Introduction

Synchronization in multimedia systems refers to temporal relationships between media objects in the multimedia systems. In future multimedia systems (based, e.g., on MPEG-4) synchronization may also refer to spatial and content relationships, as well as temporal.

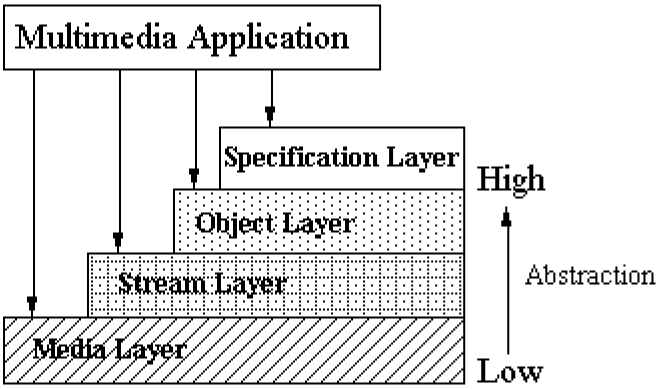
Synchronization between media objects comprises relationships between time-dependent media objects as well as time-independent media objects. Synchronization may need to occur at different levels in a multimedia system, consequently synchronization support is typically found in the operating system, communication system, databases, multimedia documents, and the application. A general scheme might involve a layered approach to achieving synchronization. For example, a Computer-Supported Collaborative Workgroup (CSCW) session might involved a multi-party video conferencing session with audio, and a shared whiteboard. Parties may make reference to objects on the shared whiteboard, using a pointer, to support what they are saying (e.g., saying "This area here..." while indicating the area with a pointer). Here, video and audio are continuous media objects which are highly periodic, whereas the shared whiteboard is a discrete media stream, as changes to it are highly irregular (the content, including the position of the pointer, depends on which participant has control of the object and when they make changes to it). The media streams must be highly synchronized, so that speech remains lip synchronized, and the whiteboard updates are synchronized with audio references to them.

The operating system and lower levels of the communication system are responsible for ensuring that jitter on individual streams does not occur during presentation of the video, audio, and whiteboard streams (intramedia synchronization). At a higher level, the runtime support for the synchronization of multiple multimedia media streams must ensure that the various media streams remain synchronized with respect to each other (intermedia synchronization). Finally, the application(s) are responsible for ensuring synchronicity between application-level events (usually initiated by the users). For example, if the application at the source does not capture timing dependencies between a user waving the pointer over part of the object in the whiteboard and the supporting audio stream, then it will be impossible for the application at the sink to know that the whiteboard and audio events need to be synchronized.

The temporal relations between media objects must be specified during capture of the media objects, if the goal of the presentation is to present media in the same way that they were originially captured. Synchronization information of events in an animation sequence or a slide show is usually specified by the designer, using, for example, a time-axis.

A Reference Model for Multimedia Synchronization

A reference model is needed to understand the requirements of multimedia synchronization, identify and structure runtime mechanisms that can support these requirements, identify interfaces between runtime mechanisms, and compare solutions for multimedia synchronization systems.
Figure 11.1 shows a reference model for multimedia synchronization systems. Each layer implements synchronization mechanisms which are provided by an appropriate interface. These interfaces can be used to specify or enforce the temporal relationships. Each interface can be used by the application directly, or by the next higher layer to implement an interface. Higher layers offer higher programming and QoS abstractions.



Media Layer

An application operates on a single continuous media stream, which is treated as a sequence of LDUs. Networking components must be taken into account. Provides access to files and devices.

Stream Layer

The stream layer operates on continuous media streams as well as groups of media streams. In a group, all streams are presented in parallel by using mechanisms for interstream synchronization. QoS parameters will specify intrastream and interstream synchronization requirements.
Continuous media is seen as a data flow with implicit time constraints;

individual LDUs are not visible. An application using the stream layer is responsible for starting, stopping and grouping the streams, and for the definition of the required QoS in terms of timing parameters supported by the stream layer. It is also responsible for the synchronization with time-independent media objects. Tasks include resource reservation and LDU process scheduling.

Object Layer

The object layer operates on all media streams and hides the differences between continuous and discrete media. An application that interacts with this layer will be presented with a view of a complete, synchronized presentation. This layer takes a complete synchronization specification as its input and is responsible for the correct schedule of the overall presentation.

Specification Layer

This layer contains applications and tools that are allowed to create synchronization specifications (e.g., authoring tools, multimedia document editors).

The specification layer is also responsible for mapping user-required QoS parameters to the qualities offered at the object layer interface.

Synchronization specifications can be:

- Interval-based: specifications of the temporal relations between the time intervals of the presentation of media objects
- Axes-based: allows presentation events to be synchronized according to shared axes, e.g., a global timer
- Control flow-based: at specified points in presentations, they are synchronized
- Event-based: Events in the presentation trigger presentation actions

Synchronization in a Distributed Environment

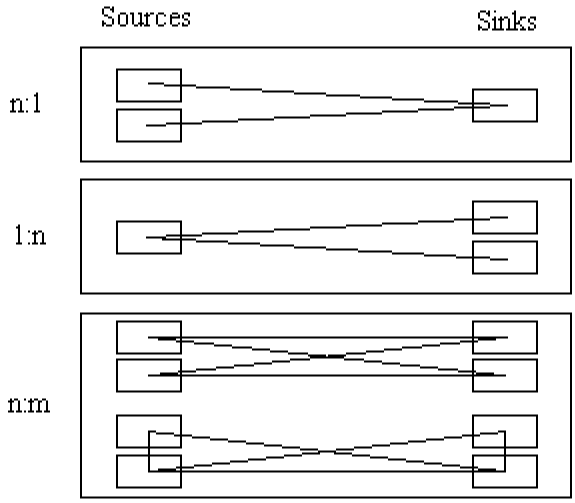
Synchronization in a distributed environment is complex, because there may be more than one source of multimedia data, and more than one sink consuming it. The synchronization information for the various media stream may also reside at different sources.

Transport of the synchronization specification

The sink needs to have the synchronization information available to correctly display an object. There are three main approaches to delivering the synchronization information to the sink:

- Delivery of the synchronization information before the start of the presentation
- Use of an additional synchronization channel
- Multiplexed data streams

If the multimedia presentation is live and multiple parties are involved, then none of the approaches above is suitable for delivering synchronization information to the sink(s) in a timely fashion. Figure 11.2 shows typical communication patterns.



Of particular interest here, is that if multiple sinks are involved, then they will receive identical data. It would be inefficient if the data were replicated at the source for separate transmission to each of the sinks. It would also be inefficient if the same operation was carried out at different sinks. *Multicasting* or broadcasting of streams is the responsibility of the stream layer, whereas efficient planning of operation execution in the different communication patterns is a responsibility of the object layer.

Multi-Step Synchronization

In a distributed environment, synchronization is typically a multi-step process, during which the synchronization must be maintained so as to enable the sink to perform the final synchronization. The synchronization steps are:

- during object acquisition, e.g., during frame digitization
- during retrieval, e.g., synchronized access to frames of a stored video
- during delivery of the LDUs to the network
- during the transport of the LDUs, e.g., using isochronous protocols
- at the sink, e.g., synchronized delivery to the output devices
- within the output device

With many different points at which synchronization must occur decisions must be made about how to implement it. A first decision is the selection of the type of transport for the synchronization specification. In runtime, decisions must be taken concerning the location of synchronization operations, keeping clocks in synchrony (if used to provide common timing information), and the handling of multicast and broadcast messages. Coherent planning of the steps in the synchronization process, together with the necessary operations of the objects, e.g., decompression, must also be done. In addition, presentation manipulation operations demand additional replanning at runtime.

Synchronization Specification

A synchronization specification should comprise:

- Intra-object synchronization specifications for the media objects of the presentation
- QoS descriptions for intra-object specifications
- Inter-object synchronization specifications for media objects of the presentation
- QoS descriptions for inter-object synchronization

In addition, the form, or alternate forms, of a multimedia object may be described. For example, a text could be presented as text on the screen or as a generated audio sequence. In the case of live synchronizations, the temporal relations are implicitly defined during capture. QoS requirements are specified before the start of the capture. In the case of synthetic synchronization, the specification must be created explicitly.

Time stamping and pack architecture.

we have studied the reference model for multimedia synchronization and the synchronization requirements in a distributed environment. The issue of synchronization has been addressed in MPEG-2 standard, where intra-media continuity and inter-media synchronization have been handles at different layers of the multimedia stream. In this lesson, we are going to study the architecture of multimedia streams and the time-stamping requirements to ensure synchronization between the digital storage medium (DSM) , i.e. the source and the systems target decoder (STD), i.e. the sink.

Requirements of media playback

There are two distinguishing requirements of media playback :

- Intra-media continuity
- Inter-media synchronization

In order to understand MPEG synchronization, it is necessary first to familiarize with the encoder's system architecture. At the time of encoding in MPEG, in the first stage, video frames and audio samples are separately encoded, and an independent stream of bytes is output for each media channel. Each media stream is then packetized independently. Packets from the different streams are interspersed to form a single multiplexed stream of packets, and the multiplexed stream is then organized into packs, with each pack comprising of an integral number of packets.

Packs and packets

In a MPEG stream, whereas continuity is handled at the pack layer, synchronization is handled at the packet layer. Fig.35.1 shows the generation of packs at the encoder. The decoder system referred to an System Target Decoder (STD), as shown

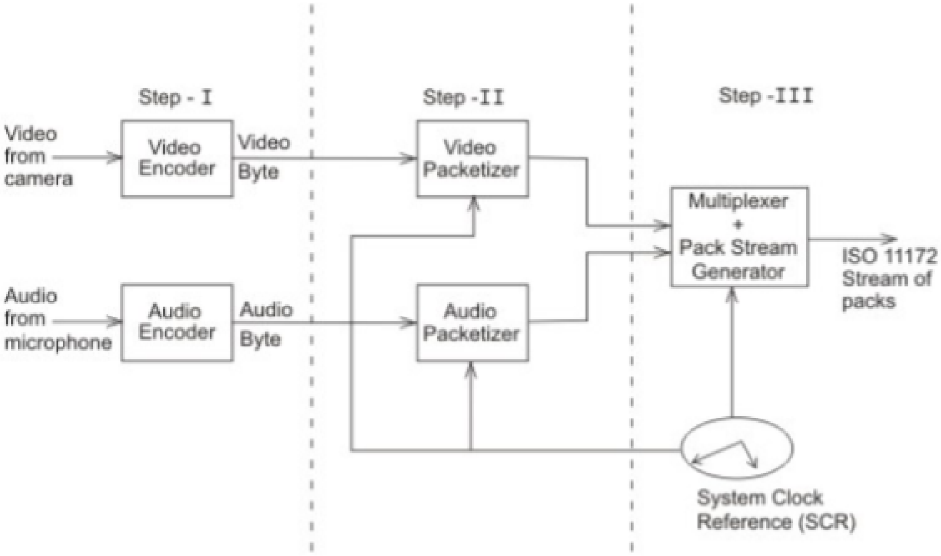


FIGURE 35.1 Generation of packs at the encoder

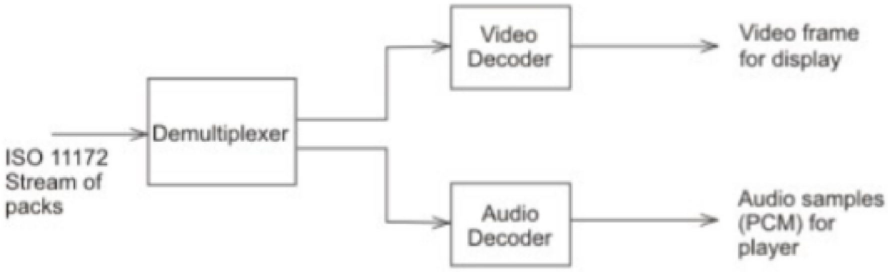


Figure 35.2 System target decoder

Requirements of System Clock Reference (SCR) In MPEG stream, packs constitute a common system layer appeared around a media specific compression layer, and house the following functions at the encoder :

- Interleaving of multiple compressed streams into a single stream
- Time stamping : insertion of the SCR SCR’s are sampled values (in Hz) of the encoder’s clock which is constrained to meet the following drift bounds on its frequency :
- Nominal frequency = 90,000 Hz
- Fractional frequency drift ρ ≤ 0.00005

- Frequency drift υ ≤ 4.5 Hz

- Rate of change of drift δυ/δt ≤ 0.00025 Hz/s.

Taking drift at the rate of 0.00025 Hz/s, the frequency can go beyond the permitted 4.5 Hz drift bound over a period of about 5 hours. Hence, continuous operation of the encoder for longer than 5 hours may need resynchronization. SCR is inserted into each pack header. The SCR value in a pack is set equal to reading of the encoder's clock at the instant the last byte of the SCR is constructed at the encoder. Successive SCR's cannot differ by more than 0.7 seconds to assure accurate clock updating at the STD. Hence, the time interval between successive packs cannot exceed 0.7 seconds. A pack header also contains the rate of the multiplexed stream as manifested in the pack and is termed the mux-rate. The mux-rate is not the bit or the byte rate but is a scaled value. Suppose, the number of bytes following the SCR in a pack p till end = lp bytes and tl is the time at which the last byte of the pack is constructed at the encoder. Then, the mux-rate is computed as

$$mux-rate=\frac{l_p}{50\left(t_l-SCR\right)}$$

The mux-rate may vary from pack to pack.

The first pack header usually contains the following additional system headers

- Rate bound : Max. mux-rate coded in any pack.

- Video bound: Integer, indicating the maximum number of video streams (upto 16).

- Audio bound: Integer, indicating the maximum number of audio streams (upto 32

- Video, Audio lock flag : indicates if there is a specific constant harmonic relationship between the media sampling rates and the SCR frequency.

- Fixed flag: defined in the following section.

Pack architecture and pack headers

shows the composition of a pack from the packets.

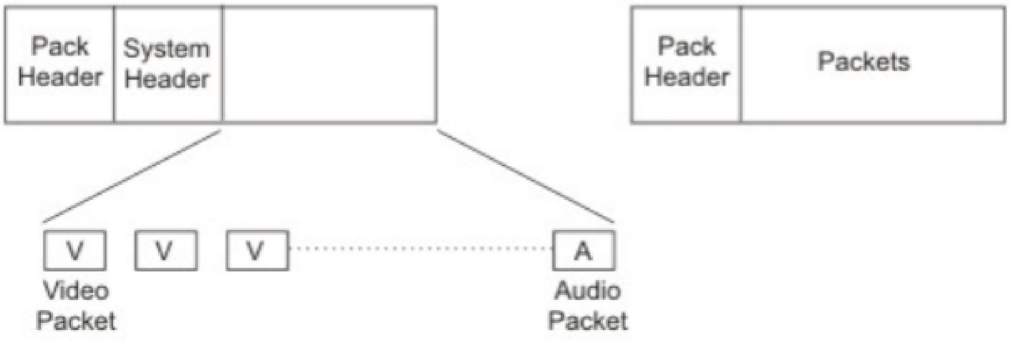


Figure 35.3 Composition of a pack

The fixed flag indicates whether the stream is of fixed bitrate, in which case, SCRs of each pack in the multiplexed stream bear the following linear relationship with the size offset of the pack within the stream.

$$SCR(i) = \lfloor (a * i + b) \rfloor \text{ modulo } 2^{33}$$

where a and b are real-valued constants, i is the index of the final byte of the SCR field counted from the start of the multiplexed stream. If the data rate averaged between the successive SCRs is equal to the max-rate field, the fixed flag is set. If the mux-rate field indicates higher than the average rate, then the stream is delivered in bursts. In order to transform a bursty stream into a constant bit rate stream, MPEG permits an encoder to insert a padding stream. Such a padding stream would consist of a sequence of packets interleaved with audio and video packets within each pack to achieve a constant bit rate over the entire pack. At the STD upon demultiplexing, the padding stream packets are simply discarded.

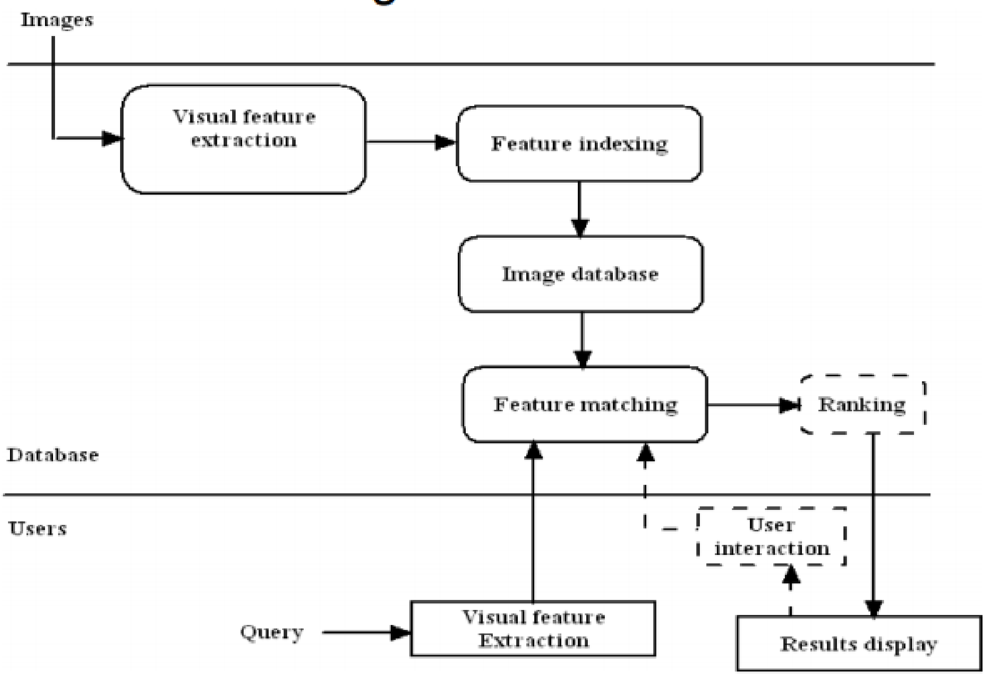
image indexing and retrieval

Four main steps to approaches the image indexing and retrieval

Content based Image Retrieval (CBIR)– low level features

- Extract low level image features (color, edge, texture and shape)
- Expand these image feature towards semantic levels
- Index on these images based on similar measurement
- Relevance feedback to refine the candidate images

- Content based image retrieval



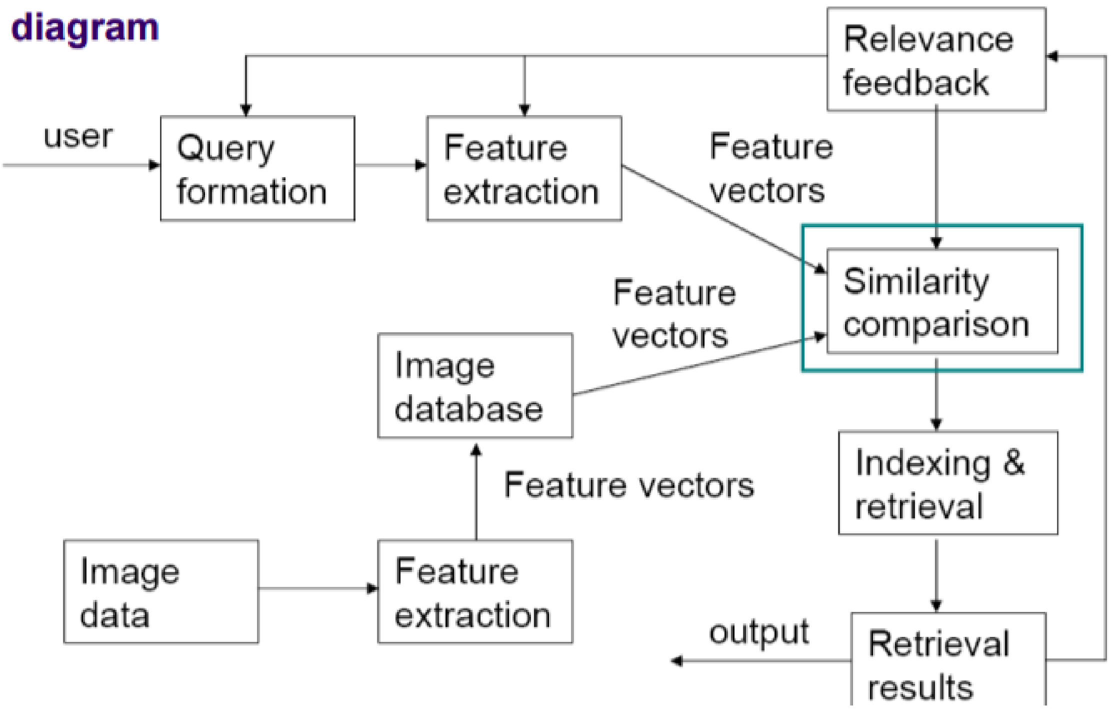


Image representation

A visual content descriptor can be either global or local.

The global descriptor uses the visual features of the whole image

A local descriptor uses the visual features of regions or objects to describe the image content, with the aid of region/object segmentation techniques

- Image representation

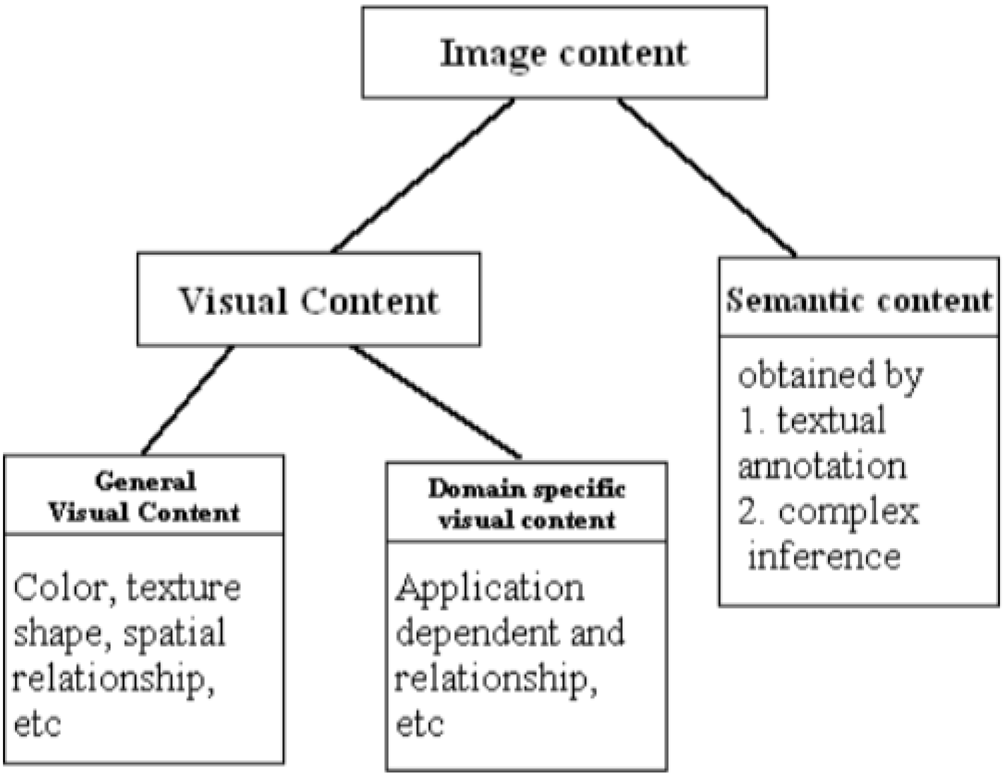


Image content

Visual Content

Semantic content

obtained by
1. textual annotation
2. complex inference

General Visual Content

Color, texture shape, spatial relationship, etc

Domain specific visual content

Application dependent and relationship, etc

Low level Feature Extraction –

Color Representation

Color Color is very powerful in description and of easy extraction from nature images in its considerable variance changes:

Illumination

Orientation of the surface

Viewing geometry of the camera

- Color fundamentals

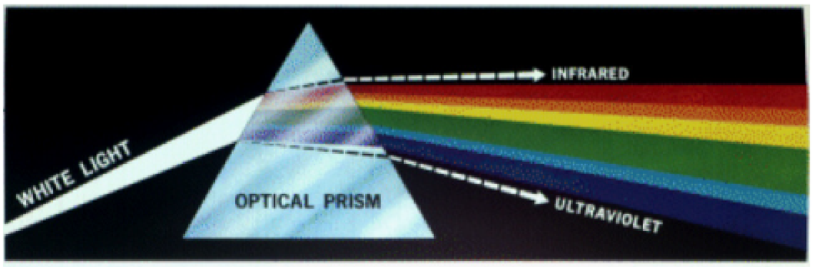


Plate 1. Color spectrum seen by passing white light through a prism. (Courtesy of General Electric Co., Lamp Business Division.)

Ref: Gonzalez and Woods, digital image processing

Color fundamentals

Light of different wavelengths produces different color sensations such as in different broad regions (violet, blue, green, yellow, orange and red)

Color fundamentals

The colors that humans perceive in an object are determined by the nature of the light reflected from the object.

Visible light is electromagnetic radiation with a spectrum wavelength ranging approximately from 400 to 780 nm.

Red, Green and Blue are the additive primary colors. Any color can be specified by just these three values, giving the weights of these three component

Color space

RGB (Red, Green and Blue) space

The RGB color space is the most important means of representing colors used in multimedia.

A color can be represented in a form (r-value,g-value,b-value). The value in here is defined as the percentage of the pure light of each primary.

Examples:

(100%,0%,0%) – pure saturated primary red

(50%,0%,0%) – a darker red

(0%,0%,0%) – black

(100%,100%,100%) – white

A Cartesian Coordinate System is defined to measure each color with a vector.

a practical system, a RGB color can hold different bits such as 24-bit, 15 -bit and 12-bit color depth. 24-bit -- full RGB color space 15-bit – 5-bit for R, 6-bit for G and 5-bit for B 12-bit – 4-bit for R, 4-bit for G and 4-bit for B

The color coherence vector (CCV) is a tool to distinguish images whose color histograms are indistinguishable The CCV is a descriptor that includes relationship between pixels – spatial information

Color Coherence Vector (CCV) A color's coherence is defined as the degree to which pixels of that color are members of large similar-color regions. These significant regions are referred as coherent regions which are observed to be of significant importance in characterizing images Coherence measure classifies pixels as either coherent or incoherent A color coherence vector represents this classification for each color in the imag

For RGB color space, if each color channel M is discretized into 16 levels, the total number of discrete color combinations called histogram bins N. H(M) is a vector , Where each represents the number of pixels in image M falling into bin i M3 = 16x16x16=4096 bins in total (h h, h, ... h)

Texture The concept of texture is intuitively obvious but has no precise definition

Texture can be described by its tone and structure Tone – based on pixel intensity properties Structure – describes spatial relationships of primitive