GPU 与视频解码

2019 年的标准摄像头配置都是 2k 以上的标准分辨率,并且非常便宜,码率会根据场景而定,运动目标很多,8M 以上码率,普通场景 2-4M 码率.1M 是 2016 以前使用的监控码率.

一般来说,当视频播放达到 4 路以上,cpu 就会感觉不太够用,这时候我们就需要 gpu 来支持.而 GPU 支持,并不会 100%的成功解码,尤其对一些非标准的 h.264,x264 视频编码,gpu 可能会遇上 441,442,424 这类光栅排列不够标准化,而发生解码错误.在实际使用 gpu 解码视频时,这经常发生.而这就是 TFFMPEG_Reader 的特殊之处.

另一方面,视频也代表了大规模的数据图片,有效利用视频的图片资源,是一种现代化的 AI 建模手段.整个 Z-AI 引擎在设计之初视频支持是核心功能之一.

下面我们来看看 Z-AI 内置的 ffmpeg 支持的 gpu 是怎样工作.

文件 mp4,mkv 文件,rtsp,rtmp,http,https,hls 等等协议的视频支持,在 Z-AI 引擎和 demo 中都使用 TFFMPEG_Reader 来驱动解码,它会将视频帧以最优性能解码成 TMemoryRaster 格式.

何谓最优性能?

TFFMPEG_Reader 对于 gpu 的支持比较特殊,因为 gpu 即使成功驱动了解码器,也不一定可以成功解码,所以在 TFFMPEG_Reader 内置了一种探测性质的解码方式,当 gpu 解码器遇到问题时(时有发生),就会自动切成 cpu 解码,当 cpu 解码也出现问题,它才会返回错误.

我们可以自行通过阅读 TFFMPEG_Reader 的实例来了解 reader 解码过程工作的细节.

```
type
  TFFMPEG_Reader = class;
  TFFMPEG_VideoStreamReader = class;

  TFFMPEG_Reader = class(TCoreClassObject)
  private
    FVideoSource: TPascalString;
    FWorkOnGPU: Boolean;
    src_filename: RawByteString;
    FormatCtx: PAVFormatContext;
    videoCodecCtx: PAVCodecContext;
    audioCodecCtx: PAVCodecContext;
    videoCodec: PAVCodec;
    audioCodec: PAVCodec;
    Frame, FrameRGB: PAVFrame;
    FrameRGB_buffer: PByte;
    Sws_Ctx: PSwsContext;
    videoStreamIndex: integer;
    audioStreamIndex: integer;
    videoStream: PAVStream;
    AVPacket_ptr: PAVPacket;
  public
    Current: Double;
    Current_Frame: int64;
    Width, Height: integer;
    property VideoSource: TPascalString read FVideoSource;
    property WorkOnGPU: Boolean read FWorkOnGPU;

    constructor Create(const VideoSource_: TPascalString); overload;
    constructor Create(const VideoSource_: TPascalString; const useGPU_: Boolean); overload;
    destructor Destroy; override;

    procedure OpenVideo(const VideoSource_: TPascalString; useGPU_: Boolean); overload;
    procedure OpenVideo(const VideoSource_: TPascalString); overload;
    procedure CloseVideo;

    function NextFrame(): Boolean;
    function ReadFrame(output: TMemoryRaster; RasterizationCopy_: Boolean): Boolean;

    procedure Seek(second: Double);
    function Total: Double;
    function CurrentStream_Total_Frame: int64;
    function CurrentStream_PerSecond_Frame(): Double;
    function CurrentStream_PerSecond_FrameRound(): integer;
    property PSF: Double read CurrentStream_PerSecond_Frame;
    property PSFRound: integer read CurrentStream_PerSecond_FrameRound;
    property RoundPSF: integer read CurrentStream_PerSecond_FrameRound;
    property PSF_I: integer read CurrentStream_PerSecond_FrameRound;
  end;
```

By.qq600585