

A METHOD AND SYSTEM FOR PROCESSING ELECTRONIC IMAGE CONTENT FOR DISPLAY

FIELD OF THE INVENTION

5

The present invention relates to a method and system for processing electronic image content for display, and particularly to an image processor; and to computer program code for performing this method. The present invention is of particular although not exclusive application in processing image data in the form of content displayed on a computer screen for display on a television, or other display device such as a projector, etc.

15

BACKGROUND OF THE INVENTION

Electronic image content is commonly displayed on a display device, such as a projector, television, etc, by transmitting the content either in a format suitable for receipt by the display device or via an intermediary device which typically receives raw electronic image content and outputs the content in a format suitable for display, such as VGA or HDMI for standard interfacing.

25

One existing intermediary device is a digital media receiver, for use within a home network to enable display of a home computer's contents on a television to enable image content to be enjoyed in a separate location to the computer, e.g. a lounge room, or for use in presenting a computer's contents on a more convenient viewing device, such as a projection screen. The digital media receiver enables content previously accessible on the computer to be displayed and accessed on any display device. However, while such a device enables the display of the home computer's contents, there are a number of problems associated with display quality and robustness.

35

Modern display devices, such as plasma, LCD screens and projectors, can display high definition image content which, when viewed as an incoming stream of data from a computer, have large bandwidth requirements. Compression techniques may be employed to reduce bandwidth requirements however these may reduce image quality and are both time consuming and processor intensive. This is especially evident when multiple computers are connected to the device and/or multiple display devices. A further technique employed is to reduce the number of image frames transmitted per second however this typically results in jerky images and/or audio being displayed.

SUMMARY OF THE INVENTION

According to one aspect of the invention there is provided a method of processing electronic image content for display, the method comprising:

a computing system receiving a first image comprising a first array of pixels;

said computing system receiving a second image comprising a second array of pixels;

a comparator of said computing system identifying which pixels of the second array are different from corresponding pixels of the first array by comparing said first array and said second array and outputting said different pixels of said second array to at least one image composer;

said image composer constructing a new image for display comprising said different pixels from said second array identified by said comparator and pixels of said first array complementary to said different pixels of the second array; and

outputting the new image for display.

The computing system may be a distributed computing system

and may include one or more computers. The may also include other devices with computing capabilities, such as a digital camera, PDA, mobile phone, etc.

- 5 In one embodiment, the image composer is connected to the computing system, which is generally a computer, over a telecommunications network. However, it will be appreciated by a person skilled in the art that the computing system may incorporate the display device.
- 10 Also, any one of the computers in a computing system may output different pixels as an image fragment to at least one image composer, where the image composer may be incorporated into the display device, or an intermediary image processing device as an image processor, remote from
- 15 the computer to receive image content for display on various types of display devices. In one example, the one computer outputting the image fragment is a source computer which may additionally output control information to control display of the image content. In this case,
- 20 the telecommunications network for transmitting image and control information may be a wired or wireless LAN and the communications protocol may be TCP/IP for reliable data transfer.
- 25 It is understood by those persons skilled in the art that the different pixels from the second array identified by the comparator form image data, in particular, an image fragment. Furthermore, it is understood that an image is formed from an array of pixels, each of which is assigned
- 30 a collection of bits dependent on pixel qualities such as colour and opacity. For example, an image may have 8 bits per pixel to assign these qualities. Thus, the image fragment only has those pixels found different from corresponding pixels in the first array and the
- 35 complementary pixels are those not found different from the first array. Typically, only a portion of the image changes between frames and thus, in preference, only that

portion is outputted via TCP/IP to the image composer.

It is also of benefit to authenticate receipt of data for security. Thus, the method further comprises the image
5 composer authenticating receipt to the computer when situated remotely. In the event the image composer is encompassed within the computing system, authentication may be performed between components of the system to ensure reliability. In another embodiment, all
10 communication of data between a remote image composer and the computing system may be performed via hand-shake authentication protocols for reliable data transfer.

It is to be understood by a person skilled in the art that
15 a display device may be referred to as a television in the specification but may include an LCD screen, projector, etc, all of which require video content, in particular the image component thereof, to be received in a standardised format using a standardised audio/video interface, such as
20 HDMI, VGA, component video, etc.

In addition, the skilled person will appreciate that the display device may also be a computer screen of a destination computer. In this case, the image content may
25 be received using a standardised interface such as VGA, or the destination computer may include the image composer to construct a new image for display on a connected computer screen using the received image fragment.

30 As described above, it is desirable to control display of the image content and, in an embodiment, control information is outputted in addition to the image data to control display of the image content on the one or more display devices. The control information may be used to
35 remotely control the display of the display device including pausing, resuming, starting and stopping display of the image content. Furthermore, the control

information may be employed to allow a source computer of the computing system adjust the display of the destination display device or devices, including adjusting the size or resolution of the display.

5

In another embodiment, the control information comprises the pixel array size of the first and second arrays. The pixel array size information facilitates the display device to change its display of different sized image content from toggled sources without a resizing delay.

10

It would also be desirable to receive audio and/or video content or data for display, in addition to the above described image content, in particular for the display of video clips. In this case, it is desirable to transmit audio, video and/or image content as streaming data across the telecommunications network to reduce bandwidth requirements. In one example, a video clip is displayed within a computer desktop image, which may be at a smaller display size than the full computer screen. In this example, the difference in pixels from a first array and a second array separated in time may apply only to the changes in the video clip being displayed rather than the rest of the computer desktop image. Thus, a skilled person will appreciate that one method of displaying the video clip is via the computer identifying which pixels of the portion of the screen displaying the video clip have changed at a suitably high refresh rate. However, such a high refresh rate would require a high bandwidth requirement and may thus be jerky. Another method may be to output the different pixels of the desktop image excluding the video clip pixels to the image composer, in addition with audio and video data of the video clip as streaming data. In this method, the audio and video data is overlaid with the data comprising the different pixels to display the image content including the video clip without lag and without need for such a high refresh rate

15

20

25

30

35

thus further reducing bandwidth requirements.

In another example, the streaming data is outputted to a memory intermediate the image composer and typically
5 located in a server. The stored data may be compressed and the memory may be random access memory to reduce processing time. The server may also be intermediate more than one image composer so that more than one image composer can receive the same streaming data for display.
10 Further, as the streaming data is recorded onto the memory, individual display devices may be customised to display different portions of the streaming data at any one time. Alternatively, the memory and image composer may be located in each image processing device so that the
15 streaming data is stored locally.

In the embodiment where there is more than one image composer, a skilled person will appreciate that there may also be at least one display device corresponding to that
20 image composer, in particular corresponding to an image composer which is located in an image processing device. In this case, output of the different pixels identified by said comparator is synchronised to each image composer to provide synchronised display on the corresponding display
25 devices. For example, one method includes a source computer displaying and controlling the display of image content across multiple display devices employing multiple image composers to process the image and control data for each corresponding display devices.

30 The method may also include monitoring the CPU utilisation rate of the computer to avoid reducing resources available to processes other than those for processing image data. In one embodiment, the computer suspends the comparator
35 from identifying which pixels of the second array are different from corresponding pixels of the first array by comparing the first array and the second array and outputs

the different pixels of the second array to the image composer when a CPU utilisation threshold is exceeded. In one embodiment, the threshold is 30%.

5 According to another aspect of the present invention there is provide a system for processing electronic image content for display, the system comprising:

a computing system arranged to receive a first image comprising a first array of pixels and a second image
10 comprising a second array of pixels;

a comparator of said computing system arranged to identify which pixels of the second array are different from corresponding pixels of the first array by comparing said first array and said second array and output said
15 different pixels of said second array to at least one image composer,

whereby said image composer is arranged to construct a new image for display comprising said different pixels from said second array identified by said comparator and
20 pixels of said first array complementary to said different pixels of the second array and output the new image for display.

According to another aspect of the present invention there
25 is provided a device for processing electronic image content for display, the device comprising:

an image composer arranged to:

receive a first image comprising a first array
of pixels;

30 receive pixels of a second image comprising a second array of pixels which differ from corresponding pixels of the first array;

construct a new image for display comprising said different pixels from said second array
35 identified by said comparator and pixels of said first array complementary to said different pixels of the second array; and

output the new image for display.

According to another aspect of the present invention there is provided a computer program code which when executed
5 implements the above method.

According to another aspect of the present invention there is provided a computer readable medium comprising the
above program code.

10 According to another aspect of the present invention there is provided a data signal comprising the above program code.

15 BRIEF DESCRIPTION OF THE DRAWINGS

In order that the invention be more clearly ascertained, embodiments will now be described, by way of example, with reference to the accompanying drawings, in which:

20 Figure 1 is a schematic view of a system for processing electronic image content for display according to an embodiment of the invention;

Figure 2 is a flow chart of the method implemented by the system of Figure 1 according to the present invention;

25 Figure 3 is a flow chart of the method of Figure 2 showing information sent over a network between a computing, image processing device and a display device;

Figure 4A is a schematic view of the system of Figure 1;

30 Figure 4B is a schematic view of the system of figure 1;

Figure 4C is a schematic view of the system of Figure 1;

35 Figure 5 is a schematic view of the system of Figure 1 showing multiple computers connected across a network to multiple devices for processing electronic image content which, in turn, is connected to multiple display devices;

Figure 6 is a schematic view of the system of Figure 5 showing a server intermediate the devices incorporating a memory for access by the corresponding devices;

Figure 7 is a schematic view of the system of Figure 5 showing the processing device with a memory; and

Figure 8 is a state diagram of the system of Figure 1.

DETAILED DESCRIPTION

10

According to an embodiment of the present invention, there is provided a system 10 for processing electronic image content for display including a computing system 12 having a comparator 14 for receiving a first and a second image, and arranged to identify which pixels of the first image differ from those of the second image and output the different pixels as an image fragment to an image composer 16. As described, the computing system may include one or more computers, or devices having computing capabilities such as a digital camera, PDA, mobile phone, etc, and thus includes the components of a display, processor, input device, hard-drive, etc. Furthermore, as the image composer 16 may be arranged remote from the computing system and the comparator, it will be appreciated by a person skilled in the art that the image composer 16 would include similar hardware, such as a processor, etc, to construct an image for display. If the image composer were located within the computing system then hardware could be shared. Furthermore, if the image composer were located within a display device, such as a projector, hardware such a power supply and network ports could also be shared.

In an embodiment, the image composer 16 constructs a new image for display by a display device, such as a television, from the received image fragment from the comparator 14 and the pixels of the first image

complementary in position in the array of pixels to the image fragment. Thus, the image composer 16 requires additional hardware to receive and forward data, particularly to receive image data and output display data
5 in a form readily accepted by a television or similar device, such as HDMI. The display may also be a plasma or LCD screen, projector, hand-held viewing device, etc.

Figure 2 is a flow chart of the method 18 implemented by
10 the system of processing electronic image content. The method 18 includes initially receiving 20 a first image comprising a first array of pixels. As described, images typically are formed from an array of pixels, each of which is assigned a collection of bits dependent on pixel
15 qualities. Also, it will be appreciated by those skilled in the art that the electronic image content for display may be the image displayed on a source computer screen of the computing system or a screen of a source electronic device such as a PDA, which may include static electronic
20 images, such as desktop images, or a video clip shown within the desktop including both video and audio data. Also, the desktop may display just an audio clip being played and in this case audio data is outputted to the display device for display. In the case where the image
25 content includes a sequence of static images or frames, such as the display of the computer screen, only a portion of the image changes between sequential frames.

In an embodiment, the method 18 further includes receiving
30 22 a second image comprising a second array of pixels. In an example, the first and second images may be received by the computer. Further, the computer may include a comparator for identifying 24 which pixels of the secondary array of pixels differ from corresponding pixels
35 of the first array. That is, comparing pixels of corresponding co-ordinates of each array to identify which pixels differ. Typically, the array size of the first and

second image is the same however, in the case where they are not, a translation algorithm may be implemented by the computer to enlarge or shrink an array to ensure an accurate comparison can be made between corresponding
5 pixels and the different pixels identified.

The comparator of the computing system may then output 26 the pixels found different to be used in constructing 28 a new image for display. In one example, the different
10 pixels are outputted across a network to an image composer remote from the computing system. However, in another example, the image composer is incorporated within the computing system. The image composer may perform the step of constructing a new image for display, the new image
15 comprising both the different pixels and pixels of the first array complementary to the different pixels. In one example the new image is formed by overlaying the different pixels into their corresponding position in the array of pixels of the first image however it is envisaged
20 that other methods may be employed to construct the new image such as combining the different pixels with pixels of the first image. In any event, the new image is outputted 30 for display. The method may also include outputting the new image for display to any number of
25 display devices, such as where multiple televisions are located in multiple viewing locations such lounge rooms and bedroom within a house.

The method of processing electronic image content may also
30 include outputting the different pixels across a network to more than one image composer remote from the computing system. An example of such a method being employed is in a teaching environment where a lecturer with a source computer wishes to display and control electronic image
35 content, in the form of desktop content, to be transmitted wirelessly, to a plurality of students' displays, such as computer screens, and projectors.

The method of processing electronic image content is also shown as a flow chart 32 in Figure 3, showing the image content, sent over a telecommunications network between a
5 computing system (source computer) 34 and one image processing device (image processor) 36, and finally to a display device (television) 38 for display. In the embodiment shown, the image composer 42 is remote from the computer and its comparator 40, and is incorporated within
10 an image processor 36. It is envisaged the image processor 36 includes features necessary for it to function independently, such as a processor and power supply, and features to enable it to communicate across the network and to any display device such as suitable
15 ports, interfaces, etc.

The flow chart 32 also shows the method implemented by the system of Figure 1 over time. In one example, initially, the computer 34 receives a first and second image of the
20 type described above but, in this case, a comparator 40 identifies that the first image is a null image and thus the different pixels outputted correspond to the second image received. In the example shown, image 44 is outputted to the image processor 36 which, in turn,
25 recognising that the complementary pixels are null, outputs the image for display by the television 38. An acknowledgement packet acknowledging either receipt of the image or a successful display of the image may be returned by the image processor 36 if required by the method.

30 In the example shown, the second image to be received by the computer forms the first image 44 to be subsequently received by the comparator 40. Also received is a subsequent second image 46. The comparator 40 may then
35 identify which pixels are different between the two images and output only the different pixels 48 as an image fragment across the network to the image processor 36 and

thus to the incorporated image composer 42, rather than outputting an image with a full array of pixels which has a larger packet size. Receipt of the image fragment 48 may be acknowledged if required by the method. By way of
5 example, a hand-shake authentication protocol may be employed between the image processor 36 and the computer 34, where packets of data are not transferred until an acknowledgement is received.

10 It will be appreciated by those skilled in the art that the telecommunication networks may be a wired or wireless LAN, and the communication protocol is typically TCP/IP. This protocol enables the image processor to be located anywhere remote from the computer, not just within the
15 same house in the example of a home network or in the same university in the example of a teaching environment. Also, it will be further appreciated that other networks and protocols may be employed such as UDP.

20 The image composer 42 may receive the image fragment 48 and construct a new image for display comprising the image fragment 48 and pixels 50 of the first array complementary in position in the array of pixels to those of the image fragment. The new image 52 is then outputted for display
25 to the television 38 as image data in the form suitable for the television, such as VGA or HDMI, over a suitable cable.

It will also be appreciated by those skilled in the art
30 that the image data outputted to the display device may also be outputted over a wired or wireless telecommunications network allowing the display device or devices to be located anywhere.

35 In another example, the system of processing electronic image content is further described by reference to Figures 4A, 4B and 4C. Thus, referring to Figure 4A, it can be

seen from the schematic view that the system 54 for processing electronic image content may be embodied in at a computer 56 for processing the electronic image content. In this embodiment, the computer 56 includes both the
5 comparator 14 arranged to identify which pixels of the first image differ from those of the second image and output the different pixels as an image fragment, and the image composer 16 arranged to construct a new image for display by the display device 58 and the display screen
10 59, which could be a television screen for example, from the received image fragment from the comparator and the pixels of the first image complementary in position in the array of pixels to the image fragment. The image composer 16, and thus the computer 56, outputs the new image to the
15 television 59 for display.

A person skilled in the art will appreciate that an example of another embodiment of the system shown in Figure 4A may be when an image processor is employed as a
20 stand alone system for processing and displaying electronic image content. For example, a user wishing to display image content from a recorded disk may insert the disk directly into the image processor and have the image content displayed as the image processor can incorporate
25 computer processing capabilities.

A further embodiment is shown in Figure 4B. It can be seen from this figure that the system 60 for processing image content also includes the computer 56 with the
30 comparator 14 however the image composer 16 is located remote from the computer and is incorporated within the display device 58. In this embodiment, the display device may be a television with a screen 59 and it is envisaged that if there is more than one display device, each
35 display device has the processing capability to construct a new image for display using separate image composers 16. Also, in this embodiment, communication between the

comparator 14 and the image composer 16, located within the television, may be over a telecommunications network of the type described above. Further, a person skilled in the art will appreciate that there may be more than one
5 computer 56, each with its own comparator 14, to identify and output the image fragment to the image composer 16 and thus the television 58. In such an example where the television incorporates the image composer 16 and, accordingly, suitable hardware and software is provided to
10 receive the image fragment from one or more of the connected computers across the network so that a new image can be constructed for display on the screen 59.

A still further embodiment is shown in Figure 4C. It can
15 be seen from the figure that that the system 62 for processing image content may include the computer 56 with the comparator 14 arranged to identify the different pixels, as described above, as an image fragment. Further, it can be seen that the comparator 14 outputs the
20 image fragment for transmission, via the capabilities of the computer 56, over a telecommunications network to the image composer 16 located within a stand-alone image processor 64. The image composer 16 then constructs the new image for display on the screen 59 of the display
25 device 58 using the same method as described above.

The embodiment shown in Figure 4C is shown in more detail in Figure 5, where it can be seen that there may be more than one computer 56, in system 66, to provide image
30 content to more than one image processor 64, over a telecommunications network 68, which may be displayed on more than one display device, such as the television or computer screen 59 and projector 70. The network is shown as an Internet cloud, but may be a LAN as described above.
35 It can be seen that the computer 56 displays an image on its computer screen 72 as desktop content and that this image is desired to be viewed on the user screen 59, and

via the projector 70, using the method implemented by the above described system of processing image content.

5 In one example, the image processor 64 may output audio content, by streaming the sequential images with audio data to the television screen 59. Thus, in an example, in addition to the computer desktop content images being outputted, audio data is outputted from the computer to the display device 58 for receipt by its speakers.

10

In another example, the image processor 64 may output video content in addition to image content, by streaming the sequential images with audio and video data to the display device. For example, the source computer may wish
15 to display a video clip on the display device. In this case, the video clip audio and video data is transmitted to the image composer 64 across the network 68 as streaming data. In the example where the video clip is displayed within the source computer desktop image on a
20 computer screen 72, it may be displayed at a smaller display size than the computer screen. In this case, the different pixels identified by the comparator 14 of the computer apply only to the portion of the desktop image excluding the video clip. Thus, in the case where the
25 video clip is full screen, the comparator outputs a null difference. In addition, the computer may output control information with the streaming audio and video data for the video clip to control the display of the video clip on the display device, such as pause, play, etc.

30

In one example, the source computer 56 and image processor 64 run applications to communicate information from a custom media player on the source computer which processes both commands and video clip data to output audio and
35 video data as streaming data. By way of example, streaming data may be communicated across the network 68 using known communication channels. The commands and

player data from the source computer are then sent in a format that is understandable by the image processor and ultimately the display device via these channels. Also, once communication is established, there is continuous
5 communication to keep the source computer and image processor synchronised.

One part of the application, which may run on either the source computer or image processor, acts as an agent,
10 which receives commands and player synchronisation data in the first instance. In this example, the agent resides on the source computer and will control the video fragment data within the image environment of the source computer desktop. If a command is received by the source computer,
15 such as connect, pause, play, resume, it acts on the player accordingly as these commands are not player controls, but are commands for the corresponding display devices. For example, in the example where the display device is a projector, if a user wishes to pause the
20 projection, then the agent will control the video fragment data accordingly. Other than these projection commands, the agent may also receive player related commands such as resize, mute, volume change and so on, which it also passes to the agent to act upon.

25 In the above example, the video fragment itself may react to above mentioned agent's controls. In this case, the video fragment is overlaid with the image fragment identified from the image environment of the source
30 computer by the comparator 14. The video fragment is comprised of data to enable both playback of the video and audio and data to synchronise data with an application running on an image processor 64. In one example, the image processor 64 is located within a user computer, such
35 as a laptop, and in another example, the image processor 64 is located remote from the laptop, which corresponds to a display device, and the laptop includes a laptop screen

59. Thus, it can be seen that the synchronising data may be used to translate resolution where the resolution of the source computer and the user laptop differ so that the image can be resized accordingly. The video fragment or clip, which may be overlaid on the image environment of the display screen, can also be resized relative to the image environment to provide a displayed image consistent with the source desktop image. The skilled person will also appreciate that further algorithms and modules may be required to implement this step.

In another example, the image processor 64 may be controlled and remotely operated via communication between software applications running on an operating system on the computer 56 and/or the image processor 64. In this example, the display device 58 and screen 59 may be remotely operated using a desired communication channel. For example, a television may receive basic operational ASCII commands like turn-on and turn-off. Also, the television may be remotely controlled using a controlling application on the computer 56 which outputs control data, in addition to the above streaming image and audio/video data, to the image processor 64. The control data, image and audio/video data may be bundled together in packets to be transmitted across the telecommunications network using TCP/IP and may further be bundled with authentication protocols to ensure secure transfer of data. In one example, hand shake protocols for all communications across the telecommunications network are used to reduce instances of unauthorised use and reduce risk of data piracy.

A still further embodiment is shown in Figure 6. It can be seen from the figure that intermediate the image processor 64 is a memory 72 for storing the streaming image and/or audio and video data for retrieval by at least one image processor 64. In the embodiment shown,

the memory is located in a server 73 connected to the computing system and thus source computer over a telecommunications network 68. The client-server arrangement of the server 73 and image processors 64 (clients) allows for each client to individually control the display of the received streaming data. An alternative embodiment is shown in Figure 7 where the image processor 64 includes the memory 72 to store the received streaming data upon the user's request for later retrieval. It will be appreciated by a person skilled in the art that the memory 72 may be located remotely from the image processor 64, for example the memory 72 may be incorporated in a stand-alone hard-drive.

In addition, the computer 56 may run further applications, such as a custom media player to provide an intuitive user interface to display image content on a display device. The media player may also have a function to enable recording of the image content displayed and in this case the memory is used for recording.

By way of example, the custom media player may be implemented with software operating on both the source computer 56 and the image processors 64 in a client-server arrangement, where the source computer acts as a server. However, it is envisaged, such as the example above, that the server may be distinct from the source computer. In this example, the display screen 59 and projector 70 display video and audio clips from the source computer after receiving streamed image data and audio and video data at each image processor 64. This client - server arrangement also provides various other functionalities such as controlling a remote media player from a server player. For instance, controlling functions such as stop, mute, play, pause, close, etc, may be transmitted from the server to control display on the client display screens. Further, connections to a particular client player running

on a client image processor or a group of client players can also be established from the server to individually control each client player.

5 In the above example, the custom media player server component is divided into three main domains, namely, content transmission, control transmission and connection transmission. There are two channels for communication from server to client, one for the content transmission
10 and the other is for control and connection transmission. The content transmission is the actual video, audio or both being streamed from server to client component. Controls like mute, stop, pause, play, resize, relocate, skew, enlarge, close, open, etc, can be established
15 remotely from the server component. Connectivity controls like connect, disconnect, pause and un-pause may also be established from one server to many other servers running the custom media player, each of which can be paused, connected, un-paused and disconnected at any point of
20 time, without disturbing the other client players. The video and audio content is transmitted via VLC streaming modules and, when there is a change in the user interface of the server component, a data packet including the control transmission is sent to the connected custom media
25 player clients via remote interfaces to be synchronised. Furthermore, each synchronisation control transmission packet and connectivity transmission packets are sent to all connected clients. These are sent in addition to the image content and control transmission packets and enable
30 a server player to start a connection with a particular client or group of clients. Thus, it will be appreciated by the skilled person that these commands include the following high priority commands: connect, disconnect, pause and resume.

35 Further, the client component of the above example typically operates on a remote system which will be

passive until it receives any information from a server. An active communication between the connected client and server components is maintained after a connection is initially established and each client receives three types
5 of data from the server, namely, content transmission, control transmission and connection transmission. Also, each client operates on its data without disturbing the other clients. The first information a client receives is connection transmission, which provides information
10 regarding the type of connection and information regarding the server to which a connection has been established. Other connectivity commands also include disconnect, pause and resume. The second information received is the content transmission, which is the actual video and audio
15 content, being streamed from the server. Lastly, the control commands are received from the server in order to synchronise the clients. For example, volume control, stop, play, pause, resume, close, resize, relocate, full screen are received at each client to completely control
20 the display from the server and thus source computer.

In a further example involving multiple computers 56, the above control information may include information detailing the size of each computer image to be displayed.
25 Distinct finite automata algorithms may be used to decide the static desktop image content's size when the connections are toggled between each source computer and the image processor 64. If, for example, a change of state occurs, such as pause, start, stop, resume, etc, the
30 image processing algorithms send image data from each computer across to the image processor 64 only when a change in desktop content is detected and not otherwise, thus avoiding unnecessary usage of network bandwidth. In addition, the distinct finite automata algorithms may also
35 decide the static computer desktop image size when connections are toggled between each computer and the image processor without any disconnections, hence avoiding

connectivity latency. This example is shown in the case where each computer is connected to multiple image processors at the same time, making the connectivity topology of the system Many-to-Many. That is, each image processor can be connected to one single output display device, where it can receive image content as input from many source computers at a time, and each source computer can be connected to multiple image processors at a time, making the connectivity topology that of Many-to-Many. If this condition exists, the choice to make any one of the computers the primary source or destination can be toggled using the above mentioned application, on the fly, in real time. Also, at any point of time, the display or projection may be paused, resumed, started and stopped on the fly.

Figure 8 shows a state diagram showing an implementation of the method including outputting control information, in addition to image data, to control one or more display devices in a Many-to-Many topology example. Here it can be seen that distinct finite automata algorithms decide the static desktop image which has to be resized when the following occurs: connections are toggled between the source computers and/or image processors on the fly without any disconnections; or a change in states like pause, start, stop, resume, etc, occurs.

The implementation of the finite automata may be implemented using the state design pattern in any suitable computer language.

A description of the finite automata is as follows:

The value *n* at each state describes if the image has to be reset to full image or not, where 0 is false and 1 is true.

Q is a set of finite states; E is finite set of symbols, d
a function from $Q \times E$ to Q .

$Q = \{S0, SC, SPS, NC, SPA, SRS, SRA, URS, UPA, URA, UPS\}$

5 $E = \{DS, DA, CS, CA, PS, PA, RS, RA\}$

q_0 is the initial state.

Legend

S0 - Start State

10 SC - Synchronised connect

SPS - Synchronised Pause Single

NC - No Connection

SPA - Synchronised Pause All

SRA - Synchronised Restart All

15 SRS - Synchronised Restart Single

URS - Unsynchronised Restart Single

URA - Unsynchronised Restart All

UPS - Unsynchronised Pause Single

UPA - Unsynchronised Pause All

20

DS - Disconnect Single

DA - Disconnect All

CS - Connect Single

CA - Connect All

25 PS - Pause Single

PA - Pause All

RS - Restart Single

RA - Restart All

30 In Figure 8, the value n at S0, SPS, SRA, SRS, UPA and UPS
is 0, and the value n at SC, NC, URS and URA is 1. Also,
the symbols, indicating controls between states are as
follows:

35 S0 - SC (CS)

S0 - NC (CA)

SC - S0 (DS, DA)

SC - SPS (PS)
SC - NC (CS, CA)
SPS - S0 (DA)
SPS - SC (DS)
5 SPS - SPS (PS, RS, DS)
SPS - SPA (PA)
SPS - NC (CS, CA)
NC - SPS (DS)
NC - NC (CS, CA)
10 NC - SPA (PA)
NC - UPS (PS)
NC - S0 (DA)
SPA - SRS (RS)
SPA - URS (CS, CA)
15 SPA - SRA (RA)
SPA - S0 (DA)
SPA - SPA (DS)
SPA - SPS (DS)
SRA - S0 (DA)
20 SRA - NC (CS, CA)
SRA - SPA (PA)
SRA - SPS (DA)
SRA - SRA (DS)
SRA - UPS (PS)
25 SRS - NC (CS, CA)
SRS - S0 (DA)
SRS - SRS (DS, PS)
SRS - URA (RA)
SRS - URS (RS)
30 URS - UPA (PA)
URS - URA (PA)
URS - UPS (RS, PS)
URS - URS (RS, CS, CA)
URA - UPS (CS, PS)
35 URA - SPA (PA)
URA - NC (CS, CA)
URA - S0 (DA)

UPS - URS (RS, CS, CA, DS)
UPS - SPS (PA)
UPS - URA (RA)
UPS - UPA (PA)
5 UPS - UPS (PS, DS)
UPS - S0 (DA)
UPA - S0 (DA)
UPA - URS (RS, CS, CA)
UPA - UPS (DS)
10 UPA - URA (RA)

In a still further example, the system may implement further algorithms that maintain a source computer's low CPU utilisation, high image clarity, easy toggle and
15 control shift in the Many-to-Many connections, in addition to the algorithms to transmit control and image packet data. The CPU utilisation algorithm initially reads the source computer's CPU utilisation and tunes other algorithms to automatically cap off the required CPU
20 utilisation to a pre configured limit. Thus, the algorithm makes apt use of resources available and makes room for other applications for the user of the source computer.

25 Further detail of the algorithms is given below.

The Auto Initial CPU cap off algorithm.

The Auto Initial CPU cap off algorithm initially reads the user's CPU utilization and tunes the above algorithms
30 automatically to a pre configured limit. Thus, making apt use of resources available and making room for other applications for the user. For the first 10 seconds, the algorithm takes records of the threads under study and computes the interval time for which the threads have to
35 wait execute within the CPU cap off value.

Pseudo Code:

Function 1.

This function takes on one of the important tasks of capturing the desktop image. This runs in synchronisation with the image difference algorithm. After executing one cycle of capturing the desktop, it waits for the image difference algorithm to complete its execution.

```
10      DesktopImageCaptureThread Function
      {
          rectangle = Screen Resolution
          _tempDesktopImage's PixelFormat = Format32bppArgb

15          while (Connected)
          {
              CaptureThread_AutoResetEvent.Wait for set signal

              Thread1_TotalWaitTime = Thread1_waitTime_2 -
20      Thread1_waitTime_2

              if (_Optimize)
                  wait for _T1_NewWaitTime seconds

25              perform CaptureDesktopScreen

              Thread1_waitTime_2 = Note Current Time

30              Set ImageDiffernce_AutoResetEvent

              Thread2_waitTime_2 = Note Current Time
          }
35      }
```

Function 2.

This function computes the image difference which has to be sent over to the remote client system. This thread works in along with image capture thread. The wait value computed by this algorithm will be used in this thread.

```
45      ImageDiffernceComputationThread Function
      {
          _rectangle = Screen Resolution
          _tempDesktopImage's PixelFormat = Format32bppArgb

50          while (Connected)
          {
              ImageDiffernce_AutoResetEvent.Wait for Desktop Image to be
              Captured

55              Thread2_TotalWaitTime = Thread2_waitTime_2 -
              Thread2_waitTime_1

              if (_Optimize)
                  wait for _T2_NewWaitTime seconds

60              perform ComputeImageDifference

              Thread2_waitTime_1 = Note Current Time
```

```
Set CaptureThread_AutoResetEvent
Thread1_waitTime_2 = Note Current Time
5      }
      }
      }
```

10 **Function 3.**

This function computes the wait time which is being used by the above mentioned threads to make their CPU utilization below the specified limit. It takes a lot of statistics on to come out with the wait time which is the arithmetic mean of ten times collected. Once attained, this computation executes no more.

```
PerformanceUpdateTimer_Tick Function
{
20      TotalCPUUsageValue = Get Total CPU usage value

      Call UpdateProcessThreadValues Function to Update Thread CPU
values

25      if _optimising
      {
          if _count < 10
          {
              increment _count by 1
30              _totalCPUAvg = _totalCPUAvg + TotalCPUUsageValue

              _CaptureThread_waitTime = _CaptureThread_waitTime +
Thread1_TotalWaitTime
35              _ImageDiffThread_waitTime = _ImageDiffThread_waitTime +
Thread2_TotalWaitTime
          }
      }
      else
      {
40          _totalCPUAvg = _totalCPUAvg / _count

          _CaptureThread_CPUAvg = _CaptureThread_CPUAvg / _count
45          _ImageDiffThread_CPUAvg = _ImageDiffThread_CPUAvg / _count

          _CaptureThread_waitTime = _CaptureThread_waitTime / _count
          _ImageDiffThread_waitTime = _ImageDiffThread_waitTime /
50          _count

          _percentCPUReduction = (_totalCPUAvg -
              (RequiredCPU_CapOff / _totalCPUAvg)
55          * 100

          _reducedT1_CPU = (_CaptureThread_CPUAvg *
              _percentCPUReduction) / 100
60          _reducedT2_CPU = (_ImageDiffThread_CPUAvg *
              _percentCPUReduction) / 100

          _reducedT1_CPU_Percent = ((_CaptureThread_CPUAvg -
              _reducedT1_CPU) /
65          _CaptureThread_CPUAvg) * 100
```

```

        _reducedT2_CPU_Percent = ((_ImageDiffThread_CPUAvg -
_reducedT2_CPU) /
5
                                _ImageDiffThread_CPUAvg) * 100
        _T1_NewWaitTime = ((_CaptureThread_waitTime * 100) /
_reducedT1_CPU_Percent)) -
                                _T1_waitTime
10
        _T2_NewWaitTime = ((_ImageDiffThread_waitTime * 100) /
_reducedT2_CPU_Percent))-
                                _T1_waitTime

        _optimising = false
15
        _Optimize = true
    }
}
20
}
```

Function 4.

25 In order to check compute the required wait time, we need a record of all the threads that are running on the current CPU, along with their CPU utilization values, which will be used in the following computations.

```

30
    GetAllProcessThreadList Function
    {
        _ThreadsList = Get all the threads in running in CPU
35
        Filter out those which are of this process only.
    }
}
```

Function 5.

40 Since this algorithm will concern the two main CPU guzzling threads, we need to filter out those threads which do not affect or use too much of the CPU. Hence, this function will filter out the unnecessary threads and take into consideration only those two threads which are CPU intensive, which are the threads described above.

```

50
    UpdateProcessThreadValues Function
    {
        For Every Thread in CPU
        {
55
            Get CPUPThreadValue

            if (CPUPThreadValue > 5.0)
            {
                if (_optimising)
                {
60
                    Record First Thread

                    if (Thread_1_Name is null)
                    {
```

```

Thread_1_Name = currentCPUThreadValueName
which contribute more than 30%
of the total CPU consumption.
}
5
}

else if (currentCPUThreadValueName equals
Thread_1_Name)
10
CaptureThread_CPUPAvg =
CaptureThread_CPUPAvg + CPUThreadValue

Now record Second Thread

15
if (Thread_2_Name is null)
{
check if this name has already been
assigned to Thread_1_Name
if Thread_1_Name is not equal
20
currentCPUThreadValueName
{
if CPUThreadValue >
(TotalCPUUsageValue * 0.3)
{
25
currentCPUThreadValueName
Thread_2_Name =

}
}
}
30
else if (currentCPUThreadValueName equals
Thread_2_Name)
Add CPUThreadValue to
_ImageDiffThread_CPUPAvg
35
}
}
```

The Partial Image Send Processing algorithm.

Image processing algorithms send image data across to the
40 image processor only when it detects a change in desktop
content and not otherwise, thus avoiding unnecessary usage
of network bandwidth. The following explains the
transition that happens to an image based upon the
decisions made in the algorithms.

45

Pseudo Code:

Function 1.

50

This Function captures the desktop image and intimates the thread
that computes the image difference. The image and its related data
are sent in _dataContainer, which a structural package.

55

```
StartCapturing Function
{
Add _unitDesktopImageData to _dataContainer
```

```
        while (true)
        {
            Wait till _ImageProcessed
5            _desktopImage = CaptureDesktop Image

            if (!_pauseCaptureThread)
                Signal that _desktopCaptured
10        }
    }
```

Function 2.

15 This function computes the image difference between the present and previous desktop images, if _resetImage is not set to true. _resetImage is set to true if the full image is needed instead of a partial one. This function returns an image which is containing the change alone and not the
20 redundant portions. The conditions which set _resetImage to true are restart connection, connect single, new connection, resend data and resolution change.

This function works along with the StartCapturing in synchronisation, one process after the other.

```
25
    ComputeImageDifference Function
    {
        while (true)
        {
            Wait till _desktopCaptured

            _desktopImage to _tempDesktopImage

            if (_resetImage)
            {
                _prevSynchronisedImage = null

                //Resolution has changed. Therefore get the new
40 screen co-ordinates.
                _imageRect = get screen resolution
            }

            if (_prevSynchronisedImage == null)
            {
                If there is a resolution change, this following
                condition will avoid sending
                _tempDesktopImage which could be of the previous
                resolution.
50 If not checked, OnDataUpdated will send the image and
                maintain
                _tempDesktopImage as the _prevSynchronisedImage,
                which is of a different resolution from the new
                _tempDesktopImage of the next
55 cycle, hence causing an exception in GetDifferenceImage
                function

                if (_tempDesktopImage.Width == _image
                processordesktopWidth &&
60 _tempDesktopImage.Height == _image
                processordesktopHeight)
                {
                    Save _tempDesktopImage as imageMemoryStream
                }
            }
        }
    }
```

```

    _tempDesktopImage.Width      _unitDesktopImageData.width =
    _tempDesktopImage.Height     _unitDesktopImageData.height =
5  imageMemoryStream.ToArray()  _differenceImageArray =
    (int)_encodingType           _unitDesktopImageData.encodingType =
10  _differenceImageArray        _unitDesktopImageData.imageBuffer =
    _unitDesktopImageData.TopLeft_X_CoOrdiante = 0
    _unitDesktopImageData.TopLeft_Y_CoOrdiante = 0

    _dataContainer[0] = _unitDesktopImageData
    Call OnDataUpdated function
    }
}

20  else
    {
        Call GetDifferenceImage Function

        if (_imageDifference)
25  {
            Save _diffImage as imageMemoryStream

            _unitDesktopImageData.width = _diffImage.Width
            _unitDesktopImageData.height = _diffImage.Height
30  _differenceImageArray =
imageMemoryStream.ToArray()
            _unitDesktopImageData.encodingType =
            (int)_encodingType
            _unitDesktopImageData.imageBuffer =
35  _differenceImageArray

            _dataContainer[0] = _unitDesktopImageData
            Call OnDataUpdated function
40  }
    }

    Stay idle for _T2_NewWaitTime seconds as computed

45  Signal _ImageProcessed
    }
}
```

50

Function 3.

Once the difference or a full image is ready, send it to the destination program and get a return value if the image has been used or updated. This is necessary to maintain the same previous image throughout, which is the source or client desktop and the receiving computer. Only by doing this, the difference image will be appended without any miss in a frame. Or else the source and destination images will not be the same.

60

```
OnDataUpdated Function
{
```

```
        Send _dataContainer to destination and get the result if
imageUpdatedAtDestination

        if (imageUpdatedAtDestination)
5          {
            //_prevSynchronisedImage will be null when the image sent
is first one or when
            resolution is changed.
            if (_prevSynchronisedImage == null)
10              _prevSynchronisedImage = create a dummy image of size
_image processordesktopWidth and
                                                    _image processordesktopHeight

            else
15              _prevSynchronisedImage = _currentDesktopImage
          }
        }

20    Function 4.

        Computing the image difference.

25    GetDifferenceImage Function
    {

        //This block gets the CoOrdinates for rectangel of differnce
        _remain1 = value of one pixel stride in image 1
30        _remain2 = value of one pixel stride in image 2

        _ptr1 = point to starting pixel in image 1
        _ptr2 = point to starting pixel in image 2

35        loop from i = 0 till i < _height
        {
            loop from j = 0 till j < _width * 3
            {
                if (_ptr1[0] != _ptr2[0])
40                {
                    _topLeft_x = j if (_topLeft_x > j) else
                    _topLeft_y = I if (_topLeft_y > i) else
                    _btmRight_x = j if (_btmRight_x < j) else
45                    _btmRight_y = I if (_btmRight_y < i) else
                    _topLeft_x
                    _topLeft_y
                    _btmRight_x
                    _btmRight_y

                    }
50                _ptr1++
                _ptr2++
            }
            _ptr1 += _remain1
            _ptr2 += _remain2
55        }

        //Validate if the above coordinates are not same as the
        cornern values of the complete
60        image.
        if (!((_topLeft_x == _width) &&
            (_topLeft_y == _height) &&
            (_btmRight_x == 0) &&
            (_btmRight_y == 0))) &&
65        (_btmRight_y - _topLeft_y) != 0 &&
```

```

        ((_btmRight_x - _topLeft_x) / 3) != 0)
    {
        //Consider the pixels in row 0 and column 0 also even if
5    they start from second row
        and second column.
        if (_topLeft_x >= 2 && _topLeft_y >= 2)
        {
            _topLeft_y = _topLeft_y - 2
10         _topLeft_x = _topLeft_x - 2
        }

        _rectOfChange.X = _topLeft_x / 3
        _rectOfChange.Y = _topLeft_y
15         _rectOfChange.Width = (_btmRight_x - _topLeft_x) / 3
        _rectOfChange.Height = _btmRight_y - _topLeft_y

        //avoid the Exception when rectangle is bigger than
20     resolution
        if ((_rectOfChange.X + _rectOfChange.Width + 2) < _width)
            _rectOfChange.Width = _rectOfChange.Width + 2

        if ((_rectOfChange.Y + _rectOfChange.Height + 2) <
25     _height)
            _rectOfChange.Height = _rectOfChange.Height + 2

        Clone _tempDesktopImage to _diffImage
30
        TopLeft_X_CoOrdinate of _unitDesktopImageData =
        _topLeft_x
        TopLeft_Y_CoOrdinate of _unitDesktopImageData =
        _topLeft_y
35
        _imageDifference = true
    }

40     else _imageDifference = false

    Copy _tempDesktopImage to _currentDesktopImage
}
45
```

Connections from single client to multiple image processors algorithm.

The user display device, for example user laptop, on which
50 a client component program is running, can connect to more
than one image processor at a time. All of these
connections are made through a single socket connection.
The laptop maintains a list of all the active image
processors acting as servers. An active image processor is
55 those which are connected at that point of time and are
actively communicating. Each and every packet is
communicated to the whole list. Additionally,

synchronising properties are in built to make sure that all the image processors receive the same packet without a difference.

5 **Pseudo Code:**

Function 1

10 This function makes a connection from the client laptop (display device) to an image processor. This method also contains the authentication mechanisms for the connection. Only if the connection is valid, that image processor is added to the list of active servers.

```
15 ConnectToImage processor(image processorIPAddress);  
On receiving connection from client, authenticate Connection with IP  
and send acknowledgement.  
if (OnSuccessfulConnection)  
    {  
20         _AddToimage processorList(image processorIPAddress)  
         _activeimage processorList.Add(image processorIPAddress)  
         _projectingimage processorCount++  
    }
```

Function 2

25 This function is responsible for sending the image and mouse data packets to all the servers that are in the list added by the Function 1. After sending each data packet, it waits for acknowledgments from the image processor, after which the subsequent packets are sent across. There is a threshold time for
30 which the client is going to wait for each image processor will acknowledge. If the timer times out, that connection to image processor is cut off and deleted from the list of image processor.

```
35 TriggerAsyncSend()  
    {  
        while (Acknowledged by all the servers)  
        {  
            _imageData = PacketProtocol.GetPacket(_ImageData)  
            foreach server in _activeServerList  
            {  
40                 _socket.SendData(_imageData) to IPAddress  
                 _projectingimage processorCount--  
            }  
45                 Start _acknowledgeTimer. If timesout, then disconnect  
                 this connection and remove from _activeimage  
                 processorList.  
        }  
50    }
```

Function 3

55 This function is responsible for removing an image processor from the connected image processor list once the threshold wait time for acknowledgement is over.

```
AcknowledgeTimeOutHandler()  
{
```

```
        foreach (IPAddress deadServer in
_serverInventory.InActiveServerList)
        {
            _serverList.RemoveAt(index);
5
            if (_activeServerList.Contains(deadimage processorIPAddress))
            {
                indx = _activeServerList.IndexOf(deadimage
processorIPAddress)
10
                Remove deadimage processorIPAddress from _activeServerList
                _projectingServerCount--
            }
        }
    }
15
```

Connections from single image processor to multiple clients.

A single image processor can connect to multiple client display devices outputting image content to be displayed. This is possible by maintaining a list of client or sources that are actively connected to this image processor. All the connections are made through one single socket. Data packets received from all the sources are sorted based on the source IP address; assembled if it has been fragmented earlier and display of that connection is updated.

Pseudo Code:

```
30
    Function 1
    This function makes a connection from to the source laptop. It
    also contains the authentication mechanisms for the connection.
    Only if the connection is valid, that source is added to the
35
    list of active sources.

    ConnectFromSourceReceived(SourceIPAddress);
    On receiving connection from client, authenticate Connection with IP
    and send acknowledgement.
40
    if (OnSuccessfulConnection)
    {
        _AddToSourceList (SourceIPAddress)
        _activeSourceList.Add(SourceIPAddress)
        _projectingSourceCount++
45
    }
```

Function 2

This function does multiple operations on the data packets received based on the kind of packet received from and also based on which source has sent it.

It does the operations of:

- It accepts a connection from a source which needs to project its content to image processor.

- On receiving the connection, it authenticates the connection and then adds this source IP address into its list of actively projecting source list.
- Segregates the packets based on the header and finds out if it's a valid packet by checking the packet size. If the packet is valid, it sends it to its respective processing function. If it isn't, it stores in the temporary buffer and appends it with the other packets until it's a valid packet.

[illegible]

```
        _receivedArray = null;

    }
    break;
5
        case PacketType.DataPacket:
            //if it is data packet, pass the data to StaticImage to get image data
            if (_receivedPacketType == DataPacket)
            {
10                _completeArray = _receivedArray;

                //Check if the image processor Client is still connected. If it is, you will
                be receiving images.
                if (_SourceConnected)
15                {
                    _ImageData =
                        (ArrayList)PacketProtocol.GetPacketData
                        (PacketType.DataPacket,
20                        _completeArray);

                    //send the image data to the view class for display.
                    _display.UpdateDesktopImage by building Image from _ImageDataList;
                }

25                //If not, connection has been cut off, set default image processorServer
                display image.
                else
                    _display.UpdateDesktopImage(null);

30                _socket.SendData(e.SourceIPAddress,
                    PacketProtocol.GetPacket(PacketType.
                        Acknowledge,
                        AcknowledgementStatus.DataReceived))
                    ;
35            }

            _receivedArray = null;

            break;
40
        case PacketType.MouseCursorPacket:
            if (_receivedPacketType == MouseCursorPacket)
            {
45                _mouseDataArray = _receivedArray;

                _mouseData =
                    PacketProtocol.GetPacketData(PacketT
50                    ype.MouseCursorPacket,
                    _mouseDataArray);

                //send the image data to the view class for
                display.
                _display.UpdateMouseData(_mouseData);
55                _receivedArray = null;
                break;

                case PacketType.MultiplePackets:
60
                    requiredLength =
                        _receivedPacketData.RequiredTrimmedBytes;

                    count = 0;
65                    for each byte in _receivedArray
```

```
{
    if (count < requiredLength)
    {
        if (_receivedTempArray_1 == null)
        {
            _receivedTempArray_1 = initialize;

            _receivedTempArray_1[count] = Byte;
        }

        else
        {
            if (_receivedTempArray_2 == null)
            {
                _receivedTempArray_2 = new byte[_receivedArray.Length -
                    requiredLength];
            }

            _receivedTempArray_2[Math.Abs(count - (requiredLength))] = Byte;
        }

        count++;
    }

    _receivedArray = _receivedTempArray_1;
    _receivedTempArray_1 = null;
    breakProcessingLoop = false;
    break;

case PacketType.IncompletePacket:

    _incompleteByteArray = true;
    _remainingIncompleteByteLength =
        _receivedPacketData.RemainingIncompleteByteLe
        ngth;
    _partialArray = _receivedArray;
    _receivedArray = null;
    break;

case PacketType.ResolutionChangePacket:

    ResolutionChangeData =
        (ResolutionChangePcktStructure)PacketProtocol
        .GetPacketData(_receivedPacketType,
        _receivedArray);

    ChangeDesktopResolution(ResolutionChangeData.
        Width, ResolutionChangeData.Height);

    _socket.SendData(e.SourceIPAddress,
        PacketProtocol.GetPacket(PacketType.Acknowledge,
        AcknowledgementStatus.ResolutionChangeSuccess
        ful));

    _receivedArray = null;
    break;
}

//Check if there are any data fragments remaining
if (_receivedArray == null && _receivedTempArray_2 !=
null)
{
    _receivedArray = new
    byte[_receivedTempArray_2.Length];
    _receivedArray = _receivedTempArray_2;
}
```

```
        _receivedTempArray_2 = null;  
        breakProcessingLoop = false;  
    }  
5      if (breakProcessingLoop)  
        break;  
    }  
}
```

10

15

CLAIMS

1. A method of processing electronic image content for display, the method comprising:
 - 5 a computing system receiving a first image comprising a first array of pixels;
said computing system receiving a second image comprising a second array of pixels;
 - a comparator of said computing system identifying
10 which pixels of the second array are different from corresponding pixels of the first array by comparing said first array and said second array and outputting said different pixels of said second array to at least one image composer;
 - 15 said image composer constructing a new image for display comprising said different pixels from said second array identified by said comparator and pixels of said first array complementary to said different pixels of the second array; and
 - 20 outputting the new image for display.
2. A method as claimed in claim 1, wherein said image composer is connected to said computing system over a telecommunications network.
- 25 3. A method as claimed in claim 2, further comprising outputting said different pixels from said second array identified by said comparator from said computing system via TCP/IP over the telecommunications network.
- 30 4. A method as claimed in claim 3, further comprising said image composer authenticating receipt of said different pixels.
- 35 5. A method as claimed in claim 1, further comprising outputting the new image for display by at least one display device.

6. A method as claimed in claim 5, further comprising outputting to the display device using an audio/video interface.

5

7. A method as claimed in claim 5, further comprising said computing system outputting control information to control display of said image content on the display device.

10

8. A method as claimed in claim 7, wherein said control information comprises a pixel array size of said first and second array.

15

9. A method as claimed in claim 1, further comprising said computing system receiving audio and/or video data and outputting said audio and/or video data in addition to said different pixels from said second array identified by said comparator as streaming data to the image composer.

20

10. A method as claimed in claim 9, further comprising said computing system outputting control information to control display of said audio and/or video data for display by at least one display device.

25

11. A method as claimed in claim 9, further comprising outputting said streaming data to a memory intermediate said image composer.

30

12. A method as claimed in claim 11, wherein said memory is located in a server connected to said computing system over a telecommunications network.

35

13. A method as claimed in claim 1, further comprising outputting from said computing system said different pixels identified by said comparator to each of the at least one image composers to construct said new image for

display by each display device corresponding to the image composers.

14. A method as claimed in claim 13, further comprising
5 said computing system synchronising output of said different pixels identified by said comparator to each image composer to provide synchronised display on each corresponding display device.

10 15. A method as claimed in claim 14, further comprising said computing system outputting control information to synchronise display of said image content on each corresponding display device.

15 16. A method as claimed in claim 1, further comprising compressing said different pixels from said second array identified by said comparator before outputting to said image composer.

20 17. A method as claimed in claim 1, wherein said computing system comprises at least one computer.

18. A method as claimed in claim 17, wherein said image content originates from the at least one computer as a
25 electronic image.

19. A method as claimed in claim 1, further comprising
said computing suspending said comparator from identifying
which pixels of the second array are different from
30 corresponding pixels of the first array by comparing said first array and said second array and outputting said different pixels of said second array to said image composer when said comparator exceeds a CPU utilisation threshold of said computing system.

35 20. A method as claimed in claim 19, wherein said threshold is 30% of said computing system CPU utilisation.

21. A system for processing electronic image content for display, the system comprising:

5 a computing system arranged to receive a first image comprising a first array of pixels and a second image comprising a second array of pixels;

10 a comparator of said computing system arranged to identify which pixels of the second array are different from corresponding pixels of the first array by comparing said first array and said second array and output said different pixels of said second array to at least one image composer,

15 whereby said image composer is arranged to construct a new image for display comprising said different pixels from said second array identified by said comparator and pixels of said first array complementary to said different pixels of the second array and output the new image for display.

20 22. A system as claimed in claim 21, wherein said image composer is connected to said computing system over a telecommunications network.

25 23. A system as claimed in claim 22, wherein said computing system outputs said different pixels from said second array identified by said comparator via TCP/IP over the telecommunications network.

30 24. A system as claimed in claim 21, wherein said image composer outputs said new image for display by at least one display device.

35 25. A system as claimed in claim 21, wherein said image composer outputs to the display device using an audio/video interface.

26. A system as claimed in claim 21, wherein said

computing system outputs said different pixels identified by said comparator to more than one image composer to construct said new image for display by more than one corresponding display device.

5

27. A system as claimed in claim 21, wherein said computing system comprises at least one computer.

28. A device for processing electronic image content for display, the device comprising:

10

an image composer arranged to:

receive a first image comprising a first array of pixels;

15

receive pixels of a second image comprising a second array of pixels which differ from corresponding pixels of the first array;

20

construct a new image for display comprising said different pixels from said second array identified by said comparator and pixels of said first array complementary to said different pixels of the second array; and

output the new image for display.

29. Computer program code which when executed implements the method of any one of claims 1 to 20.

25

30. A computer readable medium comprising the program code of claim 28.

31. A data file comprising the program code of claim 29.

30