The view frustum culling technique described herein allows Free Viewpoint Video (FVV) or other 3D spatial video rendering at a client by sending only the 3D geometry and texture data necessary for a specific viewpoint or view frustum from a server to the rendering client. The synthetic viewpoint is then rendered by the client by using the received geometry and texture data for the specific viewpoint or view frustum. In some embodiments of the view frustum culling technique, the client has both some texture data and 3D geometric data stored locally if there is sufficient local processing power. Additionally, in some embodiments, additional spatial and temporal data can be sent to the client to support changes in the view frustum by providing additional geometry and texture data that will likely be immediately used if the viewpoint is changed either spatially or temporally.
RECEIVE ONLY TEXTURE DATA AND GEOMETRIC DATA NECESSARY TO RENDER A GIVEN VIEW FRUSTUM OF A SPATIAL THREE DIMENSIONAL VIDEO FROM A SERVER AT A CLIENT.

RENDER THE GIVEN VIEWPOINT OF THE SPATIAL THREE DIMENSIONAL VIDEO AT THE CLIENT USING THE DOWNLOADED TEXTURE AND GEOMETRIC DATA NECESSARY TO RENDER THE GIVEN VIEW FRUSTUM.

FIG. 1
CAPTURE A SCENE USING AN ARRANGEMENT OF SENSORS.

INPUT AND CALIBRATE THE STREAMS OF SENSOR DATA.

GENERATE GEOMETRIC DATA AND TEXTURE DATA FROM THE CALIBRATED STREAMS OF SENSOR DATA AND STORE.

RECEIVE A CURRENT SYNTHETIC VIEWPOINT OR VIEW FRUSTUM OF THE SCENE FROM A CLIENT COMPUTING DEVICE.

RETRIEVE THE STORED GEOMETRIC DATA AND TEXTURE DATA NECESSARY TO RENDER THE CURRENT SYNTHETIC VIEWPOINT OR VIEW FRUSTUM.

TRANSMIT THE GEOMETRIC DATA AND TEXTURE DATA TO THE CLIENT COMPUTING DEVICE FOR RENDER AND DISPLAY TO AN END USER.

FIG. 2
A FREE VIEWPOINT VIDEO PLAYER IS INSTALLED ON A LOCAL CLIENT.

A USER OF THE LOCAL CLIENT REQUESTS A FREE VIEWPOINT VIDEO FROM A SERVER.

A MESSAGE FROM THE SERVER IS RECEIVED AT THE CLIENT THAT TELLS THE CLIENT TO INSTANTIATE THE FREE VIEWPOINT VIDEO PLAYER WITH CONTROLS APPROPRIATE TO THE SELECTED FREE VIEWPOINT VIDEO TYPE.

THE CLIENT INSTANTIATES THE FREE VIEWPOINT VIDEO PLAYER.

THE CLIENT REQUESTS A DESIRED VIEWPOINT OR VIEW FRUSTUM FROM THE SERVER.

THE CLIENT RECEIVES FROM THE SERVER ONLY THE NECESSARY 3D GEOMETRY DATA AND TEXTURE DATA TO RENDER THE VIEWPOINT AT THE CLIENT.

THE CLIENT COMBINES THE RECEIVED 3D GEOMETRY AND TEXTURE DATA TO RENDER THE DESIRED VIEW OF THE FREE VIEWPOINT VIDEO.

THE CLIENT CHECKS FOR USER NAVIGATION INPUT AND IF THERE IS ANY USER NAVIGATION INPUT SENDS THE NAVIGATION INPUT TO THE SERVER.

THE CLIENT RECEIVES NEW 3D GEOMETRY AND TEXTURE DATA TO RENDER A NEW VIEWPOINT OR VIEW FRUSTUM BASED ON THE UPDATED NAVIGATION INPUT FROM THE SERVER.

FIG. 3
FIG. 4
SERVER 502

3D RENDERER 514

DATABASE OF FW/SPATIAL 3D VIDEO FOR 360 DEGREE VIEWPOINTS OF SCENES AND CORRESPONDING GEOMETRIC DATA AND TEXTURE DATA FOR EACH VIEWPOINT

VIEWPOINT PREDICTOR BASED ON VIEWPOINT NAVIGATION CHANGES RECEIVED FROM CLIENT 528

LEVEL OF DETAIL 532 COMPUTATIONS

COMPUTATION OF VIEW FRUSTUM FOR GIVEN VIEWPOINT 510

REQUEST FOR A SPECIFIC FWV 534

COMMAND TO INSTANTIATE FWV PLAYER FOR CHOSEN FW 536

REQUEST FOR A CURRENT VIEWPOINT OF FW 538

GEOMETRY AND TEXTURE DATA TO RENDER ONLY CURRENT VIEWPOINT OF CHOSEN FW 540

UPDATED VIEWPOINT OR RATE OF CHANGE OF VIEWPOINT 542

GEOMETRY AND TEXTURE DATA TO RENDER PREDICTED VIEWPOINT OF CHOSEN FW BASED ON AN UPDATED VIEWPOINT OR RATE OF CHANGE 544

LEVEL OF DETAIL 530 COMPUTATIONS

FVV/3D SPATIAL VIDEO DATA STORE PREVIOUSLY RECEIVED DATA

COMPUTATION OF VIEW FRUSTUM FOR GIVEN VIEWPOINT 512

FIG. 5
SIMPLIFIED COMPUTING DEVICE 500

PROCESSING UNIT(S) 710

SYSTEM MEMORY 520

DISPLAY DEVICE(S) 755

GPU(S) 715

INPUT DEVICE(S) 740

OUTPUT DEVICE(S) 750

COMMUNICATIONS INTERFACE 730

REMOVABLE STORAGE 770

NON-REMOVABLE STORAGE 780

STORAGE DEVICES 760

FIG. 7
VIEW FRUSTUM CULLING FOR FREE VIEWPOINT VIDEO (FVV)

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of and the priority to a prior provisional U.S. patent application entitled "INTERACTIVE SPATIAL VIDEO" which was assigned Ser. No. 61/653,983 and was filed May 31, 2012.

BACKGROUND

[0002] A traditional video generally includes one or more scenes, where each scene in the video can be either relatively static (e.g., the objects in the scene do not substantially change or move over time) or dynamic (e.g., the objects in the scene substantially change and/or move over time). In a traditional video the viewpoint of each scene is chosen by the director when the video is recorded or captured and this viewpoint cannot be controlled or changed by an end user while they are viewing the video. In other words, in a traditional video the viewpoint of each scene is fixed and cannot be modified when the video is being rendered and displayed.

[0003] Free Viewpoint Video (FVV) is created from images captured by multiple cameras viewing a scene from different viewpoints. FVV generally allows a user to look at a scene from synthetic viewpoints that are created from the captured images and to navigate around the scene. More specifically, in FVV an end user can interactively control and change their viewpoint of each scene at will while they are viewing the video. In other words, in a FVV each end user can interactively generate synthetic (i.e., virtual) viewpoints of each scene on-the-fly while the video is being rendered and displayed. This creates a feeling of immersion for any end user who is viewing a rendering of the captured scene, thus enhancing their viewing experience.

[0004] The creation and playback of a FVV requires working with a substantial amount of data. The process of creating and playing back FVV or other 3D spatial video typically is as follows. First, a scene is simultaneously recorded from many different perspectives using sensors such as RGB cameras and other video and audio capture devices. Second, the captured video data is processed to extract 3D geometric information in the form of geometric proxies using 3D Reconstruction (3DR) algorithms. Finally, the original texture data (e.g., RGB data) and geometric proxies are recomposed during rendering, for example by using Image Based rendering (IBR) algorithms, to generate synthetic viewpoints of the scene.

[0005] The amount of data may vary considerably from one FVV to another FVV due to the differences in the number of sensors used to record the scene, the length of the FVV, the type of 3DR algorithms used to process the data, and the type of IBR algorithm used to generate synthetic views of the scene.

[0006] There exists a wide variety of different combinations of both bandwidth and local processing power that can be used for viewing FVV on a client.

SUMMARY

[0007] In general, embodiments of the view frustum culling technique described herein transfer data necessary to render a given viewpoint or view frustum of a FVV or other three-dimensional (3D) spatial video over a network, from one or more servers to a client that renders the FVV or 3D spatial video.

[0008] In some embodiments of the view frustum culling technique only 3D geometry and texture data (e.g., RGB texture data) necessary for rendering a specific synthetic viewpoint or view frustum for a FVV or 3D spatial video are transmitted from a server (or computing cloud) to a client. The video for the synthetic viewpoint is then rendered by the client using the received 3D geometry and texture data. One benefit of these embodiments of the view frustum culling technique is that only the data necessary to render a specific viewpoint is transferred from the server to the client. This limits the amount of bandwidth required to transfer FVV or 3D spatial video to a client.

[0009] In some embodiments of the view frustum culling technique, the client stores some texture data and 3D geometric data locally if there is sufficient local processing power. Local data at the client and sufficient processing power can lead to more fluid and seamless transitions as the virtual viewpoint is moved around within a FVV scene. In addition, for static or non-moving elements of the scene, 3D geometry can be cached locally on the client, eliminating the need for redundant data transfers.

[0010] Finally, in some embodiments of the view frustum culling technique, additional spatial and temporal data can be sent to the client from the server so that data necessary to support a desired view frustum is supplemented with additional geometry and texture data that would be immediately used if the viewpoint was changed either spatially or temporally.

[0011] It is noted that this Summary is provided to introduce a selection of concepts, in a simplified form, that are further described hereafter in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

DESCRIPTION OF THE DRAWINGS

[0012] The specific features, aspects, and advantages of the disclosure will become better understood with regard to the following description, appended claims, and accompanying drawings where:

[0013] FIG. 1 depicts a high level flow diagram of an exemplary process for practicing the view frustum culling technique described herein.

[0014] FIG. 2 depicts another flow diagram of an exemplary process for practicing the view frustum culling technique described herein from the perspective of a server.

[0015] FIG. 3 depicts another flow diagram of an exemplary process for playing FVV content at a client according to the view frustum culling technique.

[0016] FIG. 4 depicts one exemplary embodiment of the view frustum culling technique described herein wherein the geometric data and texture data of the view frustum is divided into increasingly smaller three dimensional cells.

[0017] FIG. 5 is an exemplary architecture for practicing one exemplary embodiment of the view frustum culling technique described herein.

[0018] FIG. 6 is a diagram illustrating a spatial three dimensional video pipeline in which the view frustum culling technique described herein can be practiced.
FIG. 7 is a schematic of an exemplary computing environment which can be used to practice the view frustum culling technique.

DETAILED DESCRIPTION

In the following description of the view frustum culling technique, reference is made to the accompanying drawings, which form a part thereof, and which show by way of illustration examples by which the view frustum culling technique described herein may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the claimed subject matter.

The following sections provide background information and an overview of the view frustum culling technique, as well as exemplary processes and an exemplary architecture for practicing the technique. Details of various embodiments of the view frustum culling technique are also provided, as is a description of an exemplary spatial video pipeline and a suitable computing environment for practicing the technique.

It is also noted that for the sake of clarity specific terminology will be resorted to in describing the pipeline technique embodiments described herein and it is not intended for these embodiments to be limited to the specific terms so chosen. Furthermore, it is to be understood that each specific term includes all its technical equivalents that operate in a broadly similar manner to achieve a similar purpose. Reference herein to “one embodiment”, “another embodiment”, “an exemplary embodiment”, or “an alternate embodiment”, or “one implementation”, or “another implementation”, or “an exemplary implementation”, or “an alternate implementation” means that a particular feature, a particular structure, or particular characteristics described in connection with the embodiment or implementation can be included in at least one embodiment of the pipeline technique. The appearances of the phrases “in one embodiment”, “in another embodiment”, “in an exemplary embodiment”, “in an alternate embodiment”, “in one implementation”, “in another implementation”, “in an exemplary implementation”, and “in an alternate implementation” in various places in the specification are not necessarily all referring to the same embodiment or implementation, nor are separate or alternative embodiments/implementations mutually exclusive of other embodiments/implementations. Yet furthermore, the order of process flow representing one or more embodiments or implementations of the pipeline technique does not inherently indicate any particular order not imply any limitations of the pipeline technique.

The term “sensor” is used herein to refer to any one of a variety of scene-sensing devices which can be used to generate a sensor data that represents a given scene. Each of the sensors can be any type of video capture device (e.g., any type of video camera).

The term “server” is used herein to refer to one or more server computing devices either operating in a standalone server-client mode or operating in a computing cloud infrastructure so as to provide FFV or 3D spatial video services to a client computer over a data communication network.

A view frustum is the region of space in a modeled world that might appear on a screen; it is the field of view of a notional camera. View frustum culling is the process of removing objects that lie completely outside the viewing frustum from the rendering process.

1.1 Overview of the Technique

In general, the view frustum culling technique described herein transfers Free Viewpoint Video (FVV) from a server to a client over a network, such as, for example, the Internet, or over a proprietary intranet.

The view frustum culling technique embodiments described herein generally involve providing a FVV that provides a consistent and manageable amount of data to a client despite the large amounts of data typically demanded to create and render the FVV. In one general embodiment, this is accomplished by first capturing a scene using an arrangement of sensors. This sensor arrangement includes a plurality of sensors that generate a plurality of streams of sensor data, where each stream represents the scene from a different geometric perspective. These streams of sensor data are input and calibrated, and then geometric proxies and texture data are generated from the calibrated streams of sensor data. The geometric proxies and texture data describe the scene as a function of time. Next, a current synthetic viewpoint of the scene is received from a client computing device via a data communication network. This current synthetic viewpoint was selected by an end user of the client computing device. Once a current synthetic viewpoint is received, the geometric proxies and texture data necessary to render the given synthetic viewpoint or view frustum are computed or selected by the server, for example, from a FVV database that stores that type of data generated using the scene proxies. These selected geometric proxies and texture data that depict at least a portion of the scene as viewed from the current synthetic viewpoint of the scene are transmitted to the client computing device via the data communication network for render at the client and to display to the end user of the client computing device.

From the perspective of a client computing device, a FVV produced as described above is played at the client in one general embodiment as follows. A request is received from an end user to display a FVV selection user interface screen that allows the end user to select a FVV available for playing. This FVV selection user interface screen is displayed on a display device, and an end user FVV selection is input. The end user FVV selection is then transmitted to a server via a data communication network. The client computing device then receives an instruction from the server via the data communication network to instantiate end user controls appropriate for the type of FVV selected. In response, an appropriate FVV control user interface is provided to the end user. The client computing device then monitors end user inputs via the FVV control user interface, and whenever an end user viewpoint navigation input is received, it is transmitted to the server via the data communication network. FVV geometric proxies and texture data to render the requested viewpoint or view frustum are then received from the server. These geometric proxies and texture data are rendered at the client so as to render at least a portion of the captured scene as it would be viewed from the last viewpoint the end user input, and is displayed on the aforementioned display device as it is received.

As discussed above, some embodiments of the view frustum culling technique transfer only the 3D geometry data and texture data necessary to render a specific viewpoint or view frustum from the server to the client. The synthetic viewpoint is then rendered by the client using the received 3D
geometry and texture data. This approach has the advantage of providing a consistent and manageable amount of data to a client, or several clients, because only the geometric data and texture data necessary to display a specific viewpoint or view frustum desired by a user of the client are sent to the client.

In some embodiments of the view frustum culling technique, however, some additional spatial and temporal data other than only that needed to render the client’s requested viewpoint or view frustum can be sent to the client from the server. In these embodiments the data necessary to support the view frustum is supplemented with additional geometry data and texture data that would be immediately used if the viewpoint was changed either spatially or temporally at the client. For example, geometry data and texture data at the edge of the view frustum for the selected viewpoint can be sent to the client.

Furthermore, in some embodiments of the view frustum culling technique, the FVV client has texture data and 3D geometric data stored locally if there is sufficient local processing power which can provide more fluid and seamless transitions of rendering a FVV scene as the virtual viewpoint is moved around within the scene. In addition, for static or non-moving elements of the scene, previously received 3D geometry or texture data can be cached locally on the client, eliminating the need for redundant data transfers.

An overview of the view frustum culling technique having been provided, the following paragraphs will describe exemplary processes and an exemplary architecture for practicing the view frustum culling technique.

FIG. 1 depicts one exemplary computer-implemented process 100 for streaming FVV to a client according to the view frustum culling technique. As shown in FIG. 1, block 102, only texture data (e.g., RGB data) and geometric data for a given view frustum is received at a client from a server. Next, a given viewpoint of the spatial three dimensional video is rendered and displayed at the client using only the downloaded texture and geometric data for the given view frustum, as shown in block 104. Texture data (e.g., RGB data) or geometric data which has not changed on the client does not have to be downloaded again.

A modification to the process described above is that in addition to only the data necessary to render a specific viewpoint or view frustum, some additional spatial or temporal data is also sent from the server to client. Small changes in the spatial or temporal navigation are anticipated and the data is sent to the client prior to rendering. For example, additional texture data and corresponding geometric data at the edges of the client’s requested viewpoint or view frustum is sent to the client in addition to the 3D geometry and texture data necessary to render the viewpoint requested by the client. More specifically, given a current viewpoint a user’s view of a scene will include a corresponding view frustum for which geometry data and texture data is sent. However, if the time it takes to send this data from the server to the client is known, how far a client’s position and viewpoint can change in this time can be computed. Hence it is possible to send the additional geometry and texture data corresponding to the maximum distance the user can move in the time it takes to send data from the server to the client. Additionally, additional geometry and texture data can be sent to client based on a predicted viewpoint based on the client’s rate of viewpoint change. This predicted viewpoint can be calculated, for example, by computing a maximum bounding volume that will contain the

user’s viewpoint based on the velocity the user is moving and the time it takes to transmit geometry data and texture data to the client. Additionally, a lower level of detail of geometric data can be sent to the client for viewpoints that the client has a lower probability of reaching. For example, if the user’s velocity (V) and the time it takes to send data from the server to the client (t) is known, one can compute that the most the user can move is P = P + tv, where P is the current location and P’ is the furthest they can move in time t. Furthermore, a user is less likely to see an object if need to move the entire allowable distance for it to come into view, which means that a lower level of detail can be sent for the object. Similarly, a lower level of detail of texture data and geometric data can be sent for objects in the distance of the client’s view frustum. Yet another variation of the process described above includes provisions for reducing detail based on the angular velocity of the camera required to bring objects into view, i.e., objects that are further away angularly will translate into faster camera motion, thus the rendering will be more motion blurred and less detail need be rendered.

FIG. 2 depicts another exemplary computer-implemented process 200 for sending a FVV from one or more servers to a client according to the view frustum culling technique. In the embodiment shown in FIG. 2, a scene is captured using an arrangement of sensors (block 202). This sensor arrangement includes a plurality of sensors that generate a plurality of streams of sensor data, where each stream represents the scene from a different geometric perspective. These streams of sensor data are input and calibrated (block 204), and then scene geometric data and texture data are generated via conventional means from the calibrated streams of sensor data and are stored at the server (block 206). The geometric data and texture data describe the scene as a function of time. Next, a current synthetic viewpoint of the scene or its associated view frustum is received from a client computing device via a data communication network (block 208). This current synthetic viewpoint can be accompanied by the client’s display characteristics if it is necessary to compute the view frustum for the current synthetic viewpoint. It is noted that this current synthetic viewpoint was selected by an end user of the client computing device. Once a current synthetic viewpoint is received, the geometric data and texture data to render the given synthetic viewpoint or view frustum are retrieved from the location where they were stored (e.g., from a database) at the server (block 210) and are transmitted to the client computing device via the data communication network for render and display to the end user of the client computing device (block 212).

FIG. 3 depicts another exemplary computer-implemented process 300 for playing FVV content at a client according to the view frustum culling technique. As shown in block 302, a user installs a FVV player on a local client. The user selects and requests a desired FVV stored on a server, as shown in block 304. The client receives a message from the server that tells the client to instantiate a FVV player with controls appropriate to the FVV type of the desired FVV, as shown in block 306, and the client instantiates the FVV player, as shown in block 308. The client then requests a desired viewpoint on server from the server, and if necessary sends the client’s display characteristics if it is necessary for the server to calculate the client’s view frustum, as shown in block 310. The server renders the desired viewpoint for the desired FVV, and sends the client only the 3D geometry data and texture data (e.g., RGB data) necessary to
render the client’s viewpoint/view frustum of the desired FVV, as shown in block 312. The client combines the 3D geometry data and texture data to render the desired viewpoint/view frustum at the client, as shown in block 314. The client then checks for user viewpoint navigation input and if there is any, the client sends navigation input (e.g. a request for a new viewpoint) to the server (block 316). The server can then render a viewpoint of the FVV based on the received navigation input and send the geometry data and texture data needed for the client to render the FVV for the new viewpoint which is received at the client, as shown in block 318, and blocks 310 through 318 can be repeated. For example, to change viewpoints, a new (typically user specified) viewpoint is sent from the client to the server, and a new FVV or other 3D spatial video is initiated from the new viewpoint at the server. The 3D geometry and texture data associated with the new viewpoint are retrieved, the FVV is rendered at the server and the necessary 3D geometry and texture data necessary to render the FVV or 3D spatial video for the viewpoint or viewpoint view frustum requested by the client that renders the FVV is transmitted to the client until a new viewpoint request is received.

As described with respect to FIG. 1, a modification to the exemplary process described in FIG. 3, is that in addition to only the data necessary to render a specific viewpoint or view frustum, some additional texture data and corresponding geometric data at the edges of the view frustum is sent to the client in addition to the 3D geometry and texture data necessary to render the viewpoint requested by the client. As discussed above with respect to FIG. 1, the client’s viewpoint can be predicted based on the client’s rate of viewpoint change; a lower level of detail of geometric data can be sent to the client for viewpoints that the client has a lower probability of reaching; and a lower level of detail of texture data and geometric data can be sent for objects in the distance of the client’s view frustum.

In some embodiments of the technique, the geometric data and texture data is stored as a spatial representation of all viewpoints possible. For example, the spatial representation of all viewpoints possible can be defined by three dimensional cells as shown in FIG. 4. A large three dimensional cell 402 can be subdivided into smaller three dimensional cells 404 and the four smaller three dimensional cells further be subdivided into even smaller three dimensional cells 406. The server can store the geometric data and texture data of the FVV in the increasingly subdivided three dimensional cells and the client can request specific cells corresponding to a desired viewpoint or view frustum to be rendered. Alternatively, the server can compute the cells to send to the client based on a viewpoint received from the client that the client wishes to render. In any of these embodiments, the three dimensional cells can be stored in a compressed format. The cells can also be used to provide the level of detail of texture data or geometric data desired. It should be noted that any spatial data structure can be used to represent the three dimensional cells discussed above. For example, an octree, a kd-tree or a bounding volume hierarchy structure could be used.

Exemplary processes for practicing the view frustum culling technique having been described, the following section discusses an exemplary architecture for practicing the technique.

FIG. 5 shows an exemplary architecture 500 for practicing one embodiment of the view frustum culling technique. As shown in FIG. 5, this exemplary architecture 500 includes a server 502, that can be a general purpose computing device 700, which will be discussed in greater detail with respect to FIG. 7. The server 502 includes a database 504 of FVV/spatial 3D videos 506. For each of the videos 506, the database 504 includes the texture data and geometric data for rendering all of the synthetic viewpoints of each of the FVV’s. The geometric data and texture data stored in the database 504 may have been previously calculated at the server via conventional means. Only the texture data and geometric data necessary to render a desired viewpoint or view frustum at the client is sent to the client. If the client 508 only provides a given viewpoint, the server 502 can compute the client’s viewpoint frustum in a viewpoint frustum computation module 510. Likewise, the client can compute the client’s viewpoint frustum in a viewpoint frustum computation module 512 on the client. The server 502 can determine which geometric data and texture data to send to the client by rendering the desired FVV for the desired viewpoint in a 3D renderer 514.

The client 508 includes a FVV or spatial video player 516 which can be used to view and navigate through a FVV or other 3D spatial video. The client 508 also includes a user interface 518 that includes a display and that allows a user 520 of the client 508 to input user data such as, for example, the particular video 506 that the user would like to interact with, the viewpoint or view frustum the user would like to view, changes in the viewpoint, and so forth. The client 508 also has a 3D renderer 522 that can render the given viewpoint of the desired free viewpoint video 506 at the client 508 using the downloaded texture and geometric data for the desired viewpoint. The client 508 can also include a data store 524 that can store various data, such as, for example, geometric and texture data previously sent to the client 508 from the server 502, so that the data does not have to be retransmitted from the server once it has been sent. Furthermore, the client 508 can also include a viewpoint predictor 526 that predicts a viewpoint in the free viewpoint video based on viewpoint navigation changes requested by the client or computed using a rate of change of the viewpoint that the client is viewing. If the client does not compute the predicted viewpoint, the server can also employ a viewpoint prediction module 528 to compute the predicted viewpoint based on the viewpoint navigation updates. Additionally, the client can employ a level of detail computation module 530 that can compute the level of detail for an image or geometric data best suited to display far away objects or other objects that can be displayed with less detail in the free viewpoint video. Likewise, the server can also have a level of detail computation module 532 that can compute the level of detail for an image or geometric data best suited to display objects that can be rendered with less detail in the free viewpoint video.

In one embodiment of the view frustum culling technique the architecture 500 could be used in the following manner to render a free viewpoint video at a client 508. The client 508 sends a request 534 for a specific free viewpoint video to the server 502. The server 502 then sends a command 536 to instantiate the FVV player 516 for the chosen video to the client 508. The client 508 instantiates the FVV player 516 and sends a request 538 for a current viewpoint of the FVV. The server 502 then sends the geometry and texture data necessary to render only the current viewpoint of the chosen FVV 540. The client 508 then renders the desired viewpoint of the desired FVV at the client using the received geometry and texture data. The client 508 can then send an updated
desired viewpoint or rate of change of the viewpoint 542 to the server 502, and in return the server 502 can send the geometry and texture data to render the desired updated viewpoint or a predicted viewpoint based on the viewpoint rate of change 544.

[0047] As discussed previously, some embodiments of the view frustum culling technique send, in addition to only the data necessary to render a specific viewpoint or view frustum, some additional spatial or temporal data from the server to the client. Small changes in the spatial or temporal navigation are anticipated and the geometric and texture data is sent to the client prior to rendering. For example, additional texture data and corresponding geometric data at the edges of the client’s requested viewpoint or view frustum is sent to the client in addition to the 3D geometry and texture data necessary to render the viewpoint requested by the client. In this case the client’s viewpoint can be predicted based on the client’s rate of viewpoint change in a viewpoint prediction module 528 on the server or in a viewpoint prediction module 526 on the client. Additionally, a lower level of detail of geometric data can be computed in a level of detail computation module 532 and can be sent to the client for viewpoints that the client has a lower probability of reaching. Similarly, a lower level of detail of texture data and geometric data can be sent for objects in the distance of the client’s view frustum. In one case a client may request a certain level of detail of geometric and/or texture data from the server and in this case the client may determine the level of detail desired in a level of detail computation module 530 on the client.

[0048] 1.5 Exemplary Spatial Video Pipeline
[0049] The view frustum culling technique described herein can be used in various scenarios. One way the technique can be used is in a system for generating Spatial Video (SV). The following paragraphs provide details of a spatial video pipeline in which the view frustum culling technique described herein can be used. The details of image capture, processing, storage and streaming, rendering and the user experience discussed with respect to this exemplary spatial video pipeline can apply to various similar processing actions discussed with respect to the exemplary processes and the exemplary architecture of the view frustum culling technique discussed above.

[0050] Spatial Video (SV) provides a next generation, interactive, and immersive video experiences relevant to both consumer entertainment and telepresence, leveraging applied technologies from Free Viewpoint Video (FVV). As such, SV encompasses a commercially viable system that supports features required for capturing, processing, distributing, and viewing any type of FVV media in a number of different product configurations.

[0051] It is noted, however, that view frustum culling technique embodiments described herein are not limited to only the exemplary FVV pipeline to be described. Rather, other FVV pipelines can also be employed to create and render video, as desired.

[0052] 1.5.1 Spatial Video Pipeline
[0053] SV requires an end to end processing and playback pipeline for any type of FVV that can be captured. Such a pipeline 600 is shown in FIG. 6, the primary components of which include: Capture 602; Process 604; Storage/Streaming 606; Render 608; and the User Experience 610.

[0054] The SV Capture 602 stage of the pipeline supports any hardware used in an array to record a FVV scene. This includes the use of various different kinds of sensors (including video cameras and audio) for recording data. When sensors are arranged in 3D space relative to a scene, their type, position, and orientation is referred to as the camera geometry. The SV pipeline generates the calibrated camera geometry for static arrays of sensors as well as for moving sensors at every point in time during the capture of a FVV. The SV pipeline is designed to work with any type of sensor data from any kind of an array, including, but not limited to RGB data from traditional cameras (including the use of structured light such as with Microsoft’s Corporation’s Kinect™), monochrome cameras, or time of flight (TOF) sensors that generate depth maps and RGB data directly. The SV pipeline is able to determine the intrinsic and extrinsic characteristics of any sensor in the array at any point in time. Intrinsic parameters such as the focal length, principal point, skew coefficient, and distortions are required to understand the governing physics and optics of a given sensor. Extrinsic parameters include both rotations and translations which detail the spatial location of the sensor as well as the direction the sensor is pointing. Typically, a calibration setup procedure is carried out that is specific to the type, number and placement of sensors. This data is often recorded in one or more calibration procedures prior to recording a specific FVV. If so, this data is imported into the SV pipeline in addition to any data recorded with the sensor array.

[0055] Variability associated with the FVV scene as well as playback navigation may impact how many sensors are used to record the scene as well as which type of sensors are selected and their positioning. SV typically includes at minimum one RGB sensor as well as one or more sensors that can be used in combination to generate 3D geometry describing a scene. Outdoor and long distance recording favors both wide baseline and narrow baseline RGB stereo sensor pairs. Indoor conditions favor narrow baseline stereo IR using structured light avoiding the dependency upon lighting variables. As the scene becomes more complex, for example as additional people are added, the use of additional sensors reduces the number of occluded areas within the scene—more complex scenes require better sensor coverage. Moreover, it is possible to capture both an entire scene at one sensor density and then to capture a secondary, higher resolution volume at the same time with additional moveable sensors targeting the secondary higher resolution area of the scene. As more sensors are used to reduce occlusion artifacts in the array, additional combinations of the sensors can also be used in processing such as when a specific sensor is part of both a narrow baseline stereo pair as well as a different wide baseline stereo pair involving a third sensor.

[0056] The SV pipeline is designed to support any combination of sensors in any combination of positions.

[0057] The SV Process 604 stage of the pipeline takes sensor data and extracts 3D geometric information that describes the recorded scene both spatially and temporally. Different types of 3DR algorithms are used depending on: the number and type of sensors, the input camera geometry, and whether processing is done in real time or asynchronously from the playback process. The output of the process stage is various geometric proxies which describe the scene as a function of time. Unlike video games or special effects technology, 3D geometry in the SV pipeline is created using automated computer vision 3DR algorithms with no human input required.

[0058] SV Storage and Streaming 606 methods are specific to different FVV product configurations, and these can be
segmented as: bidirectional live applications of FVV in telepresence, broadcast live applications of FVV, and asynchronous applications of FVV. Depending on details associated with these various product configurations, data is processed, stored, and distributed to end users in different manners.

[0059] The SV pipeline uses 3D reconstruction to process calibrated sensor data to create geometric proxies describing the FVV scene. The SV pipeline uses various 3D reconstruction approaches depending upon the type of sensors used to record the scene, the number of sensors, the positioning of the sensors relative to the scene, and how rapidly the scene needs to be reconstructed. 3D geometric proxies generated in this stage include depth maps, point based renderings, or higher order geometric forms such as planes, objects, billboards, models, or other high fidelity proxies such as mesh based representations. The SV Render 608 stage is based on image based rendering (IBR), since synthetic, or virtual, viewpoints of the scene are created using real images and different types of 3D geometry. SV render 608 uses different IBR algorithms to render synthetic viewpoints based on variables associated with the product configuration, hardware platform, scene complexity, end user experience, input camera geometry, and the desired degree of viewpoint navigation in the final FVV. Therefore, different IBR algorithms are used in the SV Rendering stage to maximize photorealism from any necessary synthetic viewpoints during end user playback of a FVV.

[0060] When the SV pipeline is used in real time applications, sensor data must be captured, processed, transmitted, and rendered in less than one thirtieth of a second. Because of this constraint, the types of 3D reconstruction algorithms that can be used are limited to high performance algorithms. Primarily, 3D reconstruction that is used real time includes point cloud based depictions of a scene or simplified proxies such as billboards or prior models which are either modified or animated. The use of active IR or structured light can assist in generating point clouds in real time since the pattern is known ahead of time. Algorithms that can be implemented in hardware are also favored.

[0061] Asynchronous 3D reconstruction removes the constraint of time from processing a FVV. This means that point based reconstructions of the scene can be used to generate higher fidelity geometric proxies, such as when point clouds are used as an input to create a geometric mesh describing surface geometry. The SV pipeline also allows multiple 3D reconstruction steps to be used when creating the most accurate geometric proxies describing the scene. For example, if a point cloud representation of the scene has been reconstructed, there may be some noisy or error prone stereo matches present that extend the boundary of the human silhouette, leading to the wrong textures appearing on a mesh surface. To remove these artifacts, the SV pipeline runs a segmentation process to separate the foreground from the background, so that points outside of the silhouette are rejected as outliers.

[0062] In another example of 3D reconstruction, a FVV is created with eight genlocked devices from a circular camera geometry each device consisting of: 1 IR randomized structured light projector, 2 IR cameras, and 1 RGB camera. Firstly, IR images are used to generate a depth map. Multiple depth maps and RGB images from different devices are used to create a 3D point cloud. Multiple point clouds are combined and meshed. Finally, RGB image data is mapped to the geometric mesh in the final result, using a view dependent texture mapping approach which accurately represents specular textures such as skin. [0063] The SV User Experience 610 processes data so that navigation is possible with up to 6 degrees of freedom (DOF) during FVV playback. In non-live applications, temporal navigation is possible as well—the present navigation is possible as well—this is spatiotemporal (or space-time) navigation. Viewpoint navigation means users can change their viewpoint (what is seen on a display interface) in real time, relative to moving video. In this way, the video viewpoint can be continuously controlled or updated during playback of a FVV scene.

[0064] 2.0 Exemplary Operating Environments:

[0065] The view frustum culling technique described herein is operational within numerous types of general purpose or special purpose computing system environments or configurations. FIG. 7 illustrates a simplified example of a general-purpose computer system on which various embodiments and elements of the view frustum culling technique, as described herein, may be implemented. It should be noted that any boxes that are represented by broken or dashed lines in FIG. 7 represent alternate embodiments of the simplified computing device, and that any or all of these alternate embodiments, as described below, may be used in combination with other alternate embodiments that are described throughout this document.

[0066] For example, FIG. 7 shows a general system diagram showing a simplified computing device 700. Such computing devices can be typically be found in devices having at least some minimum computational capability, including, but not limited to, personal computers, server computers, hand-held computing devices, laptop or mobile computers, communications devices such as cell phones and PDA’s, multi-processor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, audio or video media players, etc.

[0067] To allow a device to implement the view frustum culling technique, the device should have a sufficient computational capability and system memory to enable basic computational operations. In particular, as illustrated by FIG. 7, the computational capability is generally illustrated by one or more processing unit(s) 710, and may also include one or more GPUs 715, either or both in communication with system memory 720. Note that the processing unit(s) 710 of the general computing device may be specialized microprocessors, such as a DSP, a VLIW, or other micro-controller, or can be conventional CPUs having one or more processing cores, including specialized GPU-based cores in a multi-core CPU.

[0068] In addition, the simplified computing device of FIG. 7 may also include other components, such as, for example, a communications interface 730. The simplified computing device of FIG. 7 may also include one or more conventional computer input devices 740 (e.g., pointing devices, keyboards, audio input devices, video input devices, input devices, devices for receiving wired or wireless data transmissions, etc.). The simplified computing device of FIG. 7 may also include other optional components, such as, for example, one or more conventional computer output devices 750 (e.g., display device(s) 755, audio output devices, output devices, devices for transmitting wired or wireless data transmissions, etc.). Note that typical communications interfaces 730, input devices 740, output devices 750, and storage
devices 760 for general-purpose computers are well known to those skilled in the art, and will not be described in detail herein.

[0069] The simplified computing device of FIG. 7 may also include a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 700 via storage devices 760 and includes both volatile and nonvolatile media that is either removable 770 and/or non-removable 680, for storage of information such as computer-readable or computer-executable instructions, data structures, program modules, or other data. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes, but is not limited to, computer or machine readable media or storage devices such as DVD's, CD's, floppy disks, tape drives, hard drives, optical drives, solid state memory devices, RAM, ROM, EPROM, flash memory or other memory technology, magnetic cassettes, magnetic tapes, magnetic disk storage, or other magnetic storage devices, or any other device which can be used to store the desired information and which can be accessed by one or more computing devices.

[0070] Storage of information such as computer-readable or computer-executable instructions, data structures, program modules, etc., can also be accomplished by using any of a variety of the aforementioned communication media to encode one or more modulated data signals or carrier waves, or other transport mechanisms or communications protocols, and includes any wired or wireless information delivery mechanism. Note that the terms “modulated data signal” or “carrier wave” generally refer a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. For example, communication media includes wired media such as a wired network or direct-wired connection carrying one or more modulated data signals, and wireless media such as acoustic, RF, infrared, laser, and other wireless media for transmitting and/or receiving one or more modulated data signals or carrier waves. Combinations of the any of the above should also be included within the scope of communication media.

[0071] Further, software, programs, and/or computer program products embodying the some or all of the various embodiments of the view frustum culling technique described herein, or portions thereof, may be stored, received, transmitted, or read from any desired combination of computer or machine readable media or storage devices and communication media in the form of computer executable instructions or other data structures.

[0072] Finally, the view frustum culling technique described herein may be further described in the general context of computer-executable instructions, such as program modules, being executed by a computing device. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. The embodiments described herein may also be practiced in distributed computing environments where tasks are performed by one or more remote processing devices, or within a cloud of one or more devices, that are linked through one or more communications networks. In a distributed computing environment, program modules may be located in both local and remote computer storage media including media storage devices. Still further, the aforementioned instructions may be implemented, in part or in whole, as hardware logic circuits, which may or may not include a processor.

[0073] It should also be noted that any or all of the aforementioned alternate embodiments described herein may be used in any combination desired to form additional hybrid embodiments. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. The specific features and acts described above are disclosed as example forms of implementing the claims.

What is claimed is:

1. A computer-implemented process for receiving spatial three dimensional video, comprising:

   using a client computing device for:

   receiving only texture data and geometric data for a given view frustum of a spatial three dimensional video from a server at a client;

   rendering the given viewpoint of the spatial three dimensional video at the client using the downloaded texture and geometric data for the given view frustum.

2. The computer-implemented process of claim 1 wherein the client specifies the given view frustum to the server before the texture data and geometric data are downloaded to the client.

3. The computer-implemented process of claim 1 wherein the client receives texture data and geometric data computed by the client based on a viewpoint received from the client.

4. The computer-implemented process of claim 1, further comprising:

   checking if texture data or geometric data has been previously downloaded to the client; and
   not downloading the texture data or the geometric data when it has previously downloaded to the client.

5. The computer-implemented process of claim 1 wherein additional texture data and corresponding geometric data at the edges of the view frustum is received at the client.

6. The computer-implemented process of claim 1 wherein the client's viewpoint is predicted based on the client's rate of viewpoint change.

7. The computer-implemented process of claim 6 wherein the viewpoint is expanded based on the client's predicted viewpoint.

8. The computer-implemented process of claim 6 wherein the view frustum is expanded based on the client's predicted viewpoint.

9. The computer-implemented process of claim 6 wherein the view frustum is expanded based on the client's predicted viewpoint.

10. The computer-implemented process of claim 1 wherein the geometric data is stored as a spatial representation of all viewpoints possible.

11. The computer-implemented process of claim 1 wherein the spatial representation of all viewpoints possible is defined by three dimensional cells.

12. The computer-implemented process of claim 11 wherein the server stores the cells and wherein the client requests specific cells corresponding to a desired viewpoint to be rendered.

13. The computer-implemented process of claim 11 wherein the server computes the cells to send to the client based on a viewpoint the client wishes to render.
14. The computer-implemented process of claim 11 wherein the three dimensional cells are in a compressed format.

15. A computer-implemented process for receiving free viewpoint video, comprising:
   using a client computing device for:
   installing a free viewpoint video player on a local client;
   selecting a free viewpoint video stored on a server;
   receiving a message from the server that tells the client to instantiate the free viewpoint video player with controls appropriate to the selected free viewpoint video type;
   instantiating the free viewpoint video player with controls appropriate to the selected free viewpoint video type;
   requesting a desired viewpoint of the selected free viewpoint video from the server;
   receiving only the necessary geometric and texture data to render the desired viewpoint of the selected viewpoint video; and
   combining the received geometric and texture data to render the desired viewpoint of the free viewpoint video

16. The computer-implemented process of claim 15 further comprising:
   the client checking for user viewpoint navigation input; and
   if there is any user viewpoint navigation input, the client sending the navigation input to the server.

17. The computer-implemented process of claim 16 wherein the server uses the client's navigation input to determine which 3D geometry and texture data to next send to the client.

18. A system for providing free viewpoint video, comprising:
   a general purpose computing device;
   a computer program comprising program modules executable by the general purpose computing device, wherein the computing device is directed by the program modules of the computer program to,
   download only texture data and geometric data relevant to a given viewpoint of a free viewpoint video at a client;
   render the given viewpoint of the free viewpoint video at the client using only the downloaded texture and geometric data for the given viewpoint.

19. The system of claim 18 wherein the downloaded texture data and the downloaded geometric data is slightly greater than required to render the given viewpoint.