

MULTIDIMENSIONAL METADATA FOR PARALLEL PROCESSING OF THE SEGMENTED MEDIA DATA

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is based on and claims priority to U.S. Provisional Patent Application No. 63/298,534 filed on January 11, 2022, and U.S. Patent Application No. 17/993,583 filed on November 23, 2022, the disclosures of which are incorporated by reference herein in their entirety.

FIELD

[0002] The disclosure provides an extension to the one-dimensional NBMP segment metadata to support splitting and merging multidimensional media data into segments that each can be processed independently.

BACKGROUND

[0003] Network and cloud platforms are used to run various applications. The network-based media processing (NBMP) standard defines a set of tools for the independent processing of 1-dimensional media segments. However, when the media is multidimensional, there might be a need to segment the media data in multiple dimensions, which is not currently addressed by the NBMP standard.

SUMMARY

[0004] According some embodiments, there is provided a method executed by at least one processor. The method includes segmenting a multidimensional media stream into a plurality

of segments of multidimensional media in a multidimensional space. The method further includes representing each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode. The method further includes deriving a network based media processing (NBMP) workflow based on the respective sequence vectors of each segment of the plurality of segments.

[0005] According to some embodiments, an apparatus includes at least one memory configured to store program code and at least one processor configured to read the program code and operate as instructed by the program code. The program code includes segmenting code configured to cause the at least one processor to segment a multidimensional media stream into a plurality of segments of multidimensional media in a multidimensional space. The program code further includes representing code configured to cause the at least one processor to represent each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode. The program code further includes deriving code configured to cause the at least one processor to derive a network based media processing (NBMP) workflow based on the respective sequence vectors of each segment of the plurality of segments;

According to some embodiments, a non-transitory computer-readable storage medium, stores instructions that, when executed by at least one processor, cause the at least one processor

to segment a multidimensional media stream into a plurality of segments of multidimensional media in a multidimensional space. The instructions further cause the at least one processor to represent each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode. The instructions further cause the at least one processor to derive a network based media processing (NBMP) workflow based on the respective sequence vectors of each segment of the plurality of segments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] FIG. 1 is a diagram of an environment in which methods, apparatuses and systems described herein may be implemented, according to some embodiments.

[0007] FIG. 2 is a block diagram of example components of one or more devices, according to some embodiments.

[0008] FIG. 3 is a block diagram of an NBMP system, according to some embodiments.

[0009] FIG. 4 is diagram of a workflow for parallel processing of segments, according to some embodiments.

[0010] FIG. 5 is a block diagram of an apparatus for processing media content in NBMP, according to some embodiments.

DETAILED DESCRIPTION

[0011] The following detailed description of example embodiments refers to the accompanying drawings. The same reference numbers in different drawings may identify the same or similar elements.

[0012] The foregoing disclosure provides illustration and description, but is not intended to be exhaustive or to limit the implementations to the precise form disclosed. Modifications and variations are possible in light of the above disclosure or may be acquired from practice of the implementations. Further, one or more features or components of one embodiment may be incorporated into or combined with another embodiment (or one or more features of another embodiment). Additionally, in the flowcharts and descriptions of operations provided below, it is understood that one or more operations may be omitted, one or more operations may be added, one or more operations may be performed simultaneously (at least in part), and the order of one or more operations may be switched.

[0013] It will be apparent that systems and/or methods, described herein, may be implemented in different forms of hardware, firmware, or a combination of hardware and software. The actual specialized control hardware or software code used to implement these systems and/or methods is not limiting of the implementations. Thus, the operation and behavior of the systems and/or methods were described herein without reference to specific software code—it being understood that software and hardware may be designed to implement the systems and/or methods based on the description herein.

[0014] Even though particular combinations of features are recited in the claims and/or disclosed in the specification, these combinations are not intended to limit the disclosure of possible implementations. In fact, many of these features may be combined in ways not specifically recited in the claims and/or disclosed in the specification. Although each dependent claim listed below may directly depend on only one claim, the disclosure of possible

implementations includes each dependent claim in combination with every other claim in the claim set.

[0015] No element, act, or instruction used herein should be construed as critical or essential unless explicitly described as such. Also, as used herein, the articles “a” and “an” are intended to include one or more items, and may be used interchangeably with “one or more.” Where only one item is intended, the term “one” or similar language is used. Also, as used herein, the terms “has,” “have,” “having,” “include,” “including,” or the like are intended to be open-ended terms. Further, the phrase “based on” is intended to mean “based, at least in part, on” unless explicitly stated otherwise. Furthermore, expressions such as “at least one of [A] and [B]” or “at least one of [A] or [B]” are to be understood as including only A, only B, or both A and B.

[0016] Reference throughout this specification to “one embodiment,” “an embodiment,” or similar language means that a particular feature, structure, or characteristic described in connection with the indicated embodiment is included in at least one embodiment of the present solution. Thus, the phrases “in one embodiment,” “in an embodiment,” and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

[0017] Furthermore, the described features, advantages, and characteristics of the present disclosure may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize, in light of the description herein, that the present disclosure may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the present disclosure.

[0018] FIG. 1 is a diagram of an environment 100 in which methods, apparatuses and systems described herein may be implemented, according to embodiments. As shown in FIG. 1, the environment 100 may include a user device 110, a platform 120, and a network 130. Devices of the environment 100 may interconnect via wired connections, wireless connections, or a combination of wired and wireless connections.

[0019] The user device 110 includes one or more devices capable of receiving, generating, storing, processing, and/or providing information associated with platform 120. For example, the user device 110 may include a computing device (*e.g.*, a desktop computer, a laptop computer, a tablet computer, a handheld computer, a smart speaker, a server, etc.), a mobile phone (*e.g.*, a smart phone, a radiotelephone, etc.), a wearable device (*e.g.*, a pair of smart glasses or a smart watch), or a similar device. In some implementations, the user device 110 may receive information from and/or transmit information to the platform 120.

[0020] The platform 120 includes one or more devices as described elsewhere herein. In some implementations, the platform 120 may include a cloud server or a group of cloud servers. In some implementations, the platform 120 may be designed to be modular such that software components may be swapped in or out depending on a particular need. As such, the platform 120 may be easily and/or quickly reconfigured for different uses.

[0021] In some implementations, as shown, the platform 120 may be hosted in a cloud computing environment 122. Notably, while implementations described herein describe the platform 120 as being hosted in the cloud computing environment 122, in some implementations, the platform 120 may not be cloud-based (*i.e.*, may be implemented outside of a cloud computing environment) or may be partially cloud-based.

[0022] The cloud computing environment 122 includes an environment that hosts the platform 120. The cloud computing environment 122 may provide computation, software, data access, storage, etc. services that do not require end-user (*e.g.*, the user device 110) knowledge of a physical location and configuration of system(s) and/or device(s) that hosts the platform 120. As shown, the cloud computing environment 122 may include a group of computing resources 124 (referred to collectively as “computing resources 124” and individually as “computing resource 124”).

[0023] The computing resource 124 includes one or more personal computers, workstation computers, server devices, or other types of computation and/or communication devices. In some implementations, the computing resource 124 may host the platform 120. The cloud resources may include compute instances executing in the computing resource 124, storage devices provided in the computing resource 124, data transfer devices provided by the computing resource 124, etc. In some implementations, the computing resource 124 may communicate with other computing resources 124 via wired connections, wireless connections, or a combination of wired and wireless connections.

[0024] As further shown in FIG. 1, the computing resource 124 includes a group of cloud resources, such as one or more applications (“APPs”) 124-1, one or more virtual machines (“VMs”) 124-2, virtualized storage (“VSs”) 124-3, one or more hypervisors (“HYPs”) 124-4, or the like.

[0025] The application 124-1 includes one or more software applications that may be provided to or accessed by the user device 110 and/or the platform 120. The application 124-1 may eliminate a need to install and execute the software applications on the user device 110. For

example, the application 124-1 may include software associated with the platform 120 and/or any other software capable of being provided via the cloud computing environment 122. In some implementations, one application 124-1 may send/receive information to/from one or more other applications 124-1, via the virtual machine 124-2.

[0026] The virtual machine 124-2 includes a software implementation of a machine (*e.g.*, a computer) that executes programs like a physical machine. The virtual machine 124-2 may be either a system virtual machine or a process virtual machine, depending upon use and degree of correspondence to any real machine by the virtual machine 124-2. A system virtual machine may provide a complete system platform that supports execution of a complete operating system (“OS”). A process virtual machine may execute a single program, and may support a single process. In some implementations, the virtual machine 124-2 may execute on behalf of a user (*e.g.*, the user device 110), and may manage infrastructure of the cloud computing environment 122, such as data management, synchronization, or long-duration data transfers.

[0027] The virtualized storage 124-3 includes one or more storage systems and/or one or more devices that use virtualization techniques within the storage systems or devices of the computing resource 124. In some implementations, within the context of a storage system, types of virtualizations may include block virtualization and file virtualization. Block virtualization may refer to abstraction (or separation) of logical storage from physical storage so that the storage system may be accessed without regard to physical storage or heterogeneous structure. The separation may permit administrators of the storage system flexibility in how the administrators manage storage for end users. File virtualization may eliminate dependencies between data accessed at a file level and a location where files are physically stored. This may

enable optimization of storage use, server consolidation, and/or performance of non-disruptive file migrations.

[0028] The hypervisor 124-4 may provide hardware virtualization techniques that allow multiple operating systems (*e.g.*, “guest operating systems”) to execute concurrently on a host computer, such as the computing resource 124. The hypervisor 124-4 may present a virtual operating platform to the guest operating systems, and may manage the execution of the guest operating systems. Multiple instances of a variety of operating systems may share virtualized hardware resources.

[0029] The network 130 includes one or more wired and/or wireless networks. For example, the network 130 may include a cellular network (*e.g.*, a fifth generation (5G) network, a long-term evolution (LTE) network, a third generation (3G) network, a code division multiple access (CDMA) network, etc.), a public land mobile network (PLMN), a local area network (LAN), a wide area network (WAN), a metropolitan area network (MAN), a telephone network (*e.g.*, the Public Switched Telephone Network (PSTN)), a private network, an ad hoc network, an intranet, the Internet, a fiber optic-based network, or the like, and/or a combination of these or other types of networks.

[0030] The number and arrangement of devices and networks shown in FIG. 1 are provided as an example. In practice, there may be additional devices and/or networks, fewer devices and/or networks, different devices and/or networks, or differently arranged devices and/or networks than those shown in FIG. 1. Furthermore, two or more devices shown in FIG. 1 may be implemented within a single device, or a single device shown in FIG. 1 may be implemented as multiple, distributed devices. Additionally, or alternatively, a set of devices (*e.g.*,

one or more devices) of the environment 100 may perform one or more functions described as being performed by another set of devices of the environment 100.

[0031] FIG. 2 is a block diagram of example components of one or more devices of FIG.

1. The device 200 may correspond to the user device 110 and/or the platform 120. As shown in FIG. 2, device 200 may include a bus 210, a processor 220, a memory 230, a storage component 240, an input component 250, an output component 260, and a communication interface 270.

[0032] The bus 210 includes a component that permits communication among the components of the device 200. The processor 220 is implemented in hardware, firmware, or a combination of hardware and software. The processor 220 is a central processing unit (CPU), a graphics processing unit (GPU), an accelerated processing unit (APU), a microprocessor, a microcontroller, a digital signal processor (DSP), a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), or another type of processing component. In some implementations, the processor 220 includes one or more processors capable of being programmed to perform a function. The memory 230 includes a random access memory (RAM), a read only memory (ROM), and/or another type of dynamic or static storage device (*e.g.*, a flash memory, a magnetic memory, and/or an optical memory) that stores information and/or instructions for use by the processor 220.

[0033] The storage component 240 stores information and/or software related to the operation and use of the device 200. For example, the storage component 240 may include a hard disk (*e.g.*, a magnetic disk, an optical disk, a magneto-optic disk, and/or a solid state disk), a compact disc (CD), a digital versatile disc (DVD), a floppy disk, a cartridge, a magnetic tape,

and/or another type of non-transitory computer-readable medium, along with a corresponding drive.

[0034] The input component 250 includes a component that permits the device 200 to receive information, such as via user input (*e.g.*, a touch screen display, a keyboard, a keypad, a mouse, a button, a switch, and/or a microphone). Additionally, or alternatively, the input component 250 may include a sensor for sensing information (*e.g.*, a global positioning system (GPS) component, an accelerometer, a gyroscope, and/or an actuator). The output component 260 includes a component that provides output information from the device 200 (*e.g.*, a display, a speaker, and/or one or more light-emitting diodes (LEDs)).

[0035] The communication interface 270 includes a transceiver-like component (*e.g.*, a transceiver and/or a separate receiver and transmitter) that enables the device 200 to communicate with other devices, such as via a wired connection, a wireless connection, or a combination of wired and wireless connections. The communication interface 270 may permit the device 200 to receive information from another device and/or provide information to another device. For example, the communication interface 270 may include an Ethernet interface, an optical interface, a coaxial interface, an infrared interface, a radio frequency (RF) interface, a universal serial bus (USB) interface, a Wi-Fi interface, a cellular network interface, or the like.

[0036] The device 200 may perform one or more processes described herein. The device 200 may perform these processes in response to the processor 220 executing software instructions stored by a non-transitory computer-readable medium, such as the memory 230 and/or the storage component 240. A computer-readable medium is defined herein as a non-

transitory memory device. A memory device includes memory space within a single physical storage device or memory space spread across multiple physical storage devices.

[0037] Software instructions may be read into the memory 230 and/or the storage component 240 from another computer-readable medium or from another device via the communication interface 270. When executed, software instructions stored in the memory 230 and/or the storage component 240 may cause the processor 220 to perform one or more processes described herein. Additionally, or alternatively, hardwired circuitry may be used in place of or in combination with software instructions to perform one or more processes described herein. Thus, implementations described herein are not limited to any specific combination of hardware circuitry and software.

[0038] The number and arrangement of components shown in FIG. 2 are provided as an example. In practice, the device 200 may include additional components, fewer components, different components, or differently arranged components than those shown in FIG. 2. Additionally, or alternatively, a set of components (*e.g.*, one or more components) of the device 200 may perform one or more functions described as being performed by another set of components of the device 200.

[0039] In some embodiments, an NBMP system 300 is provided. With reference to FIG. 3, the NBMP system 300 comprises an NBMP source 310, an NBMP workflow manager 320, a function repository 330, one or more media processing entities 350, a media source 360, and a media sink 370.

[0040] The NBMP source 310 may receive instructions from a third party entity, may communicate with the NBMP workflow manager 320 via an NBMP workflow, and may

communicate with the function repository 330 via a function discovery API 391. For example, the NBMP source 310 may send a workflow description document(s) (WDD) to the NBMP workflow manager 320, and may read the function description of functions stored in the function repository 330, the functions being media processing functions stored in memory of the function repository 330 such as, for example, functions of media decoding, feature point extraction, camera parameter extraction, projection method, seam information extraction, blending, post-processing, and encoding. The NBMP source 310 may comprise or be implemented by at least one processor and memory that stores code configured to cause the at least processor to perform the functions of the NBMP source 310.

[0041] The NBMP source 310 may request the NBMP workflow manager 320 to create workflow including tasks 352 to be performed by the one or more media processing entities 350 by sending the workflow description document, which may include several descriptors, each of which may have several parameters.

[0042] For example, the NBMP source 310 may select functions stored in the function repository 330 and send the workflow description document to the NBMP workflow manager 320 that includes a variety of descriptors for description details such as input and output data, required functions, and requirements for the workflow. The workflow description document may include a set of task descriptions and a connection map of inputs and outputs of tasks 352 to be performed by one or more of the media processing entities 350. When the NBMP workflow manager 320 receives such information from the NBMP source 310, the NBMP workflow manager 320 may create the workflow by instantiating the tasks based on function names and connecting the tasks in accordance with the connection map.

[0043] Alternatively or additionally, the NBMP source 310 may request the NBMP workflow manager 320 to create workflow by using a set of keywords. For example, NBMP source 310 may send the NBMP workflow manager 320 the workflow description document that may include a set of keywords that the NBMP workflow manager 320 may use to find appropriate functions stored in the function repository 330. When the NBMP workflow manager 320 receives such information from the NBMP source 310, the NBMP workflow manager 320 may create the workflow by searching for appropriate functions using the keywords that may be specified in a Processing Descriptor of the workflow description document, and use the other descriptors in the workflow description document to provision tasks and connect them to create the workflow.

[0044] The NBMP workflow manager 320 may communicate with the function repository 330 via a function discovery API 393, which may be a same or different API from the function discovery API 391, and may communicate with one or more of the media processing entities 350 via an API 394 (e.g. an NBMP task API). The NBMP workflow manager 320 may comprise or be implemented by at least one processor and memory that stores code configured to cause the at least processor to perform the functions of the NBMP workflow manager 320.

[0045] The NBMP workflow manager 320 may use the API 394 to setup, configure, manage, and monitor one or more tasks 352 of a workflow that is performable by the one or more media processing entities 350. In some embodiments, the NBMP workflow manager 320 may use the API 394 to update and destroy the tasks 352. In order to configure, manage, and monitor tasks 352 of the workflow, the NBMP workflow manager 320 may send messages, such as requests, to one or more of the media processing entities 350, wherein each message may have

several descriptors, each of which have several parameters. The tasks 352 may each include media processing functions 354 and configurations 353 for the media processing functions 354.

[0046] In some embodiments, after receiving a workflow description document from the NBMP source 310 that does not include a list of the tasks (e.g. includes a list of keywords instead of a list of tasks), the NBMP workflow manager 320 may select the tasks based on the descriptions of the tasks in the workflow description document to search the function repository 330, via the function discovery API 393, to find the appropriate functions to run as tasks 352 for a current workflow. For example, the NBMP workflow manager 320 may select the tasks based on keywords provided in the workflow description document. After the appropriate functions are identified by using the keywords or the set of task descriptions that is provided by the NBMP source 310, the NBMP workflow manager 320 may configure the selected tasks in the workflow by using the API 394. For example, the NBMP workflow manager 320 may extract configuration data from information received from the NBMP source, and configure the tasks 352 based on the configuration data.

[0047] The one or more media processing entities 350 may be configured to receive media content from the media source 360, process the media content in accordance with the workflow, that includes tasks 352, created by the NBMP workflow manager 320, and output the processed media content to the media sink 370. The one or more media processing entities 350 may each comprise or be implemented by at least one processor and memory that stores code configured to cause the at least processor to perform the functions of the media processing entities 350.

[0048] The media source 360 may include memory that stores media and may be integrated with or separate from the NBMP source 310. In some embodiments, the NBMP workflow manager 320 may notify the NBMP source 310 when a workflow is prepared and the media source 360 may transmit media content to the one or more of the media processing entities 350 based on the notification that the workflow is prepared.

[0049] The media sink 370 may comprise or be implemented by at least one processor and at least one display that is configured to display the media that is processed by the one or more media processing entities 350.

[0050] As discussed above, messages from the NBMP Source 310 (e.g. a workflow description document for requesting creation of a workflow) to the NBMP workflow manager 320, and messages (e.g. for causing the workflow to be performed) from the NBMP workflow manager 320 to the one or more media processing entities 350 may include several descriptors, each of which may have several parameters. In cases, communication between any of the components of the NBMP system 300 using an API may include several descriptors, each of which may have several parameters.

[0051] Embodiments may relate to a method to identify and signal the nonessential inputs, outputs and tasks in a workflow run on the cloud platforms.

[0052] An essential output of a workflow may be an output of that workflow that must produce data for the workflow to be considered as operating properly. An essential input of the workflow may be an input that must be processed for the workflow to create its essential outputs. A properly operating workflow may be a workflow that processes all of its essential inputs and

produces all of its essential outputs. An essential task of a workflow may be a task necessary to operate properly and process data that is required for a properly operating workflow. For example, an essential task may be a task that processes an essential input, and/or produces an essential output. In some embodiments, an essential input may be an input that is needed for an essential task to operate, and an essential output may be an output that is needed as an essential input for an essential task, or an output that is needed as an output for the workflow as a whole. A nonessential input may be an input that is not needed by the workflow in order to produce the essential outputs of the workflow. For example, a workflow may produce all of the essential outputs if all of the essential inputs are provided, even if none of the nonessential inputs are provided. A nonessential task may be a task that is included in the workflow, but that is not an essential task. For example, a nonessential task may process a nonessential input and produce a nonessential output.

[0053] The NBMP standard defines the splitter and merger function template. FIG. 4 shows a diagram of a workflow 400 corresponding to an example of this process of using NBMP splitter/merger functions for parallel processing of the segments. In FIG. 4, Task T 410 may be converted to n instances of Task T, running in parallel.

[0054] In some embodiments, the media stream is continuous. The splitter 420 may convert the media stream to N media sub-streams. Each sub-stream may be processed by an instance of T 430A-C and then the sub-streams are interleaved together at merger 440 to generate the output 450, equivalent of Task T 410 output stream. 1: N splitter and N :1 merger functions work on the segment boundaries. Each segment may have a start, duration, and length metadata, or a start code and a sequence number associated with it. Since the segments are

independent, the sub-streams are independent of each other in terms of being processed by Task T 410. Task $T_0, \dots, T_{N-1}$, do not need to process the segments at the same time. Since the segments and sub-streams may be independent, each instance of Task T 410 may run at its speed. However, the current NBMP standard only addresses the 1-D segmentation of the media data.

[0055] The current NBMP standard defines the following formats for the segment metadata. Each segment satisfies the following requirements:

- A. a continuous set of samples
- B. the maximum duration of D in the scale of time-scale T, where D and T are configuration parameters

[0056] Each segment may use one of the following metadata:

1. The timing metadata:

- A. Start time s in time-scale t,
- B. time-scale $t = T$,
- C. duration d in time scale t,
- D. length l (bytes)

2. Or, the sequence metadata:

- A. Identical and unique startcode in all other segments
- B. A sequence number in increasing order.

[0057] In both cases, the media are segmented only in one dimension (e.g., typically time). However, media signals tend to be multidimensional.

[0058] The embodiments of the present disclosure include multidimensional metadata for multidimensional segment processing using:

1. The start and lengths
2. The startcode and the sequence vector.

[0059] The following definitions may be used:

- **C** is a vector $[c_0, c_1, \dots, c_{M-1}]$ with M dimension, with element c_i with index i, where index i+1 is nested in index i, meaning one increment of index i of the vector is considered a larger increase than any increment in indices i+1, i+2, ..., M-1, where $0 \leq i < M$.
- A multidimensional segment with dimension M can be defined as a segment representing the information regarding samples in space starting at point **S** = $[s_0, s_1, \dots, s_{M-1}]$ and length **D** = $[d_0, d_1, \dots, d_{M-1}]$, where s_i and d_i are non-negative integer numbers. If non-integer values are needed, then vector **T** = $[t_0, t_1, \dots, t_{M-1}]$ represents the scale factor t_i for dimension i, in which the actual starting point and length in the dimension i are s_i/t_i and d_i/t_i respectively, where t_i is a positive integer number.
- Scaling vector **T** = $[t_0, t_1, \dots, t_{M-1}]$, the scale factors for **S** and **D**
- Starting vector **S** = $[s_0, s_1, \dots, s_{M-1}]$ representing the starting point of the media segment in M dimensional space with each index s_i in the unit t_i
- Length vector **D** = $[d_0, d_1, \dots, d_{M-1}]$ representing the hyperspace the media segment covering in M dimensional space with each index d_i in the unit t_i
- The size of segment L in bytes.

[0060] The parameters s_i and d_i may be nonnegative integer numbers and t_i is positive an integer number. The segments may be ordered in increasing order as defined by the following definition:

- A segment K with starting vector $[s_{K,0}, \dots, s_{K,M-1}]$ is earlier than segment L with starting point $[s_{L,0}, \dots, s_{L,M-1}]$ if
 - $s_{K,i} < s_{L,i}$ for at least one i , $0 \leq i < M$
 - $s_{K,j} \leq s_{L,j}$ for any $0 \leq j < i$
- An increasing order set of segments is a 1-D sequence of segments in which each segment is located after all other segments earlier than itself in the sequence.

[0061] Each segment may carry the following metadata:

- $\underline{t}$, $\underline{s}$, $\underline{d}$, L

[0062] The following metadata may be defined for each segment as follows:

- Sequence vector $\underline{n} = [n_0, n_1, \dots, n_{M-1}]$ representing the sequence of the media segment in M dimensional space with each index n_i .
- Startcode C, a unique code that every segment starts with, and the code is not repeated in the middle of any segments.
- The size of segment L in bytes.

[0063] The parameters s_i and d_i may be nonnegative integer numbers and t_i may be a positive integer number. The segments may be ordered in increasing order as defined by the following definition:

- A segment K with sequence vector $[n_{K,0}, \dots, n_{K,M-1}]$ is earlier than segment L with starting point $[n_{L,0}, \dots, n_{L,M-1}]$ if
 - o $n_{K,i} < n_{L,i}$ for at least one i , $0 \leq i < M$
 - o $n_{K,j} \leq n_{L,j}$ for any $0 \leq j < i$
- In an increasing order set of segments is a 1-D sequence of segments in which each segment is located after all other segments earlier than itself in the sequence.

[0064] Each segment may carry the following metadata:

- $\underline{n}$, C, L

[0065] In the case of $M=1$ (e.g., 1 dimensional), the above design may be reduced to the existing 1-D metadata:

- a. Segment timed metadata: t, s, d, L
- b. Segment start code: n,c, L

[0066] FIG. 5 is a flowchart of example process 500 for signaling multidimensional metadata.

In some implementations, one or more process blocks of FIG. 5 may be performed by any of the elements discussed above, for example NBMP system 300 or any element included therein, for example NBMP workflow manager 320.

[0067]

[0068] As shown in FIG. 5, process 500 include segmenting the multidimensional media streams into a sequence of a plurality of segments (block 510).

[0069] As further shown in FIG. 5, the process 500 may include deriving a network based media processing workflow based on the sequence of the plurality of segments (operation 520).

[0070] As further shown in FIG. 5, the process 500 may include adding multidimensional metadata to the sequence of a plurality of segments of multidimensional media (operation 530).

[0071] In embodiments, operation 530 may further include representing a starting point for each segment of the plurality of segments using the multidimensional metadata.

[0072] In embodiments, the process 500 may further include representing a position of the each segment in a hyperspace using a starting vector, a length vector, and a scaling vector associated with the length vector and the starting vector.

[0073] In embodiments, the process 500 may further include ordering the sequence of the plurality of segments in an increasing order of sequence vectors using rules defined by the multidimensional metadata.

[0074] In embodiments, operation 530 may further include representing each segment of the plurality of segments with an identical unique code point, wherein the each segment includes the identical unique code point, and wherein the each segment starts with the identical unique code point.

[0075] In embodiments, the process 500 may further include ordering the plurality of segments based on a sequence vector representation of the plurality of segments.

[0076] In embodiments, the process 500 may further include ordering the plurality of segments in an increasing order of sequence vectors based on rules defined by the sequence vector representation.

[0077] Although FIG. 5 shows example blocks of process 500, in some implementations, process 500 may include additional blocks, fewer blocks, different blocks, or differently arranged blocks than those depicted in FIG. 5. Additionally, or alternatively, two or more of the blocks of process 500 may be performed in parallel.

[0078] The foregoing disclosure provides illustration and description, but is not intended to be exhaustive or to limit the implementations to the precise form disclosed. Modifications and variations are possible in light of the above disclosure or may be acquired from practice of the implementations.

[0079] As used herein, the term component is intended to be broadly construed as hardware, firmware, or a combination of hardware and software.

[0080] It will be apparent that systems and/or methods, described herein, may be implemented in different forms of hardware, firmware, or a combination of hardware and software. The actual specialized control hardware or software code used to implement these systems and/or methods is not limiting of the implementations. Thus, the operation and behavior of the systems and/or methods were described herein without reference to specific software code—it being understood that software and hardware may be designed to implement the systems and/or methods based on the description herein.

[0081] Even though combinations of features are recited in the claims and/or disclosed in the specification, these combinations are not intended to limit the disclosure of possible implementations. In fact, many of these features may be combined in ways not specifically recited in the claims and/or disclosed in the specification. Although each dependent claim listed

below may directly depend on only one claim, the disclosure of possible implementations includes each dependent claim in combination with every other claim in the claim set. No element, act, or instruction used herein should be construed as critical or essential unless explicitly described as such. Also, as used herein, the articles “a” and “an” are intended to include one or more items, and may be used interchangeably with “one or more.” Furthermore, as used herein, the term “set” is intended to include one or more items (*e.g.*, related items, unrelated items, a combination of related and unrelated items, etc.), and may be used interchangeably with “one or more.” Where only one item is intended, the term “one” or similar language is used. Also, as used herein, the terms “has,” “have,” “having,” or the like are intended to be open-ended terms. Further, the phrase “based on” is intended to mean “based, at least in part, on” unless explicitly stated otherwise.

WHAT IS CLAIMED IS:

1. A method executed by at least one processor, the method comprising:
segmenting a multidimensional media stream into a plurality of segments of multidimensional media in a multidimensional space;
representing each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode; and
deriving a network based media processing (NBMP) workflow based on the respective sequence vector of each segment of the plurality of segments.
2. The method according to claim 1, further comprising ordering the plurality of segments in an increasing order of sequence vectors using rules defined by the multidimensional metadata.
3. The method according to claim 1, further comprising representing each segment of the plurality of segments with an identical unique code point, wherein each segment includes the identical unique code point, and wherein each segment starts with the identical unique code point.
4. The method according to claim 3, further comprising ordering the plurality of segments based on a sequence vector representation of the plurality of segments.

5. The method according to claim 4, further comprising ordering the plurality of segments in an increasing order of sequence vectors based on rules defined by the sequence vector representation.

6. An apparatus comprising:

- at least one memory configured to store program code; and
- at least one processor configured to read the program code and operate as instructed by the computer program code, the computer program code comprising:
 - segmenting code configured to cause the at least one processor to segment a multidimensional media stream into a plurality of segments of multidimensional media in a multidimensional space;
 - representing code configured to cause the at least one processor to represent each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode; and
 - deriving code configured to cause the at least one processor to derive a network based media processing (NBMP) workflow based on the respective sequence vectors of each segment of the plurality of segments.

7. The apparatus according to claim 6, wherein the computer program code further includes first ordering code configured to cause the at least one processor to order the plurality

of segments in an increasing order of sequence vectors using rules defined by the multidimensional metadata.

8. The apparatus according to claim 6, wherein the computer program code further includes second representing code configured to cause the at least one processor to represent each segment of the plurality of segments with an identical unique code point, wherein each segment includes the identical unique code point, and wherein each segment starts with the identical unique code point.

9. The apparatus according to claim 8, wherein the computer program code further includes second ordering code configured to cause the at least one processor to order the plurality of segments based on a sequence vector representation of the plurality of segments.

10. The apparatus according to claim 9, wherein the computer program code further includes third ordering code the plurality of segments in an increasing order of sequence vectors based on rules defined by the sequence vector representation.

11. A non-transitory computer-readable storage medium, storing instructions, which, when executed by at least one processor, cause the at least one processor to:

segment a multidimensional media stream into a plurality of segments of multidimensional media in a multidimensional space;

represent each segment of the plurality of segments of multidimensional media using a respective sequence vector, the respective sequence vector comprising one or more predefined multidimensional metadata, wherein the predefined multidimensional metadata includes one of (1) a starting vector, a length vector, and a scaling vector, and (2) a startcode; and

derive a network based media processing (NBMP) workflow based on the respective sequence vectors of each segment of the plurality of segments.

12. The non-transitory computer-readable storage medium according to claim 11, further comprising instructions that cause the at least one processor to order the plurality of segments in an increasing order of sequence vectors using rules defined by the multidimensional metadata.

13. The non-transitory computer-readable storage medium according to claim 11, further comprising instructions that cause the at least one processor to represent each segment of the plurality of segments with an identical unique code point, wherein each segment includes the identical unique code point, and wherein each segment starts with the identical unique code point.

14. The non-transitory computer-readable storage medium according to claim 13, further comprising instructions that cause the at least one processor to order the plurality of segments based on a sequence vector representation of the plurality of segments.

15. The non-transitory computer-readable storage medium according to claim 14, further comprising instructions that cause the at least one processor to order the plurality of segments in an increasing order of sequence vectors based on rules defined by the sequence vector representation.