



(86) **Date de dépôt PCT/PCT Filing Date:** 2014/08/04
(87) **Date publication PCT/PCT Publication Date:** 2015/02/05
(85) **Entrée phase nationale/National Entry:** 2016/02/02
(86) **N° demande PCT/PCT Application No.:** US 2014/049605
(87) **N° publication PCT/PCT Publication No.:** 2015/017855
(30) **Priorité/Priority:** 2013/08/02 (US61/861,610)

(51) **Cl.Int./Int.Cl.** **G01C 11/00** (2006.01),
G01C 11/36 (2006.01), **G06K 9/80** (2006.01),
G06T 7/00 (2006.01), **H04N 13/00** (2006.01)
(71) **Demandeur/Applicant:**
XACTWARE SOLUTIONS, INC., US
(72) **Inventeurs/Inventors:**
TAYLOR, JEFFREY, US;
WEBECKE, EDMUND, US
(74) **Agent:** MCCARTHY TETRAULT LLP

(54) **Titre : SYSTEME ET PROCEDE POUR DETECTER DES ELEMENTS DANS DES IMAGES AERIENNES A L'AIDE DE
TECHNIQUES DE CARTOGRAPHIE D'ECART ET DE SEGMENTATION**
(54) **Title: SYSTEM AND METHOD FOR DETECTING FEATURES IN AERIAL IMAGES USING DISPARITY MAPPING AND
SEGMENTATION TECHNIQUES**

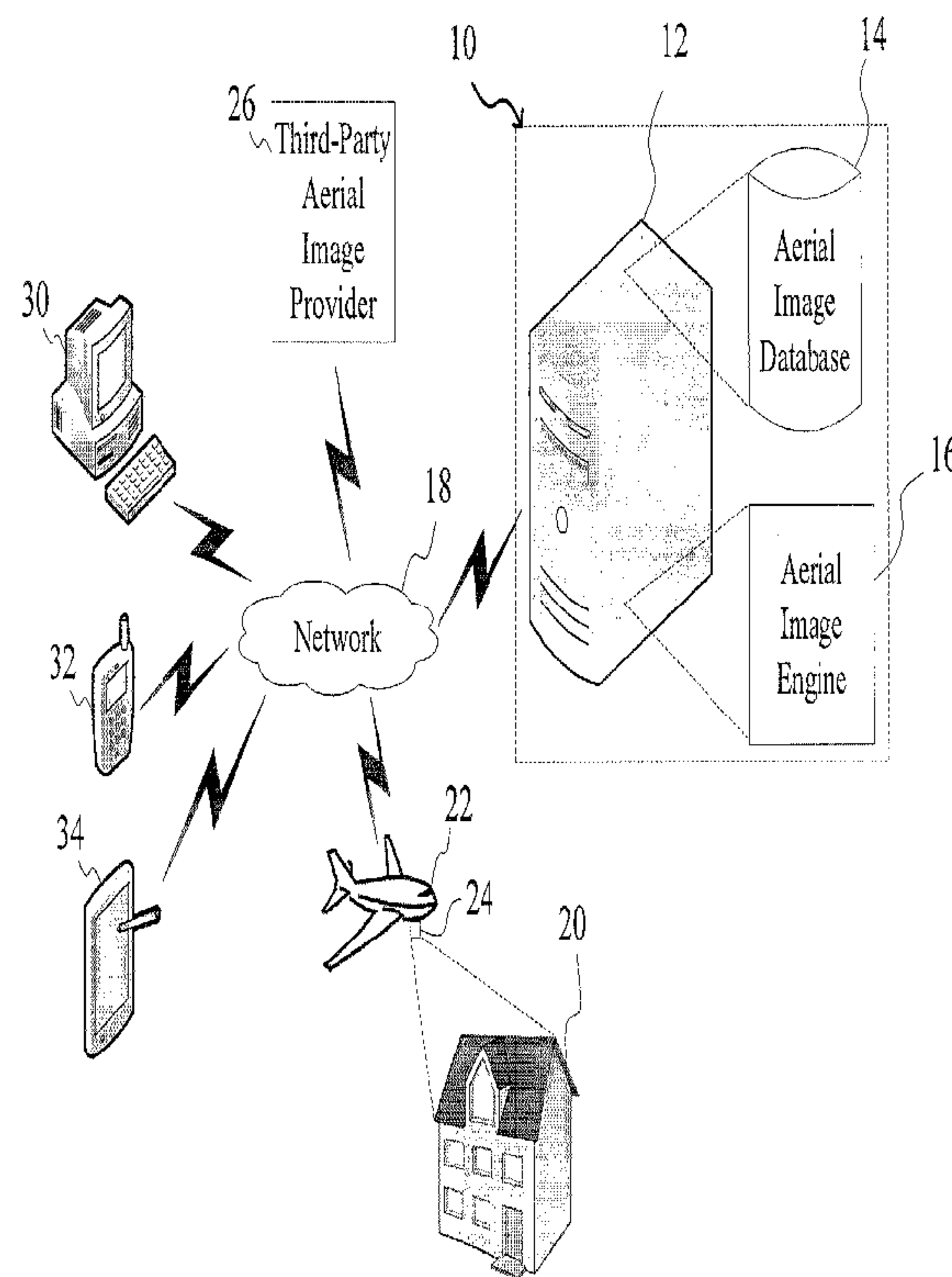


FIG. 1

(57) **Abrégé/Abstract:**

A system for aerial image detection and classification is provided herein. The system comprising an aerial image database storing one or more aerial images electronically received from one or more image providers, and an object detection pre- processing



(57) Abrégé(suite)/Abstract(continued):

engine in electronic communication with the aerial image database, the object detection pre-processing engine detecting and classifying objects using a disparity mapping generation sub-process to automatically process the one or more aerial images to generate a disparity map providing elevation information, a segmentation sub-process to automatically apply a pre-defined elevation threshold to the disparity map, the pre-defined elevation threshold adjustable by a user, and a classification sub-process to automatically detect and classify objects in the one or more stereoscopic pairs of aerial images by applying one or more automated detectors based on classification parameters and the pre-defined elevation threshold.

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

(19) World Intellectual Property
Organization
International Bureau(10) International Publication Number
WO 2015/017855 A1(43) International Publication Date
5 February 2015 (05.02.2015)(51) International Patent Classification:
H04N 13/00 (2006.01)(21) International Application Number:
PCT/US2014/049605(22) International Filing Date:
4 August 2014 (04.08.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/861,610 2 August 2013 (02.08.2013) US(71) Applicant: **XACTWARE SOLUTIONS, INC.** [US/US];
One Xactware Plaza, Orem, UT 84097 (US).(72) Inventors: **TAYLOR, Jeffrey**; 1327 E. 900 S., Pleasant
Grove, UT (US). **WEBECKE, Edmund**; 6907 W. 10300
N., Highland, UT (US).(74) Agents: **NIKOLSKY, Mark, E.** et al.; McCarter & Eng-
lish, LLP, Four Gateway Center, 100 Mulberry Street, Ne-
wark, NJ 07102 (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:**

— with international search report (Art. 21(3))

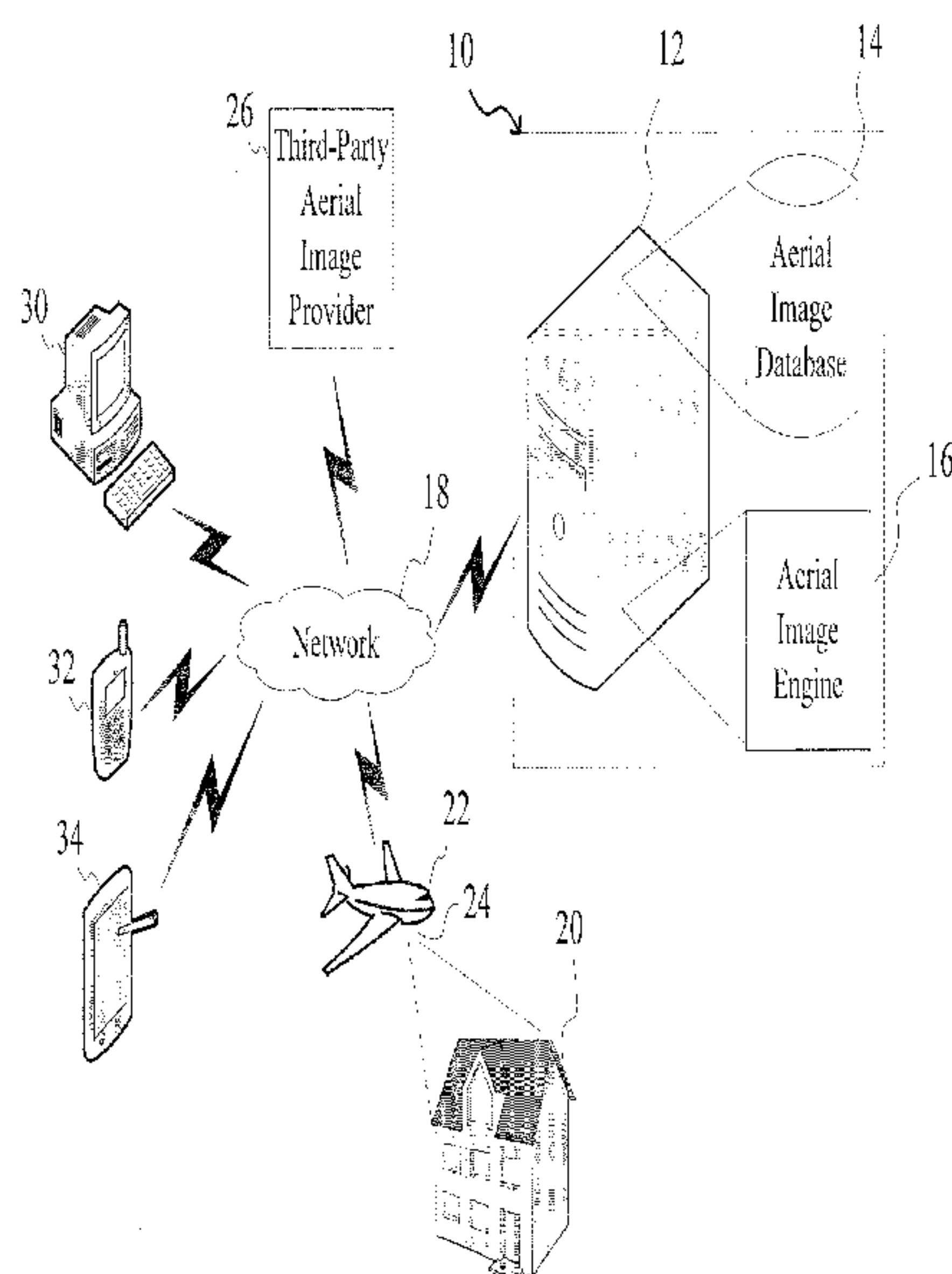
(54) Title: SYSTEM AND METHOD FOR DETECTING FEATURES IN AERIAL IMAGES USING DISPARITY MAPPING
AND SEGMENTATION TECHNIQUES

FIG. 1

(57) Abstract: A system for aerial image detection and classification is provided herein. The system comprising an aerial image database storing one or more aerial images electronically received from one or more image providers, and an object detection pre-processing engine in electronic communication with the aerial image database, the object detection pre-processing engine detecting and classifying objects using a disparity mapping generation sub-process to automatically process the one or more aerial images to generate a disparity map providing elevation information, a segmentation sub-process to automatically apply a pre-defined elevation threshold to the disparity map, the pre-defined elevation threshold adjustable by a user, and a classification sub-process to automatically detect and classify objects in the one or more stereoscopic pairs of aerial images by applying one or more automated detectors based on classification parameters and the pre-defined elevation threshold.

SYSTEM AND METHOD FOR DETECTING FEATURES IN AERIAL IMAGES
USING DISPARITY MAPPING AND SEGMENTATION TECHNIQUES

SPECIFICATION

5

BACKGROUND

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 61/861,610 filed on August 2, 2013, the entire disclosure of which is expressly
10 incorporated herein by reference

FIELD OF THE DISCLOSURE

The present disclosure relates generally to the field of aerial image detection and classification. More specifically, the present disclosure relates to a system and method for
15 detecting features in aerial images using disparity mapping and segmentation techniques.

RELATED ART

Accurate and rapid identification and estimation of objects in aerial images is increasingly important for a variety of applications. For example, roofing information is
20 often used by construction professionals to specify materials and associated costs for both newly-constructed buildings, as well as for replacing and upgrading existing structures. Further, in the insurance industry, accurate information about construction materials and costs is critical to determining the proper costs for insuring buildings/structures.

Various software systems have been implemented to process aerial images to
25 identify building structures and associated features thereof. However, such systems are often time-consuming and difficult to use, and require a great deal of manual input by a user. Further, such systems may not have the ability to improve results through continued usage over time.

In view of existing technology in this field, what would be desirable is a system
30 that automatically and efficiently processes aerial images to automatically identify various types of objects in the images. Moreover, what would be desirable is a system that self-improves over time to become more accurate and efficient. Accordingly, what would be desirable, but has not yet been provided, is a system and method for detecting features in

aerial images using disparity mapping and segmentation techniques which addresses these needs.

SUMMARY

The present system of the current disclosure detects features in aerial images using disparity mapping and segmentation techniques. More specifically, the system includes an object detection pre-processing engine for object detection and classification using one or
5 more aerial images. The object detection pre-processing engine includes disparity map generation, segmentation, and classification to identify various objects and types of objects in an aerial image. Detection algorithms, including region growing algorithms and split-and-merge segmentation algorithms, are applied to an image to identify structures. These component-based algorithms can evolve and become more efficient over time. The
10 information derived from these pre-processed images can then be used by the mass production engine for the manual and/or automated production of drawings, sketches, and models. A quality control engine could also be used for ensuring the accuracy of any drawings, sketches, or models generated by the system.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing features will be apparent from the following Detailed Description, taken in connection with the accompanying drawings, in which:

FIG. 1 is a diagram showing the system of the present disclosure for detecting
5 features in aerial images;

FIG. 2 is a diagram showing software components of the aerial image engine;

FIG. 3 is a flowchart showing the steps for processing one or more images by the object detection preprocessing engine;

FIG. 4 is an image depicting object classification carried out by the system using a
10 pre-defined elevation threshold;

FIG. 5 is a diagram of various classification parameters utilized by the system;

FIG. 6 illustrates processing of an aerial image by the system using a region growing algorithm to identify structures, as well as footprints of such structures;

FIG. 7 illustrates processing of an aerial image by the system using a split-and-
15 merge segmentation algorithm to identify structures;

FIG. 8 is a flowchart showing steps for evaluating and improving models generated by the system;

FIGS. 9-11 are screenshots showing various user interface screens generated by the system;

FIG. 12 is a diagram showing hardware and software components of the system;
20

FIG. 13 is a diagram showing processing steps carried out by the aerial image engine of the present invention for generating a point cloud from a pair of aerial images;

FIG. 14 is a disparity map generated by the engine using a depth from disparity method;

FIG. 15 is a point cloud generated by the engine using the depth from disparity
25 method;

FIG. 16 is a disparity map generated by the engine using the eye-ray method;

FIG. 17 is a point cloud generated by the engine using the eye-ray method;

FIG. 18 is a diagram showing processing steps for generating a composite point
30 cloud from multiple image pairs;

FIG. 19 is a nadir view of a point cloud based on a nadir pair of images;

FIG. 20 is an oblique view of a point cloud based on a nadir pair of images;

FIG. 21 is a slice of the point cloud that crosses the front windows of the building shown based on a nadir pair of images;

FIG. 22 is a nadir view of a point cloud based on an oblique pair of images;

FIG. 23 is an oblique view of a point cloud based on an oblique pair of images;

5 **FIG. 24** is a slice of the point cloud that crosses the front windows of the building shown based on an oblique pair of images;

FIG. 25 is a nadir view of a point cloud based on a combination of nadir and oblique pairs of images;

10 **FIG. 26** is an oblique view of a point cloud based on a combination of nadir and oblique pairs of images;

FIG. 27 is a slice of the point cloud that crosses the front windows the building shown based on a combination of nadir and oblique pairs of images; and

FIG. 28 is a diagram illustrating modules and associated processing steps for automated building recognition carried out by the aerial image engine.

DETAILED DESCRIPTION

The present disclosure relates to a system and method for detecting features in aerial images using disparity mapping and segmentation techniques, as discussed in detail below in connection with **FIGS. 1-28**.

5 **FIG. 1** is a diagram showing the system of the present disclosure for detecting features in aerial images, indicated generally at **10**. The system **10** comprises a computer system **12** (e.g., a server) having an aerial image database **14** stored therein and a software aerial image engine (module) **16**. The database **14** could be stored on the computer system **12**, or located externally (e.g., in a separate database server in communication with the
10 system **10**). As will be discussed in greater detail below, the aerial image engine **16** allows users to detect features in aerial images and generate three-dimensional models therefrom.

The system **10** can communicate through a network **18** with one or more of a variety of image providers to obtain aerial images or photographs of a building structure **20** and can store them in the aerial image database **14** in any suitable format, such as JPEG,
15 TIFF, GIF, etc. Network communication could be over the Internet using standard TCP/IP communications protocols (e.g., hypertext transfer protocol (HTTP), secure HTTP (HTTPS), file transfer protocol (FTP), electronic data interchange (EDI), etc.), through a private network connection (e.g., wide-area network (WAN) connection, e-mails, electronic data interchange (EDI) messages, extensible markup language (XML) messages,
20 Javascript Object Notation messages (JSON) file transfer protocol (FTP) file transfers, etc.), or any other suitable wired or wireless electronic communications format. Image providers that the computer system **12** could communicate with include, but are not limited to, an airplane **22** (or unmanned autonomous vehicle (UAV)) having a camera **24** capable of capturing images of the structure **20**, and/or a third-party aerial image provider **26**, such
25 as Pictometry, Google, or Bing.

The computer system **12** could be any suitable computer server (e.g., a server with an INTEL microprocessor, multiple processors, multiple processing cores) running any suitable operating system (e.g., Windows by Microsoft, Linux, etc.). The computer system **12** includes non-volatile storage, which could include disk (e.g., hard disk), flash memory,
30 read-only memory (ROM), erasable, programmable ROM (EPROM), electrically-erasable, programmable ROM (EEPROM), or any other type of non-volatile memory. The aerial image engine **16**, discussed in greater detail below, could be embodied as computer-readable instructions stored in computer-readable media (e.g., the non-volatile memory

mentioned above), and programmed in any suitable programming language (e.g., C, C++, Java, etc.).

The system **10** could be web-based and could allow for remote access to the system **10** over a network **18** (e.g., Internet, WAN, LAN, etc.) by one or more devices, such as a personal computer system **30**, a smart cellular telephone **32**, a tablet computer **34**, or other devices. It is also contemplated that at least some of the functionality of the system **10** could run locally on devices (e.g., personal computer **30**, smart cellular telephone **32**, tablet computer **34**, etc.) programmed with software in accordance with the present disclosure. It is conceivable that, in such circumstances, the device could communicate with a remote aerial image database over a network **18**.

FIG. 2 is a diagram showing software components **50** of the aerial image engine **16**. The components **50** include a project and task management system **52**, object detection preprocessing engine **60**, mass production engine **62**, and quality control (QC) engine **64** (discussed in more detail below). The project and task management system **52** allows for the creation, processing, and monitoring of projects relating to the processing of aerial images. The projects could be based on high resolution aerial imagery of areas, where the images have been captured, copied, and pre-processed. Each project is tracked in detail (e.g., steps toward completion, time spent on each task, etc.) to help pinpoint target areas for optimization, simplification, and/or improvement.

The project and task management system **52** includes several distinct modules. More specifically, the system includes a management server **54**, work manager **56**, and web manager **58**. The management server **54** is a set of web services that store and serve geo-referenced data, including raw data (e.g., data generated by computer vision (CV)) and elaborated data (e.g., new and previous sketches, ITV's (insurance-to-value), insurance claims, and other related data). The management server **54** also provides a feedback mechanism that lets users quickly and efficiently return new and improved training data to the object detection preprocessing engine **60**.

The work manager **56** is a set of web services that dispatches tasks to low-cost, highly-trained operators, and then processes and stores the results of the work that they accomplish. The work manager **56** ensures that projects and tasks are assigned orderly based on priority and urgency levels. For example, customer requests could be assigned the highest priority, followed by customer PIF (policy in force) addresses, and then AOI's (areas of interest) with the most historical significance. The web manager **58** is a full web

application user interface that allows managers to handle creating projects, managing property contractors and operators, work monitoring, and tracking of historic data, productivity statistics (e.g., per operator, team, and/or project block, etc.), and other statistics.

5 The object detection preprocessing engine **60**, discussed in more detail below, detects structures in images, and then processes the images to identify different types of objects. More specifically, the object preprocessing engine **60** processes imagery to analyze stereoscopic pairs of images and detect various objects of interest (e.g., buildings, trees, pools, noise (elements with a significant level of entropy), etc.). For example, the
10 object detection preprocessing engine **60** could take preprocessed building structure perimeter information, add automatic line finding capabilities, and provide the ability to gather height information from stereoscopic pairs.

 The mass production engine **62** (e.g., mass production client application) is an automatically updated smart client (e.g., desktop, mobile, or web application) for quickly
15 creating aerial models (e.g., 3D models) of one or more structures and accompanying prefill and metadata for an aerial image library (which could be address-based). The mass production engine **62** includes software tools to support the manual and automated process of creating a roof and/or property report. The mass production engine **62** could be a closed system which works in conjunction with designated web services and is programmed to
20 protect any personally identifiable information (PII) data (e.g., the system could withhold from operations of the system actual address information or geocode information of a structure, remove imagery that is no longer needed from the local cache, etc.).

 The quality control engine **64** ensures the accuracy of the model and related data generated from the images. The quality control engine **64** could be automated and/or could
25 guide a technician in review and verification.

FIG. 3 is a flowchart showing the steps for processing one or more images by the object detection preprocessing engine **60**. Broadly, the process includes disparity map and/or point cloud generation sub-process **74**, segmentation sub-process **84**, and classification sub-process **90**. In step **72**, the system electronically receives one or more
30 aerial images (e.g., imagery intake, imagery input data, etc.). High-level imagery specifications (e.g., proprietary satellite images) could be used for maximum automation and efficiency in relation to production, but the system could use lower-quality data inputs

(e.g., third-party data) as well. The clearer the aerial images that are used as input, the better the results of the automated detection algorithms.

In sub-process 74, the system generates a disparity map and/or point cloud, which provides information about the elevation of the structures (e.g., objects, elements, etc.) present in the stereoscopic pair of images. To generate a disparity map and/or point cloud, in step 76, the system uses world file information to process the overlapped region between stereoscopic images. One or more image pairs can be used in this process, and the resulting disparity maps and/or point clouds can be combined to gain additional information. In step 78, the orientation of each image (e.g., left and right images) is processed, such as by using the camera position. In step 80, if needed (e.g., particularly if the overlapping images are from different flight lines), the brightness of the images is normalized. A disparity map and/or point cloud is then generated in step 82. The parameters used to generate the disparity map and/or point cloud are fine-tuned to account for differences between imagery data (e.g., differences produced by different camera systems, differences in sun angles, etc.) and other factors. The system could use other in-flight or post-flight processing systems capable of producing accurate disparity maps and/or point clouds.

In sub-process 84, segmentation is applied to the image, which allows the system to detect changes in different parts of the image that are later grouped together into areas based on similarities. These areas are subsequently classified (e.g., as structures, trees, pools, etc.), as discussed below in more detail. To apply segmentation, in step 86, a height threshold is applied to the disparity map and/or point cloud. This threshold is adjustable, but (for reasons relating to classification) should be taller than the height of a standard house or the tallest tree in a given area. In step 88, one or more automated detectors (e.g., algorithms) are applied to objects in the image that are below the threshold to initially detect other objects (e.g., buildings). Automated detectors become more accurate and efficient over time and can be tuned and continually added. When one or more new detectors are added, the database could be reprocessed to run just the new detectors. Algorithms that could be used include region growing algorithms and/or split-and-merge segmentation algorithms (which could be used to find blobs that may be subsequently identified as structures, trees, noise, etc.), as well as object/feature detection algorithms. These algorithms are discussed in more detail in FIGS. 6-7.

In step 90, classification is applied to detect and classify objects (e.g., buildings, trees, pools, noise, etc.). To apply classification, in step 92, objects higher and/or taller than the certain predefined threshold (based on the height information derived by the disparity map) are automatically added as structures (e.g., automatically classified as buildings). In step 94, areas are classified based on classification parameters using (and training) machine learning algorithms, such as neural networks. Machine learning algorithms and neural networks are powerful mechanisms which provide the system with the ability to learn and acquire experience from existing data and processes. For example, the network could be trained using an image database containing any number of stereoscopic image pairs, where the images are taken from different locations (including residential, industrial and commercial areas) and from datasets that have been captured using different types of sensor technology. The trained network could be tested using a test image database and an automated test tool. After the images have been pre-processed, a data package containing all information derived from the aerial images could be stored in a property database for future use by users or software applications.

FIG. 4 is an image 170 depicting object classification carried out by the system using a pre-defined elevation threshold. As shown, the height threshold 174 is adjustable to be a certain distance from the ground 172. The threshold 174 is set to be larger than trees 176 and houses 178 in the area. Anything above that threshold 174, such as a building tower 180, is automatically classified as a building, because it is too tall to be anything else.

FIG. 5 is a diagram of various classification parameters 96 that can be utilized by the system. Classification is based on a set of specific parameters 96 that can be modified and evolved to refine and improve the classification or detection of new elements (e.g., cars, vegetation, etc.). Classification parameters 96 could include aspect 98, disparity 114, color 126, and/or texture 132 parameters. More specifically, the aspect parameter 98 could include area 100 (e.g., blob area in pixels, where small blob areas are automatically discarded), aspect ratio 102 (e.g., best blob fitting oriented rectangle width and height ratio), rectangle fitting 104 (e.g., best blob fitting oriented rectangle area and blob area ratio, which tends to separate buildings from trees and noise), contour defects 106 (e.g., sum of defects of the blob convex polygon, which tends to separate buildings from trees and noise, where the noise generates more defects), HOG (histogram of oriented gradient) maximum 108 (e.g., distance between two main peaks in the HOG, and is ideally used for

buildings having a distance of 90°), HOG property **110** (e.g., contour pixel proportion that belongs to the two main peaks in the HOG, which tends to distinguish buildings from trees), and/or HOG uniform **112** (e.g., Kolmogorov test of similarity between the HOG and the uniform distribution, which shows a low similarity in buildings and a high value in
5 trees due to spread orientations).

The disparity parameter **114** could include height mean **116** (e.g., mean of the blob disparity values, because noise has lower values than buildings or trees), height deviation **118** (e.g., standard deviation of the blob disparity values), distance to height **120** (e.g., sum of contour pixel distance to the edges of the disparity map, because noise usually presents
10 a high distance value), contour correspondence **122** (e.g., sum of contour correspondences with contrasted disparity, because buildings and trees present a high contour correspondence value), ground prop **124** (e.g., analysis of the disparity between a reference point (ground point) and a studied point of a given blob, because noise usually belongs to ground). The color parameter **126** could include RGB (red green blue) **128** (e.g., mean
15 value of color channels, such as to separate buildings from trees, which are usually green) and HSV (hue, saturation value) **130** (e.g., mean value of HSV channels). The texture parameter **132** could include deviation mean **134** (e.g., mean of the deviation of a window ceiling of the blob, which could separate trees from buildings due to contrasted lighting in leaves) and/or Sobel mean **136** (e.g., mean of the deviation of a window ceiling of the blob
20 with a high pass filter applied to increase contrast).

FIG. 6 illustrates processing of an aerial image **150** by the system using a region growing algorithm to identify structures, as well as footprints **152** of such structures. The algorithm uses the highest points found in the disparity map as seed values and begins the region growing process from there outward. The region growing algorithm uses not only
25 color information, but also disparity map information, which prevents growing over low-elevation areas. This works particularly well on high-resolution imagery (though the processing time per image is relatively large). However, if the image resize parameter is too large, it can cause inaccuracies in the structure footprint detected. Parameters of the region growing algorithm could include:

30

Parameter	Default Value
Image resize	30%
Ground proportion	45%
Similar RGB & HSV	Fuzzy set {0, 5, 10}
Building detection threshold	Disparity = 200
Noise threshold	Area = 700 pixels
Thresholding analysis step	Disparity = 10
Minimum seed area	100 pixels

Table 1

FIG. 7 illustrates processing of an aerial image **162** by the system using a split-and-merge segmentation algorithm to identify structures. The split-and-merge segmentation algorithm (which is faster than the region growing algorithm) looks at the whole image **162** and determines if it is non-uniform or uniform. If it is non-uniform, the image is split into four squares. Then, it looks at each of those four squares and repeats the process on the squares until a complete segmentation map is created, where all squares are 100% uniform within them, as shown in image **164**. This algorithm is useful for finding differences between different areas of an image and detecting the location of the changes. This, as shown in image **166**, can be used to define the roof lines that separate the different sections of a roof. The split-and-merge algorithm uses not only color information but also the disparity map information (e.g., height map information) as input data. In this way, the interior lines are detected based on both the imagery and the disparity map information.

FIG. 8 is a flowchart showing steps **200** for evaluating and improving models generated by the system. This process is optional and not required for object detection. The basic workflow includes determining the dimensions of the structure and creating a sketch or drawing. For example, the system provides the ability to determine and define a geocode at a front door (or one of the front doors) of a structure. Further, the system provides the ability to determine the footprint of the structure, roof dimensions (e.g., number of faces and other roof attributes), default area living space and non-living space (e.g., square feet), number of stories, number of car garage attributes, number and size of housing attributes (e.g., windows and doors), type of roofing material (e.g., shingles, metal, wood shake, etc.), exterior wall materials (e.g., stucco, brick, stone, etc.), and the number,

size, and material of exterior property attributes (e.g., swimming pools, decks, etc.), among other things.

In step **204**, the project and task management system (e.g., web manager application) guides a manager **202** in creating and publishing one or more projects. Publishing the project (automatically or manually) assigns it to a specific team (or operator) and makes the tasks in the project available in a queue. The manager can prioritize tasks within a project and across projects, thereby controlling the priority (on a per project basis) of how the models and metadata are processed.

In step **206**, an operator of the assigned team, once available, is automatically assigned the highest priority task from the queue. The necessary pre-processed data, including data defaults and imagery, is then retrieved from one or more databases for the operator. These secondary methods are provided for operators to derive information where required and where automated detectors yield inaccurate or undetected results. Generally, the mass production engine guides an operator through the following steps: define **210**, perimeter **212**, interior lines **214**, metadata **216**, and submit **218**, as discussed below. In step **210**, the mass production engine allows the operator **208** to define the property by displaying for his/her review one or more images from the specified location and the default data thereof. When required, the operator **208** marks which buildings and related structures belong to that particular property. This provides the operator **208** with the ability to separate and combine structures and/or to identify new structures, which is useful if the object preprocessing engine did not accurately find a structure or merged together separate structures. If a new structure is created, a new task will be added to the appropriate queue and subsequently assigned to another operator. Where the parcel boundary geographic accuracy and/or detector default data is acceptable, this step would only require a quick review and verification by the operator **208**.

In step **212**, the mass production engine allows/guides the operator **208** to verify and/or edit (e.g., creates, adjusts, etc.) the roof perimeter. Although preprocessing would most likely have accurately identified the perimeter, it may be necessary to adjust the perimeter (e.g., moving the points defining the perimeter) to match the exact contour of the building. In step **214**, the mass production engine allows/guides the operator **208** to verify and/or edit (e.g., correct, add, remove, etc.) the interior lines of the structure.

In step **216**, the mass production engine allows/guides the operator **208** in creating the metadata associated with the property. The operator could examine the imagery and answer a specific set of questions about the property. The user interface would guide the operator **208** through desired attribute specifications or to verify automated pre-fill results.

5 Answering the question could require clicking on a point on the model, such as marking the front door geo-code or verifying roof features. The metadata could include roof material (e.g., shingle, shake, metal, tile/slate or membrane), number and placement of roof features (e.g., chimneys, roof vents, turtle vents, skylights, etc.), front door geocode location, number of levels, walls, exterior material(s)/percentages, default area living

10 compared with nonliving space, number, size, and placement of doors, windows, garage stalls, rain gutters, air conditioner units, trees, swimming pools, etc. After all phases of work have been successfully completed, in step **218**, the operator **208** submits the model and metadata.

In step **220**, automated QC checks (automated algorithms and/or operator input

15 prompts) are implemented by the mass production engine to verify the validity and accuracy of the model and related data (e.g., metadata). This ensures that the images and related data will successfully import into other products (e.g., programs, engines, etc.). If the checks fail, the operator is notified of the error and the submission process is canceled. Otherwise, depending on the parameters of the project, the operator is given a new task,

20 and the model is selected and added to the QC queue (or alternatively published for use).

In step **224**, when the QC technician **222** is ready, the system (e.g., web application) pulls the highest priority structure from the QC queue and displays it on top of the appropriate nadir and oblique imagery. The system also allows any other imagery of the particular location to be displayed with the model. In step **226**, the QC engine prompts

25 the QC technician **222** to review both the model and related data (e.g., metadata). In step **228**, the engine prompts the QC technician **222** to mark the structure as either verified (to be published to the library) or rejected and returned to the operator for review (its priority increased to push it higher in the operator's queue). When rejecting the model, the QC technician **222** can specify the reason from a canned list of possible issues provided by the

30 system and/or draft a custom message. Multiple levels of quality assurance (e.g., teams) could be configured with varying responsibilities.

FIGS. 9-11 are showing various user interface screens generated by the system. More specifically, **FIGS. 8-9** are screenshots **250, 260** showing a split pane interface allowing multiple views (e.g., left view **252, 262** and right view **254, 264**) of the structure to help better evaluate and analyze details. The system provides an operator with basic tools including pan and zoom, the ability to display all images covering the structure, line and vertices adjustments (e.g., adding, editing and deleting vertices and lines), snapping appropriate angles between lines (90°, 45°, etc.), ensuring lines are parallel and/or horizontal, snapping to existing lines/vertices, etc. The interface could also provide upper toolbar **256, 266** and lower toolbar **258, 268**, providing further functionality (e.g., define property, create contour, create break lines, submit model, new vertex, new rectangular boundary, find geometries by color, autodetect model, force coplanarity, display filling color, show detected polygon, new block, remove block, etc.). This allows the operator to view and edit the model frame with the model frame superimposed on one or more aerial different aerial images of the roof (e.g., orthographic, etc.).

FIG. 11 is another screenshot of a user interface of the system. The screenshot **270** shows the split pane interface, with an aerial view of a housing structure in the left pane **272**, a three-dimensional model of the housing structure in the right pane **274**, and a submission window **276** overlaying both. The submission window **276** includes a comment area **282** for a user to manually write a report, a submit button **278**, and a list of possible issues or submission alternatives **280** to select (e.g., accept with warning, bad images, tree was found, pool was found, unrecognized area, etc.). The system could also present the operator with questions to record any useful feedback (e.g., for quality assurance).

FIG. 12 is a diagram showing hardware and software components of the system **300**. The system **300** comprises a processing server **302** which could include a storage device **304**, a network interface **308**, a communications bus **310**, a central processing unit (CPU) (microprocessor) **312**, a random access memory (RAM) **314**, and one or more input devices **316**, such as a keyboard, mouse, etc. The server **302** could also include a display (e.g., liquid crystal display (LCD), cathode ray tube (CRT), etc.). The storage device **304** could comprise any suitable, computer-readable storage medium such as disk, non-volatile memory (e.g., read-only memory (ROM), eraseable programmable ROM (EPROM), electrically-eraseable programmable ROM (EEPROM), flash memory, field-programmable gate array (FPGA), etc.). The server **302** could be a networked computer system, a

personal computer, a smart phone, tablet computer etc. It is noted that the server **302** need not be a networked server, and indeed, could be a stand-alone computer system.

The functionality provided by the present disclosure could be provided by an aerial image engine **306**, which could be embodied as computer-readable program code stored on the storage device **304** and executed by the CPU **312** using any suitable, high or low level computing language, such as Python, Java, C, C++, C#, .NET, etc. The network interface **308** could include an Ethernet network interface device, a wireless network interface device, or any other suitable device which permits the server **302** to communicate via the network. The CPU **312** could include any suitable single- or multiple-core microprocessor of any suitable architecture that is capable of implementing and running the program **306** (e.g., Intel processor). The random access memory **314** could include any suitable, high-speed, random access memory typical of most modern computers, such as dynamic RAM (DRAM), etc.

FIG. 13 is a diagram showing processing steps **400** for generating a point cloud from a pair of aerial images, performed by an aerial image engine, and more specifically the object detection pre-processing engine. It is noted that point cloud data could be acquired during data capture and/or during image processing.

At step **402**, a pair of aerial images (e.g., image pair) is selected/identified (automatically by the engine or manually by the user). The pair of aerial images could be electronically received from a computer system, electronically transmitted from a database, etc. The engine could utilize a number of constraints in selecting the pair of aerial images. For example, the engine could require the pair of aerial images to have the same basic orientation (e.g., both are vertical images, both are oblique west images, etc.), the engine could require that the images have a large overlap area, and/or the engine could require that there is a small difference in capture time between both images (e.g., to avoid the effect of illumination changes).

At step **404**, the engine projects an overlapping area on ground plane data (using the pair of aerial images of step **402**). More specifically, the engine calculates the overlapping area of the images, and projects the relevant area from both images onto a horizontal ground plane of an approximate height (e.g., where the height could be extracted from the image parameters). This corrects small scale and tilt differences between the two images.

At step **406**, the engine aligns the images to the direction of flight of an aircraft or other flying vehicle from which aerial images are being taken (e.g., the line joining the observation points of both images) to produce an aligned stereoscopic pair of aerial images at step **408**. This allows the engine to find horizontal correspondences between the two
5 images. The engine rotates the images to align them to the flight direction to assimilate them to a canonical stereoscopic pair. The engine could also apply template matching to finely adjust the overlapping images.

At step **410**, the engine computes dense correspondence mapping (e.g., disparity map, using the aligned stereoscopic pair of images of step **408**). More specifically, the
10 engine applies a multi-scale disparity map module to the stereoscopic pair of images. This provides a measurement of the discrepancy distance between corresponding features on both images. The engine assigns a disparity value to each pixel on at least one of the stereoscopic images (e.g., the left image).

Using a depth from disparity method, the engine calculates depth from disparity at
15 step **412**, which then generates a point cloud at step **414** (and electronically transmits point cloud data to another computer system). More specifically, the engine calculates a height map by applying an algorithm to compute depth (e.g., distance from an observation point) as a linear function of the disparity value at each pixel (e.g., the focal distance of the camera and the distance between observation points). To generate the point cloud, the
20 engine transforms coordinates of the height map from the aligned stereoscopic pair back to the original image pair.

Alternatively (or additionally), the engine computes point heights using an eye-ray method at step **416** (based on the pair of images of step **402**), and the produces the point cloud **414**. More specifically, the engine transforms disparity values of the disparity map
25 from the aligned stereoscopic pair back to the original image pair. Then the engine applies the eye-ray method, which triangulates each point using the vision rays from both observation points. This provides a more precise height map than the depth from disparity method.

FIGS. 14-15 are views of a disparity map and point cloud generated from the depth
30 from disparity method of **FIG. 13**. More specifically, **FIG. 14** is a disparity map **418** generated by the engine using the depth from disparity method. **FIG. 15** is a point cloud **420** generated by the engine using the depth from disparity method.

FIGS. 16-17 are views of a disparity map and point cloud generated from the depth from disparity method of **FIG. 13**. More specifically, **FIG. 16** is a disparity map **422** generated by the engine using the eye-ray method. **FIG. 17** is a point cloud **424** generated by the engine using the eye-ray method.

5 **FIG. 18** is a diagram showing processing steps **500** for generating a composite point cloud from multiple image pairs, such as those used in **FIGS. 13-17**. At step **502**, multiple pairs of aerial images are selected/identified (automatically by the engine or manually by the user). The multiple pairs of aerial images could be electronically received from a computer system, electronically transmitted from a database, etc. The engine could
10 utilize a number of constraints in selecting the pairs of aerial images (as similarly described above). For example, the engine could require that each pair of aerial images have the same basic orientation (e.g., both are vertical images, both are oblique west images, etc.), the engine could require that the images have a large overlap area, and/or the engine could require that there is a small difference in capture time between both images (e.g., to avoid
15 the effect of illumination changes).

Once multiple image pairs have been selected, the engine applies a Levenberg-Marquadt optimization module **504** to the multiple image pairs. More specifically, at step **506**, the module **504** generates point clouds for each image pair (using the process described in **FIG. 13**). There are overlapping zones between different point clouds,
20 although the overlap may not be a perfect fit. In step **508**, multiple point clouds are generated.

In step **510**, the engine calculates the error resulting from the multiple point clouds (e.g., discrepancy between overlapping zones). More specifically, the engine calculates 3D features for each point cloud. The engine evaluates the discrepancy between point clouds
25 as an error metric that uses distances between corresponding 3D features. The engine accumulates the error metric to include overlaps between all point clouds.

At step **512** the engine determines whether the error is low (e.g., a pre-defined threshold). If no, the process proceeds to step **514**, and the engine calculates an error gradient according to image parameters. More specifically, the engine adjusts the camera
30 parameters to each image covering a large area and containing many different buildings. The discrepancies between point clouds are expected to be produced by minor camera parameter errors (e.g., as the camera parameters may not be the best for each single

building on the image). The engine checks the change of error gradient against minor changes in camera parameters (e.g., using a Jacobain matrix and determinant).

Then, in step **516**, the engine modifies projection parameters toward a lower error value. More specifically, the engine makes small changes to the camera parameters so that
 5 the error is reduced in a new computation of the point clouds. The process then reverts back to step **506**, and new point clouds are generated. The process is repeated until the generated point clouds are calculated by the engine to have a low error. In this way, this process is an iterative gradient-descent optimization.

If, in step **512**, the engine makes a positive determination that the error is low
 10 (thereby concluding the Levenberg-Marquadt optimization), then the process proceeds to step **518** and the engine removes redundant points. More specifically, the engine removes redundant points by using the ones with higher confidence according to the orientation of each point cloud region. Then the engine generates a composite point cloud at step **520**. Redundant points are removed because a composite point cloud (including all points from
 15 each individual point cloud) contains a large amount of information, and discrepancies in overlapping areas (although low) may be seen as noise by other engines (e.g., modules, algorithms, etc.), such as by a plane detection module.

FIGS. 19-21 are views related to a point cloud based on a nadir pair of images. More specifically, **FIG. 19** is a nadir (vertical) view **522** of a point cloud based on a nadir
 20 pair of images. **FIG. 20** is an oblique view **524** of a point cloud based on a nadir pair of images. **FIG. 21** is a slice **526** of the point cloud that cross the front windows of the building shown based on a nadir pair of images.

FIGS. 22-24 are views related to a point cloud based on an oblique pair of images. More specifically, **FIG. 22** is a nadir (vertical) view **528** of a point cloud based on an
 25 oblique pair of images. **FIG. 23** is an oblique view **530** of a point cloud based on an oblique pair of images. **FIG. 24** is a slice **532** of the point cloud that cross the front windows of the building shown based on an oblique pair of images.

FIGS. 25-27 are views related to a point cloud based on a combination of nadir and oblique pairs of images. More specifically, **FIG. 25** is a nadir (vertical) view **534** of a
 30 point cloud based on a combination of nadir and oblique pairs of images. **FIG. 26** is an oblique view **536** of a point cloud based on a combination of nadir and oblique pairs of images. **FIG. 27** is a slice **538** of the point cloud that cross the front windows the building shown based on a combination of nadir and oblique pairs of images.

FIG. 28 is a diagram of modules and associated processing steps for automated building recognition (and/or recognition of any other object) carried out by an aerial image engine **600**, and more specifically the object detection pre-processing engine. The aerial image engine **600** could include one or more modules, such as a stereo processing module **602**, a segment-based induction module **604**, a contour detection module **605**, a roof model induction model **606**, an optimization module **608**, and an evaluation module **610**.

In step **612**, the stereo processing module **602** automatically selects/identifies/receives (or a user manually selects/identifies) a set of calibrated aerial images (e.g., as input). The calibrated aerial images could be received electronically from another computer system, a database, etc. In step **614**, the segment-based induction module **604** uses the set of calibrated aerial images to detect 2D line segments on each image. The segment-based induction module **604** matches lines and generates candidate 3D lines at step **616**, and detects and discards ground lines at step **618**. Then, the segment-based induction module **604** detects horizontal lines by finding parallel clusters at step **620**, and could concurrently, detect oblique lines by finding clusters of line intersections at step **622**. In step **624**, the segment-based induction module **604** induces a set of roof model primitives, which are subsequently used at step **678** by the optimization module **608**, discussed in more detail below.

Returning to step **612**, once the set of calibrated aerial images are selected/identified, the process (concurrently) proceeds to step **626**, where the stereo processing module **602** selects image pairs in any orientation, and then the image pairs are rectified in step **628**. The stereo processing module **602** computes a multiscale disparity map at step **630**, then computes and merges pairwise point clouds at step **632**, and then generates a global point cloud at step **634**. The global point cloud generated is used at step **656** by the roof model induction module **606**, discussed in more detail below.

Returning to step **612**, once the set of calibrated aerial images are selected/identified, the process (concurrently) proceeds such that the stereo processing module **602** selects a pair of nadir images in step **636**, and then generates a stereo pair of images in step **638**. The stereo processing module **602** rectifies the stereo pair of images at step **640**, and then (concurrently) projects and aligns the stereo images at step **642**. The stereo processing module **602** then computes a multiscale disparity map at step **644**, and computes and filters a point cloud at step **646**.

The process then proceeds to the contour detection module **605**. The contour detection module **605** includes one or more algorithms to detect contours. More specifically, the contour detection module **605** could include a grabcut approach algorithm **648**, an MSER (maximally stable extremal regions) approach algorithm **650**, and/or a point cloud approach algorithm **652**. The grabcut approach **648** and the MSER approach **650** each receive the selected pair of nadir images of step **636** and the computed multiscale disparity map of step **644** as inputs. The point cloud approach **652** receives the selected pair of nadir images of step **636** and the computed and filtered point cloud of step **646** as inputs. Each of the approaches then generates an output to be used by the roof model induction module **606**.

Processing proceeds to the roof model induction module **606** which builds contours at step **654** (based on the output of the contour detection module **605**), and detects planes at step **656** (based on the global point cloud generated at step **634**). Then the roof model induction module **606** finds intersecting lines at step **660** and generates an intersecting line adjacency graph at step **662**. Concurrently with steps **660**, **662**, the roof model induction module **606** generates a plane image adjacency graph at step **658**. The roof model induction module **606** then generates a set of roof model primitives at step **664**.

The process then proceeds to the optimization module **608**, which extracts segments from images in all views at step **666** (based on the set of calibrated aerial images of step **612** and based on the set of roof model primitives of step **664**). The optimization module **608** then applies a distance transform per image at step **668** and (concurrently) applies a distance to the nearest segment at step **670**. The results/outputs of steps **668** and **6670** are then used as inputs in one or more optimization algorithms of the optimization module **608**. More specifically, the optimization algorithms could include a Lavenberg-Marquadt optimization algorithm **672**, a differential evolution optimization algorithm **674**, and/or a variable neighborhood search optimization algorithm **676**. Then at step **678** a set of adjusted primitives is generated by the optimization module **608** (based on the set of roof model primitives induced at step **624** and based on the output of the one or more optimization algorithms **672**, **674**, **676**).

The optimization module **608** then calculates overlapping with 2D lines at **680** (using the set of adjusted primitives **678**), and then applies one or more high overlapping transformation options at step **682**. Additionally, the optimization module **608** generates a model containing a roof and extensions at step **684**. The optimization module **608** applies

VNS (variable neighborhood search) optimization at step **686** and generates an adjusted model at step **688**. The adjusted model and VNS optimization are then outputted to the evaluation module **610**.

5 The process then proceeds to the evaluation module **610**, which measures error by comparing roof segments at step **690** (based on the adjusted model of step **688**, and based on a collection of blueprints with validated sketches of step **692**). The evaluation module **610** then generates an error metric at step **694**. Additionally, the evaluation module **610** generates confidence estimation at step **696** (based on the VNS optimization of steps **676** and **686**). The evaluation module **610** then generates a confidence metric at step **698**.

10 Having thus described the system and method in detail, it is to be understood that the foregoing description is not intended to limit the spirit or scope thereof. It will be understood that the embodiments of the present disclosure described herein are merely exemplary and that a person skilled in the art may make any variations and modification without departing from the spirit and scope of the disclosure. All such variations and
15 modifications, including those discussed above, are intended to be included within the scope of the disclosure.

CLAIMS

1. A system for aerial image detection and classification, comprising:
an aerial image database storing one or more aerial images electronically received from one or more image providers; and
5 a computer system in communication with the aerial image database, the computer system including an object detection pre-processing engine in electronic communication with the aerial image database, the object detection pre-processing engine detecting and classifying objects using:
a disparity mapping generation module for automatically processing the one
10 or more aerial images to generate a disparity map providing elevation information;
a segmentation module for automatically applying a pre-defined elevation threshold to the disparity map, the pre-defined elevation threshold adjustable by a user; and
a classification module for automatically detecting and classifying objects in the one or more stereoscopic pairs of aerial images by applying one or more automated
15 detectors based on classification parameters and the pre-defined elevation threshold.
2. The system of Claim 1, wherein the object detection pre-processing engine automatically classifies as an object as a building if the object is above the pre-defined elevation threshold.
3. The system of Claim 1, wherein the classification parameters include at least one of
20 aspect parameters, disparity parameters, color parameters, or texture parameters.
4. The system of Claim 1, further comprising a mass production engine in electronic communication with the object detection pre-processing engine, the mass production engine automatically producing drawings, sketches, and models using the elevation information generated by the object detection pre-processing engine.
- 25 5. The system of Claim 1, further comprising a project and task management system in electronic communication with the object detection pre-processing engine, the project and task management system guiding a user in creating and publishing one or more projects using the elevation information generated by the object detection pre-processing engine.
- 30 6. The system of Claim 1, wherein the one or more automated detects includes at least one of a region growing algorithm or a split-and-merge segmentation algorithm.

7. The system of Claim 1, wherein the disparity mapping generation module of the object detection pre-processing engine generates a point cloud.

8. The system of Claim 7, wherein the object detection pre-processing engine generates the point cloud by computing point heights using an eye-ray method.

5 9. A method for aerial image detection and classification, comprising:

electronically receiving at a computer system including an object detection pre-processing engine one or more aerial images from an aerial image database, the one or more aerial images electronically received at the aerial image database from one or more image providers;

10 automatically processing, by the object detection pre-processing engine, the one or more aerial images to generate a disparity map providing elevation information;

automatically applying, by the object detection pre-processing engine, a pre-defined elevation threshold to the disparity map, the pre-defined elevation threshold adjustable by a user; and

15 automatically detecting and classifying, by the object detection pre-processing engine, objects in the one or more stereoscopic pairs of aerial images by applying one or more automated detectors based on classification parameters and the pre-defined elevation threshold.

10. The method of Claim 9, wherein the object detection pre-processing engine
20 automatically classifies an object as a building if the object is above the pre-defined elevation threshold.

11. The method of Claim 9, wherein the classification parameters include at least one of aspect parameters, disparity parameters, color parameters, or texture parameters.

12. The method of Claim 9, further comprising automatically producing, by a mass
25 production engine, drawings, sketches, and models by using the elevation information generated by the object detection pre-processing engine, the mass production engine in electronic communication with the object detection pre-processing engine.

13. The method of Claim 9, further comprising guiding, by a project and task
30 management system, a user in creating and publishing one or more projects using the elevation information generated by the object detection pre-processing engine, the project and task management system in electronic communication with the object detection pre-processing engine.

14. The method of Claim 9, wherein the engine executes at least one of a region growing algorithm or a split-and-merge segmentation algorithm.

15. The method of Claim 9, further comprising generating a point cloud by the object detection pre-processing engine.

5 16. The method of Claim 15, wherein the object detection pre-processing engine generates the point cloud by computing point heights using an eye-ray method.

17. A non-transitory computer-readable medium having computer-readable instructions stored thereon which, when executed by a computer system having an object detection pre-processing engine, cause the computer system to perform the steps of:

10 electronically receiving one or more aerial images from an aerial image database, the one or more aerial images electronically received at the aerial image database from one or more image providers;

automatically processing, by the object detection pre-processing engine, the one or more aerial images to generate a disparity map providing elevation information;

15 automatically applying, by the object detection pre-processing engine, a pre-defined elevation threshold to the disparity map, the pre-defined elevation threshold adjustable by a user; and

automatically detecting and classifying, by the object detection pre-processing engine, objects in the one or more stereoscopic pairs of aerial images by applying one or more automated detectors based on classification parameters and the pre-defined elevation threshold.

20

18. The non-transitory computer-readable medium of Claim 17, wherein the object detection pre-processing engine automatically classifies an object as a building if the object is above the pre-defined elevation threshold.

25 19. The non-transitory computer-readable medium of Claim 17, wherein the classification parameters include at least one of aspect parameters, disparity parameters, color parameters, or texture parameters.

20. The non-transitory computer-readable medium of Claim 17, further comprising automatically producing, by a mass production engine, drawings, sketches, and models by using the elevation information generated by the object detection pre-processing engine, the mass production engine in electronic communication with the object detection pre-processing engine.

30

21. The non-transitory computer-readable medium of Claim 17, further comprising guiding, by a project and task management system, a user in creating and publishing one or more projects using the elevation information generated by the object detection pre-processing engine, the project and task management system in electronic communication
5 with the object detection pre-processing engine.
22. The non-transitory computer-readable medium of Claim 17, wherein the engine executes at least one of a region growing algorithm or a split-and-merge segmentation algorithm.
23. The non-transitory computer-readable medium of Claim 17, further comprising
10 generating a point cloud by the object detection pre-processing engine.
24. The non-transitory computer-readable medium of Claim 23, wherein the object detection pre-processing engine generates the point cloud by computing point heights using an eye-ray method.

1/28

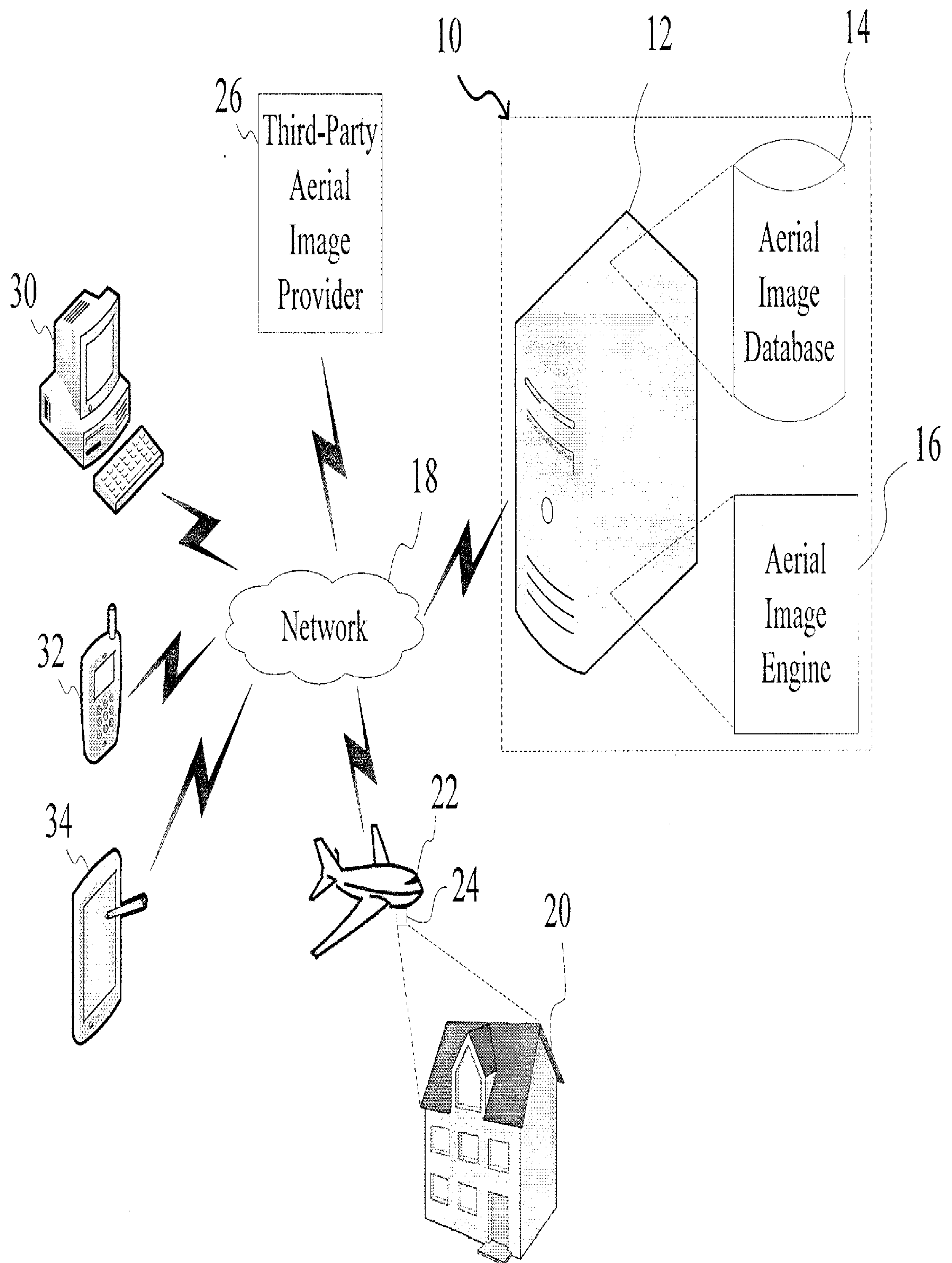


FIG. 1

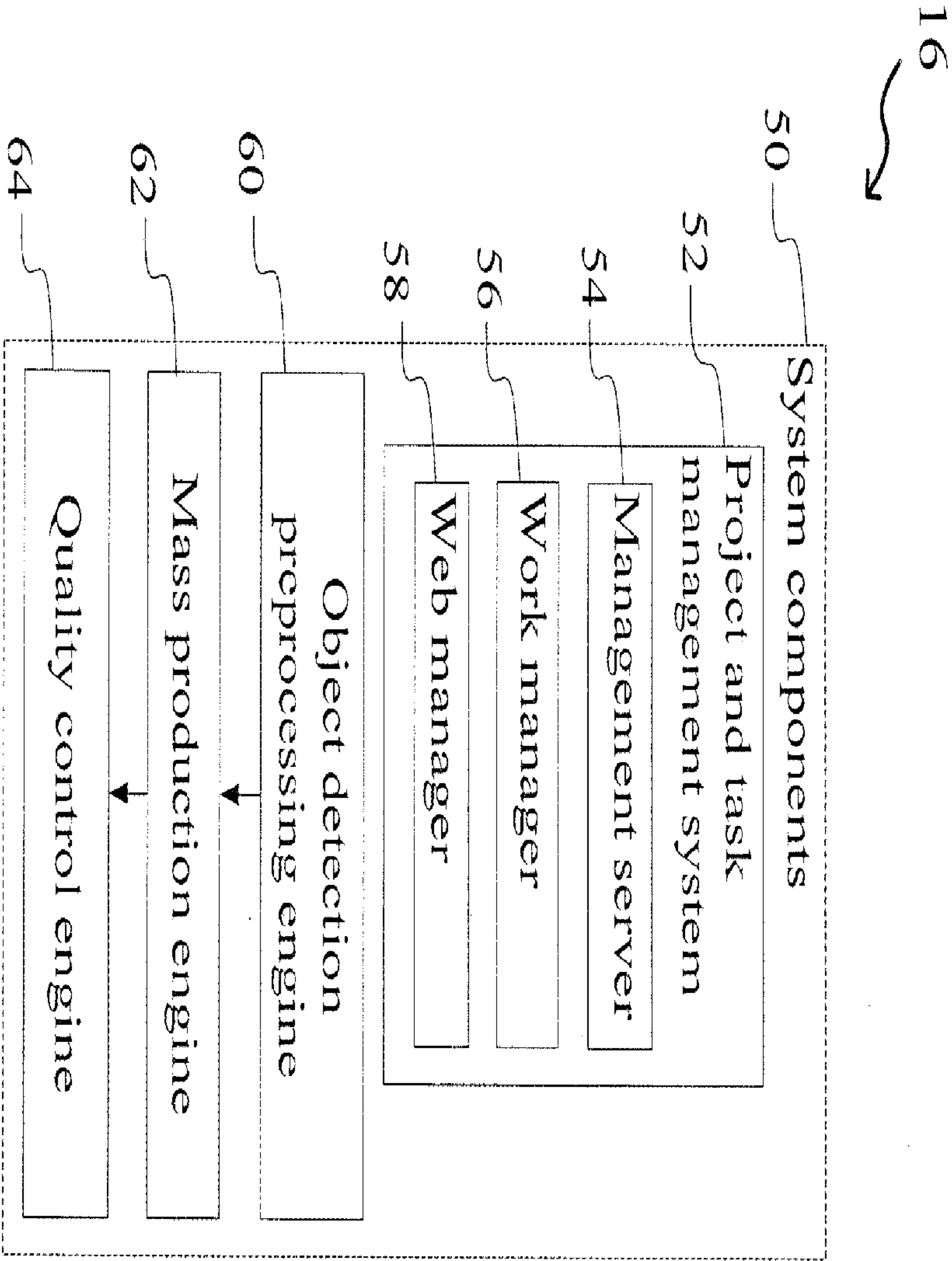


FIG. 2

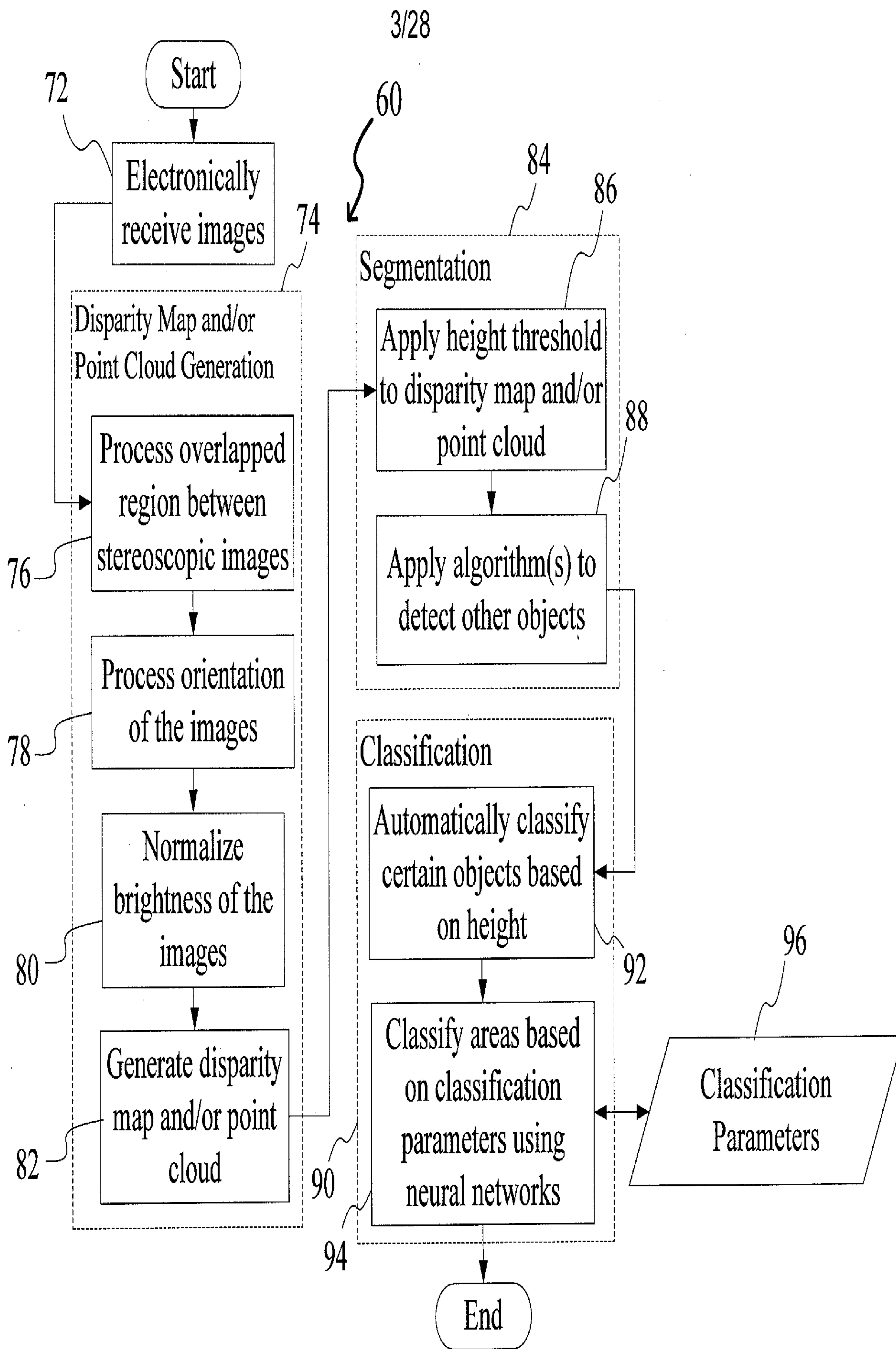


FIG. 3

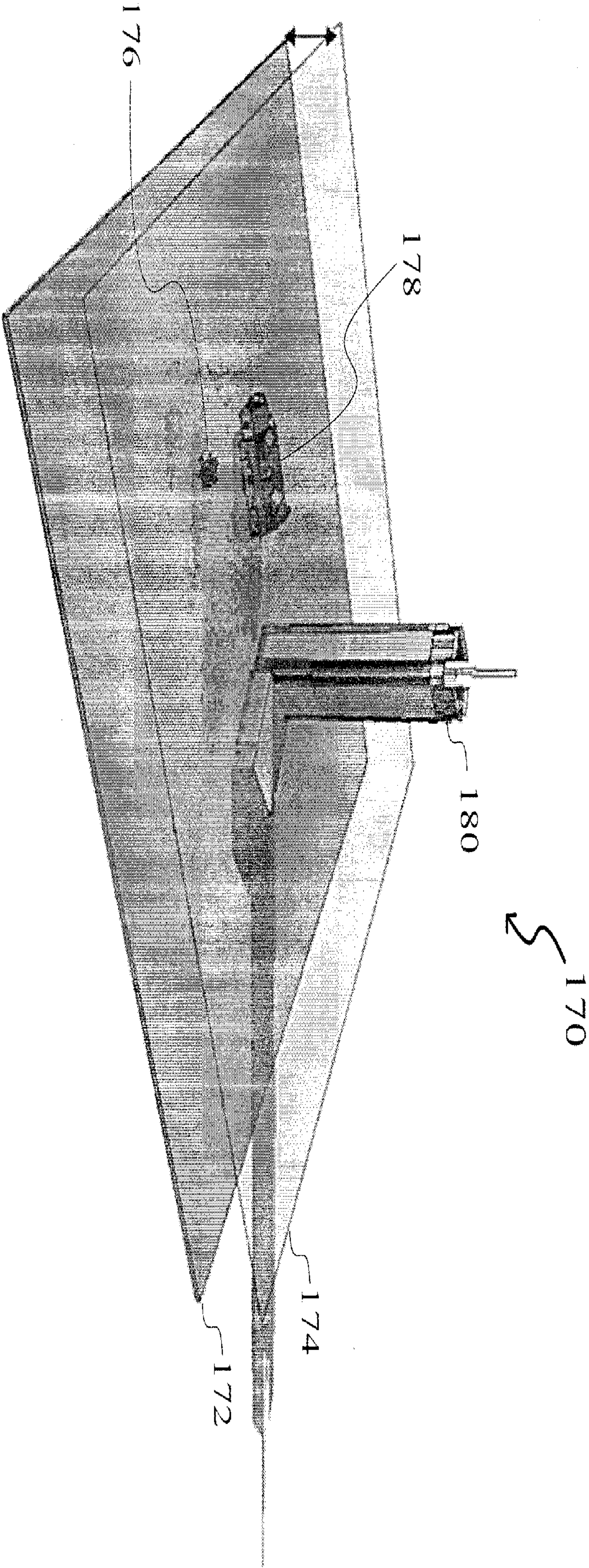


FIG. 4

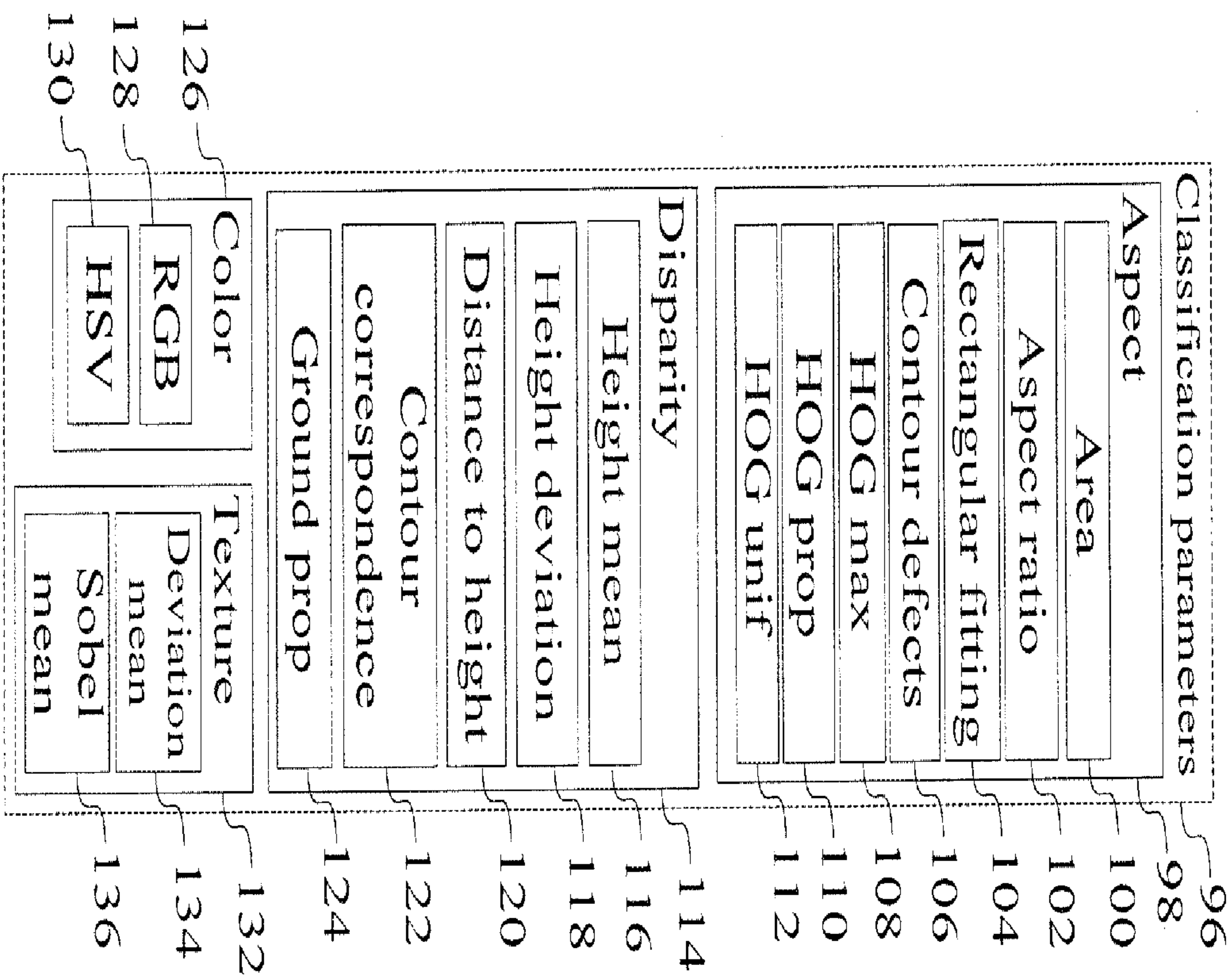


FIG. 5

6/28

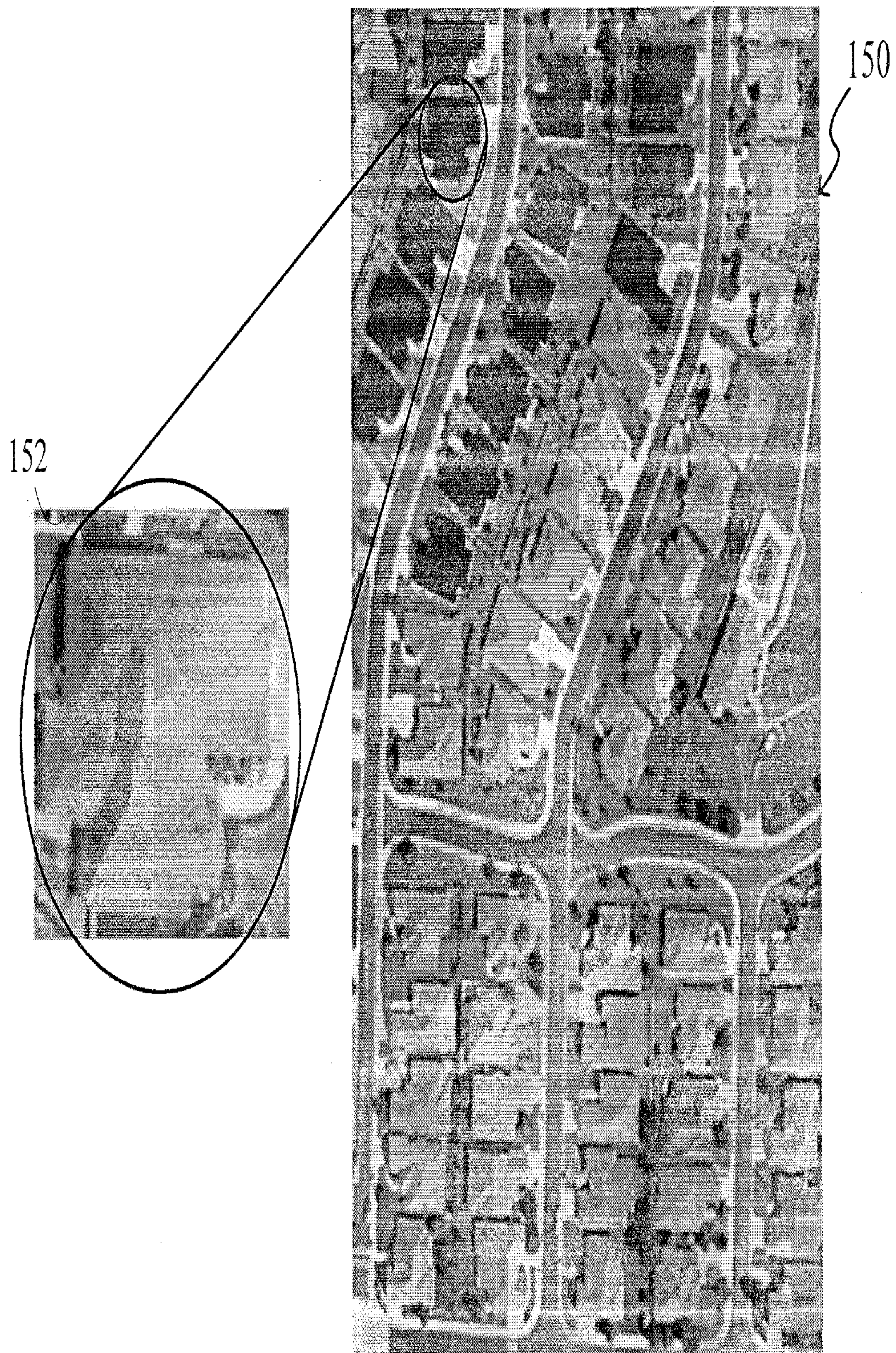
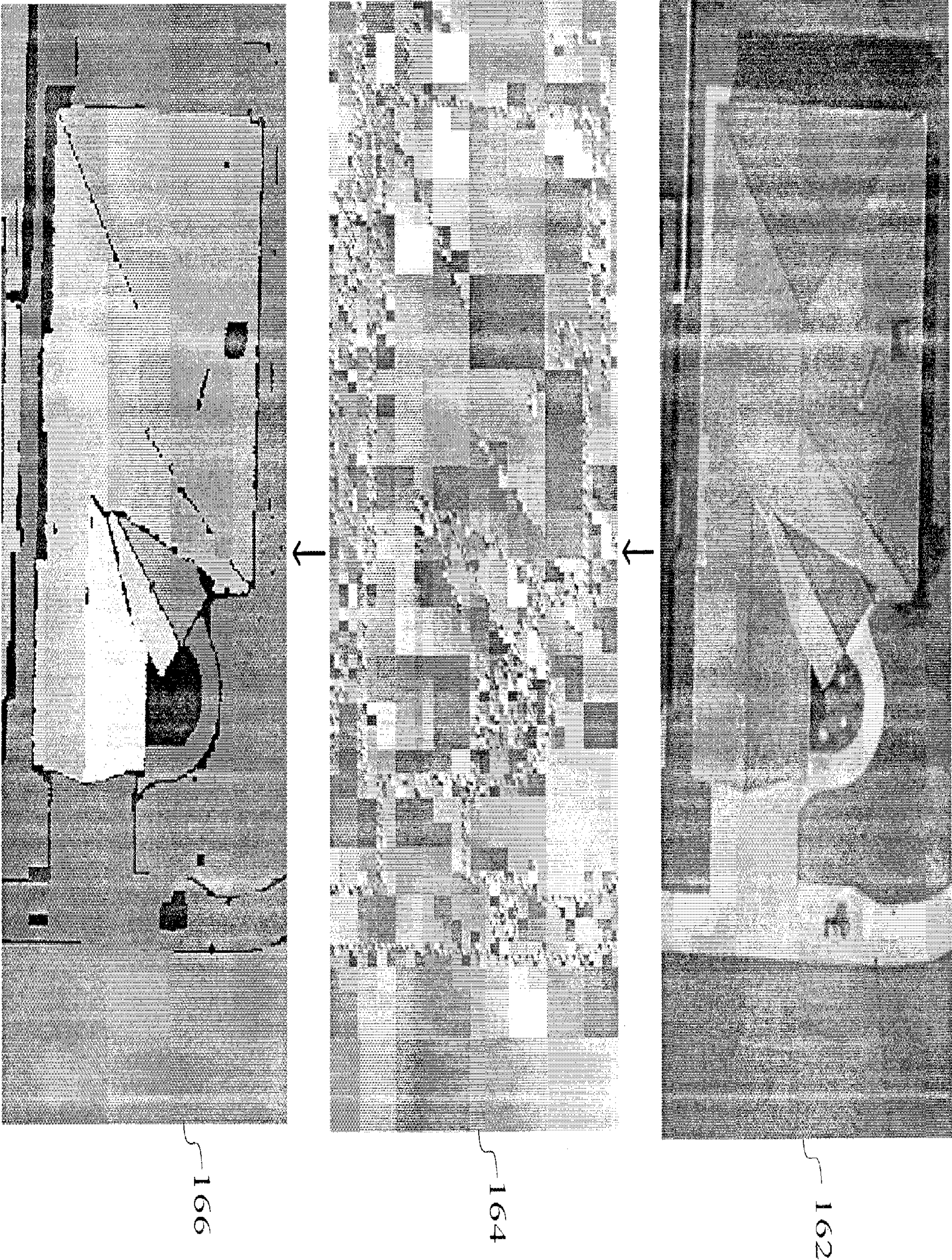


FIG. 6



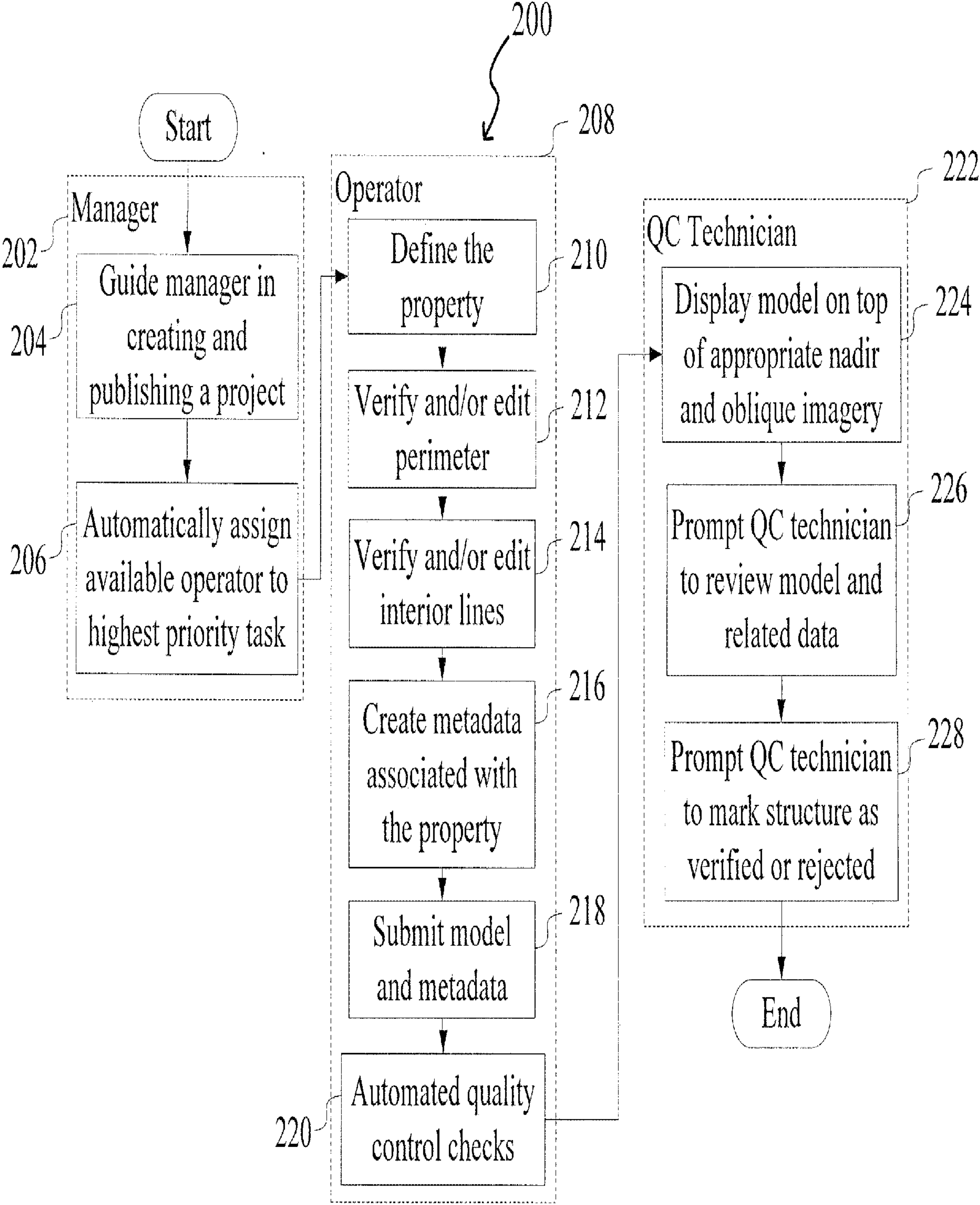


FIG. 8

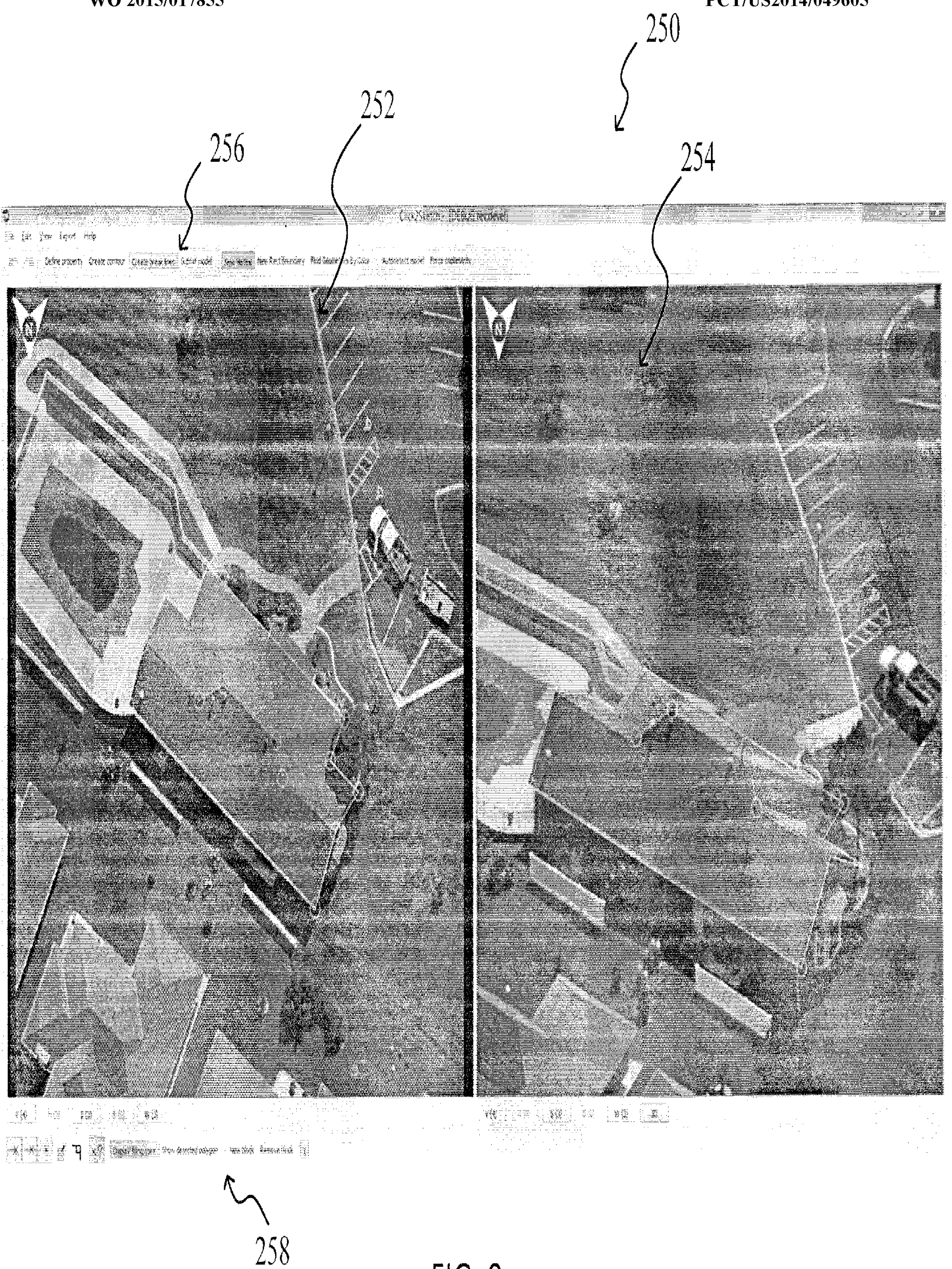


FIG. 9

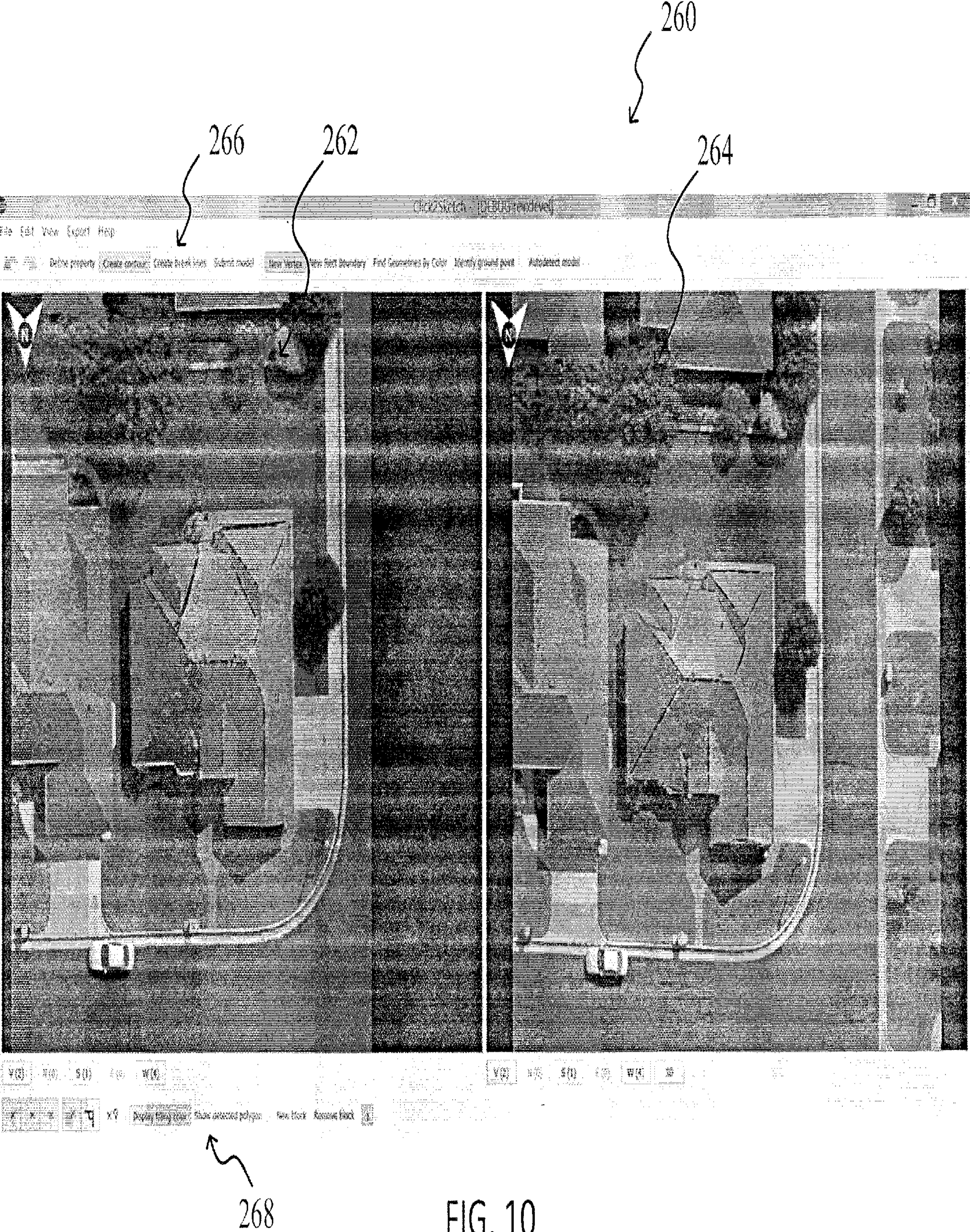


FIG. 10

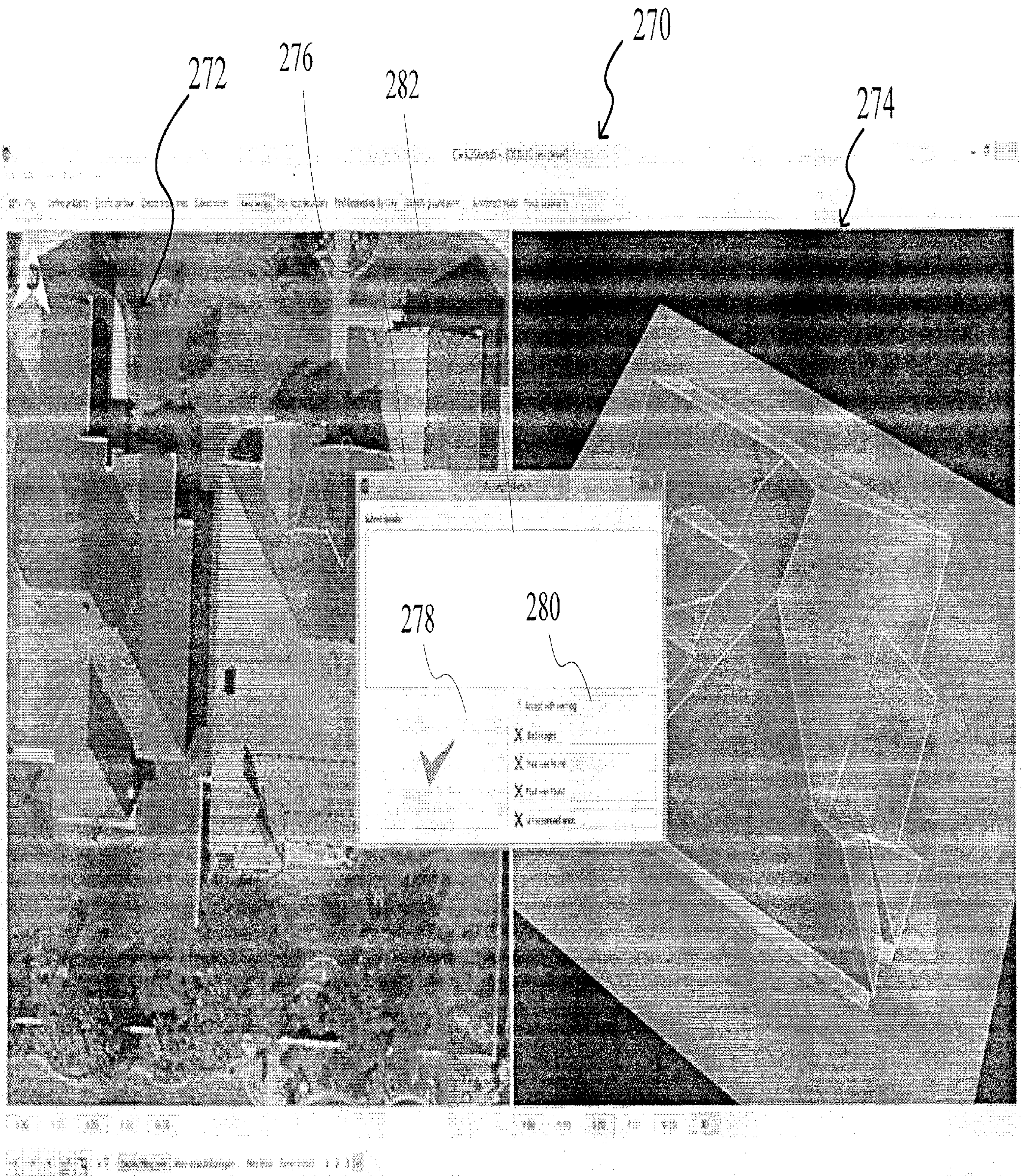


FIG. 11

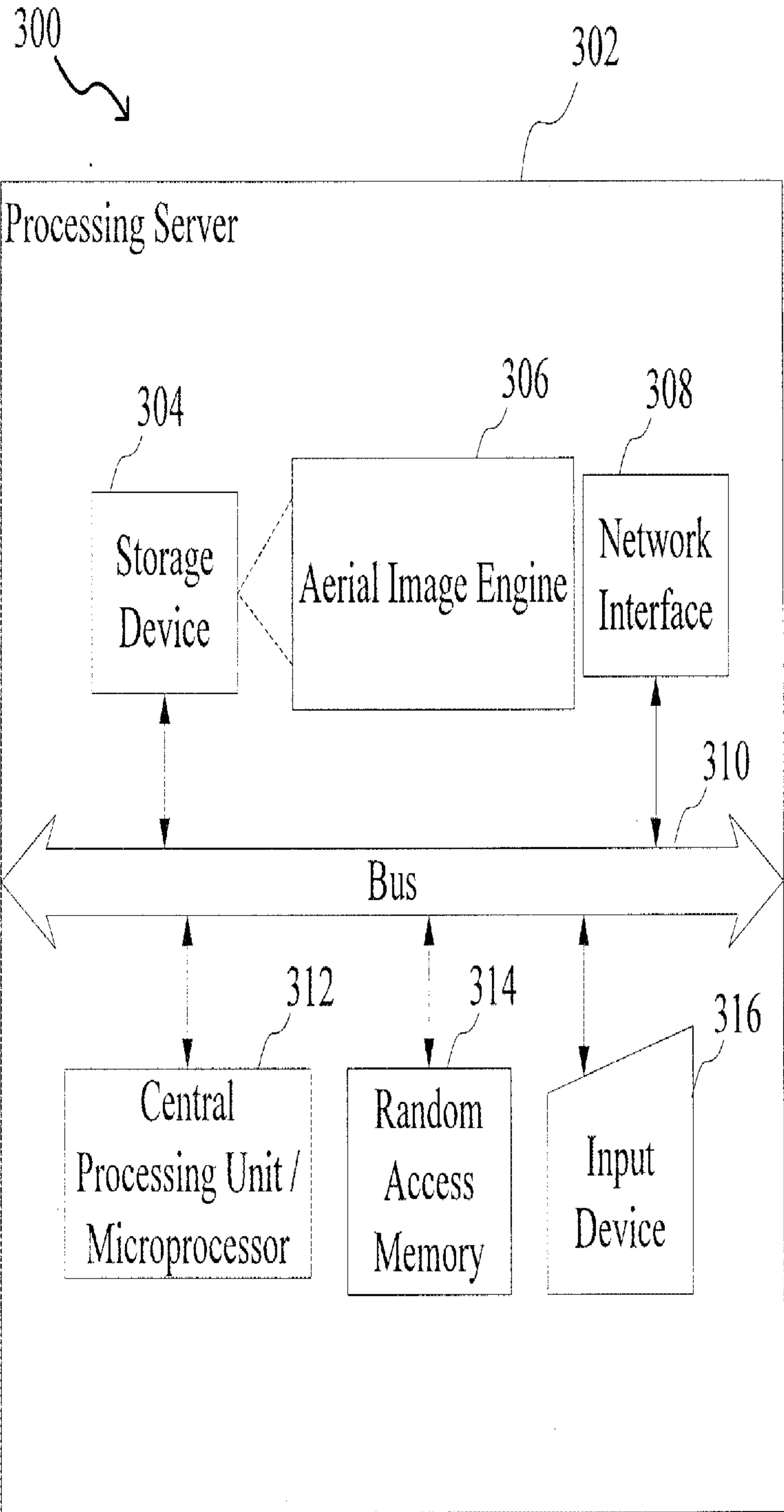


FIG. 12

13/28

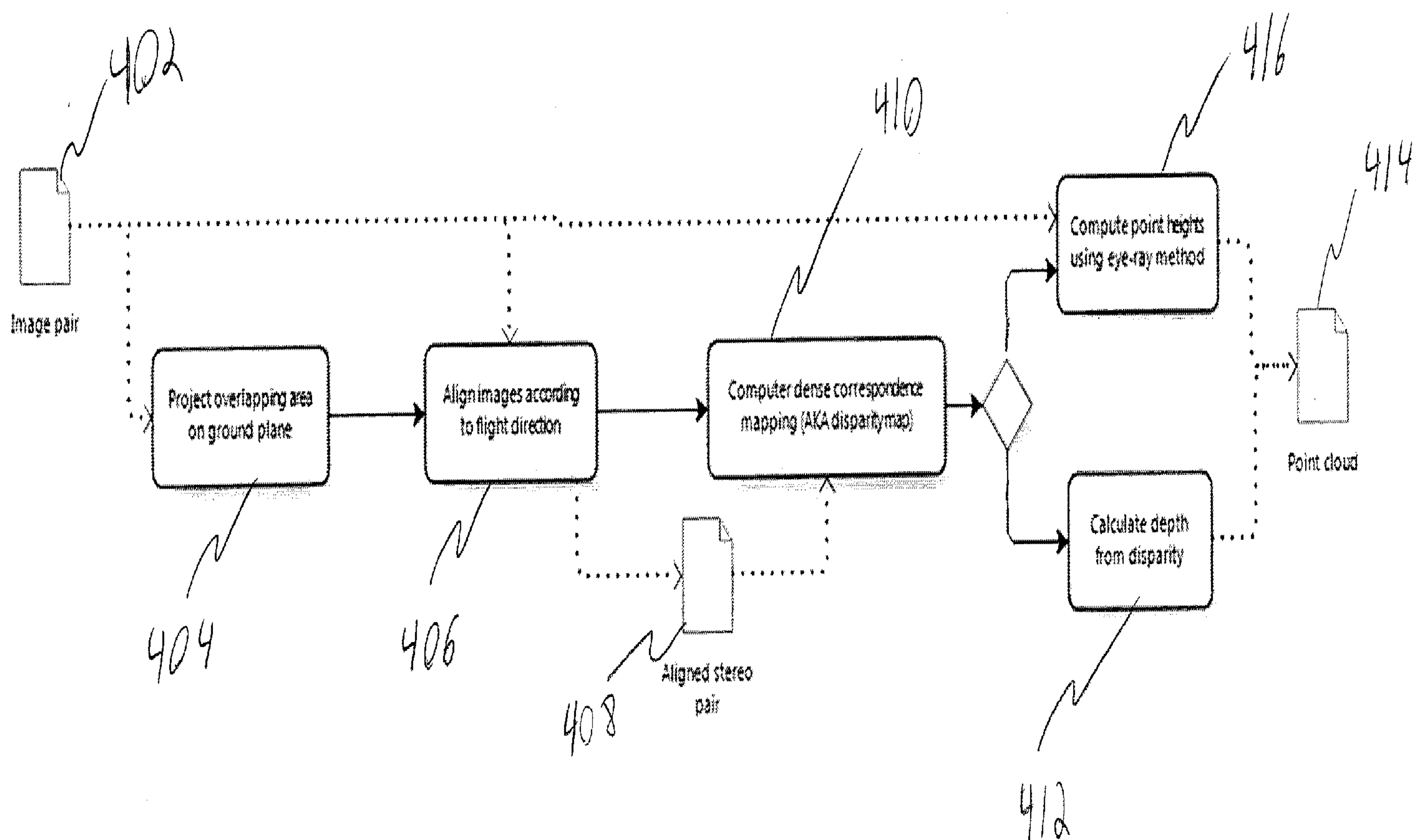


FIG. 13

14/28



418

FIG. 14

15/28

420

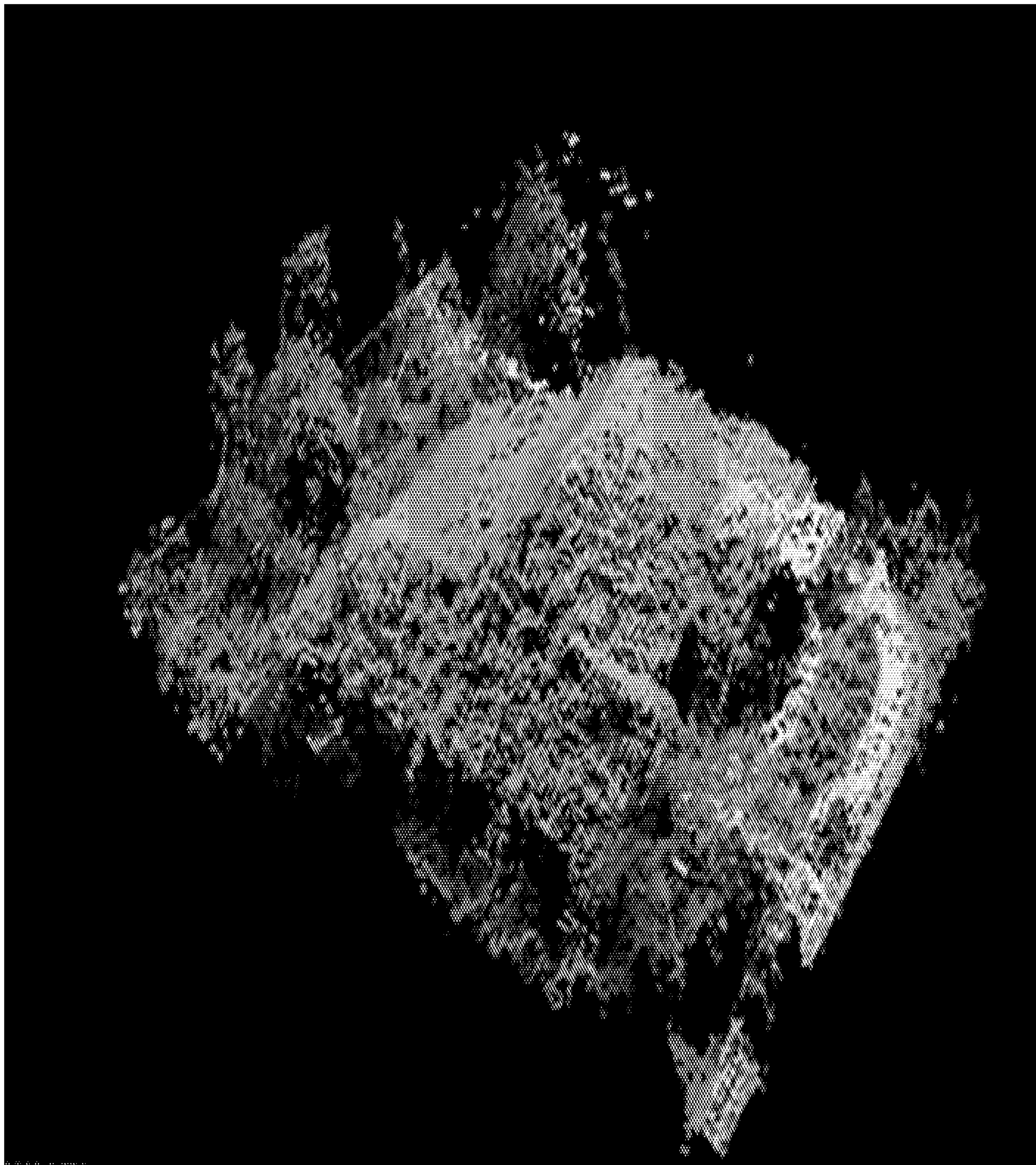


FIG. 15

16/28

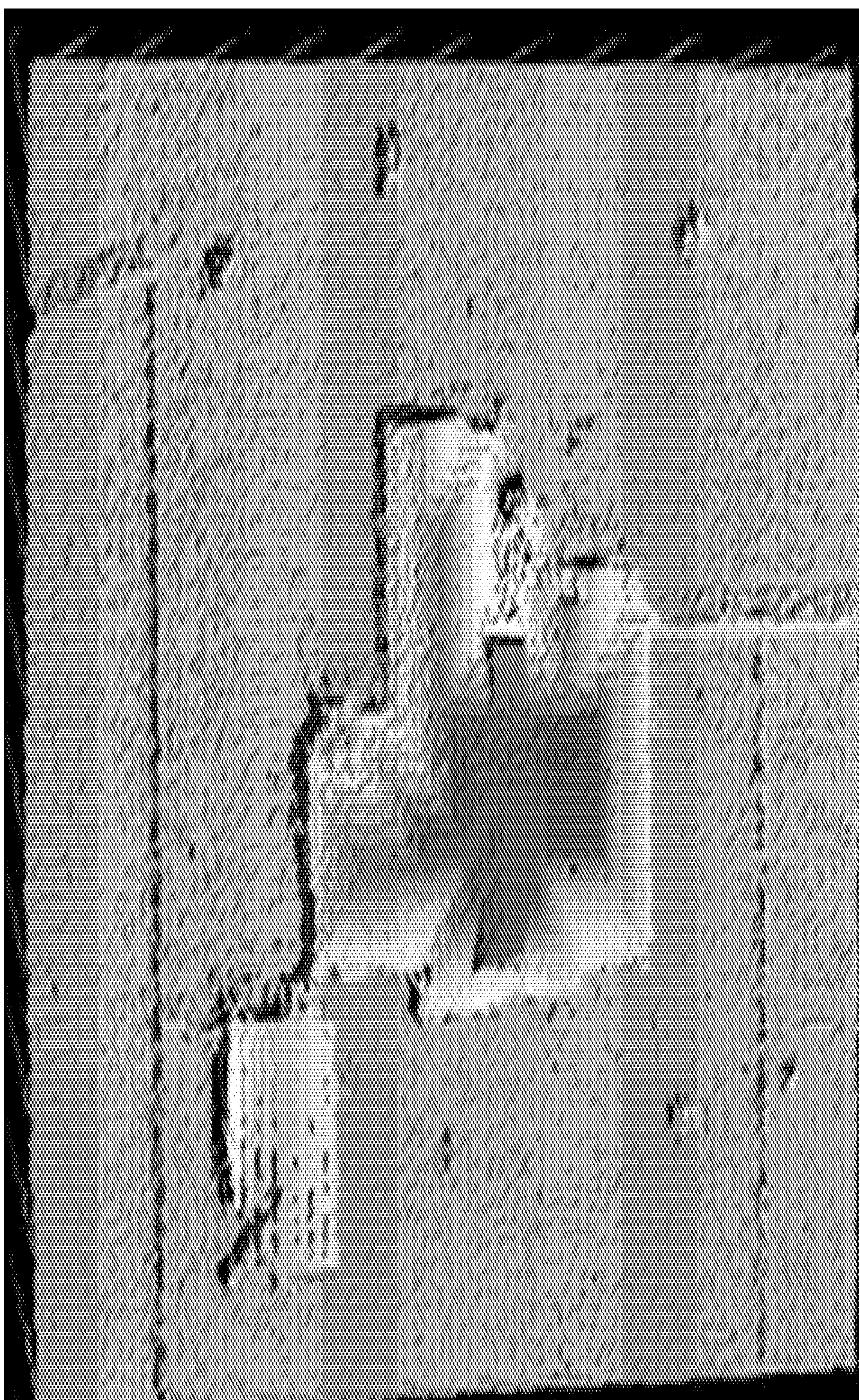


FIG. 16

17/28

424

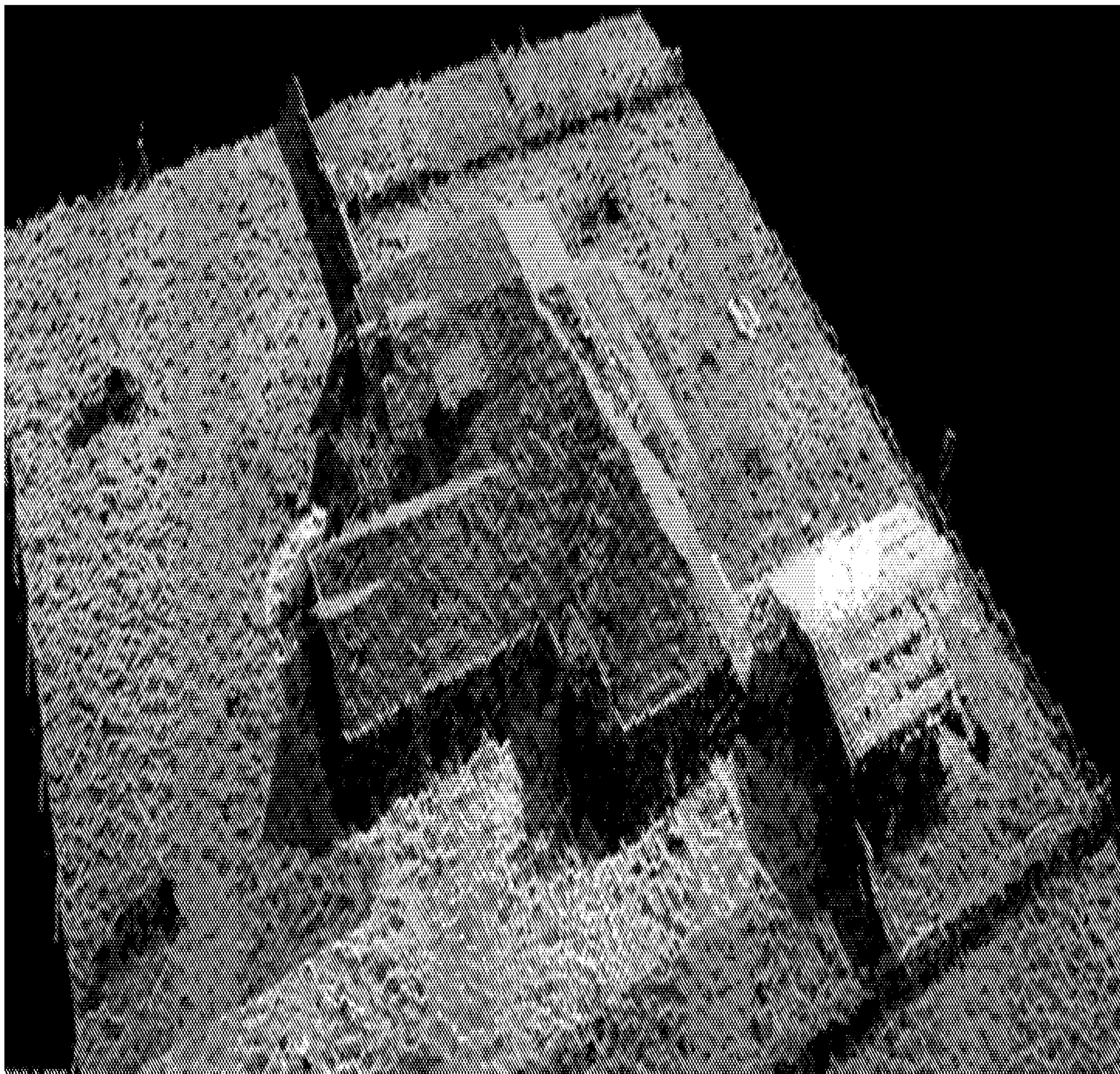
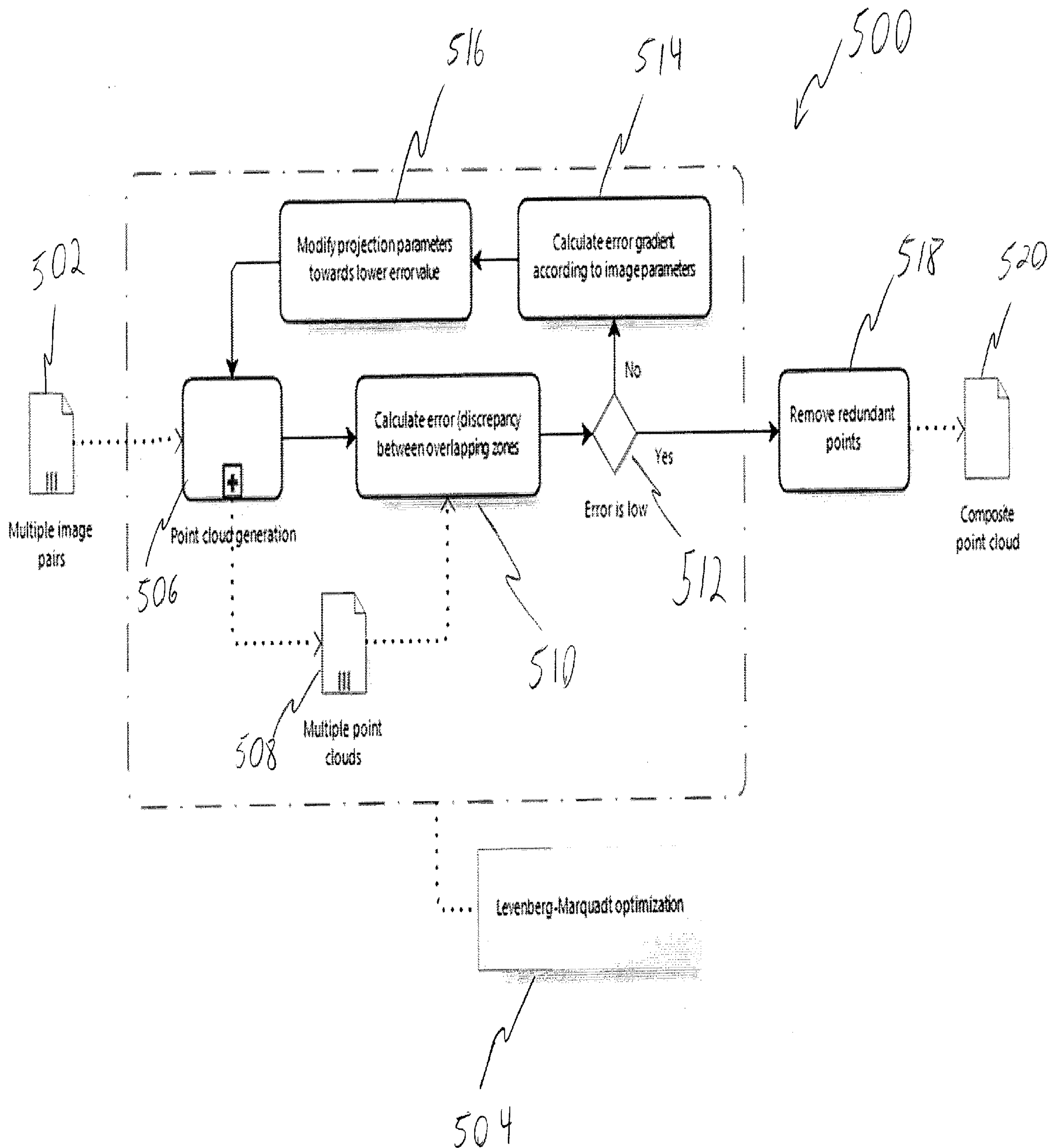


FIG. 17

18/28



19/28

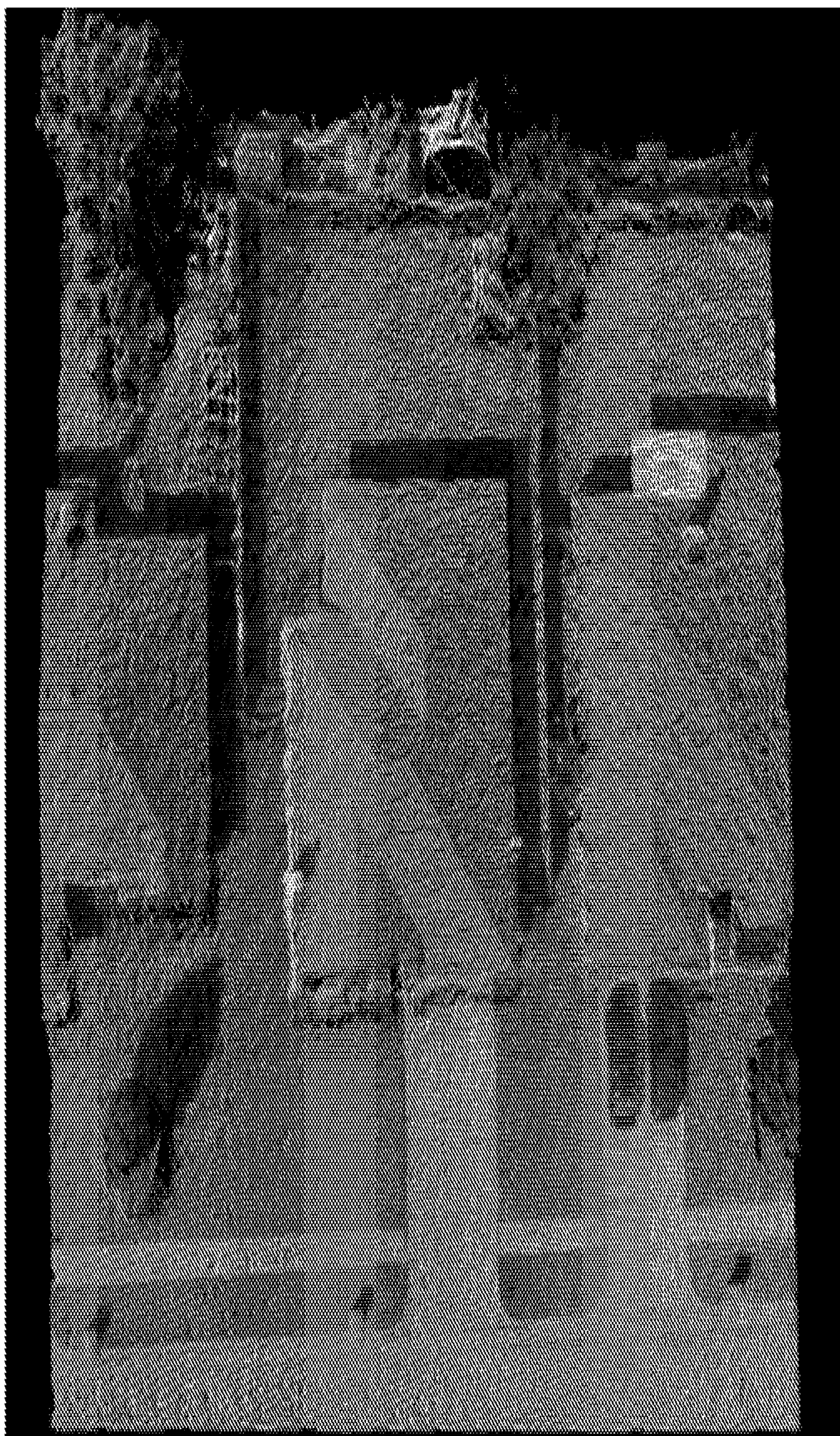


FIG. 19

20/28

524

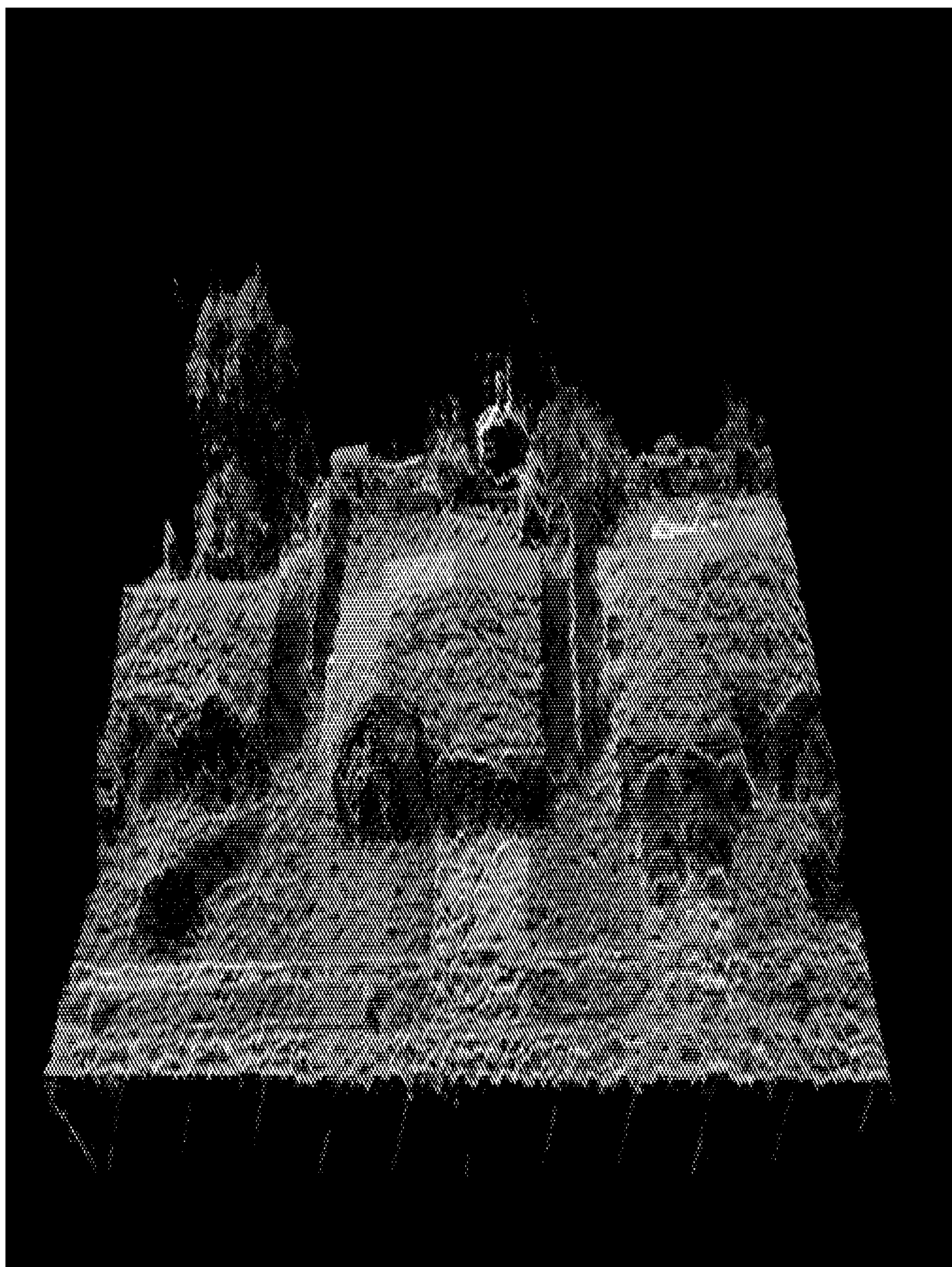


FIG. 20

21/28

526

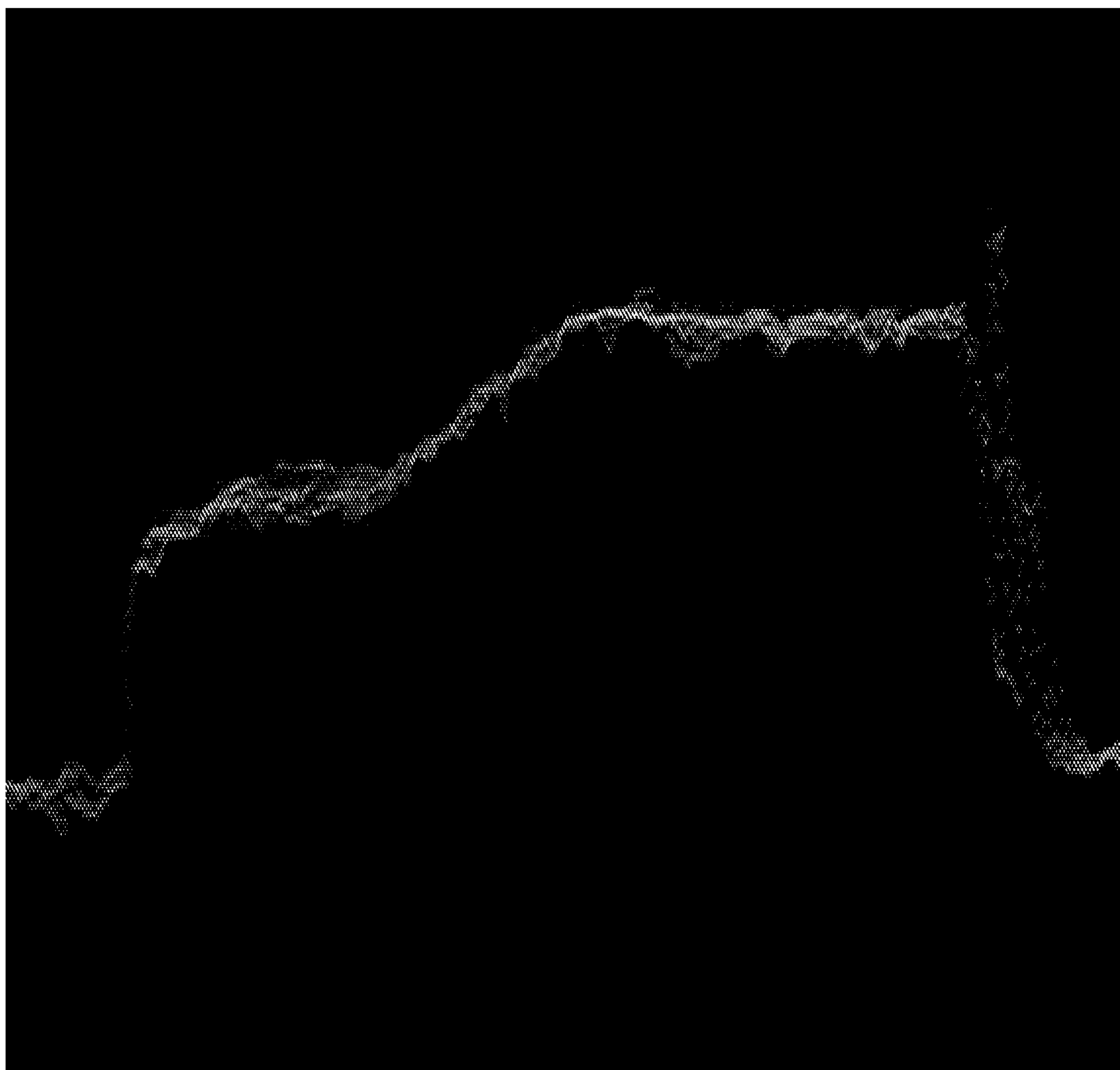
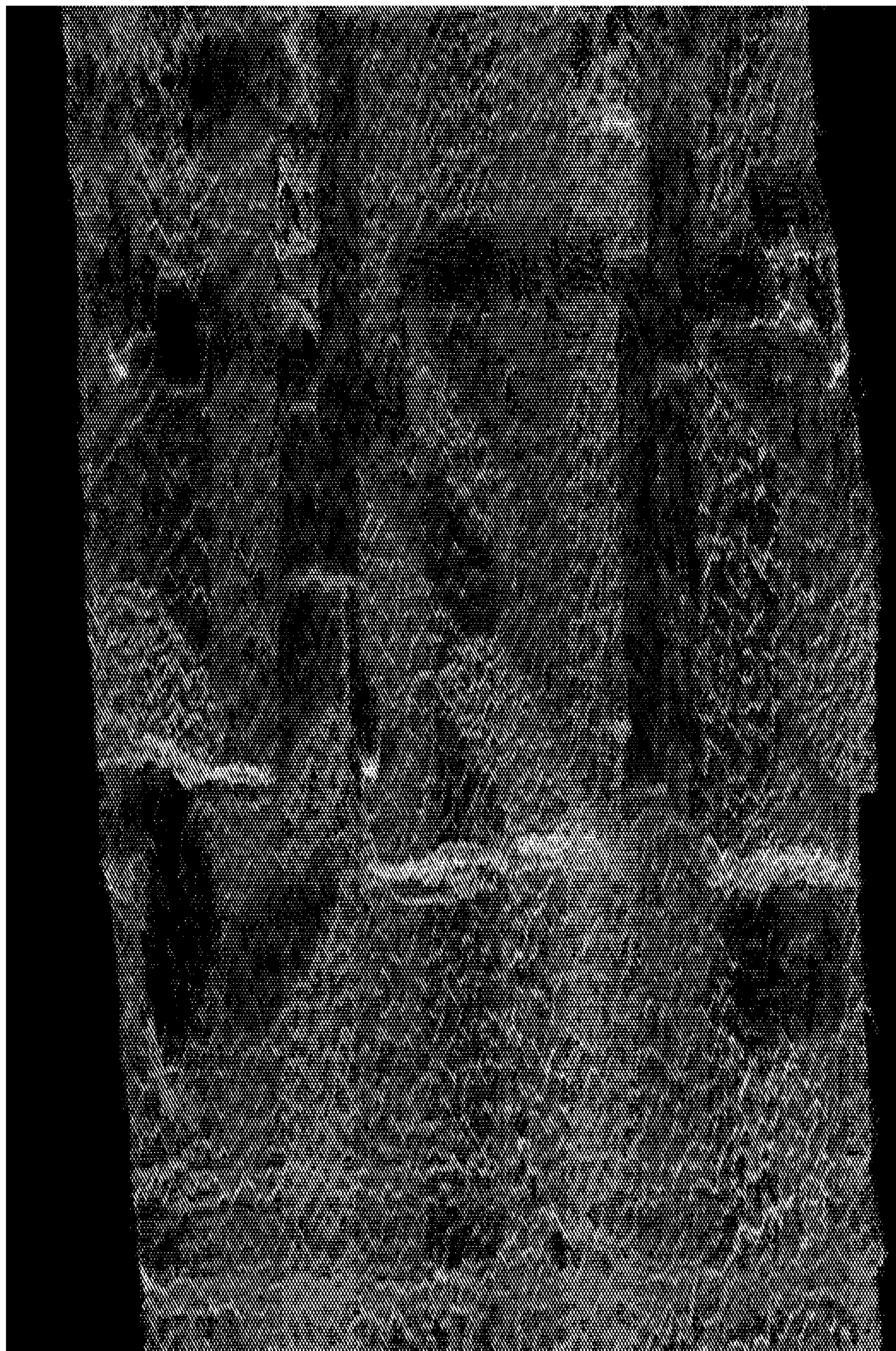


FIG. 21



528

FIG. 22

23/28

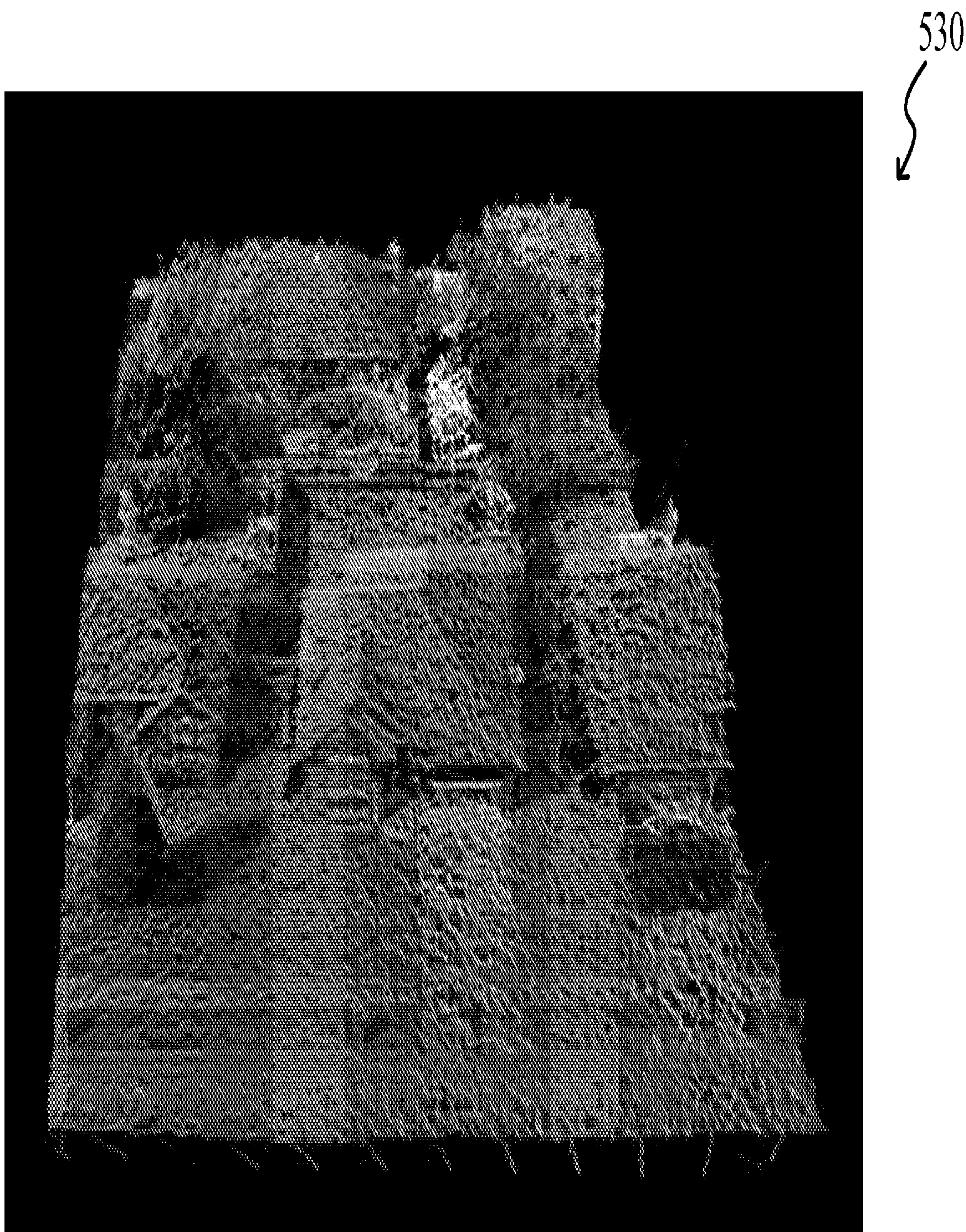


FIG. 23

24/28

532
↓

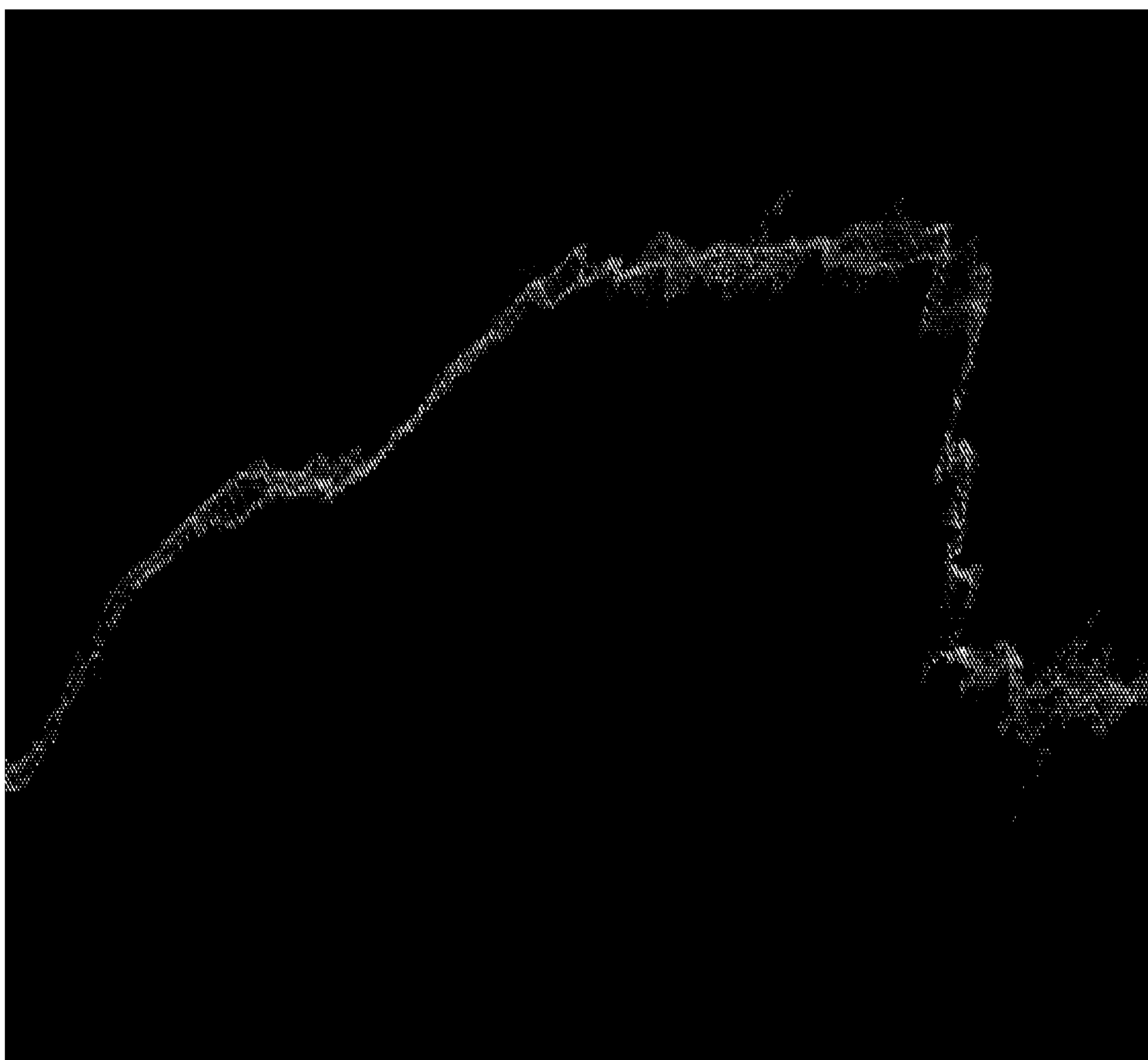
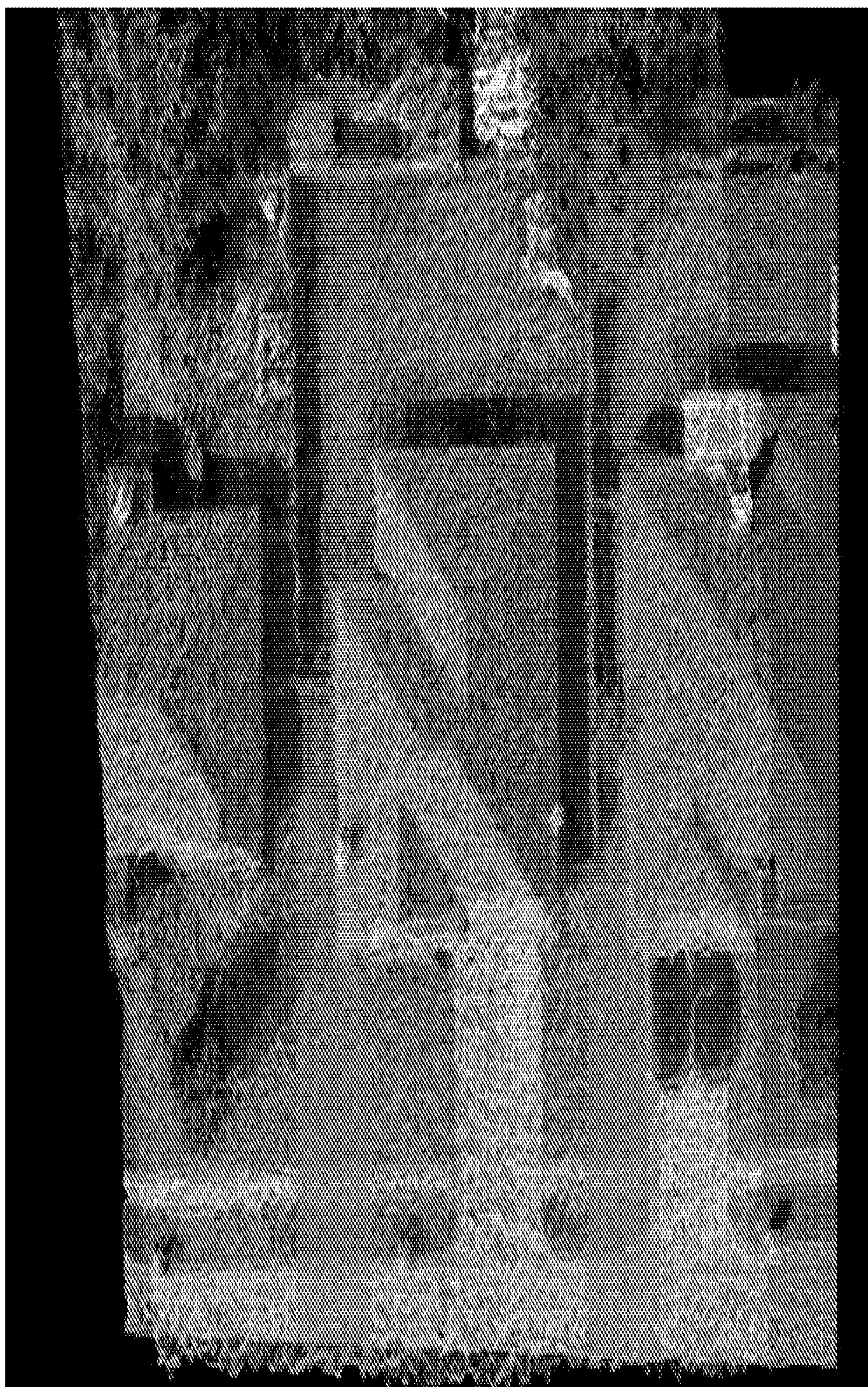


FIG. 24

25/28



534
↓

FIG. 25

26/28

536

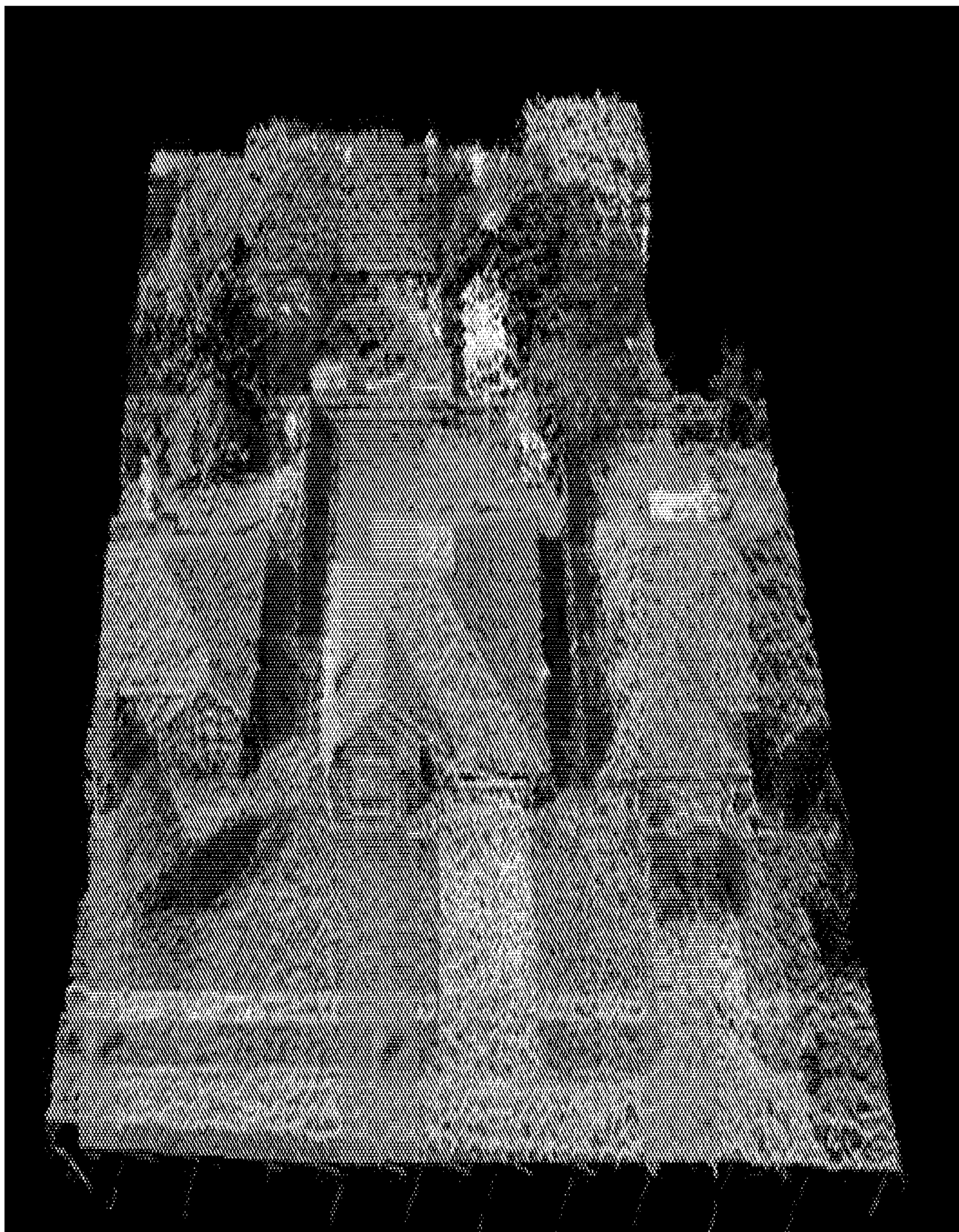


FIG. 26

27/28

538

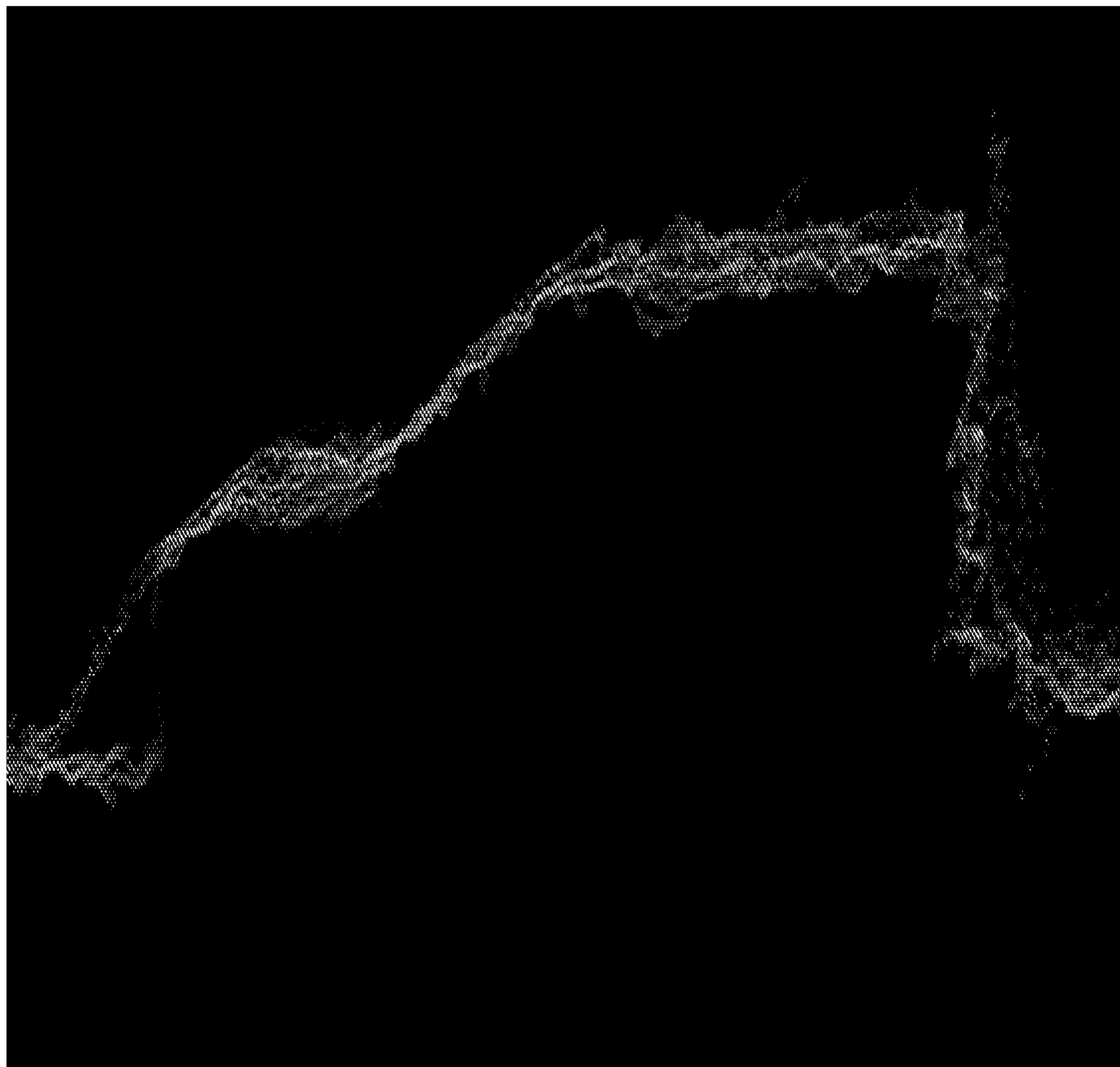


FIG. 27

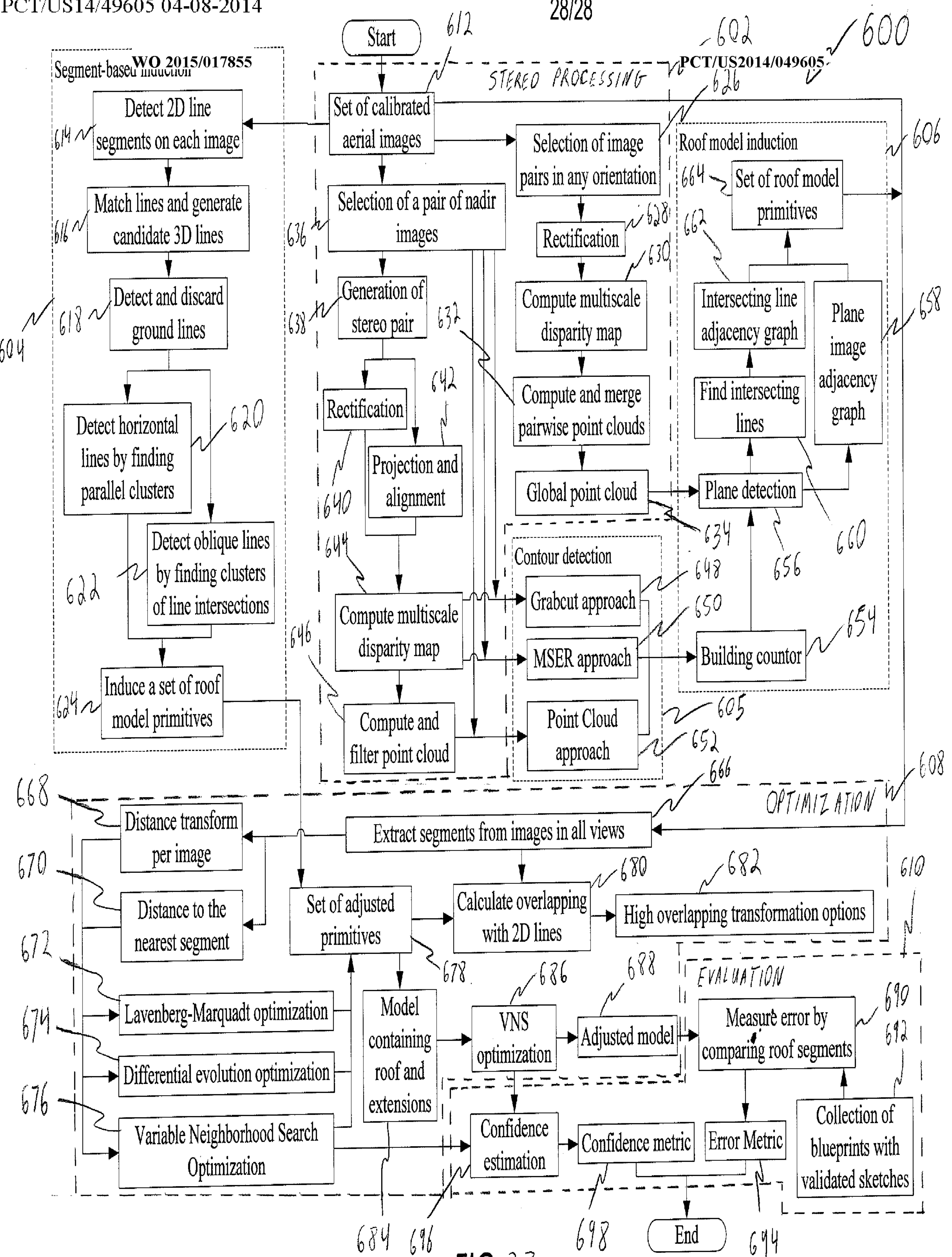


FIG. 28

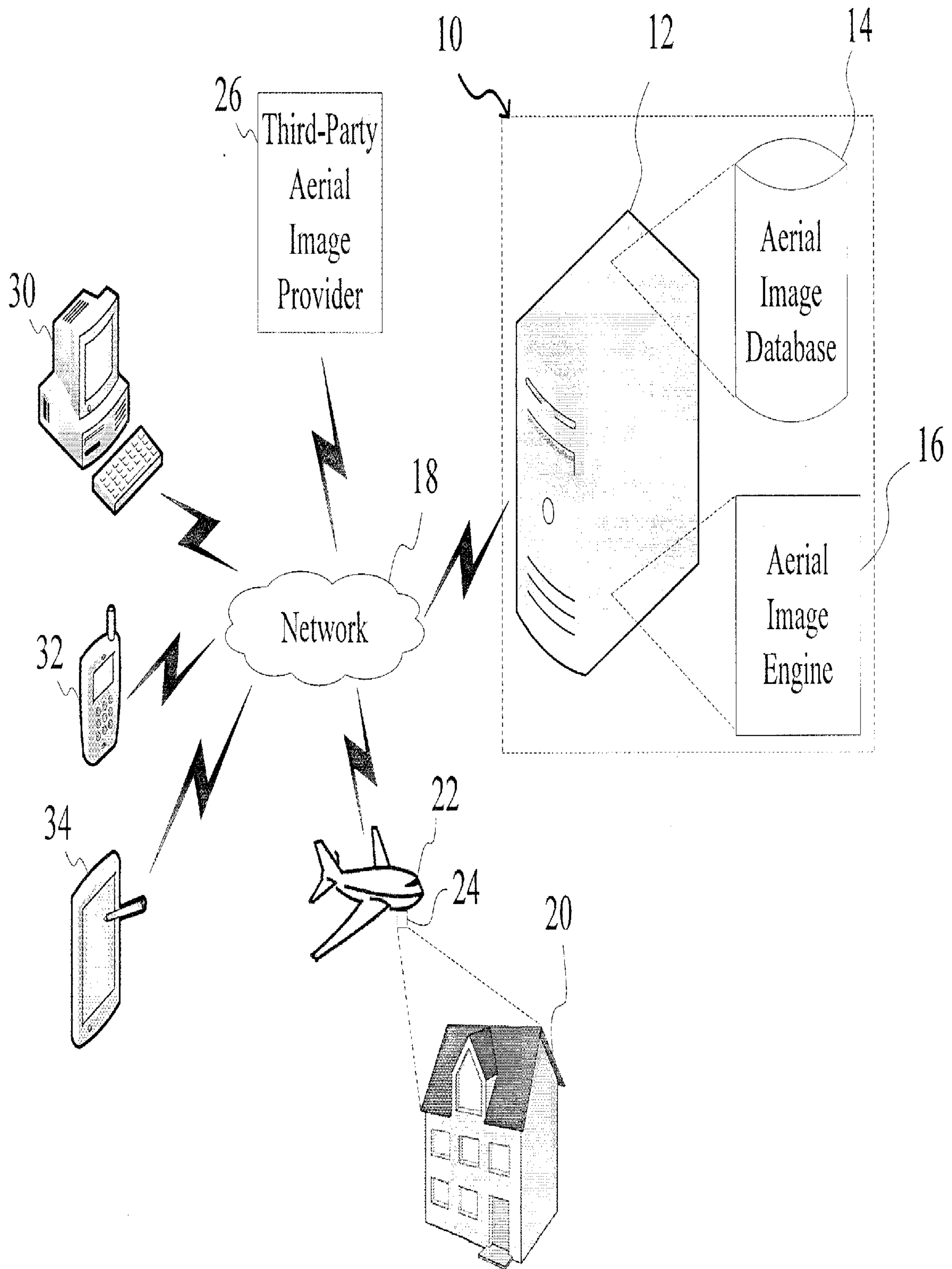


FIG. 1