



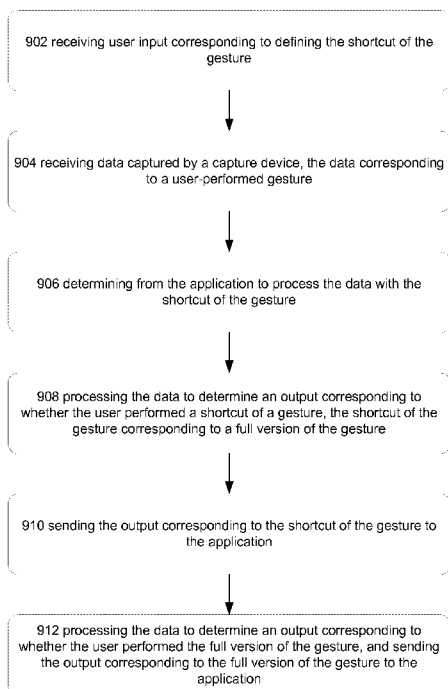
- (51) **International Patent Classification:**  
**G06F 3/01** (2006.01) **G06F 3/03** (2006.01)
- (21) **International Application Number:**  
PCT/US2010/036774
- (22) **International Filing Date:**  
29 May 2010 (29.05.2010)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**  
12/474,781 29 May 2009 (29.05.2009) US
- (71) **Applicant** (for all designated States except US): **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).
- (72) **Inventors:** **LATTA, Stephen**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **GEISNER, Kevin**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **CLAVIN, John**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond,

Washington 98052-6399 (US). **TSUNODA, Kudo**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **PEREZ, Kathryn Stone**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **MARKOVIC, Relja**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SNOOK, Gregory N.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **KIP-MAN, Alex**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (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, 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, IS, JP, KE, KG, KM, 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, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,

[Continued on next page]

(54) **Title:** GESTURE SHORTCUTS



**Fig. 9**

(57) **Abstract:** Systems, methods and computer readable media are disclosed for gesture shortcuts. A user's movement or body position is captured by a capture device of a system, and is used as input to control the system. For a system-recognized gesture, there may be a full version of the gesture and a shortcut of the gesture. Where the system recognizes that either the full version of the gesture or the shortcut of the gesture has been performed, it sends an indication that the system-recognized gesture was observed to a corresponding application. Where the shortcut comprises a subset of the full version of the gesture, and both the shortcut and the full version of the gesture are recognized as the user performs the full version of the gesture, the system recognizes that only a single performance of the gesture has occurred, and indicates to the application as such.



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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Declarations under Rule 4.17:**

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

**Published:**

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

## GESTURE SHORTCUTS

### BACKGROUND

[0001] Many computing applications such as computer games, multimedia  
5 applications, office applications or the like use controls to allow users to manipulate game  
characters or other aspects of an application. Typically such controls are input using, for  
example, controllers, remotes, keyboards, mice, or the like. Unfortunately, such controls  
can be difficult to learn, thus creating a barrier between a user and such games and  
applications. Furthermore, such controls may be different than actual game actions or  
10 other application actions for which the controls are used. For example, a game control  
that causes a game character to swing a baseball bat may not correspond to an actual  
motion of swinging the baseball bat.

### SUMMARY OF THE INVENTION

[0002] Disclosed herein are systems and methods for gesture shortcuts.

15 [0003] In an embodiment, a computing device receives a series of image data  
from a camera. This camera may comprise a color camera (such as red-green-blue or  
RGB), a depth camera, and a three-dimensional (3D) camera. This data may comprise  
separate depth and color images, a combined image that incorporates depth and color  
information, or a parsed image where objects are identified, such as people that are  
20 skeletal mapped. This data captures motions or poses made by at least one user. These  
motions or poses may be recognized by the computing device as a type of input – gesture  
input. For a given gesture (e.g. navigate up), there may be a full version of the gesture that  
the user may make and a shortcut of the gesture that the user may make, the shortcut of the  
gesture generally requiring less time, movement, or difficulty of movement for the user.

25 [0004] Where the computing device recognizes that either the shortcut of the  
gesture or the full version of the gesture has been performed by the user, it sends an  
indication of this to an application that uses gestures as input.

[0005] In an embodiment where the shortcut comprises a subset of the full  
version of the gesture, and both the shortcut and the full version of the gesture are  
30 recognized as the user performs the full version of the gesture, the computing device  
recognizes that only a single performance of the gesture has occurred, and indicates to the  
application as such.

[0006] The foregoing is a summary and thus contains, by necessity, simplifications, generalizations and omissions of detail. It may be appreciated that the summary is illustrative only and is not intended to be in any way limiting.

#### **BRIEF DESCRIPTION OF THE DRAWINGS**

5           [0007] The systems, methods, and computer readable media for gesture shortcuts in accordance with this specification are further described with reference to the accompanying drawings in which:

          [0008] FIGs. 1A and 1B illustrate an example embodiment of a target recognition, analysis, and tracking system with a user playing a game.

10          [0009] FIG. 2 illustrates an example embodiment of a capture device that may be used in a target recognition, analysis, and tracking system.

          [0010] FIG. 3A illustrates an example embodiment of a computing environment that may be used to interpret one or more gestures in a target recognition, analysis, and tracking system.

15          [0011] FIG. 3B illustrates another example embodiment of a computing environment that may be used to interpret one or more gestures in a target recognition, analysis, and tracking system.

          [0012] FIG. 4A illustrates a skeletal mapping of a user that has been generated from the target recognition, analysis, and tracking system of FIG. 2.

20          [0013] FIG. 4B illustrates further details of the gesture recognizer architecture shown in FIG. 2.

          [0014] FIGs. 5A and 5B illustrates how gesture filters may be stacked to create more complex gesture filters.

25          [0015] FIGs. 6A, 6B, 6C, 6D, and 6E illustrate an example gesture that a user may make to signal for a “fair catch” in football video game.

          [0016] FIGs. 7A, 7B, 7C, 7D and 7E illustrate the example “fair catch” gesture of FIGs. 6A-E as each frame of image data has been parsed to produce a skeletal map of the user.

          [0017] FIG. 8A illustrates a user making a full running gesture.

30          [0018] FIG. 8B illustrates a user making a shortcut running gesture, the shortcut gesture comprising a subset of the movement of the full running gesture of FIG. 8A.

          [0019] FIG. 8C illustrates a user making a second type of shortcut running gesture, the second type of shortcut running gesture comprising movement separate from the full running gesture of FIG. 8A.

[0020] FIG. 9 depicts example operational procedures for gesture shortcuts.

## DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0021] As will be described herein, a user may control an application executing  
5 on a computing environment such as a game console, a computer, or the like by  
performing one or more gestures. According to one embodiment, the gestures may be  
received by, for example, a capture device. For example, the capture device may capture a  
depth image of a scene. In one embodiment, the capture device may determine whether  
one or more targets or objects in the scene corresponds to a human target such as the user.  
10 To determine whether a target or object in the scene corresponds a human target, each of  
the targets may be flood filled and compared to a pattern of a human body model. Each  
target or object that matches the human body model may then be scanned to generate a  
skeletal model associated therewith. The skeletal model may then be provided to the  
computing environment such that the computing environment may track the skeletal  
15 model, render an avatar associated with the skeletal model, and may determine which  
controls to perform in an application executing on the computer environment based on, for  
example, gestures of the user that have been recognized from the skeletal model. A  
gesture recognizer engine, the architecture of which is described more fully below, is used  
to determine when a particular gesture has been made by the user.

20 [0022] FIGs. 1A and 1B illustrate an example embodiment of a configuration  
of a target recognition, analysis, and tracking system 10 with a user 18 playing a boxing  
game. In an example embodiment, the target recognition, analysis, and tracking system 10  
may be used to recognize, analyze, and/or track a human target such as the user 18.

[0023] As shown in FIG. 1A, the target recognition, analysis, and tracking  
25 system 10 may include a computing environment 12. The computing environment 12 may  
be a computer, a gaming system or console, or the like. According to an example  
embodiment, the computing environment 12 may include hardware components and/or  
software components such that the computing environment 12 may be used to execute  
applications such as gaming applications, non-gaming applications, or the like.

30 [0024] As shown in FIG. 1A, the target recognition, analysis, and tracking  
system 10 may further include a capture device 20. The capture device 20 may be, for  
example, a camera that may be used to visually monitor one or more users, such as the  
user 18, such that gestures performed by the one or more users may be captured, analyzed,

and tracked to perform one or more controls or actions within an application, as will be described in more detail below.

[0025] According to one embodiment, the target recognition, analysis, and tracking system 10 may be connected to an audiovisual device 16 such as a television, a monitor, a high-definition television (HDTV), or the like that may provide game or application visuals and/or audio to a user such as the user 18. For example, the computing environment 12 may include a video adapter such as a graphics card and/or an audio adapter such as a sound card that may provide audiovisual signals associated with the game application, non-game application, or the like. The audiovisual device 16 may receive the audiovisual signals from the computing environment 12 and may then output the game or application visuals and/or audio associated with the audiovisual signals to the user 18. According to one embodiment, the audiovisual device 16 may be connected to the computing environment 12 via, for example, an S-Video cable, a coaxial cable, an HDMI cable, a DVI cable, a VGA cable, or the like.

[0026] As shown in FIGs. 1A and 1B, the target recognition, analysis, and tracking system 10 may be used to recognize, analyze, and/or track a human target such as the user 18. For example, the user 18 may be tracked using the capture device 20 such that the movements of user 18 may be interpreted as controls that may be used to affect the application being executed by computer environment 12. Thus, according to one embodiment, the user 18 may move his or her body to control the application.

[0027] As shown in FIGs. 1A and 1B, in an example embodiment, the application executing on the computing environment 12 may be a boxing game that the user 18 may be playing. For example, the computing environment 12 may use the audiovisual device 16 to provide a visual representation of a boxing opponent 22 to the user 18. The computing environment 12 may also use the audiovisual device 16 to provide a visual representation of a player avatar 24 that the user 18 may control with his or her movements. For example, as shown in FIG. 1B, the user 18 may throw a punch in physical space to cause the player avatar 24 to throw a punch in game space. Thus, according to an example embodiment, the computer environment 12 and the capture device 20 of the target recognition, analysis, and tracking system 10 may be used to recognize and analyze the punch of the user 18 in physical space such that the punch may be interpreted as a game control of the player avatar 24 in game space.

[0028] Other movements by the user 18 may also be interpreted as other controls or actions, such as controls to bob, weave, shuffle, block, jab, or throw a variety of

different power punches. Furthermore, some movements may be interpreted as controls that may correspond to actions other than controlling the player avatar 24. For example, the player may use movements to end, pause, or save a game, select a level, view high scores, communicate with a friend, etc.

5           **[0029]** In example embodiments, the human target such as the user 18 may have an object. In such embodiments, the user of an electronic game may be holding the object such that the motions of the player and the object may be used to adjust and/or control parameters of the game. For example, the motion of a player holding a racket may be tracked and utilized for controlling an on-screen racket in an electronic sports game. In  
10 another example embodiment, the motion of a player holding an object may be tracked and utilized for controlling an on-screen weapon in an electronic combat game.

**[0030]** According to other example embodiments, the target recognition, analysis, and tracking system 10 may further be used to interpret target movements as operating system and/or application controls that are outside the realm of games. For  
15 example, virtually any controllable aspect of an operating system and/or application may be controlled by movements of the target such as the user 18.

**[0031]** FIG. 2 illustrates an example embodiment of the capture device 20 that may be used in the target recognition, analysis, and tracking system 10. According to an example embodiment, the capture device 20 may be configured to capture video with  
20 depth information including a depth image that may include depth values via any suitable technique including, for example, time-of-flight, structured light, stereo image, or the like. According to one embodiment, the capture device 20 may organize the calculated depth information into “Z layers,” or layers that may be perpendicular to a Z axis extending from the depth camera along its line of sight.

25           **[0032]** As shown in FIG. 2, the capture device 20 may include an image camera component 22. According to an example embodiment, the image camera component 22 may be a depth camera that may capture the depth image of a scene. The depth image may include a two-dimensional (2-D) pixel area of the captured scene where each pixel in the 2-D pixel area may represent a length in, for example, centimeters, millimeters, or the  
30 like of an object in the captured scene from the camera.

**[0033]** As shown in FIG. 2, according to an example embodiment, the image camera component 22 may include an IR light component 24, a three-dimensional (3-D) camera 26, and an RGB camera 28 that may be used to capture the depth image of a scene. For example, in time-of-flight analysis, the IR light component 24 of the capture device 20

may emit an infrared light onto the scene and may then use sensors (not shown) to detect the backscattered light from the surface of one or more targets and objects in the scene using, for example, the 3-D camera 26 and/or the RGB camera 28. In some embodiments, pulsed infrared light may be used such that the time between an outgoing light pulse and a corresponding incoming light pulse may be measured and used to determine a physical distance from the capture device 20 to a particular location on the targets or objects in the scene. Additionally, in other example embodiments, the phase of the outgoing light wave may be compared to the phase of the incoming light wave to determine a phase shift. The phase shift may then be used to determine a physical distance from the capture device to a particular location on the targets or objects.

[0034] According to another example embodiment, time-of-flight analysis may be used to indirectly determine a physical distance from the capture device 20 to a particular location on the targets or objects by analyzing the intensity of the reflected beam of light over time via various techniques including, for example, shuttered light pulse imaging.

[0035] In another example embodiment, the capture device 20 may use a structured light to capture depth information. In such an analysis, patterned light (i.e., light displayed as a known pattern such as grid pattern or a stripe pattern) may be projected onto the scene via, for example, the IR light component 24. Upon striking the surface of one or more targets or objects in the scene, the pattern may become deformed in response. Such a deformation of the pattern may be captured by, for example, the 3-D camera 26 and/or the RGB camera 28 and may then be analyzed to determine a physical distance from the capture device to a particular location on the targets or objects.

[0036] According to another embodiment, the capture device 20 may include two or more physically separated cameras that may view a scene from different angles, to obtain visual stereo data that may be resolved to generate depth information

[0037] The capture device 20 may further include a microphone 30. The microphone 30 may include a transducer or sensor that may receive and convert sound into an electrical signal. According to one embodiment, the microphone 30 may be used to reduce feedback between the capture device 20 and the computing environment 12 in the target recognition, analysis, and tracking system 10. Additionally, the microphone 30 may be used to receive audio signals that may also be provided by the user to control applications such as game applications, non-game applications, or the like that may be executed by the computing environment 12.



[0038] In an example embodiment, the capture device 20 may further include a processor 32 that may be in operative communication with the image camera component 22. The processor 32 may include a standardized processor, a specialized processor, a microprocessor, or the like that may execute instructions that may include instructions for receiving the depth image, determining whether a suitable target may be included in the depth image, converting the suitable target into a skeletal representation or model of the target, or any other suitable instruction.

[0039] The capture device 20 may further include a memory component 34 that may store the instructions that may be executed by the processor 32, images or frames of images captured by the 3-D camera or RGB camera, or any other suitable information, images, or the like. According to an example embodiment, the memory component 34 may include random access memory (RAM), read only memory (ROM), cache, Flash memory, a hard disk, or any other suitable storage component. As shown in FIG. 2, in one embodiment, the memory component 34 may be a separate component in communication with the image capture component 22 and the processor 32. According to another embodiment, the memory component 34 may be integrated into the processor 32 and/or the image capture component 22.

[0040] As shown in FIG. 2, the capture device 20 may be in communication with the computing environment 12 via a communication link 36. The communication link 36 may be a wired connection including, for example, a USB connection, a Firewire connection, an Ethernet cable connection, or the like and/or a wireless connection such as a wireless 802.11b, g, a, or n connection. According to one embodiment, the computing environment 12 may provide a clock to the capture device 20 that may be used to determine when to capture, for example, a scene via the communication link 36.

[0041] Additionally, the capture device 20 may provide the depth information and images captured by, for example, the 3-D camera 26 and/or the RGB camera 28, and a skeletal model that may be generated by the capture device 20 to the computing environment 12 via the communication link 36. The computing environment 12 may then use the skeletal model, depth information, and captured images to, for example, recognize user gestures and in response control an application such as a game or word processor. For example, as shown, in FIG. 2, the computing environment 12 may include a gestures recognizer engine 190. The gestures recognizer engine 190 may include a collection of gesture filters, each comprising information concerning a gesture that may be performed by the skeletal model (as the user moves). The data captured by the cameras 26, 28 and

device 20 in the form of the skeletal model and movements associated with it may be compared to the gesture filters in the gesture recognizer engine 190 to identify when a user (as represented by the skeletal model) has performed one or more gestures. Those gestures may be associated with various controls of an application. Thus, the computing environment 12 may use the gesture recognizer engine 190 to interpret movements of the skeletal model and to control an application based on the movements.

[0042] FIG. 3A illustrates an example embodiment of a computing environment that may be used to implement computing environment 12 of Figures 1A-2. The computing environment may be a multimedia console 100, such as a gaming console. As shown in FIG. 3A, the multimedia console 100 has a central processing unit (CPU) 101 having a level 1 cache 102, a level 2 cache 104, and a flash ROM (Read Only Memory) 106. The level 1 cache 102 and a level 2 cache 104 temporarily store data and hence reduce the number of memory access cycles, thereby improving processing speed and throughput. The CPU 101 may be provided having more than one core, and thus, additional level 1 and level 2 caches 102 and 104. The flash ROM 106 may store executable code that is loaded during an initial phase of a boot process when the multimedia console 100 is powered ON.

[0043] A graphics processing unit (GPU) 108 and a video encoder/video codec (coder/decoder) 114 form a video processing pipeline for high speed and high resolution graphics processing. Data is carried from the graphics processing unit 108 to the video encoder/video codec 114 via a bus. The video processing pipeline outputs data to an A/V (audio/video) port 140 for transmission to a television or other display. A memory controller 110 is connected to the GPU 108 to facilitate processor access to various types of memory 112, such as, but not limited to, a RAM (Random Access Memory).

[0044] The multimedia console 100 includes an I/O controller 120, a system management controller 122, an audio processing unit 123, a network interface controller 124, a first USB host controller 126, a second USB controller 128 and a front panel I/O subassembly 130 that are preferably implemented on a module 118. The USB controllers 126 and 128 serve as hosts for peripheral controllers 142(1)-142(2), a wireless adapter 148, and an external memory device 146 (e.g., flash memory, external CD/DVD ROM drive, removable media, etc.). The network interface 124 and/or wireless adapter 148 provide access to a network (e.g., the Internet, home network, etc.) and may be any of a wide variety of various wired or wireless adapter components including an Ethernet card, a modem, a Bluetooth module, a cable modem, and the like.

[0045] System memory 143 is provided to store application data that is loaded during the boot process. A media drive 144 is provided and may comprise a DVD/CD drive, hard drive, or other removable media drive, etc. The media drive 144 may be internal or external to the multimedia console 100. Application data may be accessed via the media drive 144 for execution, playback, etc. by the multimedia console 100. The media drive 144 is connected to the I/O controller 120 via a bus, such as a Serial ATA bus or other high speed connection (e.g., IEEE 1394).

[0046] The system management controller 122 provides a variety of service functions related to assuring availability of the multimedia console 100. The audio processing unit 123 and an audio codec 132 form a corresponding audio processing pipeline with high fidelity and stereo processing. Audio data is carried between the audio processing unit 123 and the audio codec 132 via a communication link. The audio processing pipeline outputs data to the A/V port 140 for reproduction by an external audio player or device having audio capabilities.

[0047] The front panel I/O subassembly 130 supports the functionality of the power button 150 and the eject button 152, as well as any LEDs (light emitting diodes) or other indicators exposed on the outer surface of the multimedia console 100. A system power supply module 136 provides power to the components of the multimedia console 100. A fan 138 cools the circuitry within the multimedia console 100.

[0048] The CPU 101, GPU 108, memory controller 110, and various other components within the multimedia console 100 are interconnected via one or more buses, including serial and parallel buses, a memory bus, a peripheral bus, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include a Peripheral Component Interconnects (PCI) bus, PCI-Express bus, etc.

[0049] When the multimedia console 100 is powered ON, application data may be loaded from the system memory 143 into memory 112 and/or caches 102, 104 and executed on the CPU 101. The application may present a graphical user interface that provides a consistent user experience when navigating to different media types available on the multimedia console 100. In operation, applications and/or other media contained within the media drive 144 may be launched or played from the media drive 144 to provide additional functionalities to the multimedia console 100.

[0050] The multimedia console 100 may be operated as a standalone system by simply connecting the system to a television or other display. In this standalone mode, the

multimedia console 100 allows one or more users to interact with the system, watch movies, or listen to music. However, with the integration of broadband connectivity made available through the network interface 124 or the wireless adapter 148, the multimedia console 100 may further be operated as a participant in a larger network community.

5           **[0051]** When the multimedia console 100 is powered ON, a set amount of hardware resources are reserved for system use by the multimedia console operating system. These resources may include a reservation of memory (e.g., 16MB), CPU and GPU cycles (e.g., 5%), networking bandwidth (e.g., 8 kbs), etc. Because these resources are reserved at system boot time, the reserved resources do not exist from the application's  
10 view.

**[0052]** In particular, the memory reservation preferably is large enough to contain the launch kernel, concurrent system applications and drivers. The CPU reservation is preferably constant such that if the reserved CPU usage is not used by the system applications, an idle thread will consume any unused cycles.

15           **[0053]** With regard to the GPU reservation, lightweight messages generated by the system applications (e.g., popups) are displayed by using a GPU interrupt to schedule code to render popup into an overlay. The amount of memory required for an overlay depends on the overlay area size and the overlay preferably scales with screen resolution. Where a full user interface is used by the concurrent system application, it is preferable to  
20 use a resolution independent of application resolution. A scaler may be used to set this resolution such that the need to change frequency and cause a TV resynch is eliminated.

**[0054]** After the multimedia console 100 boots and system resources are reserved, concurrent system applications execute to provide system functionalities. The system functionalities are encapsulated in a set of system applications that execute within  
25 the reserved system resources described above. The operating system kernel identifies threads that are system application threads versus gaming application threads. The system applications are preferably scheduled to run on the CPU 101 at predetermined times and intervals in order to provide a consistent system resource view to the application. The scheduling is to minimize cache disruption for the gaming application running on the  
30 console.

**[0055]** When a concurrent system application requires audio, audio processing is scheduled asynchronously to the gaming application due to time sensitivity. A multimedia console application manager (described below) controls the gaming application audio level (e.g., mute, attenuate) when system applications are active.

[0056] Input devices (e.g., controllers 142(1) and 142(2)) are shared by gaming applications and system applications. The input devices are not reserved resources, but are to be switched between system applications and the gaming application such that each will have a focus of the device. The application manager preferably controls the switching of input stream, without knowledge the gaming application's knowledge and a driver maintains state information regarding focus switches. The cameras 26, 28 and capture device 20 may define additional input devices for the console 100.

[0057] FIG. 3B illustrates another example embodiment of a computing environment 220 that may be the computing environment 12 shown in FIGs. 1A-2 used to interpret one or more gestures in a target recognition, analysis, and tracking system. The computing system environment 220 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the presently disclosed subject matter. Neither should the computing environment 220 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 220. In some embodiments the various depicted computing elements may include circuitry configured to instantiate specific aspects of the present disclosure. For example, the term circuitry used in the disclosure can include specialized hardware components configured to perform function(s) by firmware or switches. In other examples embodiments the term circuitry can include a general purpose processing unit, memory, etc., configured by software instructions that embody logic operable to perform function(s). In example embodiments where circuitry includes a combination of hardware and software, an implementer may write source code embodying logic and the source code can be compiled into machine readable code that can be processed by the general purpose processing unit. Since one can appreciate that the state of the art has evolved to a point where there is little difference between hardware, software, or a combination of hardware/software, the selection of hardware versus software to effectuate specific functions is a design choice left to an implementer. More specifically, one of skill in the art can appreciate that a software process can be transformed into an equivalent hardware structure, and a hardware structure can itself be transformed into an equivalent software process. Thus, the selection of a hardware implementation versus a software implementation is one of design choice and left to the implementer.

[0058] In FIG. 3B, the computing environment 220 comprises a computer 241, which typically includes a variety of computer readable media. Computer readable media

can be any available media that can be accessed by computer 241 and includes both volatile and nonvolatile media, removable and non-removable media. The system memory 222 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 223 and random access memory (RAM) 260.

5 A basic input/output system 224 (BIOS), containing the basic routines that help to transfer information between elements within computer 241, such as during start-up, is typically stored in ROM 223. RAM 260 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 259. By way of example, and not limitation, FIG. 3B illustrates operating system 225, application  
10 programs 226, other program modules 227, and program data 228.

[0059] The computer 241 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 3B illustrates a hard disk drive 238 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 239 that reads from or writes to a removable, nonvolatile  
15 magnetic disk 254, and an optical disk drive 240 that reads from or writes to a removable, nonvolatile optical disk 253 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM,  
20 solid state ROM, and the like. The hard disk drive 238 is typically connected to the system bus 221 through an non-removable memory interface such as interface 234, and magnetic disk drive 239 and optical disk drive 240 are typically connected to the system bus 221 by a removable memory interface, such as interface 235.

[0060] The drives and their associated computer storage media discussed above  
25 and illustrated in FIG. 3B, provide storage of computer readable instructions, data structures, program modules and other data for the computer 241. In FIG. 3B, for example, hard disk drive 238 is illustrated as storing operating system 258, application programs 257, other program modules 256, and program data 255. Note that these components can either be the same as or different from operating system 225, application  
30 programs 226, other program modules 227, and program data 228. Operating system 258, application programs 257, other program modules 256, and program data 255 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 241 through input devices such as a keyboard 251 and pointing device 252, commonly referred to as a mouse, trackball or

touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 259 through a user input interface 236 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). The cameras 26, 28 and capture device 20 may define additional input devices for the console 100. A monitor 242 or other type of display device is also connected to the system bus 221 via an interface, such as a video interface 232. In addition to the monitor, computers may also include other peripheral output devices such as speakers 244 and printer 243, which may be connected through a output peripheral interface 233.

[0061] The computer 241 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 246. The remote computer 246 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 241, although only a memory storage device 247 has been illustrated in FIG. 3B. The logical connections depicted in FIG. 3B include a local area network (LAN) 245 and a wide area network (WAN) 249, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0062] When used in a LAN networking environment, the computer 241 is connected to the LAN 245 through a network interface or adapter 237. When used in a WAN networking environment, the computer 241 typically includes a modem 250 or other means for establishing communications over the WAN 249, such as the Internet. The modem 250, which may be internal or external, may be connected to the system bus 221 via the user input interface 236, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 241, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 3B illustrates remote application programs 248 as residing on memory device 247. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0063] FIG. 4A depicts an example skeletal mapping of a user that may be generated from the capture device 20. In this embodiment, a variety of joints and bones are identified: each hand 302, each forearm 304, each elbow 306, each bicep 308, each shoulder 310, each hip 312, each thigh 314, each knee 316, each foreleg 318, each foot

320, the head 322, the torso 324, the top 326 and bottom 328 of the spine, and the waist 330. Where more points are tracked, additional features may be identified, such as the bones and joints of the fingers or toes, or individual features of the face, such as the nose and eyes.

5           **[0064]** Through moving his body, a user may create gestures. A gesture comprises a motion or pose by a user that may be captured as image data and parsed for meaning. A gesture may be dynamic, comprising a motion, such as mimicking throwing a ball. A gesture may be a static pose, such as holding one's crossed forearms 304 in front of his torso 324. A gesture may also incorporate props, such as by swinging a mock  
10 sword. A gesture may comprise more than one body part, such as clapping the hands 302 together, or a subtler motion, such as pursing one's lips.

**[0065]** Gestures may be used for input in a general computing context. For instance, various motions of the hands 302 or other body parts may correspond to common system wide tasks such as navigate up or down in a hierarchical list, open a file, close a  
15 file, and save a file. Gestures may also be used in a video-game-specific context, depending on the game. For instance, with a driving game, various motions of the hands 302 and feet 320 may correspond to steering a vehicle in a direction, shifting gears, accelerating, and breaking.

**[0066]** A user may generate a gesture that corresponds to walking or running, by  
20 walking or running in place himself. The user may alternately lift and drop each leg 312-320 to mimic walking without moving. The system may parse this gesture by analyzing each hip 312 and each thigh 314. A step may be recognized when one hip-thigh angle (as measured relative to a vertical line, wherein a standing leg has a hip-thigh angle of 0°, and a forward horizontally extended leg has a hip-thigh angle of 90°) exceeds a certain  
25 threshold relative to the other thigh. A walk or run may be recognized after some number of consecutive steps by alternating legs. The time between the two most recent steps may be thought of as a period. After some number of periods where that threshold angle is not met, the system may determine that the walk or running gesture has ceased.

**[0067]** Given a "walk or run" gesture, an application may set values for  
30 parameters associated with this gesture. These parameters may include the above threshold angle, the number of steps required to initiate a walk or run gesture, a number of periods where no step occurs to end the gesture, and a threshold period that determines whether the gesture is a walk or a run. A fast period may correspond to a run, as the user will be moving his legs quickly, and a slower period may correspond to a walk.



[0068] A gesture may be associated with a set of default parameters at first that the application may override with its own parameters. In this scenario, an application is not forced to provide parameters, but may instead use a set of default parameters that allow the gesture to be recognized in the absence of application-defined parameters.

5 [0069] There are a variety of outputs that may be associated with the gesture. There may be a baseline “yes or no” as to whether a gesture is occurring. There also may be a confidence level, which corresponds to the likelihood that the user’s tracked movement corresponds to the gesture. This could be a linear scale that ranges over floating point numbers between 0 and 1, inclusive. Wherein an application receiving this  
10 gesture information cannot accept false-positives as input, it may use only those recognized gestures that have a high confidence level, such as at least .95. Where an application must recognize every instance of the gesture, even at the cost of false-positives, it may use gestures that have at least a much lower confidence level, such as those merely greater than .2. The gesture may have an output for the time between the two  
15 most recent steps, and where only a first step has been registered, this may be set to a reserved value, such as -1 (since the time between any two steps must be positive). The gesture may also have an output for the highest thigh angle reached during the most recent step.

[0070] Another exemplary gesture is a “heel lift jump.” In this, a user may  
20 create the gesture by raising his heels off the ground, but keeping his toes planted. Alternatively, the user may jump into the air where his feet 320 leave the ground entirely. The system may parse the skeleton for this gesture by analyzing the angle relation of the shoulders 310, hips 312 and knees 316 to see if they are in a position of alignment equal to standing up straight. Then these points and upper 326 and lower 328 spine points may be  
25 monitored for any upward acceleration. A sufficient combination of acceleration may trigger a jump gesture.

[0071] Given this “heel lift jump” gesture, an application may set values for parameters associated with this gesture. The parameters may include the above acceleration threshold, which determines how fast some combination of the user’s  
30 shoulders 310, hips 312 and knees 316 must move upward to trigger the gesture, as well as a maximum angle of alignment between the shoulders 310, hips 312 and knees 316 at which a jump may still be triggered.

[0072] The outputs may comprise a confidence level, as well as the user’s body angle at the time of the jump.

[0073] Setting parameters for a gesture based on the particulars of the application that will receive the gesture is important in accurately identifying gestures. Properly identifying gestures and the intent of a user greatly helps in creating a positive user experience. Where a gesture recognizer system is too sensitive, and even a slight forward motion of the hand 302 is interpreted as a throw, the user may become frustrated because gestures are being recognized where he has no intent to make a gesture, and thus, he lacks control over the system. Where a gesture recognizer system is not sensitive enough, the system may not recognize conscious attempts by the user to make a throwing gesture, frustrating him in a similar manner. At either end of the sensitivity spectrum, the user becomes frustrated because he cannot properly provide input to the system.

[0074] Another parameter to a gesture may be a distance moved. Where a user's gestures control the actions of an avatar in a virtual environment, that avatar may be arm's length from a ball. If the user wishes to interact with the ball and grab it, this may require the user to extend his arm 302-310 to full length while making the grab gesture. In this situation, a similar grab gesture where the user only partially extends his arm 302-310 may not achieve the result of interacting with the ball.

[0075] A gesture or a portion thereof may have as a parameter a volume of space in which it must occur. This volume of space may typically be expressed in relation to the body where a gesture comprises body movement. For instance, a football throwing gesture for a right-handed user may be recognized only in the volume of space no lower than the right shoulder 310a, and on the same side of the head 322 as the throwing arm 302a-310a. It may not be necessary to define all bounds of a volume, such as with this throwing gesture, where an outer bound away from the body is left undefined, and the volume extends out indefinitely, or to the edge of scene that is being monitored.

[0076] FIG. 4B provides further details of one exemplary embodiment of the gesture recognizer engine 190 of FIG. 2. As shown, the gesture recognizer engine 190 may comprise at least one filter 418 to determine a gesture or gestures. A filter 418 comprises information defining a gesture 426 (hereinafter referred to as a "gesture") along with parameters 428, or metadata, for that gesture. For instance, a throw, which comprises motion of one of the hands from behind the rear of the body to past the front of the body, may be implemented as a gesture 426 comprising information representing the movement of one of the hands of the user from behind the rear of the body to past the front of the body, as that movement would be captured by the depth camera. Parameters 428 may then be set for that gesture 426. Where the gesture 426 is a throw, a parameter 428 may be a

threshold velocity that the hand has to reach, a distance the hand must travel (either absolute, or relative to the size of the user as a whole), and a confidence rating by the recognizer engine that the gesture occurred. These parameters 428 for the gesture 426 may vary between applications, between contexts of a single application, or within one  
5 context of one application over time.

[0077] Filters may be modular or interchangeable. In an embodiment, a filter has a number of inputs, each of those inputs having a type, and a number of outputs, each of those outputs having a type. In this situation, a first filter may be replaced with a second filter that has the same number and types of inputs and outputs as the first filter without  
10 altering any other aspect of the recognizer engine architecture. For instance, there may be a first filter for driving that takes as input skeletal data and outputs a confidence that the gesture associated with the filter is occurring and an angle of steering. Where one wishes to substitute this first driving filter with a second driving filter – perhaps because the second driving filter is more efficient and requires fewer processing resources – one may  
15 do so by simply replacing the first filter with the second filter so long as the second filter has those same inputs and outputs – one input of skeletal data type, and two outputs of confidence type and angle type.

[0078] A filter need not have a parameter. For instance, a “user height” filter that returns the user’s height may not allow for any parameters that may be tuned. An  
20 alternate “user height” filter may have tunable parameters – such as to whether to account for a user’s footwear, hairstyle, headwear and posture in determining the user’s height.

[0079] Inputs to a filter may comprise things such as joint data about a user’s joint position, like angles formed by the bones that meet at the joint, RGB color data from the scene, and the rate of change of an aspect of the user. Outputs from a filter may  
25 comprise things such as the confidence that a given gesture is being made, the speed at which a gesture motion is made, and a time at which a gesture motion is made.

[0080] A context may be a cultural context, and it may be an environmental context. A cultural context refers to the culture of a user using a system. Different cultures may use similar gestures to impart markedly different meanings. For instance, an  
30 American user who wishes to tell another user to “look” or “use his eyes” may put his index finger on his head close to the distal side of his eye. However, to an Italian user, this gesture may be interpreted as a reference to the mafia.

[0081] Similarly, there may be different contexts among different environments of a single application. Take a first-person shooter game that involves operating a motor

vehicle. While the user is on foot, making a fist with the fingers towards the ground and extending the fist in front and away from the body may represent a punching gesture. While the user is in the driving context, that same motion may represent a “gear shifting” gesture. There may also be one or more menu environments, where the user can save his  
5 game, select among his character’s equipment or perform similar actions that do not comprise direct game-play. In that environment, this same gesture may have a third meaning, such as to select something or to advance to another screen.

[0082] The gesture recognizer engine 190 may have a base recognizer engine 416 that provides functionality to a gesture filter 418. In an embodiment, the functionality  
10 that the recognizer engine 416 implements includes an input-over-time archive that tracks recognized gestures and other input, a Hidden Markov Model implementation (where the modeled system is assumed to be a Markov process – one where a present state encapsulates any past state information necessary to determine a future state, so no other past state information must be maintained for this purpose – with unknown parameters,  
15 and hidden parameters are determined from the observable data), as well as other functionality required to solve particular instances of gesture recognition.

[0083] Filters 418 are loaded and implemented on top of the base recognizer engine 416 and can utilize services provided by the engine 416 to all filters 418. In an embodiment, the base recognizer engine 416 processes received data to determine whether  
20 it meets the requirements of any filter 418. Since these provided services, such as parsing the input, are provided once by the base recognizer engine 416 rather than by each filter 418, such a service need only be processed once in a period of time as opposed to once per filter 418 for that period, so the processing required to determine gestures is reduced.

[0084] An application may use the filters 418 provided by the recognizer engine 190, or it may provide its own filter 418, which plugs in to the base recognizer engine 416.  
25 In an embodiment, all filters 418 have a common interface to enable this plug-in characteristic. Further, all filters 418 may utilize parameters 428, so a single gesture tool as described below may be used to debug and tune the entire filter system 418.

[0085] These parameters 428 may be tuned for an application or a context of an  
30 application by a gesture tool 420. In an embodiment, the gesture tool 420 comprises a plurality of sliders 422, each slider 422 corresponding to a parameter 428, as well as a pictorial representation of a body 424. As a parameter 428 is adjusted with a corresponding slider 422, the body 424 may demonstrate both actions that would be recognized as the gesture with those parameters 428 and actions that would not be

recognized as the gesture with those parameters 428, identified as such. This visualization of the parameters 428 of gestures provides an effective means to both debug and fine tune a gesture.

[0086] FIG. 5 depicts more complex gestures or filters 418 created from stacked gestures or filters 418. Gestures can stack on each other. That is, more than one gesture may be expressed by a user at a single time. For instance, rather than disallowing any input but a throw when a throwing gesture is made, or requiring that a user remain motionless save for the components of the gesture (e.g. stand still while making a throwing gesture that involves only one arm). Where gestures stack, a user may make a jumping gesture and a throwing gesture simultaneously, and both of these gestures will be recognized by the gesture engine.

[0087] FIG. 5A depicts a simple gesture filter 418 according to the stacking paradigm. The IFilter filter 502 is a basic filter 418 that may be used in every gesture filter. IFilter 502 takes user position data 504 and outputs a confidence level 506 that a gesture has occurred. It also feeds that position data 504 into a SteeringWheel filter 508 that takes it as an input and outputs an angle to which the user is steering (e.g. 40 degrees to the right of the user's current bearing) 510.

[0088] FIG. 5B depicts a more complex gesture that stacks filters 418 onto the gesture filter of FIG. 5A. In addition to IFilter 502 and SteeringWheel 508, there is an ITracking filter 512 that receives position data 504 from IFilter 502 and outputs the amount of progress the user has made through a gesture 514. ITracking 512 also feeds position data 504 to GreaseLightning 516 and EBrake 518, which are filters 418 regarding other gestures that may be made in operating a vehicle, such as using the emergency brake.

[0089] FIG. 6 depicts an example gesture that a user 602 may make to signal for a "fair catch" in a football video game. These figures depict the user at points in time, with FIG. 6A being the first point in time, and FIG. 6E being the last point in time. Each of these figures may correspond to a snapshot or frame of image data as captured by a depth camera 402, though not necessarily consecutive frames of image data, as the depth camera 402 may be able to capture frames more rapidly than the user may cover the distance. For instance, this gesture may occur over a period of 3 seconds, and where a depth camera captures data at 40 frames per second, it would capture 60 frames of image data while the user 602 made this fair catch gesture.

[0090] In FIG. 6A, the user 602 begins with his arms 604 down at his sides. He then raises them up and above his shoulders as depicted in FIG. 6B and then further up, to the approximate level of his head, as depicted in FIG. 6C. From there, he lowers his arms 604 to shoulder level, as depicted in FIG. 6D, and then again raises them up, to the  
5 approximate level of his head, as depicted in FIG. 6E. Where a system captures these positions by the user 602 without any intervening position that may signal that the gesture is cancelled, or another gesture is being made, it may have the fair catch gesture filter output a high confidence level that the user 602 made the fair catch gesture.

[0091] FIG. 7 depicts the example “fair catch” gesture of FIG. 5 as each frame of  
10 image data has been parsed to produce a skeletal map of the user. The system, having produced a skeletal map from the depth image of the user, may now determine how that user’s body moves over time, and from that, parse the gesture.

[0092] In FIG. 7A, the user’s shoulders 310, are above his elbows 306, which in turn are above his hands 302. The shoulders 310, elbows 306 and hands 302 are then at a  
15 uniform level in FIG. 7B. The system then detects in FIG. 7C that the hands 302 are above the elbows, which are above the shoulders 310. In FIG. 7D, the user has returned to the position of FIG. 7B, where the shoulders 310, elbows 306 and hands 302 are at a uniform level. In the final position of the gesture, shown in FIG. 7E, the user returns to the position of FIG. 7C, where the hands 302 are above the elbows, which are above the  
20 shoulders 310.

[0093] While the capture device 20 captures a series of still images, such that in any one image the user appears to be stationary, the user is moving in the course of performing this gesture (as opposed to a stationary gesture, as discussed *supra*). The system is able to take this series of poses in each still image, and from that determine the  
25 confidence level of the moving gesture that the user is making.

[0094] In performing the gesture, a user is unlikely to be able to create an angle as formed by his right shoulder 310a, right elbow 306a and right hand 302a of, for example, between 140° and 145°. So, the application using the filter 418 for the fair catch gesture 426 may tune the associated parameters 428 to best serve the specifics of the  
30 application. For instance, the positions in FIGs. 7C and 7E may be recognized any time the user has his hands 302 above his shoulders 310, without regard to elbow 306 position. A set of parameters that are more strict may require that the hands 302 be above the head 310 and that the elbows 306 be both above the shoulders 310 and between the head 322 and the hands 302. Additionally, the parameters 428 for a fair catch gesture 426 may

require that the user move from the position of FIG. 7A through the position of FIG. 7E within a specified period of time, such as 1.5 seconds, and if the user takes more than 1.5 seconds to move through these positions, it will not be recognized as the fair catch 418, and a very low confidence level may be output.

5           **[0095]** FIGs. 8A-C illustrate a user making the same system-recognized running gesture through different captured movements and poses.

**[0096]** FIG. 8A illustrates a user making a full running gesture. User 18 is captured by capture device 20. User 18 creates the full running gesture by running in place – alternately lifting each of his knees to approximately waist height then dropping  
10 the leg down to the ground. This version of the full running gesture is a periodic gesture in that user 18 repeats the motions that comprise the gesture for the duration that he wants the gesture to last.

**[0097]** FIG. 8B illustrates a user making a shortcut running gesture, the shortcut gesture comprising a subset of the movement of the full running gesture of FIG. 8A. To  
15 make this version of the shortcut running gesture, user 18 lifts one of his legs such that his knee is approximately at hip level, and holds this pose. This gesture shortcut comprises a subset of the movement of the full gesture of FIG. 8A – where the user in FIG. 8A repeatedly lifts and drops his knees, here the user lifts his knee once and holds that pose. While the full gesture of FIG. 8A involves periodic movement, in this embodiment, the  
20 shortcut gesture involves a series of non-repeated movements, or a series of movements where the series as a whole is not repeated. In an embodiment, user 18 drops his knee down to a standing pose when he wishes to end the gesture. In an embodiment, this act of dropping the knee may also comprise a subset of the full gesture. In an embodiment, computing environment 12 determines that this movement is to end the gesture shortcut  
25 rather than produce the full gesture when user 18 holds his knee at approximately hip level for more than a specified amount of time. serial

**[0098]** FIG. 8C illustrates a user making a second type of shortcut running gesture, the second type of shortcut running gesture comprising movement separate from the full running gesture of FIG. 8A. Here, user 18 takes one step forward and holds this  
30 pose with one foot in front of the other, both feet on the ground, for the duration that he wishes to produce the running gesture. This position is not found in the full running gesture of FIG. 8A. User 18 may end the gesture by stepping back to a standing pose. This gesture is similar to that of FIG. 8B in that both involve movement to initiate the gesture, then holding a pose to maintain the gesture, and movement to end the gesture.

[0099] FIG. 9 illustrates example operating procedures for gesture shortcuts. As discussed above, one gesture input to a computing device may be recognized by the computing device as a result of a plurality of ways performed by a user. In an embodiment, this plurality of ways that a gesture may be performed by a user comprises a full version of the gesture and a shortcut of the gesture.

[0100] Gesture shortcuts may be used in a variety of application contexts. For instance, running gesture shortcuts may be used in applications that involve running, like a track and field game. Text input shortcuts may be used in a text input context of an application. For example, the user may use sign language gestures to input text. A full version of a word gesture may comprise signing each letter of the word, such as H-E-A-R-T. A shortcut for the “heart” word gesture may comprise a single gesture for heart, such from the user forming hands into a representation of a heart. Such a sign language may comprise American Sign Language (ASL).

[0101] A gesture shortcut may involve different body parts than the corresponding full version of a gesture. For instance, where a user lacks use of his legs, and the full version of a running gesture involves running in place, the shortcut of the gesture may involve mimicking a running motion with the user’s hands.

[0102] Optional operation 902 depicts receiving user input corresponding to defining the shortcut of the gesture. For instance, the user, either by being prompted by the computing device or through indicating to the computing device his desire to do so, may make a motion or pose that is captured by a capture device and stored as a way to perform the gesture.

[0103] In an embodiment, where the user has defined a gesture shortcut through his movement or pose, he may then refine the gesture shortcut on the system. For instance, where gestures are recognized using filters and corresponding parameters, he may tune the parameters of his gesture shortcut in ways as discussed above.

[0104] In an embodiment, the shortcut of the gesture corresponds to a full version of a second gesture. A gesture shortcut may correspond to a plurality of full gestures, and where the user defines a shortcut, he may indicate that the shortcut is to correspond to a plurality of full gestures. For instance, in a context of printing a text file, the user may define one gesture shortcut that corresponds to the full gesture of selecting the paper orientation to be portrait, the full gesture of selecting four copies to print, and the full gesture of selecting a specific printer to print from.



[0105] In an embodiment where gestures are recognized through gesture filters and parameters, the shortcut of a gesture and the full version of a gesture may use the same gesture filter, but a different value for one or more parameters. For instance, the full version of a “ball throw” gesture may require that the user move his hand from behind his torso to approximately arm’s length in front of his torso. The shortcut may reduce the required distance that the hand must travel such that the hand must neither be extended as far back nor as far forward. This may be effectuated for changing a parameter value or values, such as one for “minimum hand distance.”

[0106] Operation 904 depicts receiving data captured by a capture device, the data corresponding to a user-performed gesture. The capture device may capture a scene that contains all of the user, such as from the floor to the ceiling and to the wall on each side of a room at the distance in which the user is located. The capture device may also capture a scene that contains only part of the user, such as the user from the abdomen up as he or she sits at a desk. The capture device may also capture an object controlled by the user, such as a prop camera that the user holds in his or her hand.

[0107] Optional operation 906 depicts determining from the application to process the data with the shortcut of the gesture. An application may limit the shortcuts that are used as input in some way. For instance, in a track and field game, running may be considered integral to the process and the application may disallow or disable a shortcut for a running gesture, requiring a user to make a full running gesture when he wishes to run. In contrast, in a first-person shooter game, running may be considered ancillary to use of the game, so use of a shortcut for a running gesture may be allowed. In this first-person shooter game, mechanics for discharging a firearm may be considered integral to the process, and the application may disallow or disable shortcuts for an aiming or firing gesture.

[0108] In an embodiment, a user may perform both shortcuts for gestures and full versions of gestures that are recognized, in the same manner that a user may simultaneously perform multiple gestures, as discussed previously. Using the first-person shooter example, the user may simultaneously make the shortcut for the running gesture, and the full version of the aiming gesture.

[0109] In an embodiment, this determination to process the data with the shortcut of the gesture originates from the user. For instance, which shortcuts to process may correspond to a difficulty level of the application that the user selects. Where the user chooses the lowest difficulty level, all shortcuts may be processed. As the user increases

the difficulty level, the number of allowed shortcuts to process may decrease, until the highest difficulty level where no shortcuts are processed.

[0110] This determination may change over time to be adaptive to user ability. For instance, a default setting of allowed shortcuts may be implemented at the start of an application session, and the allowed shortcuts may be increased or decreased during the session as the user shows his ability to perform gestures well, or lack of such ability. Further, as the user tires during the course of a session, or increases in competence, the allowed shortcuts may be increased or decreased to correspond to his current state of ability.

[0111] Operation 908 depicts processing the data to determine an output corresponding to whether the user performed a shortcut of a gesture, the shortcut of the gesture corresponding to a full version of the gesture. In an embodiment, this output may comprise a confidence level that the gesture occurred. In an embodiment, this may comprise an indication as to whether a full version of a gesture or a gesture shortcut was observed.

[0112] Operation 910 depicts sending the output corresponding to the shortcut of the gesture to the application. Where the present operations are performed by the application, the output may be sent to a component of the application that takes processed user input and maps it to in-application actions.

[0113] Optional operation 912 depicts processing the data to determine an output corresponding to whether the user performed the full version of the gesture, and sending the output corresponding to the full version of the gesture to the application.

[0114] In an embodiment, the shortcut of the gesture comprises user movement that comprises a subset of user movement that comprises the full version of the gesture.

[0115] In an embodiment, the output corresponding to the shortcut of the gesture corresponds to a high likelihood that the user triggered the gesture, the output corresponding to the full version of the gesture corresponds to a high likelihood that the user triggered the gesture, and the application recognizes only one user gesture. Where the shortcut of the gesture comprises a subset of the full version of the gesture, when the user performs the full version of the gesture he will also perform the shortcut of the gesture. Thus, for one intended gesture input, two gestures may be recognized. In an embodiment, where both the shortcut of the gesture and the full version of the gesture are recognized within a prescribed period of time (one that may be gesture and/or user specific), only one is used as input and the other is disregarded.

[0116] In an embodiment, the output corresponding to the shortcut of the gesture corresponds to a high likelihood that the user triggered the gesture, the output corresponding to the full version of the gesture corresponds to a high likelihood that the user triggered the gesture, and the application uses an output corresponding to the full version of the gesture to add detail to the gesture. Where the shortcut of the gesture is recognized and processed (such as a corresponding animation or result is displayed on a display device) as an input by the application, and then the full version of the gesture is recognized while the shortcut of the gesture is being processed, output from the full version of the gesture may be used in the processing.

[0117] For example, a full version of a “jump” gesture may comprise the user jumping. A shortcut of that “jump” gesture may comprise the initial motions of the full version of the gesture – a crouch and rise – and output a confidence level that the gesture was performed. As a result of observing the shortcut of the jump gesture, the application may process this by displaying the user’s avatar as jumping. While this occurs, if the user completes the full version of the jump gesture by continuing to rise and leaving the ground with both feet, the application may use a height that the user physically jumps to display the avatar as jumping a corresponding height to add detail to the currently-processed jump shortcut. If the user performed only the shortcut of the jump gesture, the application may have the avatar jump a default height.

[0118] In an embodiment, where the user performs a shortcut of a gesture, it may correspond to lesser in-application accomplishment than if where the user performs the full version of the gesture. For instance, in a skateboarding game that ranks a user’s performance with points scored, where the user performs a given trick using the shortcut of a skateboarding trick gesture, the user may receive fewer points than had he performed the given trick using the full version of the skateboarding trick gesture.

## **Conclusion**

[0119] While the present disclosure has been described in connection with the preferred aspects, as illustrated in the various figures, it is understood that other similar aspects may be used or modifications and additions may be made to the described aspects for performing the same function of the present disclosure without deviating therefrom. Therefore, the present disclosure should not be limited to any single aspect, but rather construed in breadth and scope in accordance with the appended claims. For example, the various procedures described herein may be implemented with hardware or software, or a combination of both. Thus, the methods and apparatus of the disclosed embodiments, or

certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium. When the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus configured for

5 practicing the disclosed embodiments. In addition to the specific implementations explicitly set forth herein, other aspects and implementations will be apparent from consideration of the specification disclosed herein. It is intended that the specification and illustrated implementations be considered as examples only.

**What is Claimed:**

1. A method for using a gesture shortcut in a system that takes user (18) gestures as input to an application, comprising:

receiving data captured by a capture device (20), the data corresponding to a user-  
5 performed gesture (902);

processing the data to determine an output corresponding to whether the user performed a shortcut of a gesture, the shortcut of the gesture corresponding to a full version of the gesture (908); and

10 sending the output corresponding to the shortcut of the gesture to the application (910).

2. The method of claim 1, further comprising:

processing the data to determine an output corresponding to whether the user performed the full version of the gesture; and

15 sending the output corresponding to the full version of the gesture to the application.

3. The method of claim 2, wherein the shortcut of the gesture comprises user movement that comprises a subset of user movement that comprises the full version of the gesture.

4. The method of claim 3, wherein the output corresponding to the shortcut of the  
20 gesture corresponds to a high likelihood that the user triggered the gesture, the output corresponding to the full version of the gesture corresponds to a high likelihood that the user triggered the gesture, and the application recognizes that the user has performed only one user gesture.

5. The method of claim 3, wherein the output corresponding to the shortcut of the  
25 gesture corresponds to a high likelihood that the user triggered the gesture, the output corresponding to the full version of the gesture corresponds to a high likelihood that the user triggered the gesture, and the application uses an output corresponding to the full version of the gesture to add detail to the gesture.

6. The method of claim 1, further comprising:

30 receiving user input corresponding to defining the shortcut of the gesture.

7. The method of claim 1, wherein the shortcut of the gesture also corresponds to a full version of a second gesture.

8. The method of claim 1, wherein the shortcut of the gesture corresponds to a gesture filter and at least one parameter, the full version of the gesture corresponds to the gesture

filter and the at least one parameter, and the value of at least one parameter corresponding to the shortcut of the gesture differs from the value of at least one parameter corresponding to the full version of the gesture.

9. The method of claim 1, further comprising:

5 determining from the application to process the data with the shortcut of the gesture before processing the data to determine the output corresponding to whether the user performed the shortcut of the gesture.

10. The method of claim 9, wherein the operation of processing the data to determine an output corresponding to whether the user performed a shortcut of a gesture is  
10 performed only after receiving an indication from a user that such processing is desired.

11. The method of claim 1, wherein the user performing the shortcut of the gesture corresponds to a smaller achievement than the user performing the full version of the gesture.

12. A system for using a gesture shortcut in a system that takes user (18) gestures as  
15 input to an application, comprising:

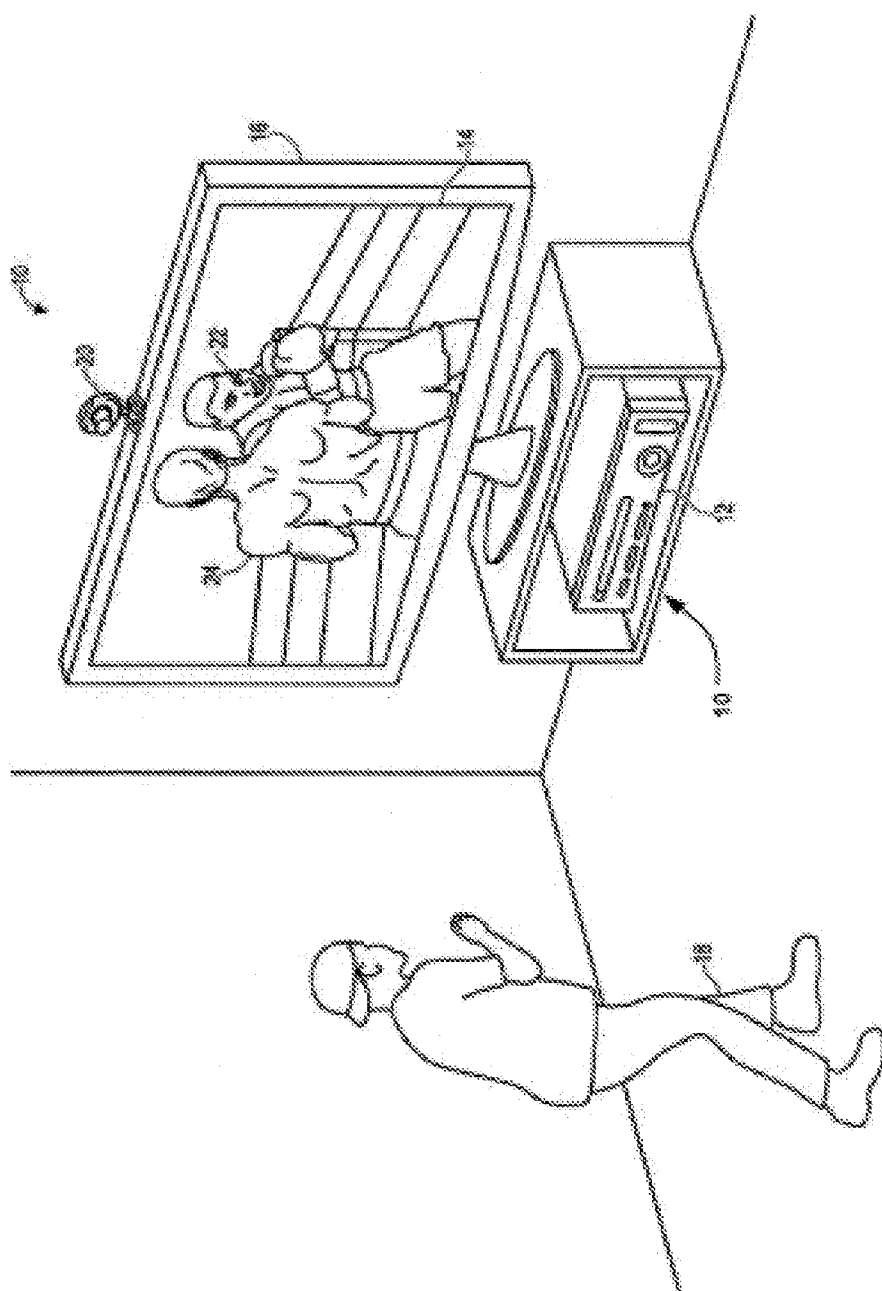
a capture device (20) for generating data corresponding to a user-performed gesture;

a processor (101) that receives the data captured by the capture device, processes the data to determine an output corresponding to whether the user performed a shortcut of  
20 a gesture, the shortcut of the gesture corresponding to a full version of the gesture, and sends the output corresponding to the shortcut of the gesture to the application.

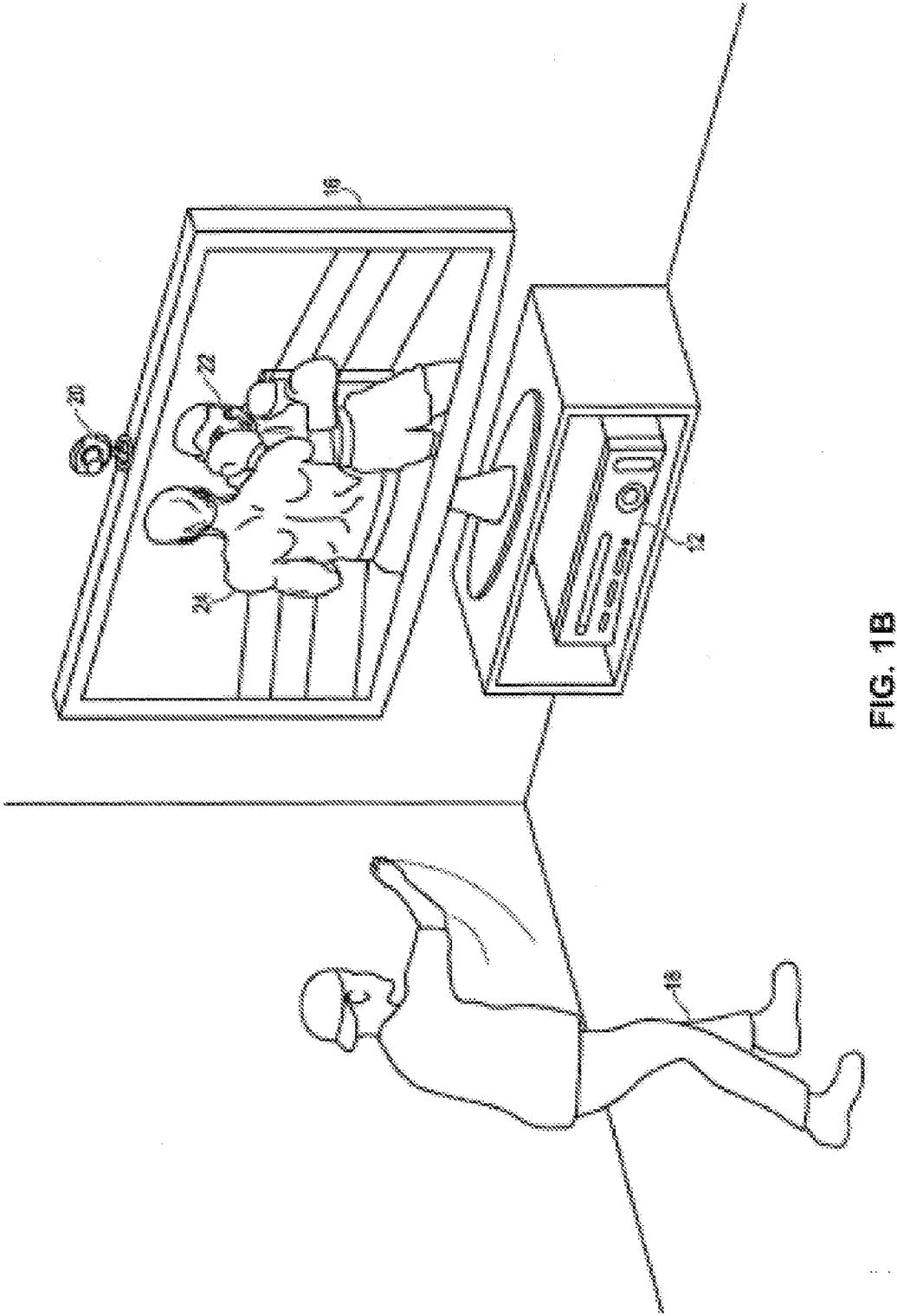
13. The system of claim 12, wherein the processor also processes the data to determine an output corresponding to whether the user performed the full version of the gesture, and sends the output corresponding to the full version of the gesture to the application.

25 14. The system of claim 13, wherein the shortcut of the gesture comprises user movement that comprises a subset of user movement that comprises the full version of the gesture.

15. The system of claim 13, wherein the output corresponding to the shortcut of the gesture corresponds to a high likelihood that the user triggered the gesture, the output  
30 corresponding to the full version of the gesture corresponds to a high likelihood that the user triggered the gesture, and the application recognizes only one user gesture.



**FIG. 1A**





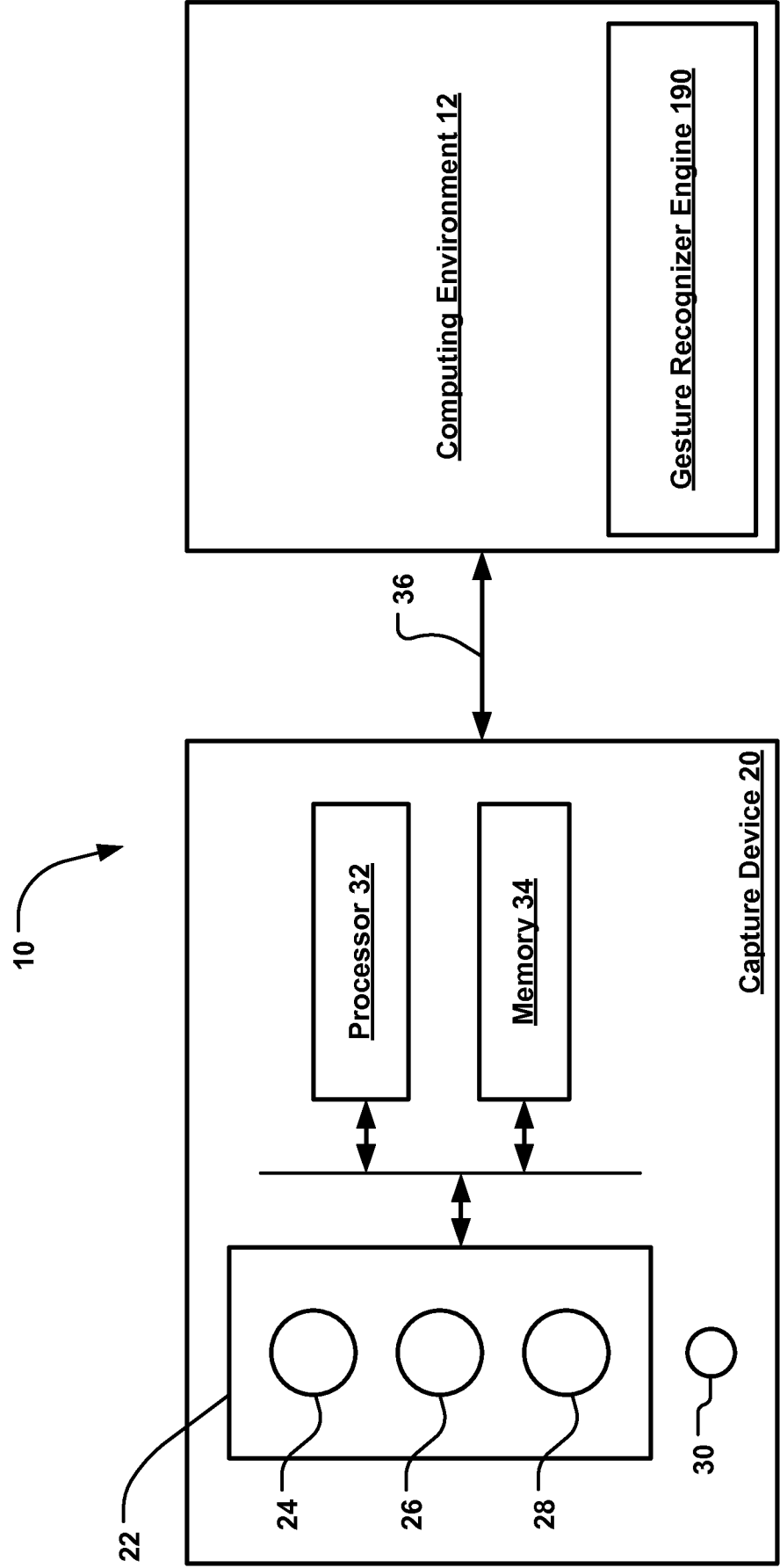


FIG. 2

4/12

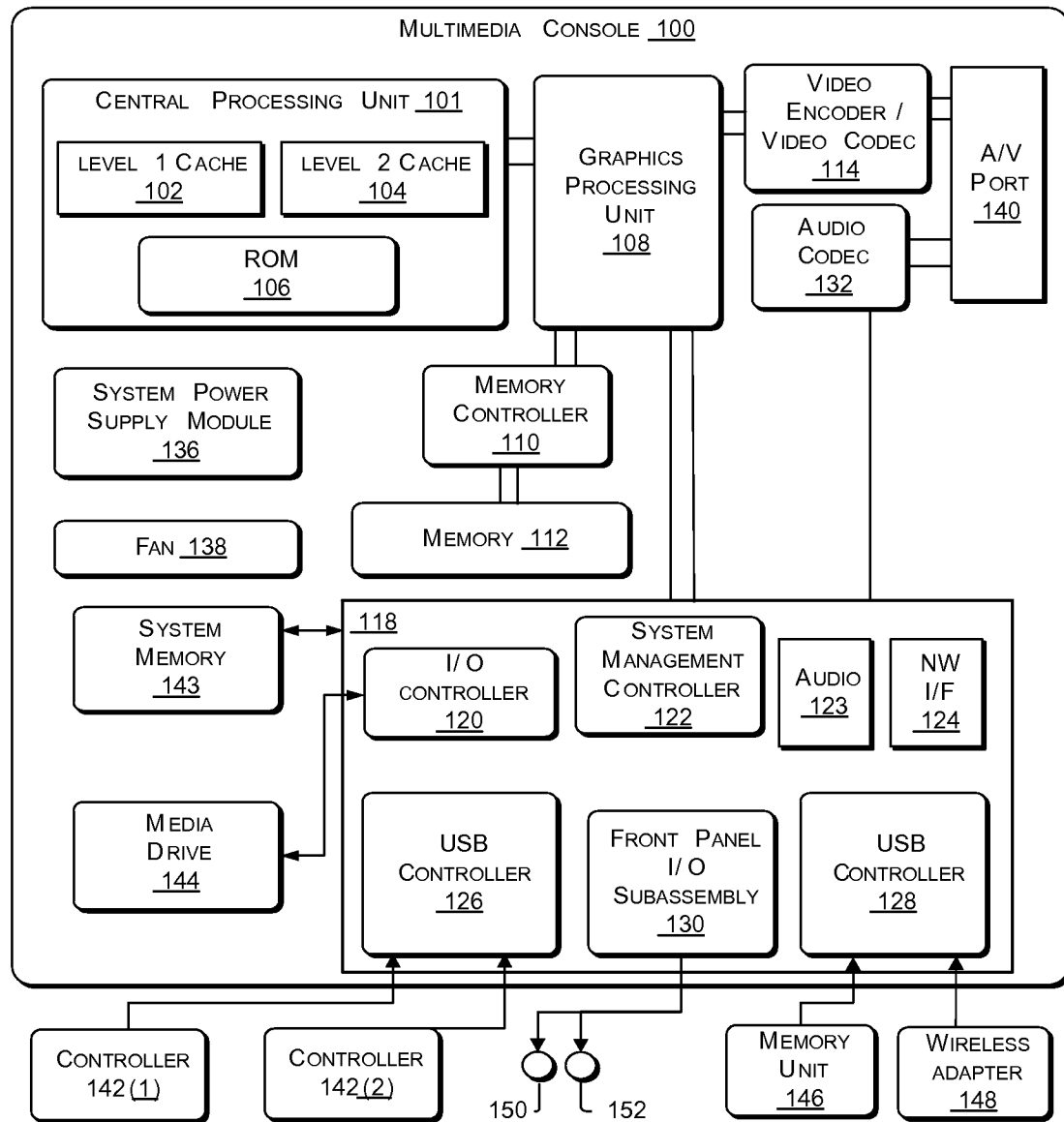
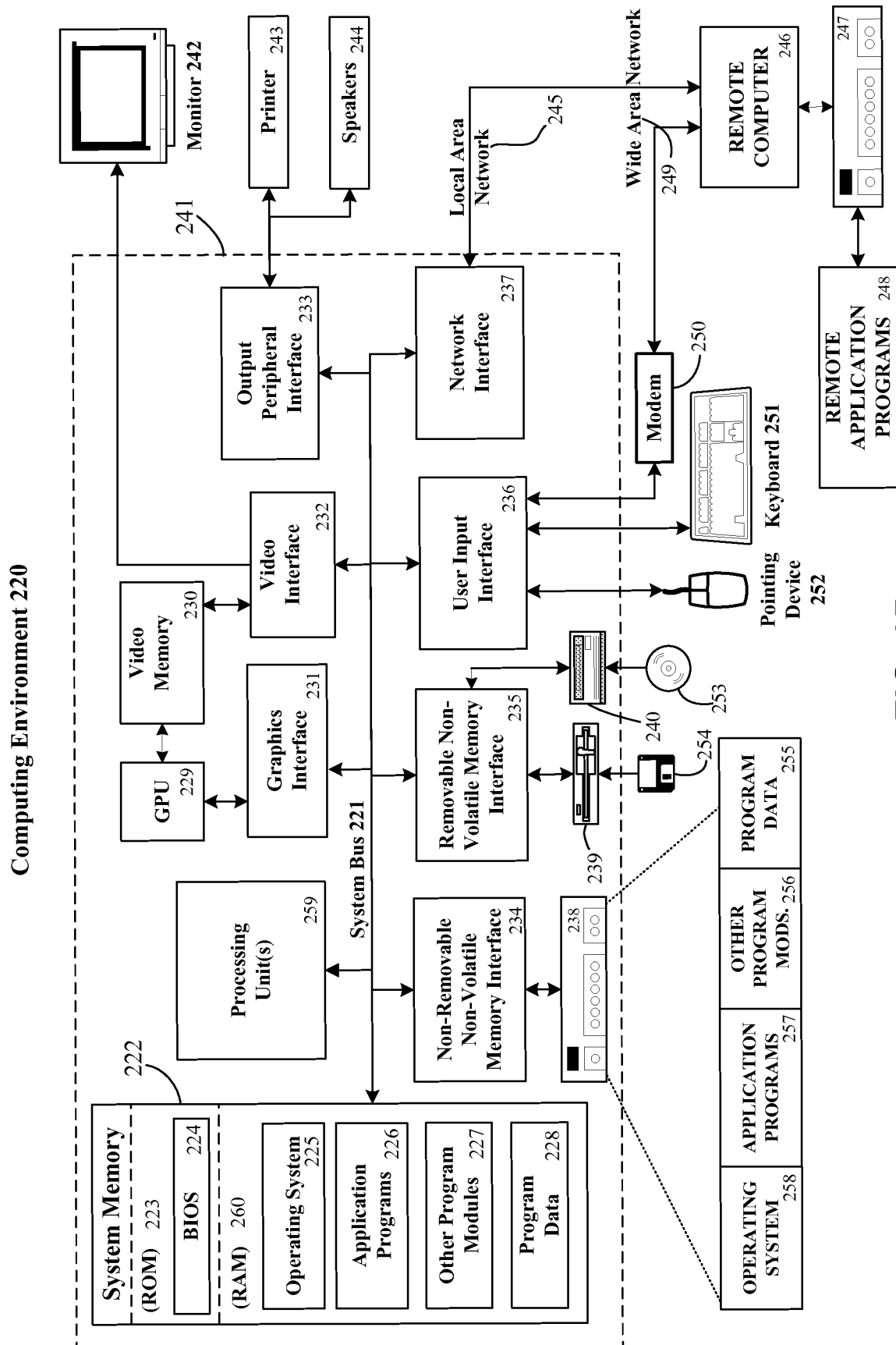


FIG. 3A

5/12



**FIG. 3B**

6/12

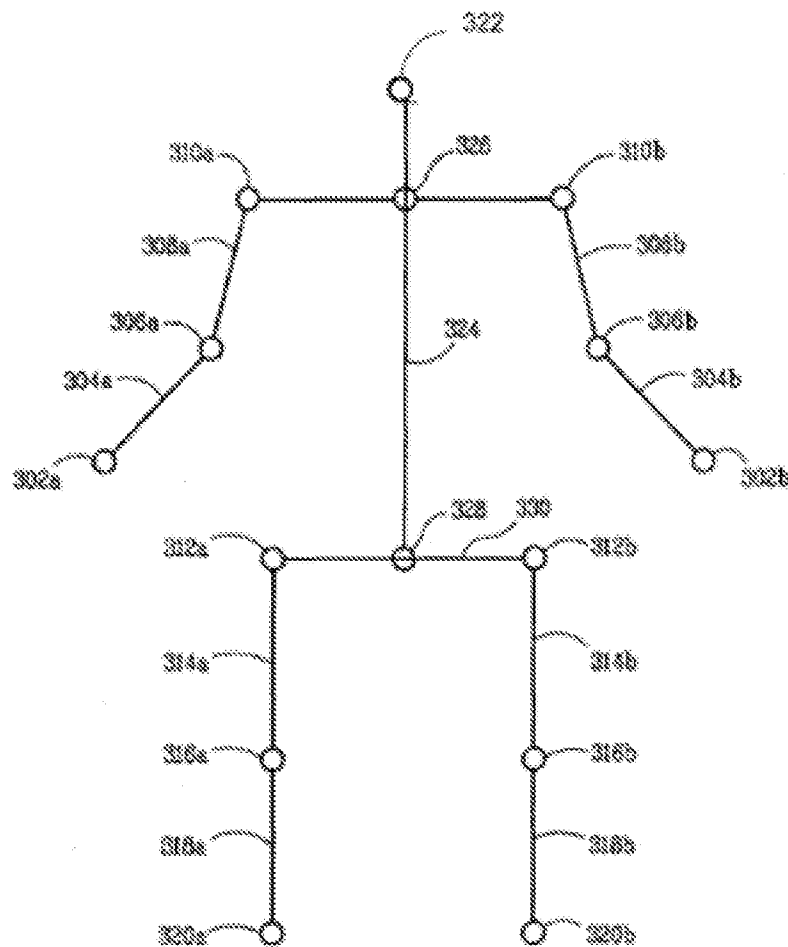


FIG. 4A

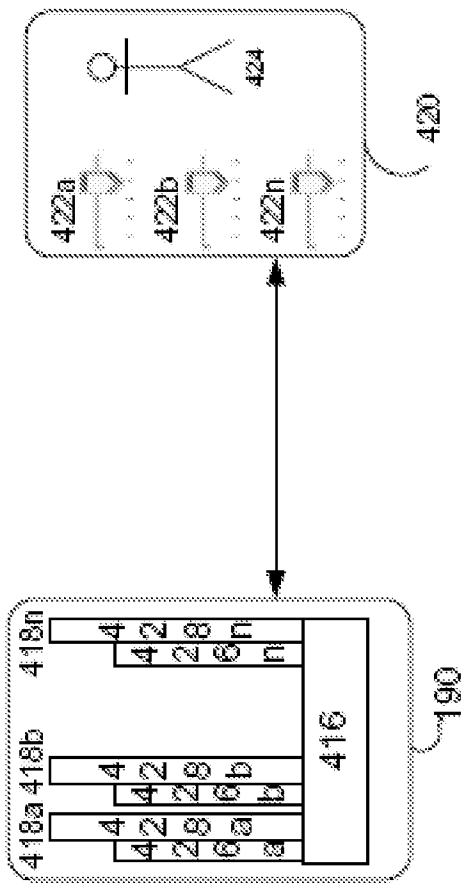


FIG. 4B

8/12

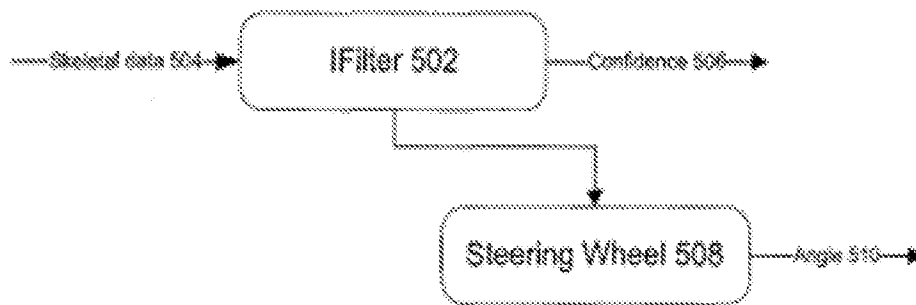


Fig. 5A

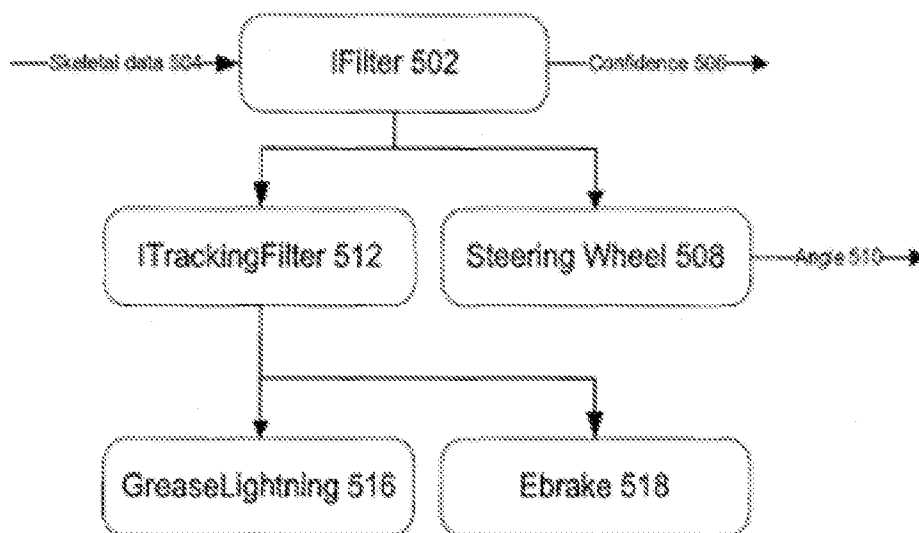


Fig. 5B

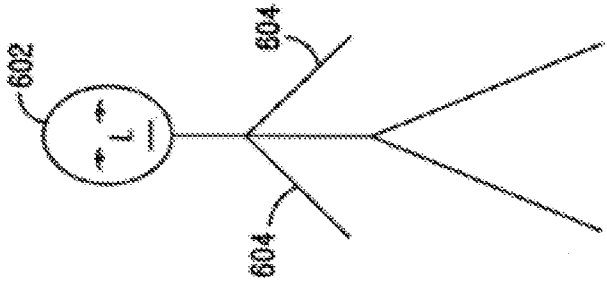


FIG. 6A

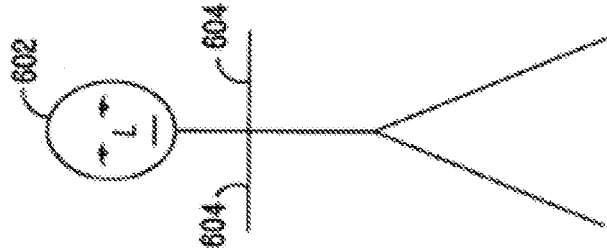


FIG. 6B

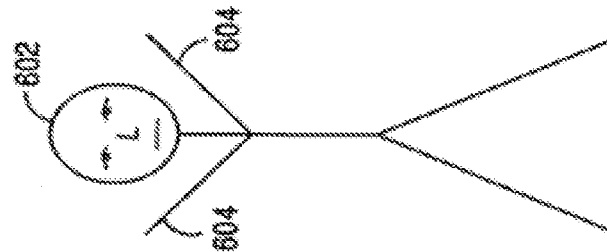


FIG. 6C

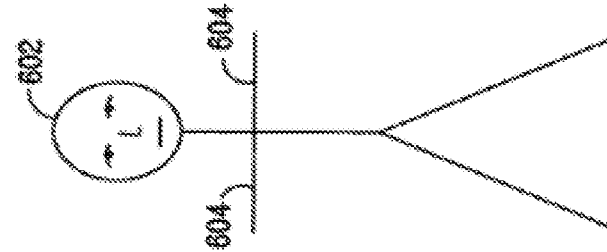


FIG. 6D

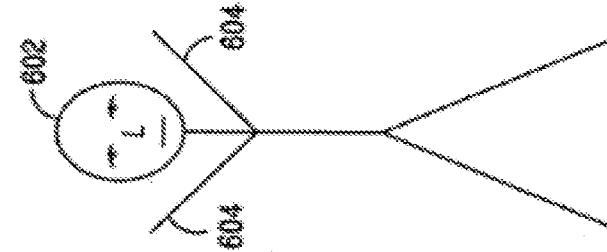


FIG. 6E

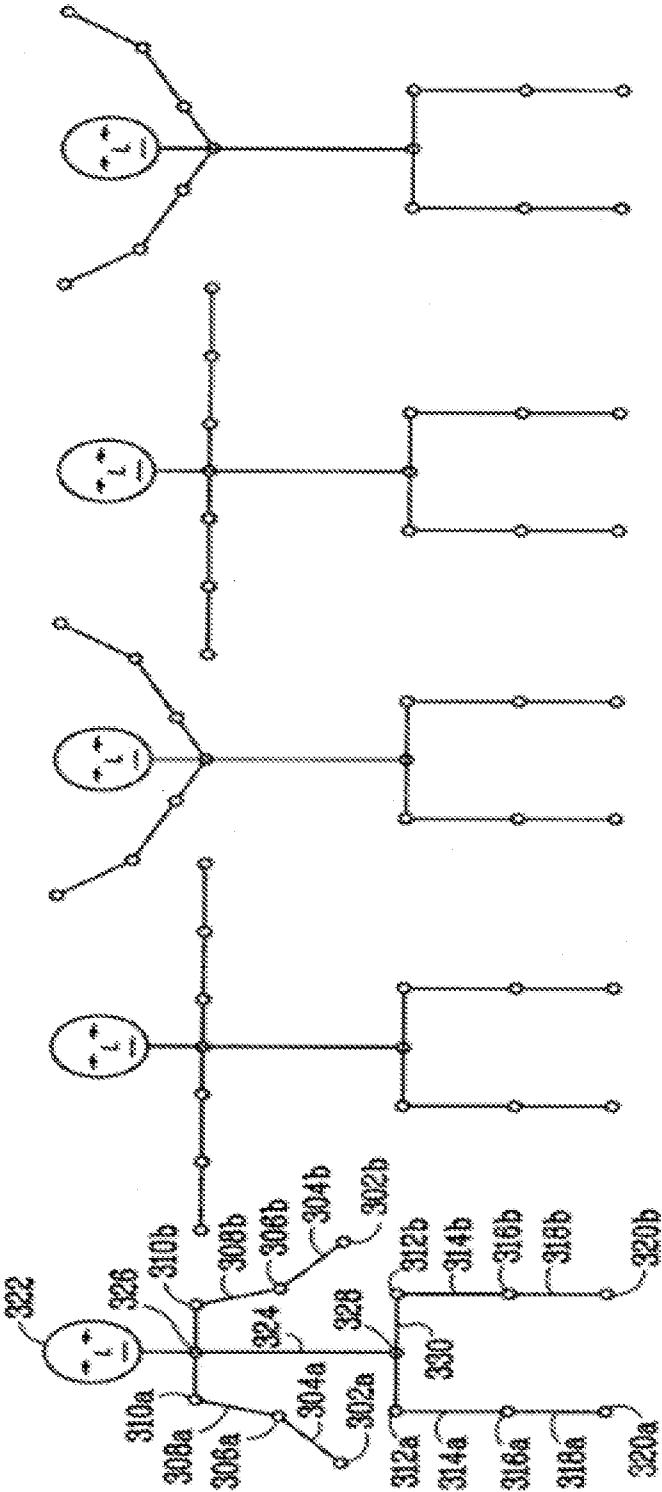


FIG. 7E

FIG. 7D

FIG. 7C

FIG. 7B

FIG. 7A



11/12

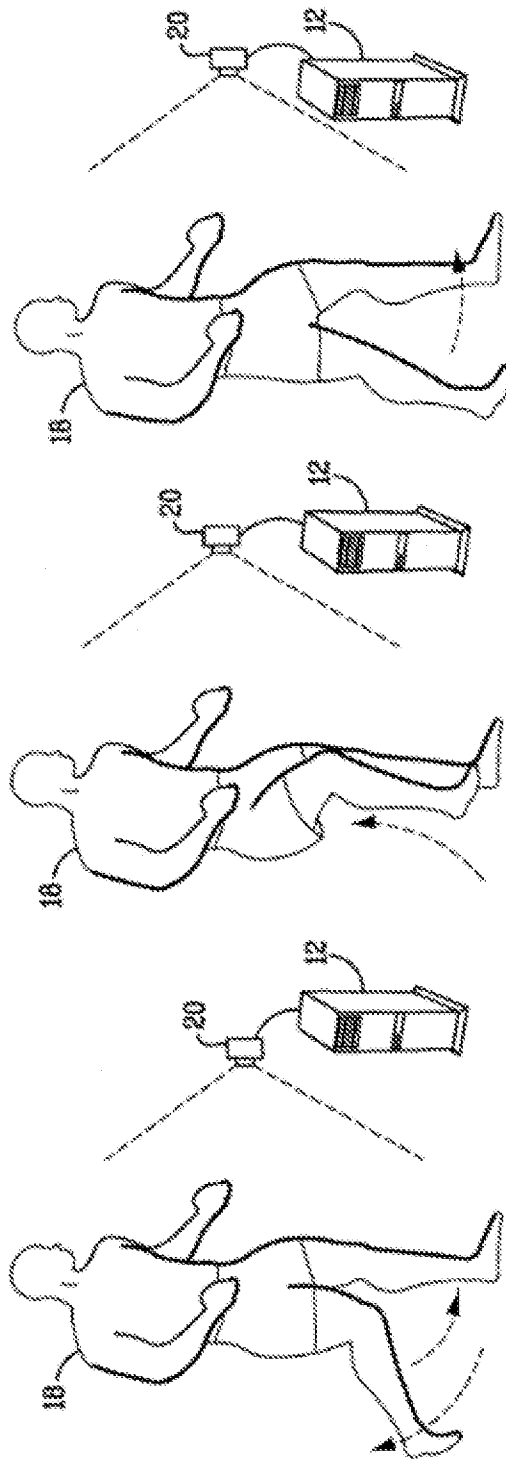
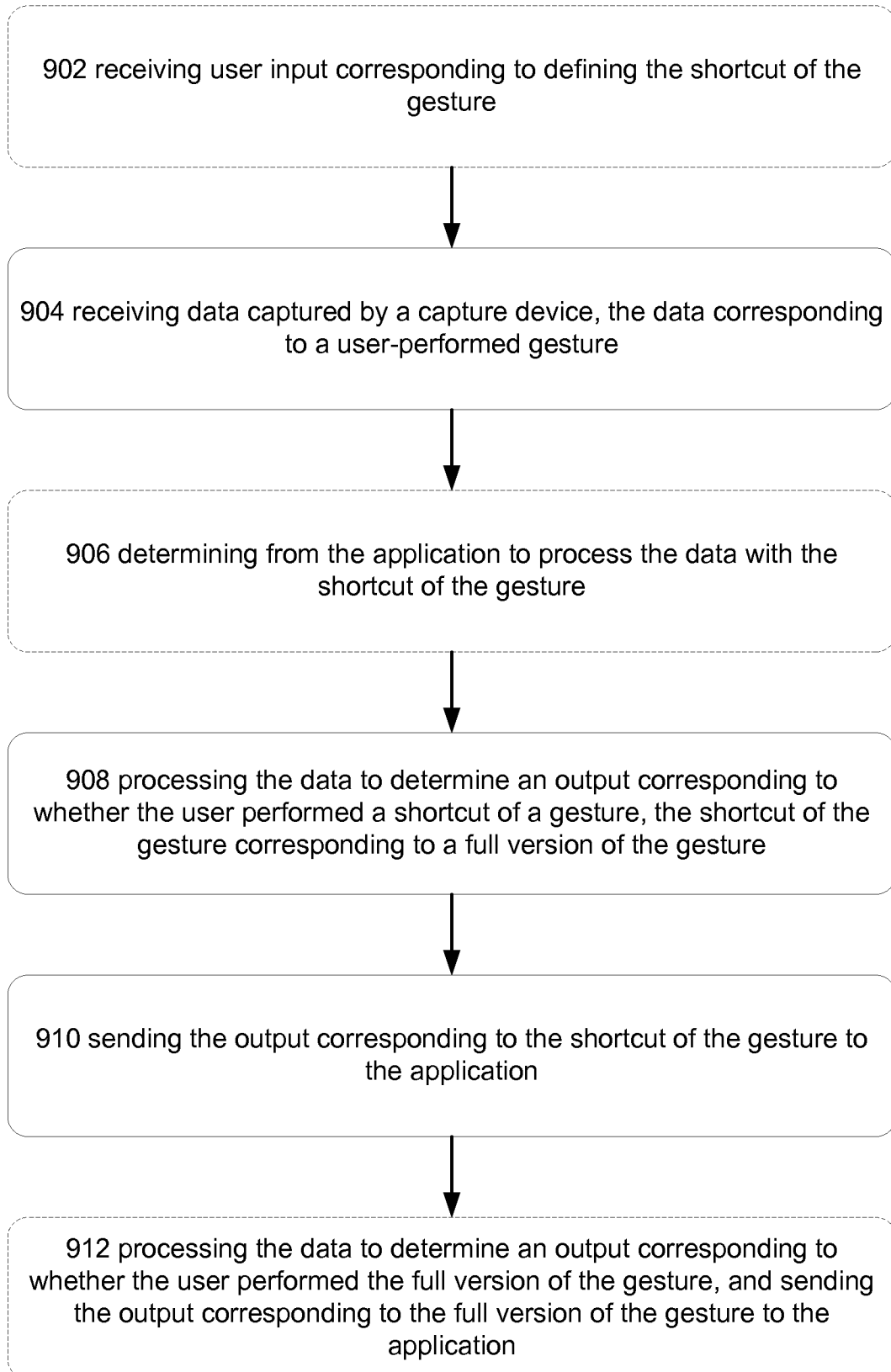


FIG. 8A

FIG. 8B

FIG. 8C

12/12

**Fig. 9**