A MULTI-GPU SCALABLE SDK FOR REAL-TIME STEREO STITCHING OF 360 VIDEO AND AUDIO

Thomas True

NVIDIA.
AGENDA

- Stereo Panoramas
- VRWorks 360 Video SDK Overview
- Calibration
- Video Stitching
- Audio Processing
- Multi-GPU Optimization
- VRWorks SDK Usage Examples
- Q&A
STEREO PANORAMAS
STEREO PANORAMAS
Stereo Video in VR

- 360 Video
- Immersive
- Provide sense of depth and scale
- Capture reality
STEREO PANORAMAS

Workflow

VRWORKS 360 VIDEO

Ingest ➔ Decode ➔ Calibrate ➔ Stitch ➔ Encode ➔ Output
360 STEREO STITCHING PIPELINE

High-Level API

Low-Level API

Core Video

Core Audio

Input:
- Rig Parameters (Estimated)
- MP4/MOV (H.264 with audio)
- Uncompressed Video (RGBA)
- Separate video and audio streams
- Stitcher XML
- Rig XML

Decide
Decode (GPU)
Demux (CPU)

Output:
- MP4/MOV (H.264 with blended audio)
- Uncompressed Video (RGBA)
- Elementary H.264 stream blended audio stream

Generate rig XML
Calibration
Save to XML

Generate rig XML

Rig Parameters (Calibrated)

Rig Parameters (Calibrated)

Calibrated Rig XML

Calibrated Rig XML

Stitcher XML
Rig XML

Generate rig XML
Calibration
Save to XML

Calibrated Rig XML

Stitcher XML
Rig XML

Output:
- MP4/MOV (H.264 with blended audio)
- Uncompressed Video (RGBA)
- Elementary H.264 stream blended audio stream
VRWORKS 360 VIDEO SDK
VRWORKS 360 VIDEO SDK

Features

- GPU-optimized stereo pipeline
- GPU-accelerated calibration
- GPU-accelerated encode and decode
- Audio mix down and stretch
- Audio encode / decode
- Audio resampling
- Realtime and offline
- Stereo and mono stitching
- Compressed and uncompressed inputs and outputs
  - MP4, H.264 and RGBA8
  - PCM and AAC audio

VRWORKS 360 VIDEO SDK
Works With Multiple Rigs
VRWORKS 360 VIDEO SDK

Workflow Flexibility

File Based Workflow
- MP4
  - Demux
  - Video
  - Audio

Stream Based Workflow
- H.264 + Audio
  - Stitch
  - Video
  - Audio
  - MP4

Buffer Based Workflow
- RGBA8UI + Audio
  - Stitch
  - Video
  - Audio
  - MP4
VRWORKS 360 VIDEO SDK
High-Level Video and Audio Pipeline
VRWORKS 360 VIDEO SDK
High-Level Video and Audio Pipeline

Advantages:

- GPU-accelerated demux, mux, decode and encode
- File and stream oriented API
- Includes calibration API
- Support for additional audio and video formats
VRWORKS 360 VIDEO SDK
Low-Level Video and Audio Pipeline

- RGBA8UI
- FP32 PCM
- Stitch
- RGBA8
- FP32 PCM
- File
Advantages:

- Works directly with CUDA device memory
- Plugs directly into a larger CUDA pipeline
- Interops with graphics API to keep frames on GPU
- Application flexibility to control threading and data transfers
CALIBRATION
1. Input initial rig specs and a set of images from the cameras
2. GPU-accelerated feature extraction
3. Construct feature correspondence pairs from overlapped regions
4. Optimize rig parameters to minimize the correspondence distances (on the virtual sphere)
CALIBRATION
GPU-Based Feature Extraction and Correspondence Matching
CALIBRATION

Features

- Works with any rig configuration
- High error tolerance in input instances.
- Accommodates lenses with up to or greater that 180° FOV
- Support standard distortion characteristics as well as custom fisheye lenses
- Provides accuracy metric based upon cross correlation [0:1]
- GPU-accelerated
VIDEO STITCHING
VIDEO STITCHING
Creating Virtual Cameras Between Physical Cameras

1. Compute pixel motion between adjacent cameras
   - Lambertian: Appearance not view dependent
   - Cameras photometrically consistent
   - Rig calibrated
   - Surface point same when viewed by adjacent cameras
   - Compute disparity

2. Interpolate to the virtual camera position, reprojection the virtual sphere and blend.

3. Apply post processing
IMAGE ALIGNMENT CHALLENGES

- Textureless regions
- Periodic textures
- Specular surfaces
- Occlusion
VIDEO STITCHING
Overcoming Pixel Alignment Challenges

- Detect occlusion boundaries
- Reduce pixel motion noise between occlusion boundaries
- Fill holes (textureless regions) with plausible motion
- Enforce temporal coherence
AUDIO
AUDI O PROCESSING

Overview

- Stereo mixdown with adjustable spread effect
- MP4 embedded audio, separate audio streams or individual audio buffers
- PCM and AAC audio streams
- Audio resampling
- Performed in parallel on the CPU while the video is stitching on the GPU
- Unlimited number of internal and external audio sources
- Push-pull model
Audio Processing

Stereo Mixdown With Spread Effect

1. Convert all inputs to mono.
2. Add together
3. Spread to stereo. Application programs depth of spread
4. Add gain

Result: Output volume same as input
MULTI-GPU OPTIMIZATIONS
**CHALLENGE**

How can we decode and stitch in real-time?

- Need to decode 8 separate 4K streams at 30 fps (similar to 240 fps!)
- After getting frames to GPU, will we have any time left to stitch?
PIPELINING FOR PERFORMANCE

Pipelining Ingest, Stitch, and Output for Better Throughput

Decode Frame N → Stitch Frame N → Output Frame N

Decode Frame N+1 → Stitch Frame N+1 → Output Frame N+1

Decode Frame N+2 → Stitch Frame N+2 → Output Frame N+2

Time
CHALLENGE
Minimizing Memory Copy Latency

- Must copy input/output between GPUs and CPU
- Synchronous memory copy injects bubbles in compute workload
HIDING MEMORY LATENCY
Using CUDA Streams and Asynchronous Memory Copies
CHALLENGE
Maximum Quality Achievable in Real-Time

- We can stitch lower-quality on one GPU
- We want more cameras, higher quality, higher output resolution

Project ➔ Disparity Map ➔ Post-Process ➔ Interpolate & Blend
MULTI-GPU SCALING
Distribute Stereo Pairs Among Available Devices
## STITCHING PERFORMANCE

### 8x 4K input streams

<table>
<thead>
<tr>
<th>STITCH TYPE</th>
<th>GPU</th>
<th>OUTPUT</th>
<th>STITCHING RATE (FPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereo</td>
<td>2x Quadro P6000</td>
<td>Uncompressed</td>
<td>32</td>
</tr>
<tr>
<td>Stereo</td>
<td>2x Quadro P6000</td>
<td>MP4</td>
<td>30</td>
</tr>
<tr>
<td>Mono</td>
<td>Quadro P6000</td>
<td>Uncompressed</td>
<td>40</td>
</tr>
<tr>
<td>Mono</td>
<td>GTX 1060/1080</td>
<td>Uncompressed</td>
<td>40</td>
</tr>
</tbody>
</table>
SAMPLE APPLICATIONS
VRWORKS 360 VIDEO SDK

Low-Level Video API Example

```c
// Initialize stitcher instance
RETURN_NVSS_ERROR(nvssVideoCreateInstance(&stitcher_props, &params->rig_properties, &stitcher));

// Load image frames for each camera
for (uint32_t camera = 0; camera < params->rig_properties.num_cameras; camera++) {
    nvstitchImageBuffer_t input_image;
    RETURN_NVSS_ERROR(nvssVideoGetInputBuffer(stitcher, camera, &input_image));

    // Read image from file
    getRgbaImage(image_file_path, &rgba_bitmap_ptr, image_width, image_height);

    // Copy image data to CUDA device memory
    cudaMemcpy2D(input_image.dev_ptr, input_image.pitch, rgba_bitmap_ptr, input_image.row_bytes, 
                  input_image.row_bytes, input_image.height, cudaMemcpyHostToDevice);
}

// Stitch
RETURN_NVSS_ERROR(nvssVideoStitch(stitcher));
```
for (int eye = 0; eye < num_eyes; eye++) {
    // Fetch stitched panorama(s)
    if (stereo)
        RETURN_NVSS_ERROR(nvssVideoGetOutputBuffer(stitcher, nvstitchEye(eye), &output_image));
    else
        RETURN_NVSS_ERROR(nvssVideoGetOutputBuffer(stitcher, NVSS_EYE_MONO, &output_image));

    if (out_stacked == nullptr)
        out_stacked = (unsigned char *)malloc(output_image.row_bytes * output_image.height * num_eyes);

    // Copy to host memory
    cudaMemcpy2D(out_stacked + out_offset, output_image.row_bytes, output_image.dev_ptr,
                  output_image.pitch, output_image.row_bytes, output_image.height, cudaMemcpyDeviceToHost);

    out_offset += output_image.height * output_image.row_bytes;
}
// Write to file
putRgbaImage(params->out_file, out_stacked, output_image.width, output_image.height * num_eyes);
// Clean up
RETURN_NVSS_ERROR(nvssVideoDestroyInstance(stitcher));
VRWORKS 360 VIDEO SDK

High-Level API Example

// Create video rig
RETURN_NVSTITCH_ERROR(nvstitchCreateVideoRigInstance(&rig_properties, &stitcher_properties.video_rig));

// Create audio rig
RETURN_NVSTITCH_ERROR(nvstitchCreateAudioRigInstance(&audio_rig_properties, &stitcher_properties.audio_rig, NULL));

// Create stitcher instance
RETURN_NVSTITCH_ERROR(nvstitchCreateStitcher(&params->stitcher_properties, &stitcher));

// Start stitcher
RETURN_NVSTITCH_ERROR(nvstitchStartStitcher(stitcher, nullptr, nullptr));

// Load video inputs
for (uint32_t i{}; i < (uint32_t)params->rig_properties.num_cameras; ++i)
  RETURN_NVSTITCH_ERROR(nvstitchFeedStitcherAudioVideo(0, stitcher, i, i, &params->payloads[i], false));

// Stop stitcher
RETURN_NVSTITCH_ERROR(nvstitchStopStitcher(stitcher));

// Clean up
RETURN_NVSTITCH_ERROR(nvstitchDestroyStitcher(stitcher));
VRWORKS 360 VIDEO SDK

Calibration

// Create calibration instance
RETURN_NVSTITCH_ERROR(nvstitchCreateCalibrationInstance(calib_prop, &h_calib));

// Read and load input images for calibration
for (uint32_t cam_index = 0; cam_index < camera_count; cam_index++)
{
    getRgbaImage(image_file_path, &rgba_bitmap_ptr, image_width, image_height);
    nvstitchPayload_t calib_payload = nvstitchPayload_t{ calib_prop.input_form,{ width, height } },
    calib_payload.payload.buffer.ptr = rgba_bitmap_ptr;
    calib_payload.payload.buffer.pitch = rig_properties.cameras[cam_index].image_size.x * num_channels;
    RETURN_NVSTITCH_ERROR(nvstitchFeedCalibrationInput(frame_index, h_calib, cam_index, &calib_payload));
}

// Calibrate
RETURN_NVSTITCH_ERROR(nvstitchCalibrate(h_calib, &h_calibrated_video_rig));

// Destroy calibration instance
RETURN_NVSTITCH_ERROR(nvstitchDestroyCalibrationInstance(h_calib));
VRWORKS 360 VIDEO SDK

XML Based Configuration

CreateVideoRig → Calibrate → CreateVideoRig → CreateStitcher → Stitch

CreateAudioRig → CreateAudioStitcher

rig.xml → rig_calibrated.xml → stitcher.xml → footage.xml
VRWORKS 360 VIDEO SDK

XML Examples

// Read from Xml
bool readCameraRigXml(const std::string& xmlPath,
        std::vector<nvstitchCameraProperties_t>& cameraProperties,
        nvstitchVideoRigProperties_t* videoRigProperties);

// Read filenames from Xml // For nvcalib
bool readInputCalibFilenamesXml(const std::string& xmlPath,
        std::vector<std::vector<std::string>>& filenames);

// Read media filenames Xml
bool readInputMediaFeedFilenamesXml(const std::string& xmlPath,
        std::vector<std::string>& filenames);

bool readInputMediaFeedXml(const std::string& inputFeedXmlPath,
        std::vector<nvstitchPayload_t>& mediaPayloadArray, int* arrayCount);

bool readInputAudioFeedXml(const std::string& inputFeedXmlPath,
        std::vector<nvstitchAudioPayload_t>& audioPayloadArray, int* arrayCount);

bool readStitcherPropertiesXml(const std::string& stitcherPropertiesXmlPath,
        const std::string& inputFeedXmlPath,
        nvstitchStitcherProperties_t* stitcherProperties);
VRWORKS 360 VIDEO SDK

Utilities

Image I/O

bool getRgbImage(const string& imgPath, unsigned char** imgRgb, int& imgWidth, int& imgHeight)

bool getRgbaImage(const string& imgPath, unsigned char** imgRgb, int& imgWidth, int& imgHeight)

bool putRgbaImage(const string& imgPath, unsigned char* imgRgba, int imgWidth, int imgHeight)

bool showImageRgb(int width, int height, unsigned char* imgRgb)
EXHIBITION
Come See Our Demo in the VR Village!
VRWORKS 360 VIDEO SDK
Available Now

- GPU-optimized stereo pipeline
- GPU-accelerated calibration
- GPU-accelerated encode and decode
- Audio mix down and stretch
- Audio encode / decode
- Audio resampling

- Realtime and offline
- Stereo and mono stitching
- Compressed and uncompressed inputs and outputs
  - MP4, H.264 and RGBA8
  - PCM and AAC audio

Q & A