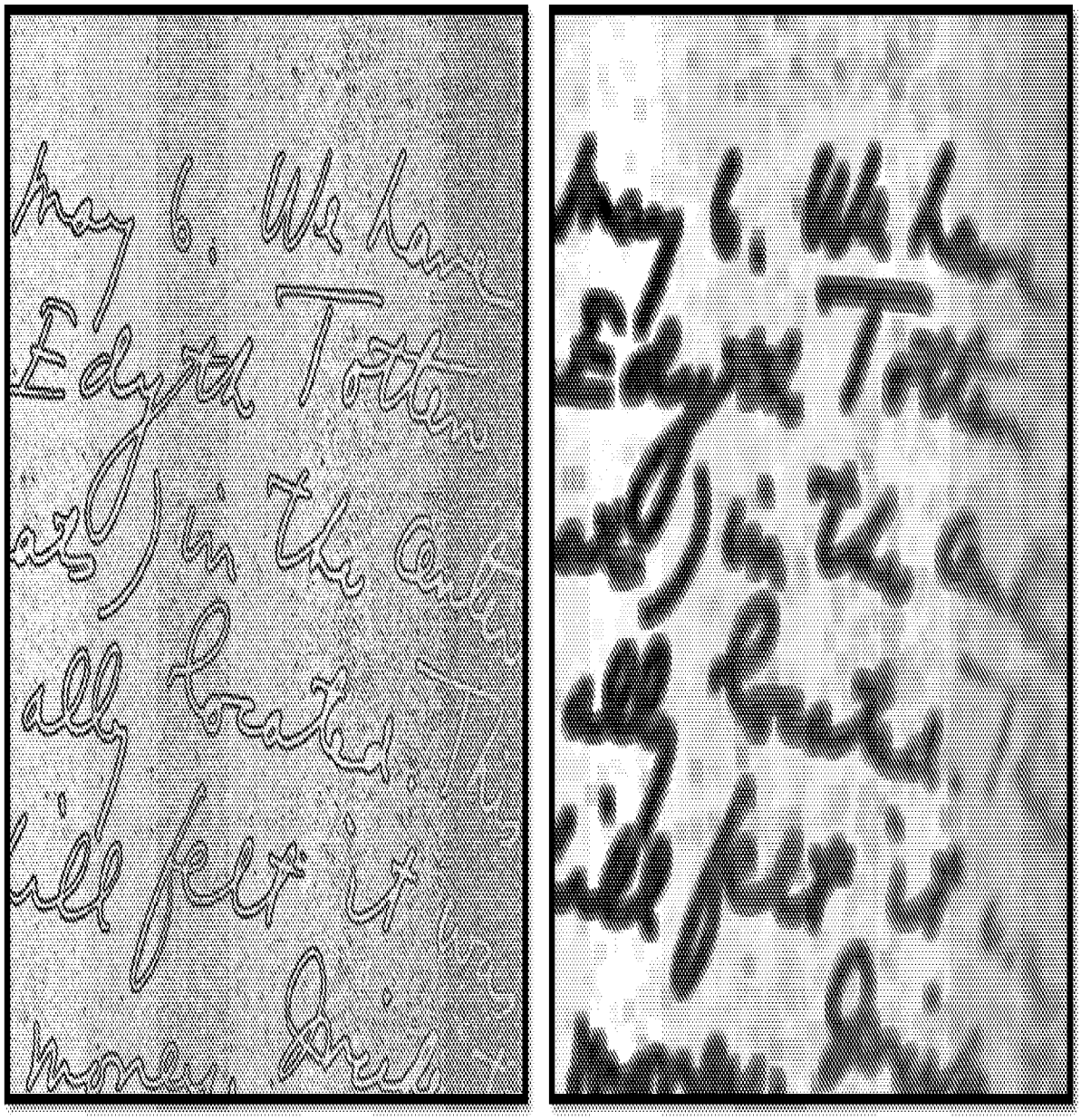
202a 206

FIG. 3A



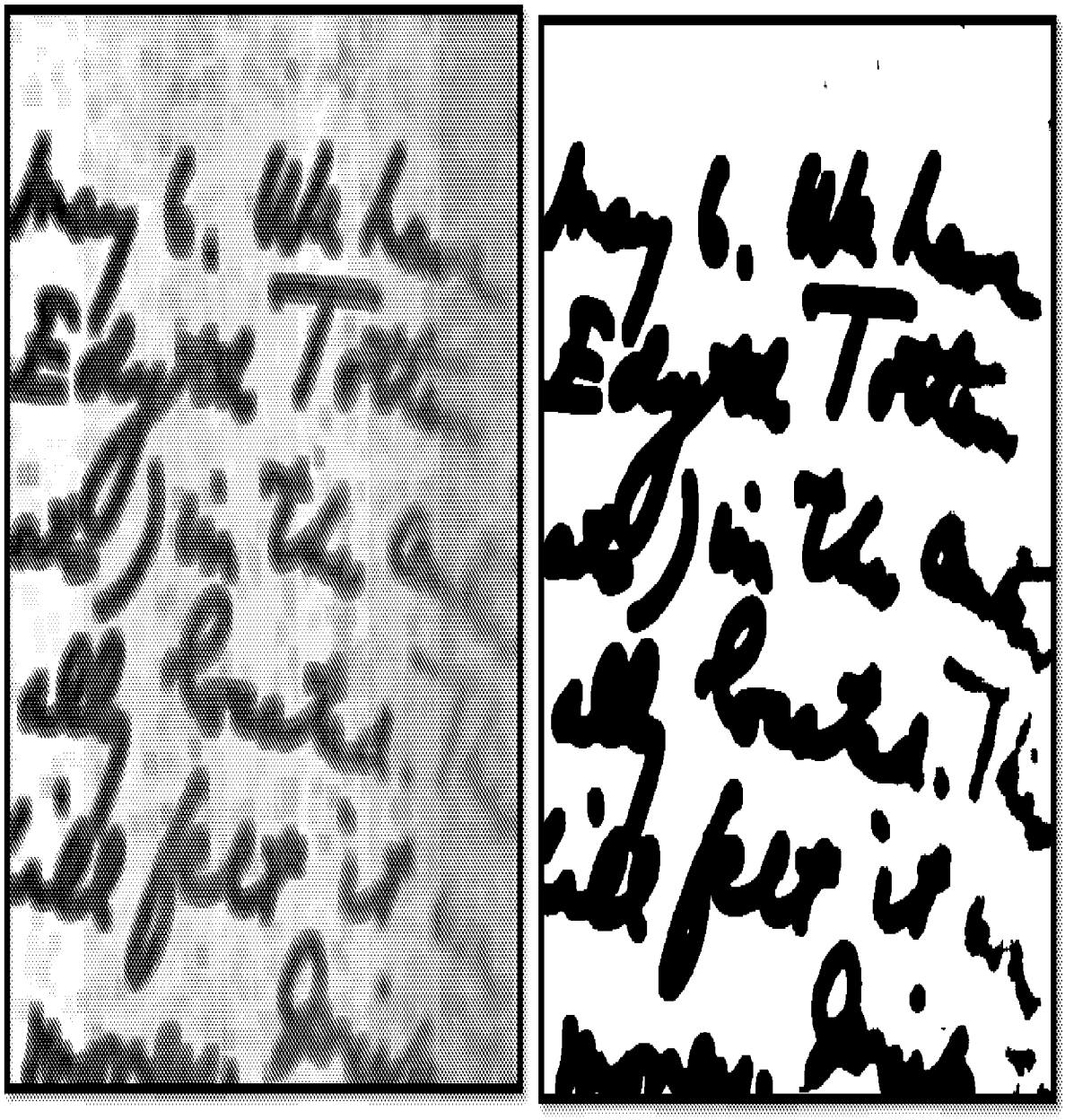
206 208

FIG. 3B



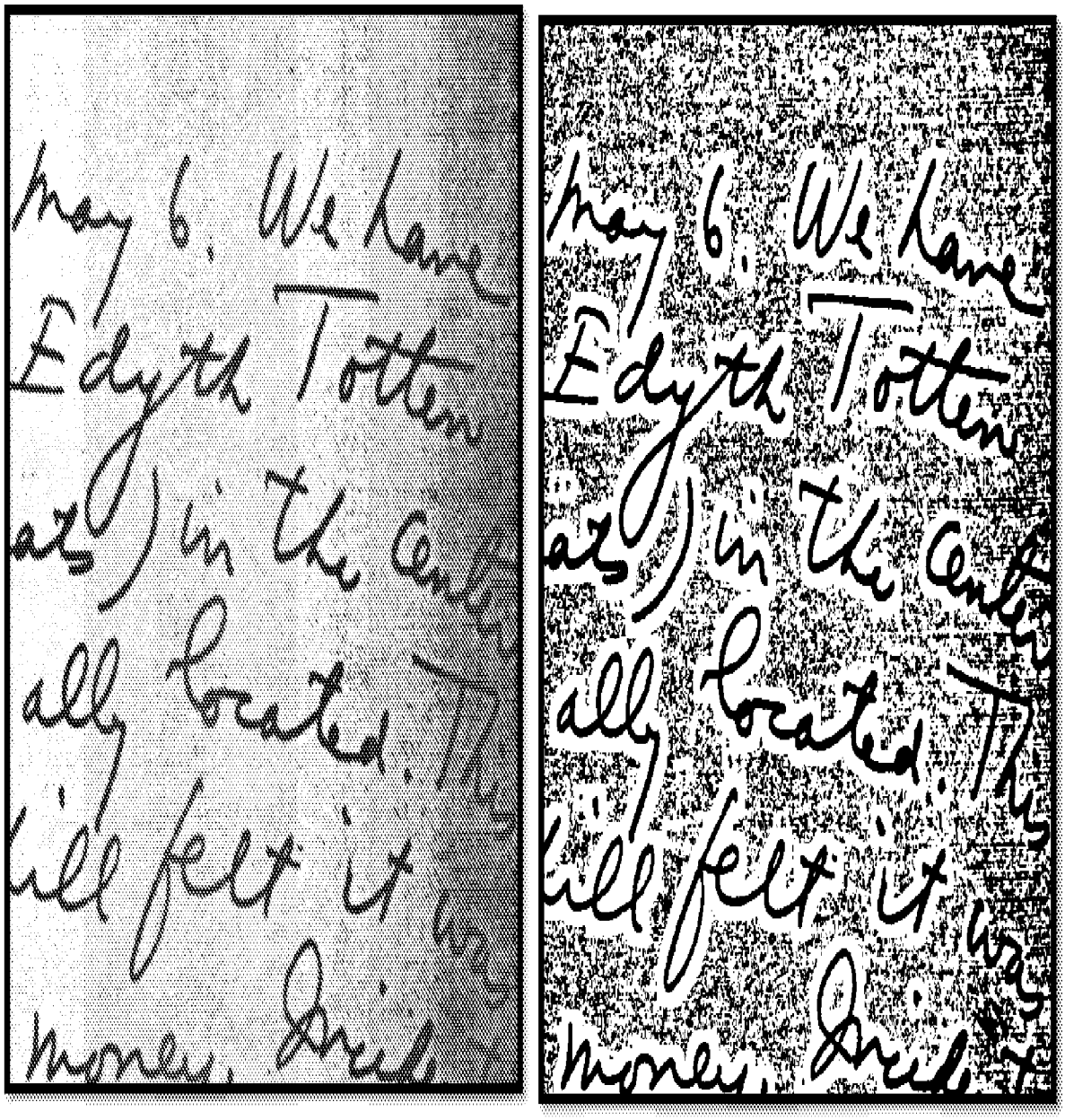
208 210

FIG. 3C



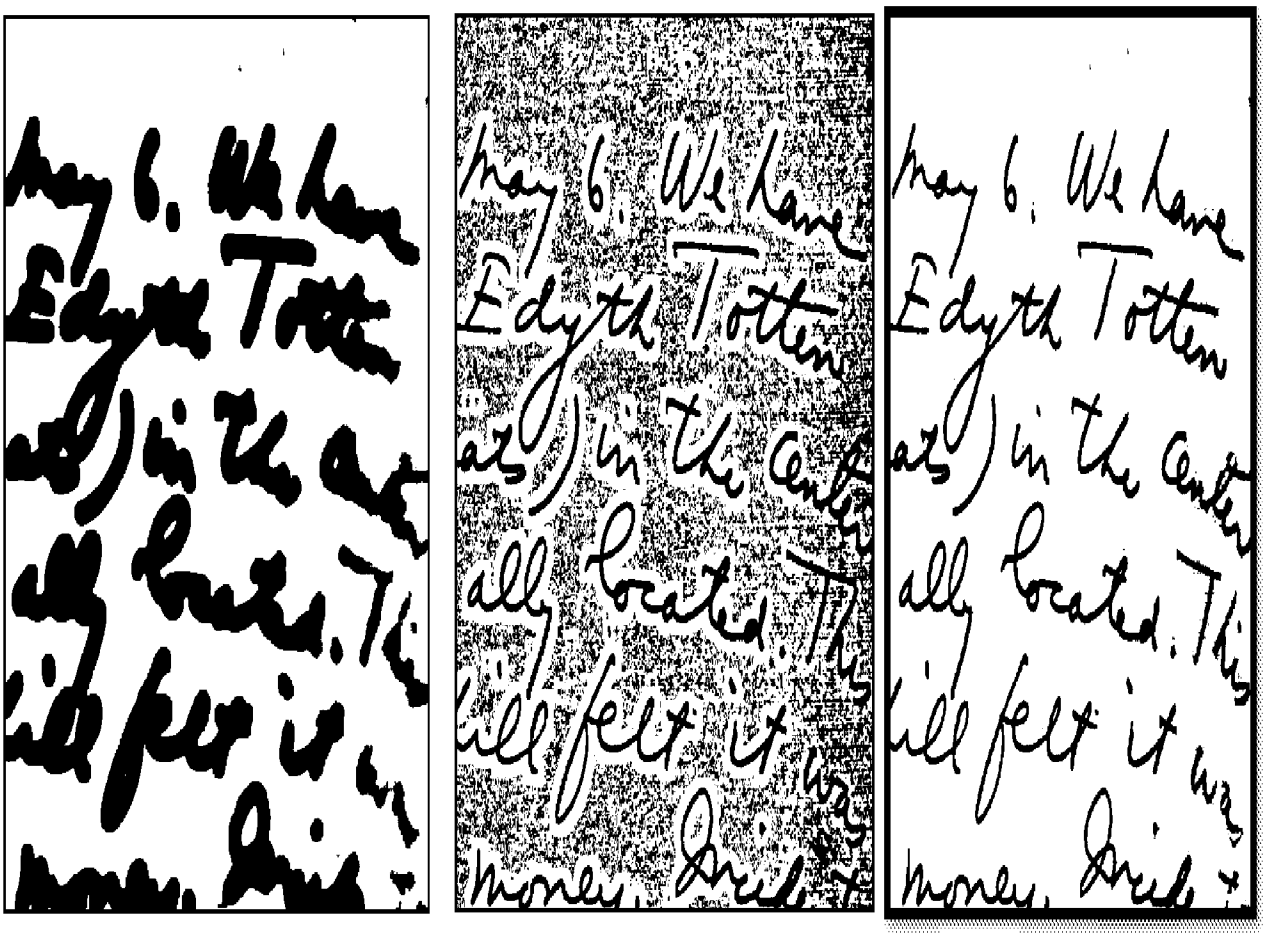
210 212

FIG. 3D



202 214

FIG. 4



212 214 216

FIG. 5

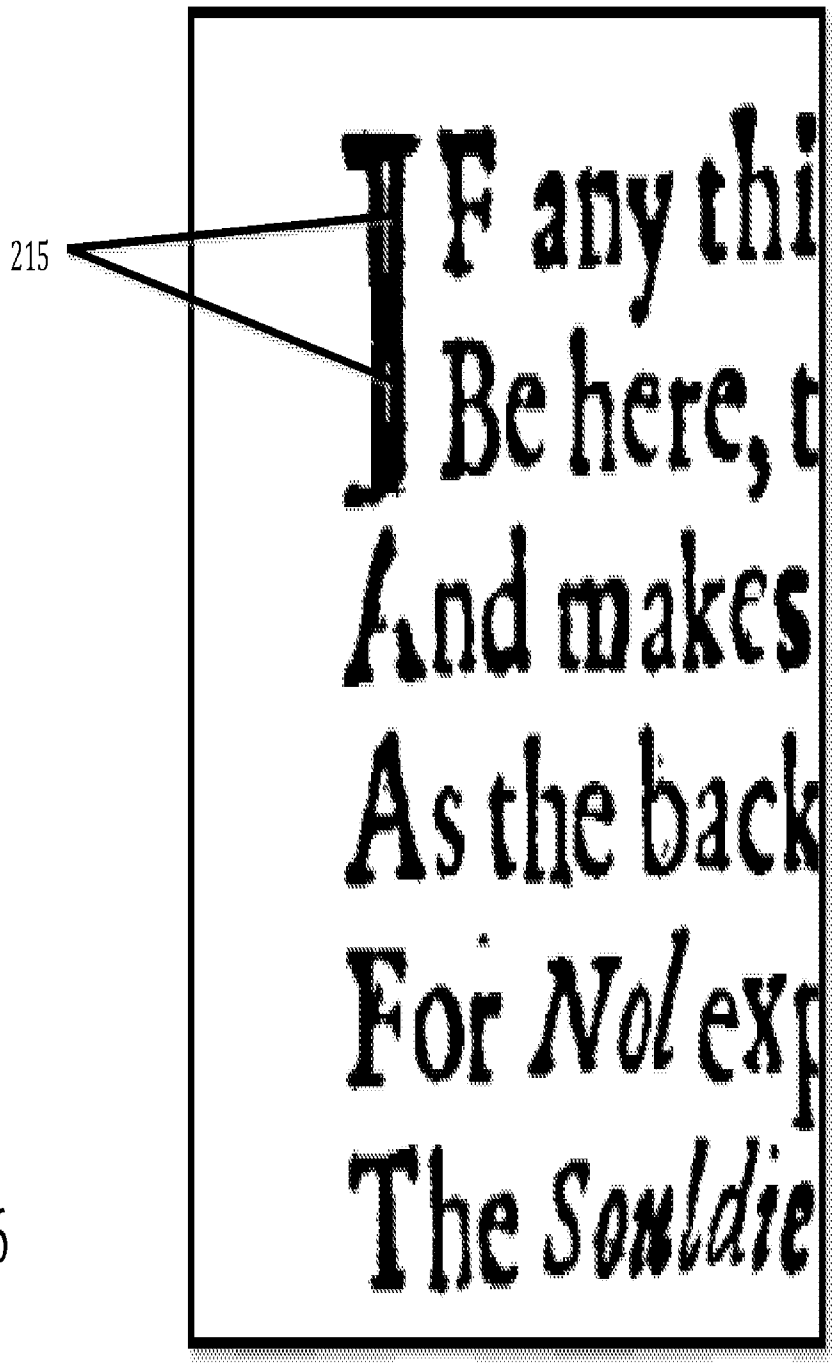
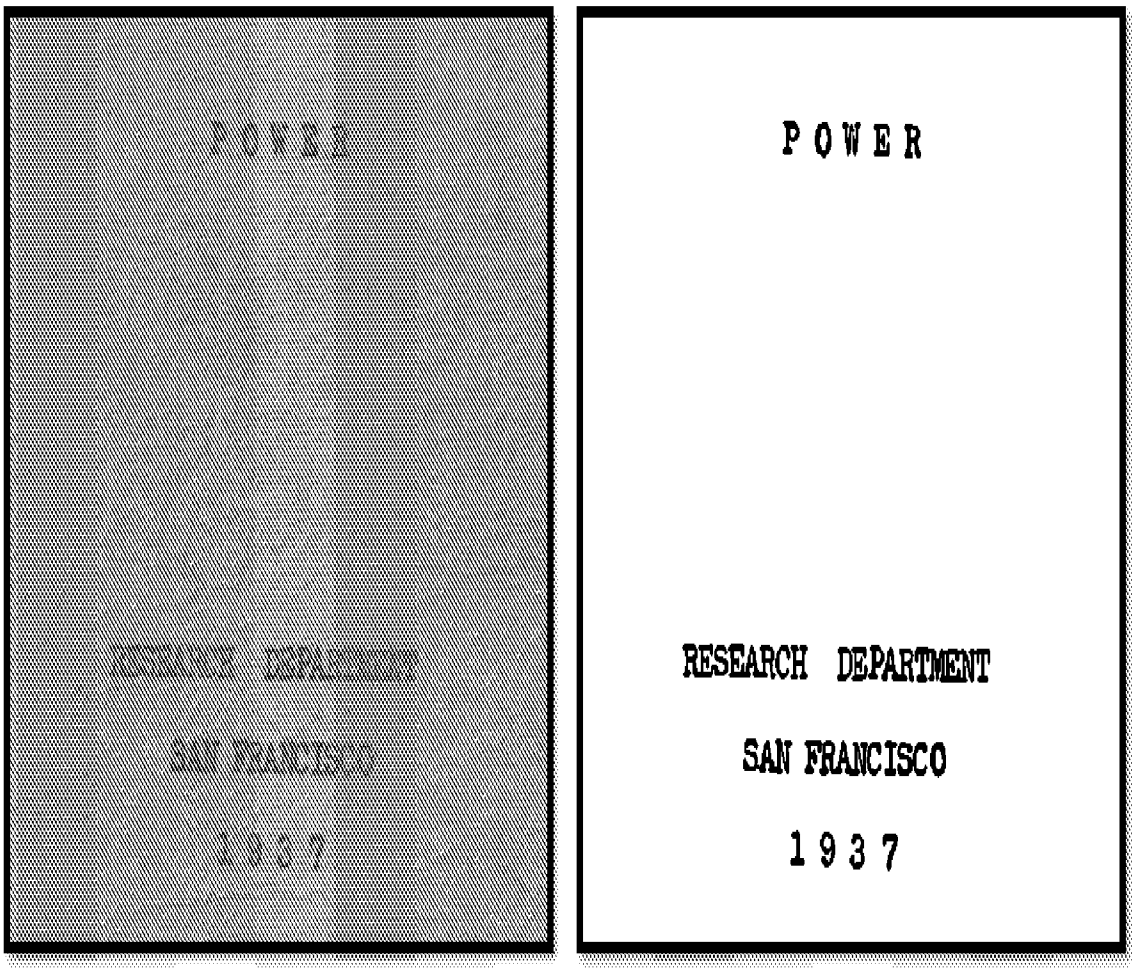
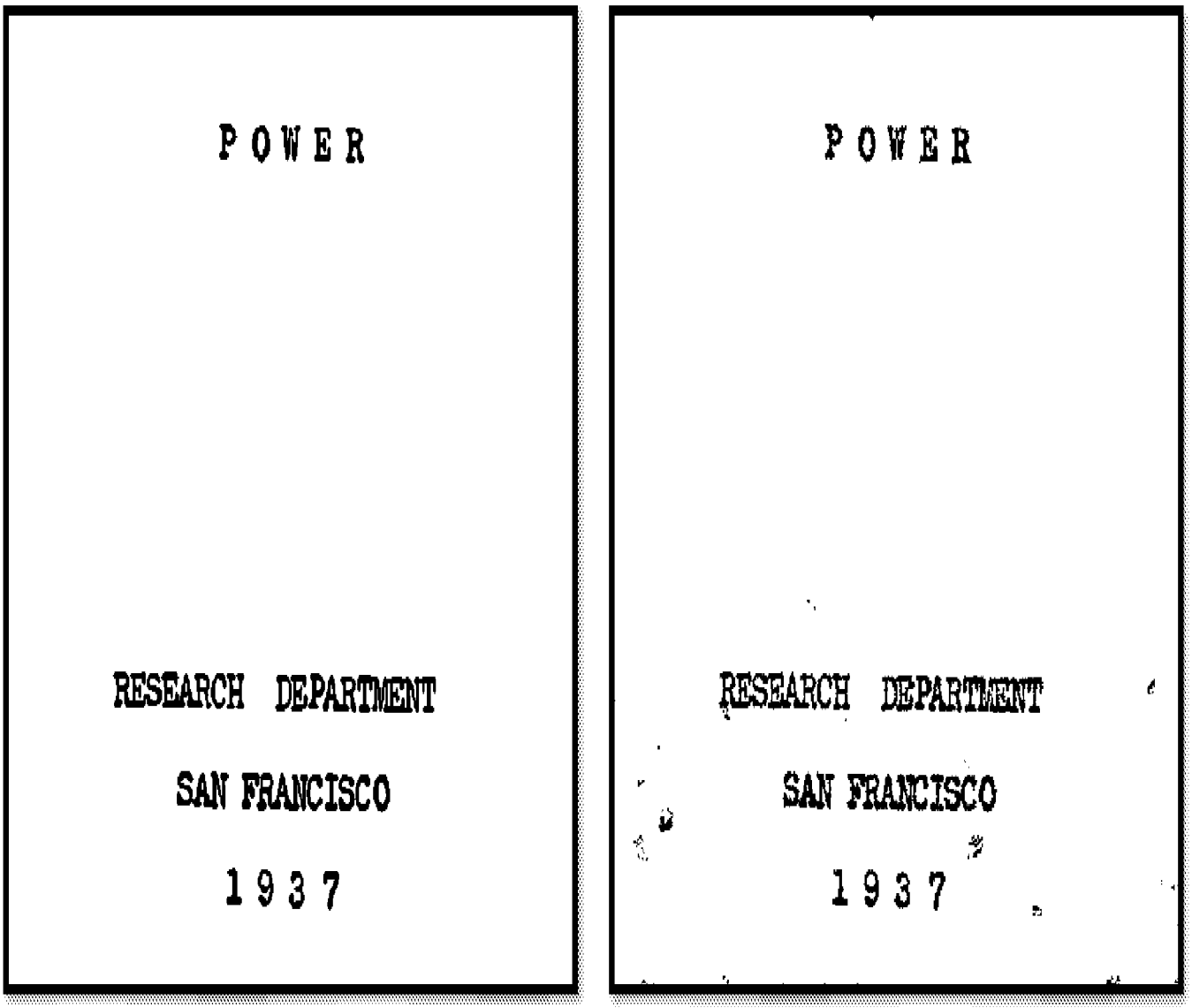

215
F any thi
Be here, t
And makes
As the back
For Nol exp
The Souldie
216

FIG. 6



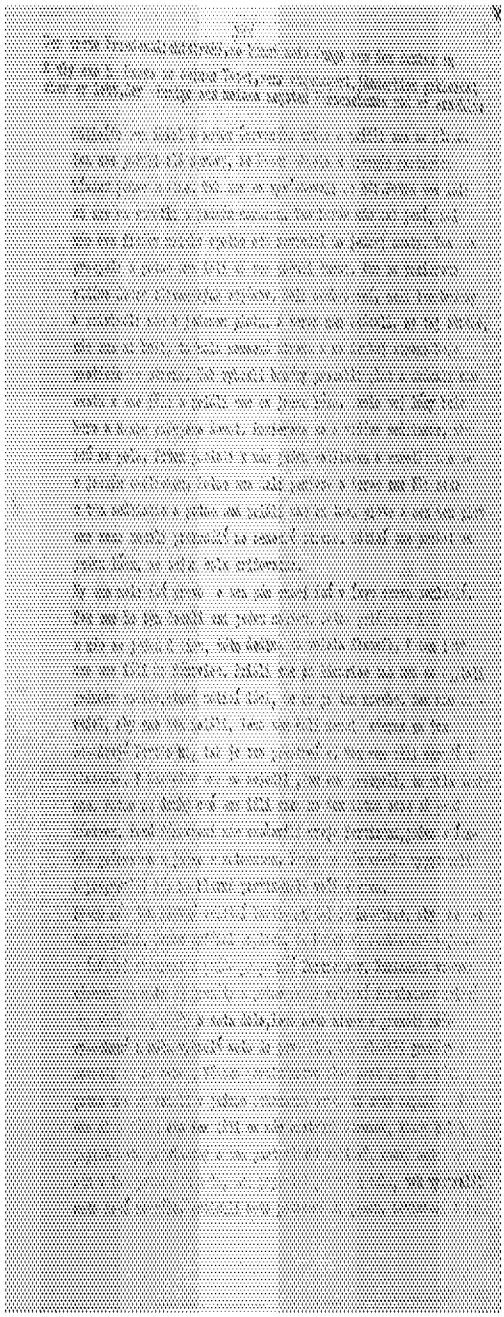
200x 217

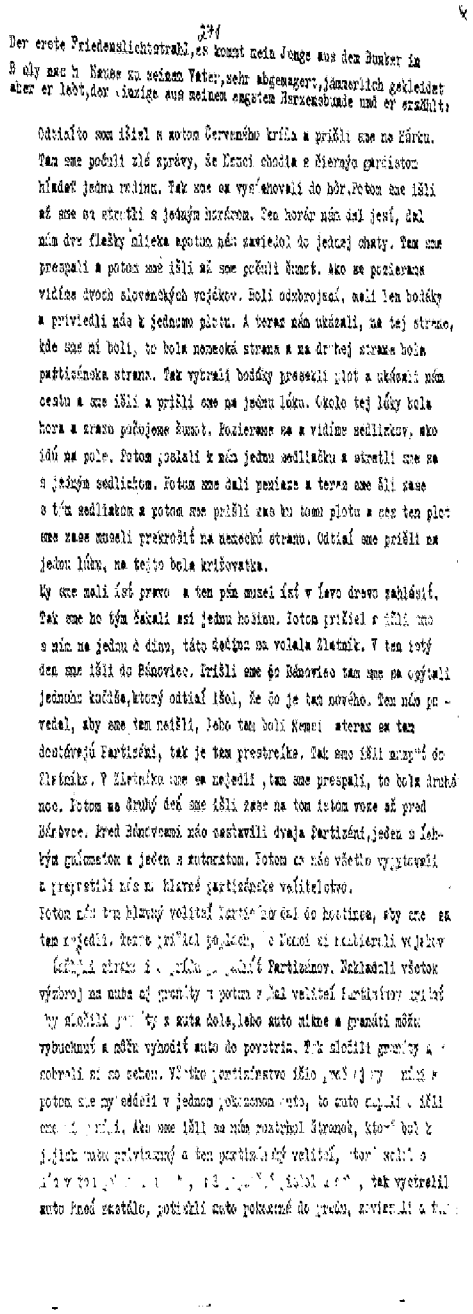
FIG. 7A



217 218

FIG. 7B



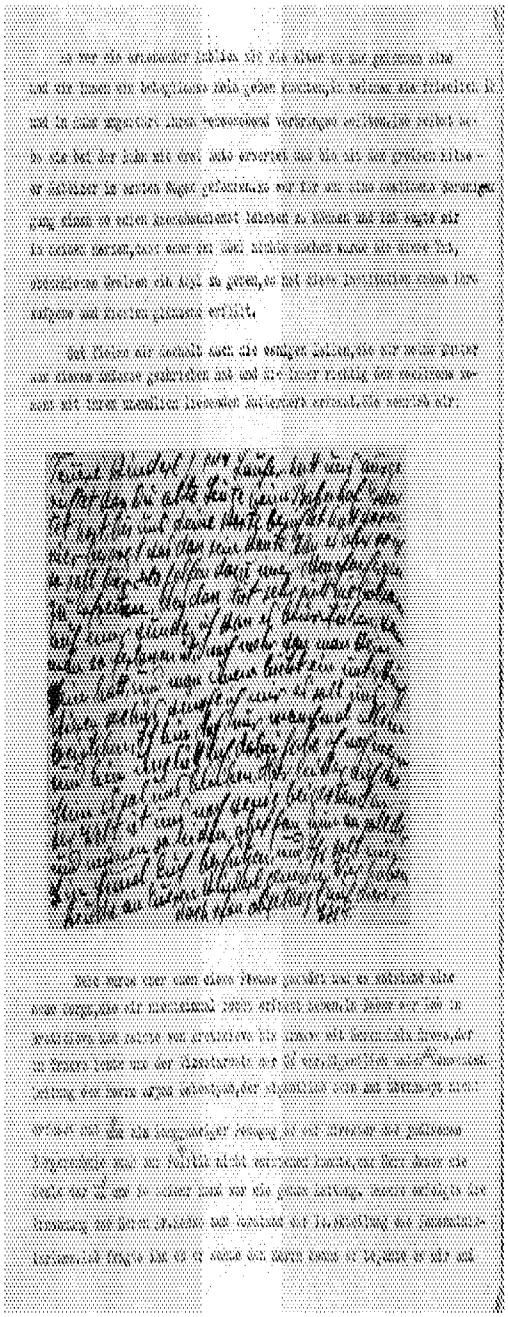


Der erste Friedensnachtstrahl,es kommt mein Junge aus dem Bunker in
S oly nac h Hause zu seinem Vater,sehr abgemagert,jämmerlich gekleidet
aber er lebt,der einzige aus meinen engsten Verwandtenkreis und er erzählt:

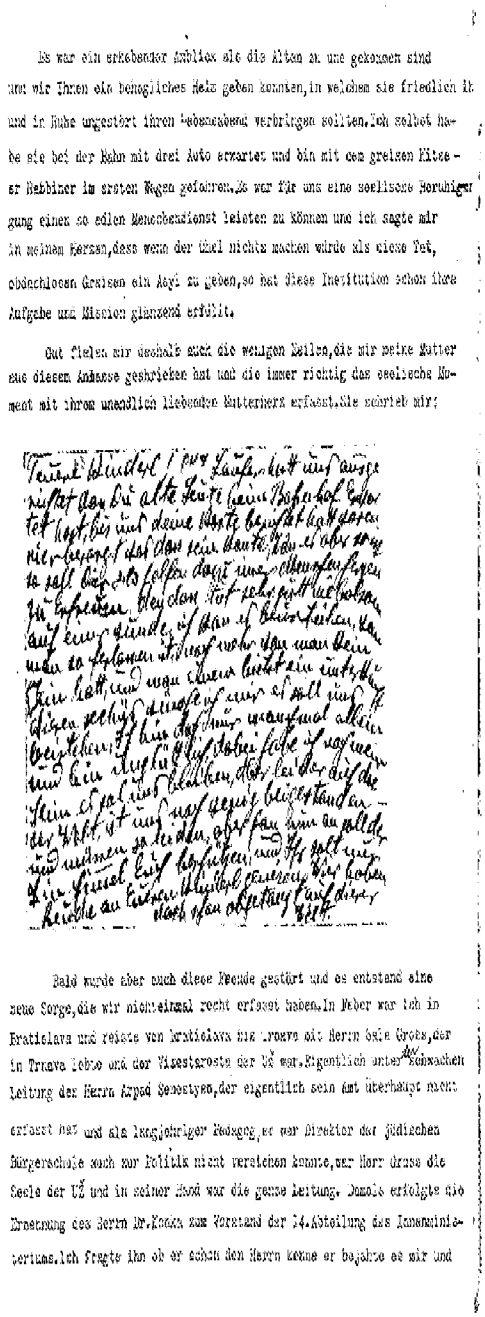
Odtiaľto sme išiel s svoton Červeného kríža a prišli sme na Kúrcu.
Tam sme počuli zlé správy, že Nemci chodia s Kiernym gardistom
hľadať jednu rodinu. Tak sme sa vyšmykovali do hôr.Potom sme išli
až sme sa stretli s jedným horárom. Ten horár nám dal jesť, dal
nám dve fľašky mlieka a potom nás zaviedol do jednej chaty. Tam sme
prespali a potom sme išli až sme počuli šumot. Ako sa pozeráme
vidíme dvoch slovenských vojakov. Boli odzbrojení, mali len bodáky
a priviedli nás k jednemu plotu. A teraz nám ukázali, na tej strane,
kde sme mi boli, to bola nemecká strana a na druhej strane bola
partizánska strana. Tak vybrali bodáky preseki plot a ukázali nám
cestu a sme išli a prišli sme na jednu lúku. Okolo tej lúky bola
hora a zrazu počujeme šumot. Pozierame sa a vidíme sedliakov, ako
idú na pole. Potom poslali k nám jedného sedliačku a stretli sme sa
s jedným sedliakom. Potom sme dali peniaze a tetka sme išli sme
s tým sedliakom a potom sme prišli sme ku tomu plotu a cez ten plot
sme zase museli prekročiť na nemecku stranu. Odtiaľ sme prišli na
jednu lúku, na tejto bola križovatka.
My sme mali ísť pravo a ten pán musel ísť v ľavo drevo zahliadiť.
Tak sme ho tým čakali asi jednu hodinu. Potom prišiel s dňa mo
s nim sme jednu ženu, táto dedina sa volala Zlatník. V ten istý
deň sme išli do Bánoviec. Prišli sme do Bánoviec tam sme sa opýtali
jednoho kučiša,ktorý odtiaľ išol, že čo je tam nového. Ten nám po-
vedal, aby sme tam nešli, lebo tam boli Nemci ateras sa tam
dostávajú Partizáni, tak je tam prestrelka. Tak sme išli naspäť do
Zlatníka. V Zlatníku sme sa nejedli ,tam sme prespali, to bola druhá
noc. Potom na druhý deň sme išli zase na tom istom voze až pred
Bôrbvce. Pred Bánovcami nás zastavili dvaja Partizáni,jeden s ľah-
kým guľomatom a jeden s automatom. Potom sa nás všetko vypytovali
a prepustili nás na hlavné partizánske veliteľstvo.
Potom nás tam hlavný veliteľ Partizánov dal do kostinca, aby sme sa
tam najedli. Keďže prišiel jeden z Nemci sa rozbehli vojaky
[illegible] Partizánov. Nakladali všetok
výzbroj na nuta aj granáty a potom v deň veliteľ Partizánov mal
by sa odísť [illegible] a auta dole,keby auto stlkne a granáti môžu
vybuchnúť a môžu vyhodiť auto do povetria. Tak sa odišli granáty a
odpovedali si so sebou. Všetko partizánstvo išlo [illegible]
potom sme vyšetrili v jednom [illegible] auto, to auto [illegible]
[illegible]. Ako sme išli na nám rozbehol žltanok, ktorý bol z
jedlom nám privítaný a ten partizánsky veliteľ, [illegible]
[illegible], tak vyčerpali
auto hneď zastalo, potiahli auto pokazené do [illegible]

200a 218a

FIG. 8A



200b



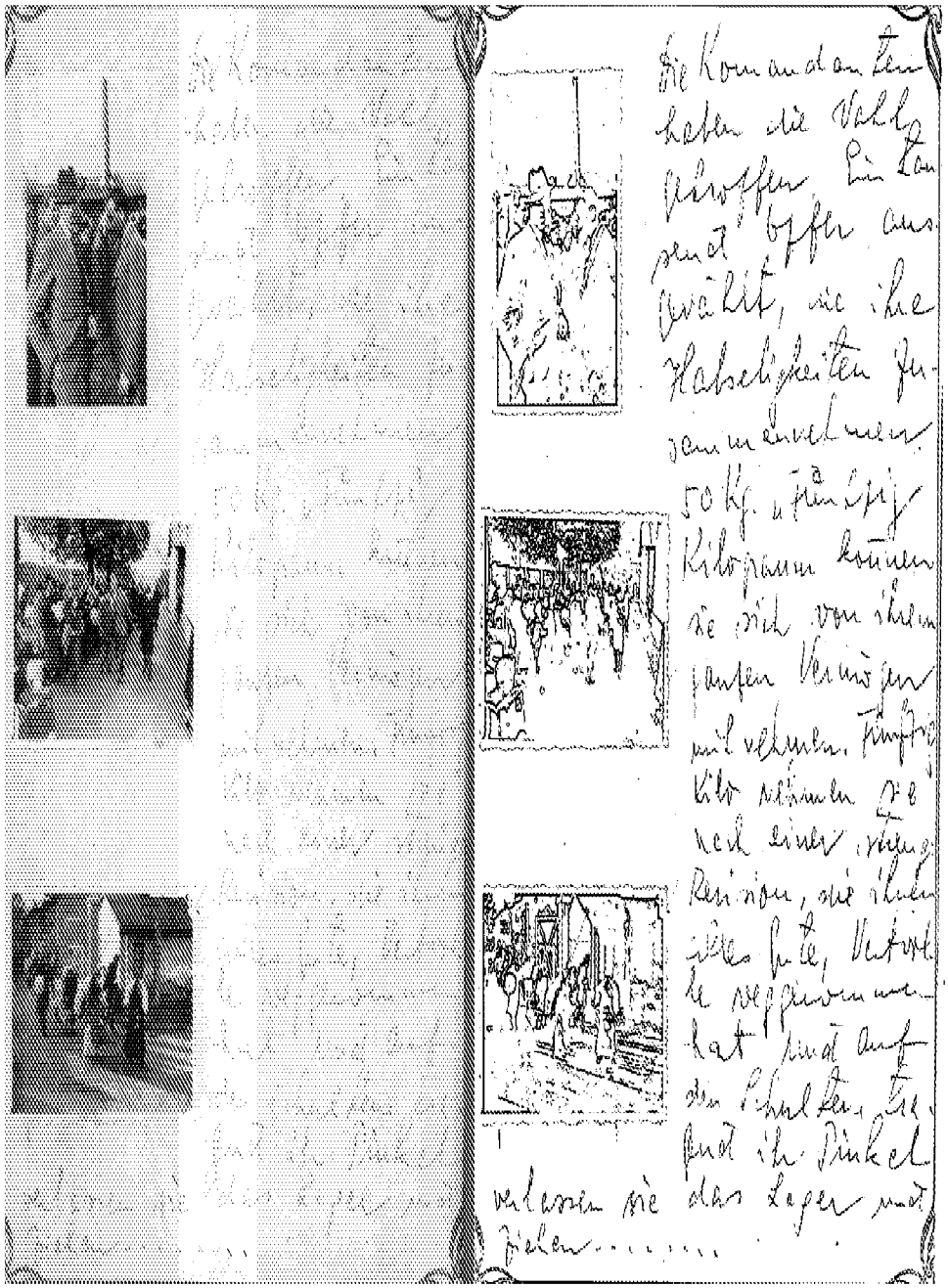
Es war ein erhebender Anblick als die Alten an uns gekommen sind und wir Ihnen ein bescheidenes Mahl geben konnten,in welchem sie friedlich i und in Ruhe ungestört ihren Lebensabend verbringen sollten.Ich selbst habe sie bei der Kahn mit drei Auto erwartet und bin mit den greisen Ehtes - er Rabbiner in ersten Wagen gefahren.Es war für uns eine seelische Beruhigung einen so edlen Menschendienst leisten zu können und ich sagte mir in meinem Herzen,dass wenn der Chol nichts machen würde als diese Tat, obschlossen Greisen ein Asyl zu geben,so hat diese Institution schon ihre Aufgabe um Mission glanzend erfüllt.

Gut fielen mir deshalb auch die wenigen Zeilen,die mir meine Mutter aus diesem Anlasse geschrieben hat und die immer richtig das seelische Moment mit ihrem unendlich liebenden Mutterherz erfasst.Sie schrieb mir:

Bald wurde aber auch diese Freude gestört und es entstand eine neue Sorge,die wir nicht einmal recht erfasst haben.In Neker war ich in Pratislava und reiste von Pratislava mit Trnava ob Herrn Sein Gross,der in Trnava lebte und der Vizestarosta der Ú war.Eigentlich unter der schwachen Leitung des Herrn Arpad Szmetyan,der eigentlich sein Amt überhaupt nicht erfasst hat und als langjähriger Beisitzer der Direktor der jüdischen Bürgerschule auch nur Politik nicht verstehen konnte,war Herr Grass die Seele der Ú und in seiner Hand war die ganze Leitung. Damals erfolgte die Ernennung des Herrn Dr.Knoch zum Vorstand der 24.Abteilung des Innenministeriums.Ich fragte ihn ob er schon den Herrn kenne er bejahte es mir und

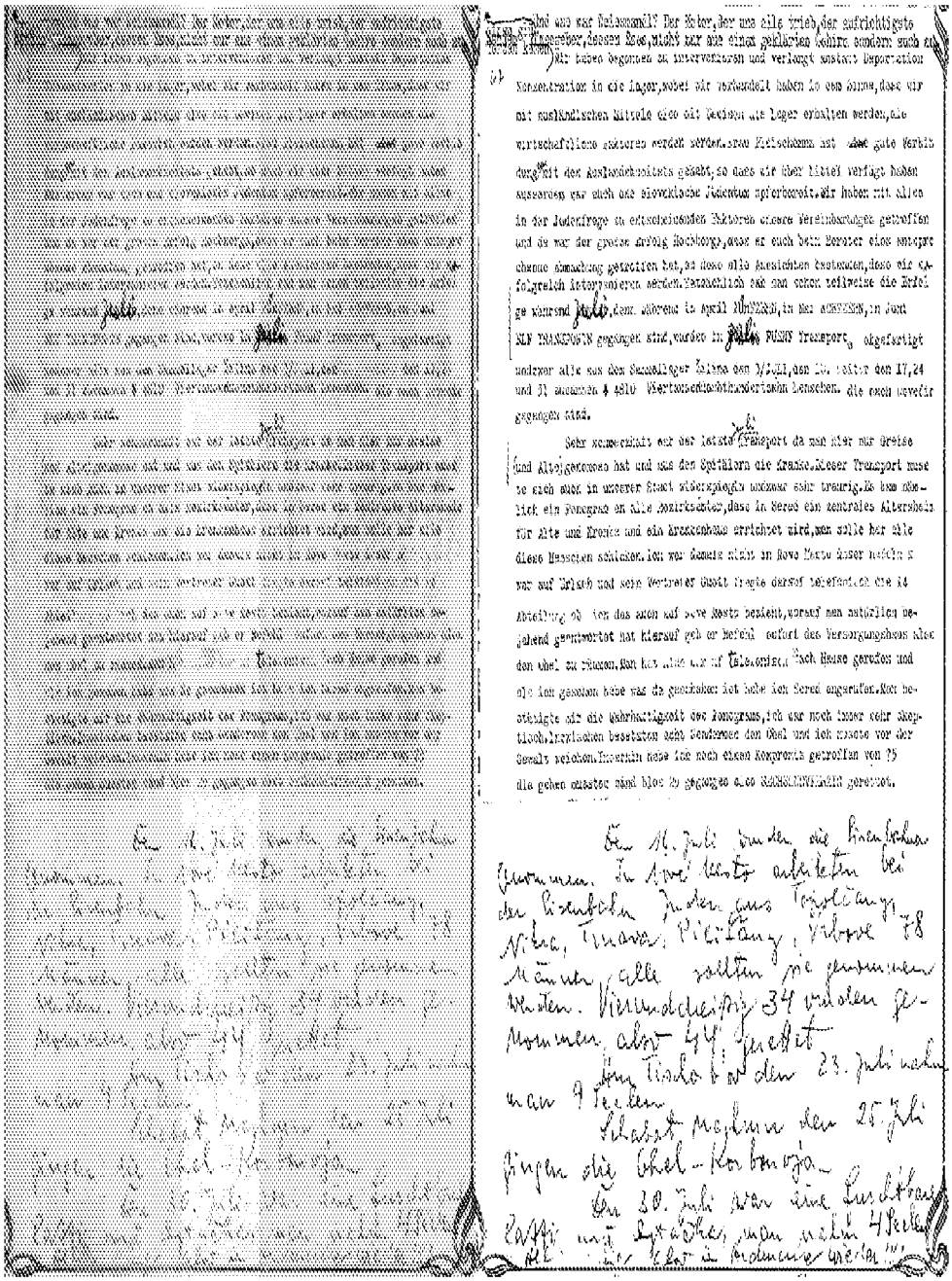
218b

FIG. 8B



200c 218c

FIG. 8C



200d 218d

FIG. 8D

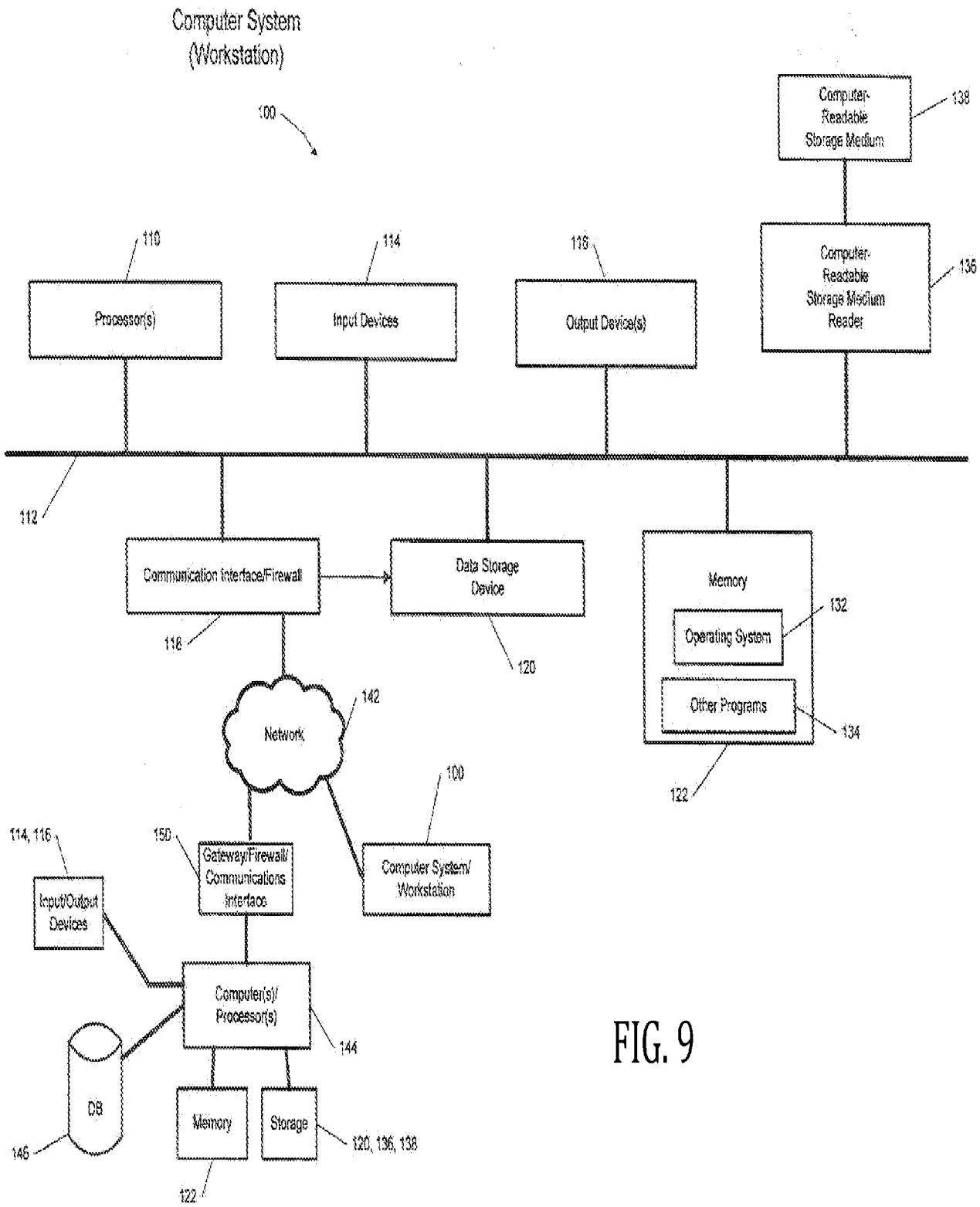

Computer System
(Workstation)
100
Computer-Readable Storage Medium
138
Processor(s)
110
Input Devices
114
Output Device(s)
116
Computer-Readable Storage Medium Reader
136
112
Communication Interface/Firewall
118
Data Storage Device
120
Memory
Operating System
132
Other Programs
134
122
Network
142
100
114, 116
Input/Output Devices
150
Gateway/Firewall/ Communications Interface
Computer System/ Workstation
Computer(s)/ Processor(s)
144
DB
146
Memory
Storage
120, 136, 138
122

FIG. 9

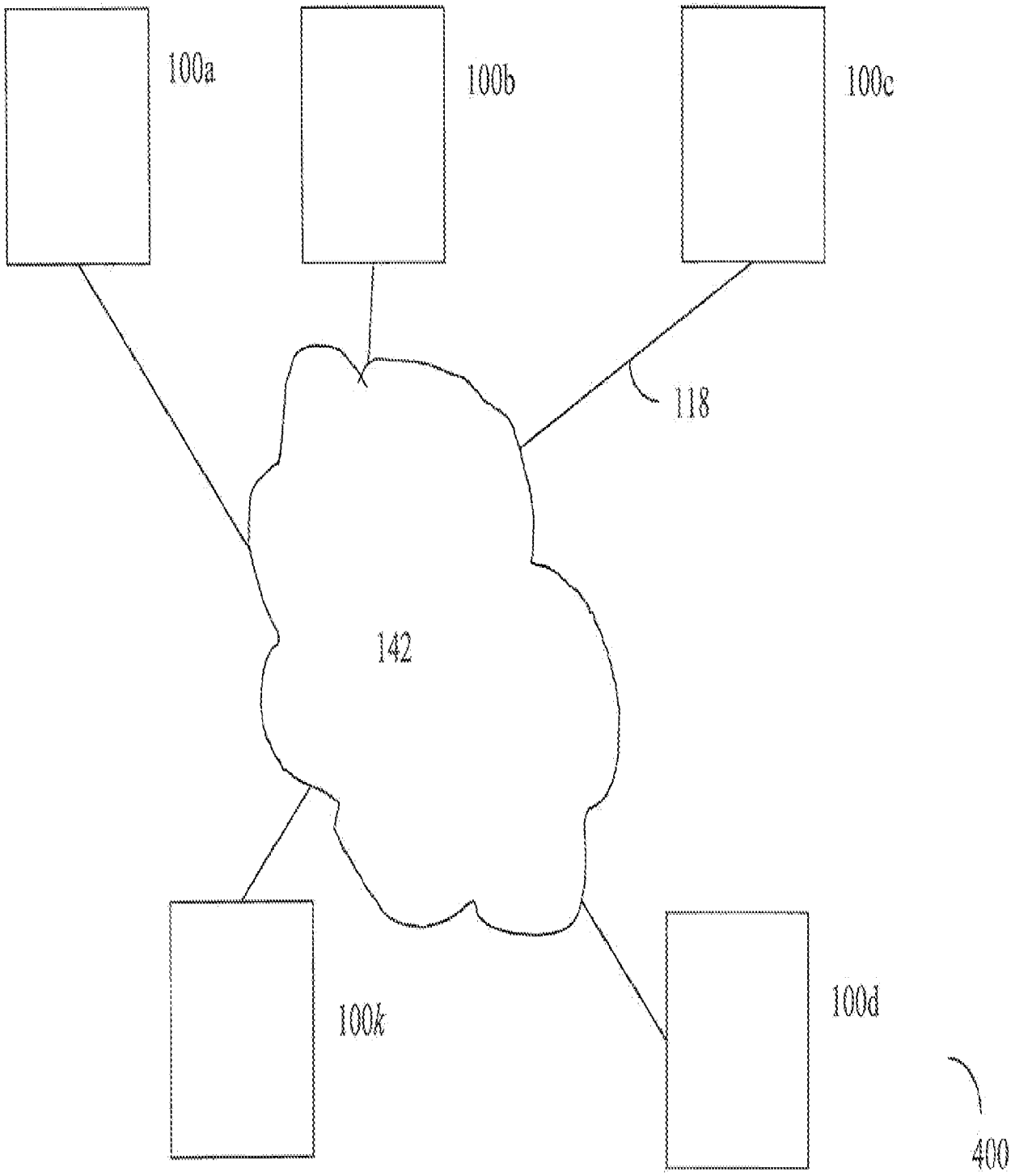


FIG. 10

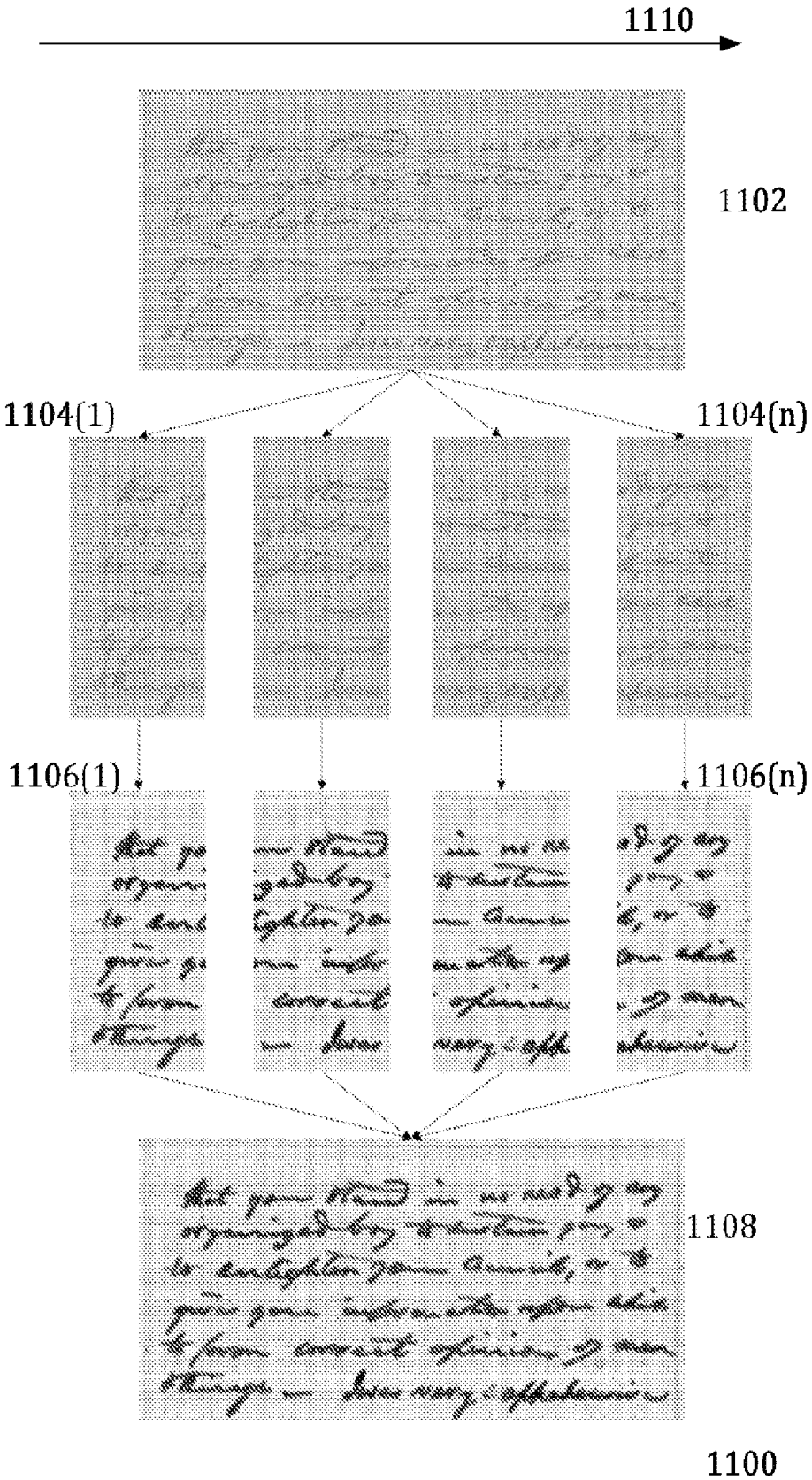


FIG. 11

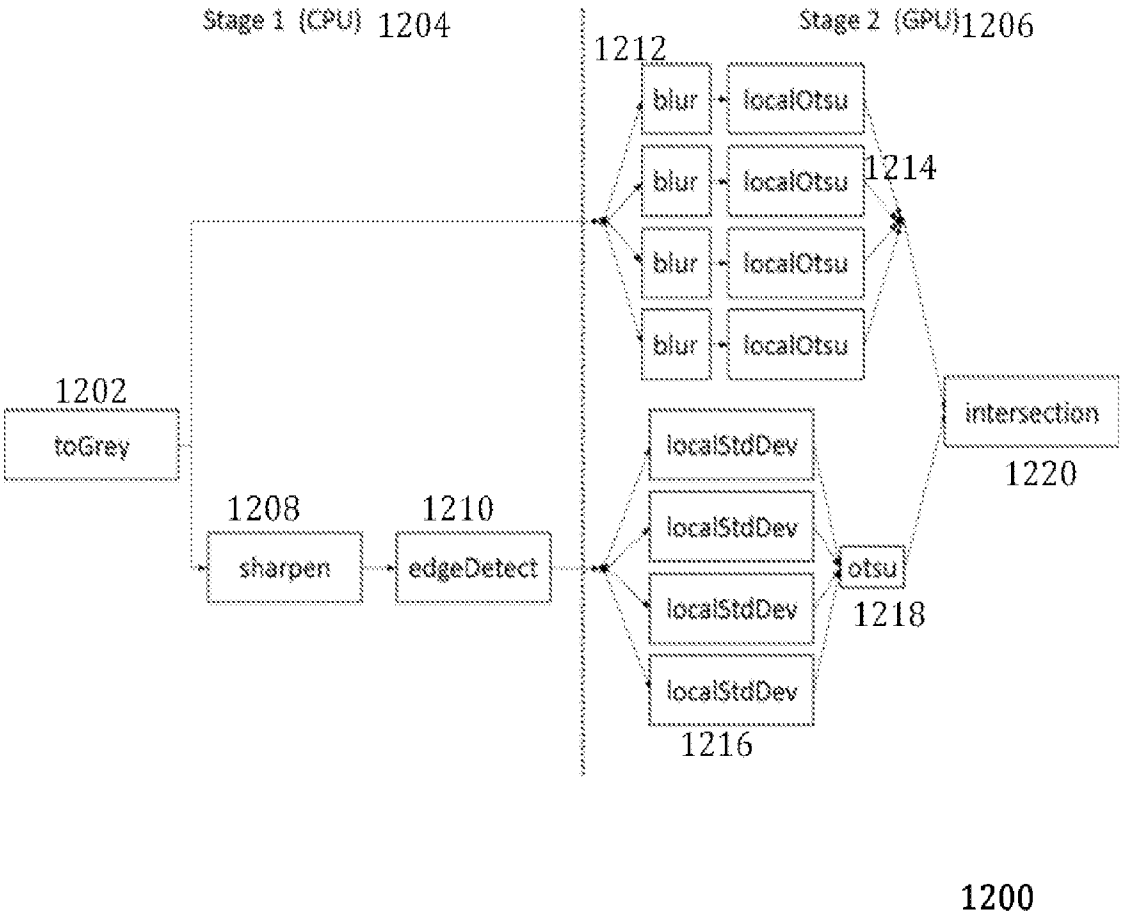

Stage 1 (CPU) 1204
Stage 2 (GPU) 1206
1212
blur
localOtsu
blur
localOtsu
blur
localOtsu
blur
localOtsu
1214
1202
toGrey
intersection
1220
localStdDev
localStdDev
localStdDev
localStdDev
1208
1210
sharpen
edgeDetect
otsu
1218
1216
1200

FIG. 12

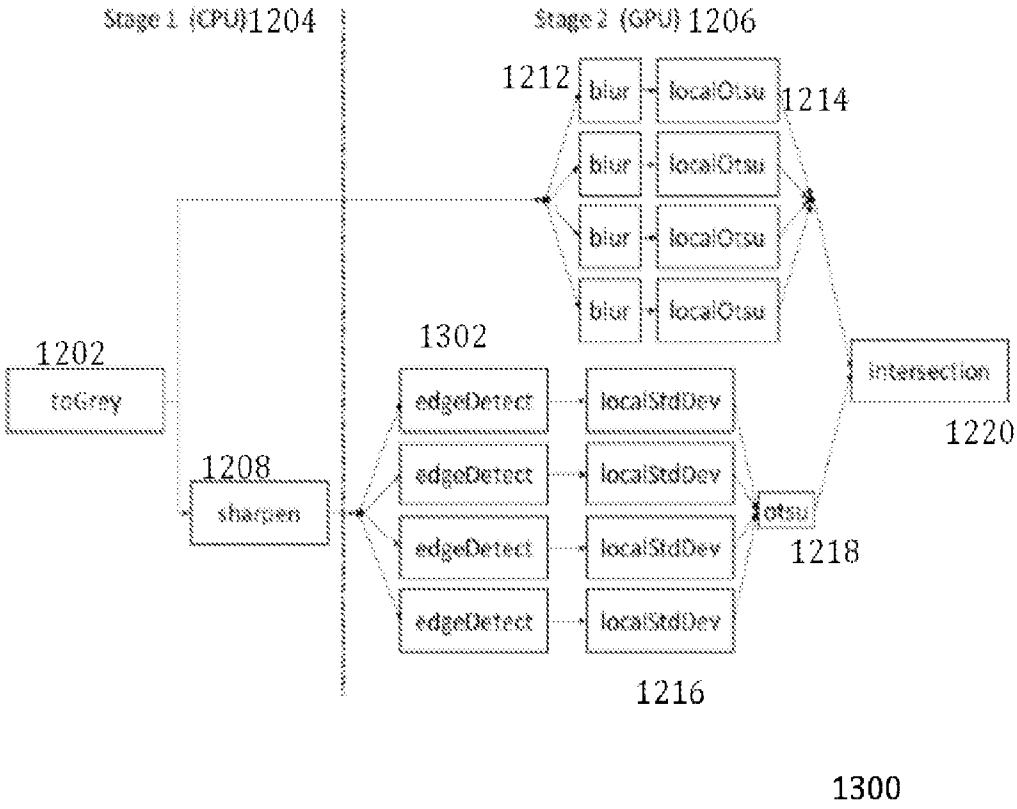

Stage 1 (CPU) 1204
Stage 2 (GPU) 1206
1212
blur
localOtsu
1214
1202
toGray
1302
edgeDetect
localStdDev
intersection
1220
1208
sharpen
otsu
1218
1216
1300

FIG. 13

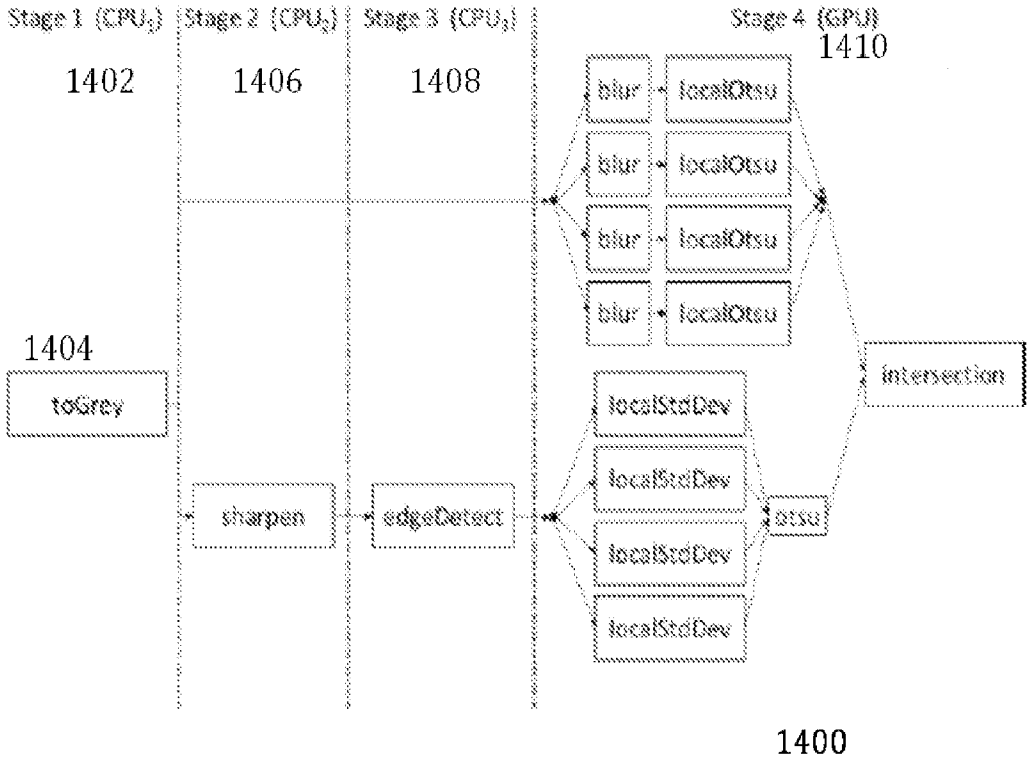

Stage 1 (CPU1)
Stage 2 (CPU2)
Stage 3 (CPU3)
Stage 4 (GPU)
1402
1406
1408
1410
1404
toGrey
sharpen
edgeDetect
blur
localOtsu
localStdDev
otsu
intersection
1400

FIG. 14

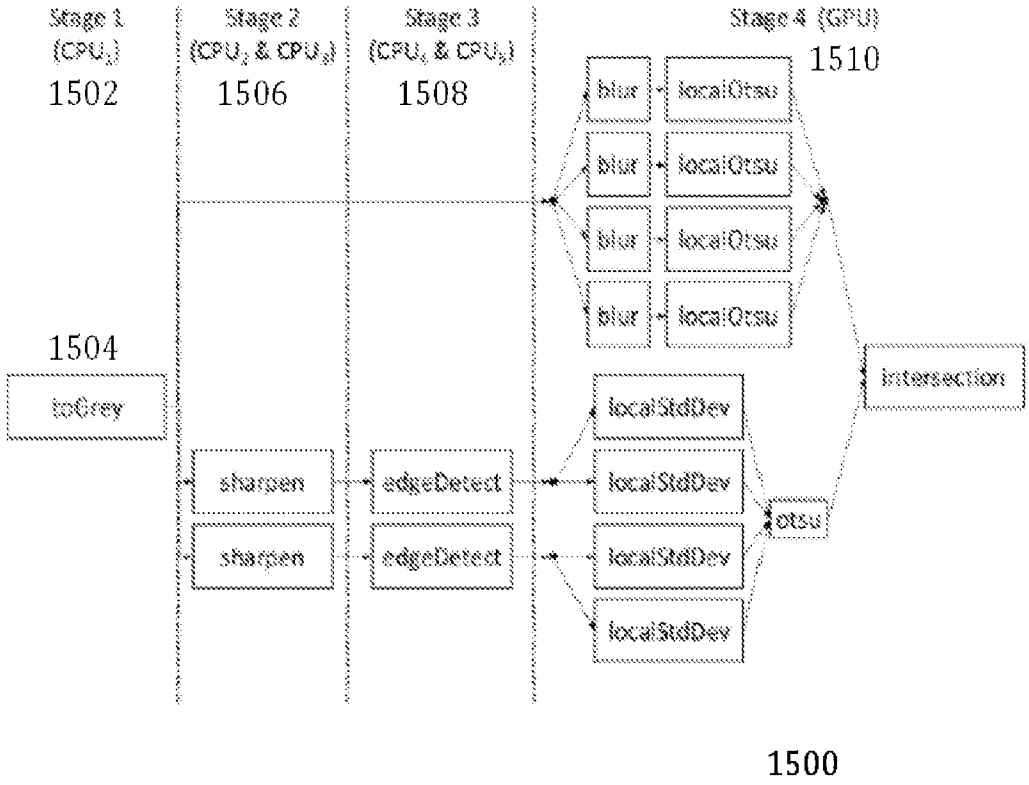

Stage 1
(CPU1)
1502
Stage 2
(CPU2 & CPU3)
1506
Stage 3
(CPU4 & CPU5)
1508
Stage 4 (GPU)
1510
1504
toGrey
sharpen
sharpen
edgeDetect
edgeDetect
blur
blur
blur
blur
localOtsu
localOtsu
localOtsu
localOtsu
localStdDev
localStdDev
localStdDev
localStdDev
otsu
intersection
1500

FIG. 15

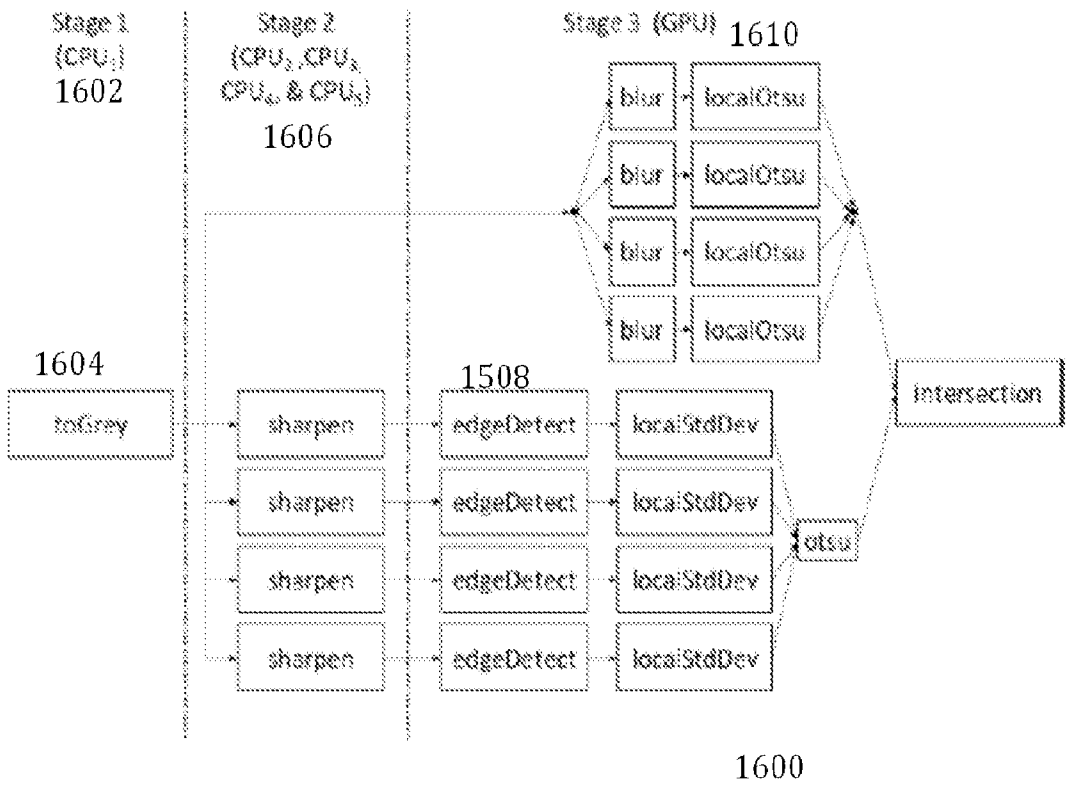

Stage 1
(CPU1)
1602
Stage 2
(CPU2, CPU3,
CPUn & CPUn-1)
1606
Stage 3 (GPU)
1610
1604
1508
toGrey
sharpen
sharpen
sharpen
sharpen
edgeDetect
edgeDetect
edgeDetect
edgeDetect
localStdDev
localStdDev
localStdDev
localStdDev
blur
blur
blur
blur
localOtsu
localOtsu
localOtsu
localOtsu
otsu
intersection
1600

FIG. 16