INTERACTIVE ZOOM AND PANNING FROM LIVE PANORAMIC VIDEO

NOSSDAV '14: Proceedings Of Network And Operating System Support On Digital Audio And Video Workshop, ACM

March 2014
OVERVIEW

• The scenario is an end-to-end real-time streaming system that deploy in a soccer stadium in Alfheim, Norway
• The goal is to help thousands of concurrent users with their own operations such as zoom or panning the camera view
• The paper is focus on the design of panoramic texture projection and the building of virtual camera on the client side
SYSTEM INTRODUCTION
SERVER-SIDE
VIDEO CAPTURE

• Use 5 cameras with resolution of 2046*1086 pixels to do the filming
• Rotate and integrate them in a circular pattern, i.e., to look directly through a point in front of the lenses in order to reduce parallax effects
• Transfer the streams to panorama processing machine with point-to-point Ethernet network
SERVER-SIDE CYLINDRICAL PROJECTION

- Retrieve a full set of synchronized frames from the camera streams with a frame synchronizer
- Generate the cylindrical panorama frames with a panorama stitcher
- Use a H.264 video encoder for the immediate frame delivery to the client side
In panorama stitcher:

- \( r = \frac{W_s}{2 \times \tan\left(\frac{f_{ov}}{2}\right)} \)
- \( r \) : radius of cylinder.
- \( W_s \) : width of source image.
- \( f_{ov} \) : field of view.
- Center is where the cameras are
The unrolled cylinder forms a Cartesian coordinate system.

Each pixel \((T_x, T_y)\) on the unrolled cylinder determines the corresponding horizontal \((\theta)\) and vertical \((\phi)\) angle of a ray from the camera center through this coordinate.
And from the previous determination, we see that:

- \( \theta = \arctan \left( \frac{T_x}{r} \right) \) and \( \phi = \arctan \left( \frac{T_y}{r} \right) \)

Then, for \( x, y, z \), in 3D space where the ray intersects the image is:

- \( z = r \) and \( x = \tan(\theta) \ast z \)
- \( y = \tan(\phi) \ast \sqrt{z^2 + x^2} \)

And we can finally get the transform function:

- \( \theta = \arctan \left( \frac{z}{x} \right) \) and \( \phi = \arctan \left( \frac{y \ast \sin(\theta)}{x} \right) \)
HTTP segment streaming is used for video downloading

- Served by Apache server along with a manifest file for informing client when the next file is ready
- Processing runs in the background without blocking the display thread and the user input thread
CLIENT-SIDE
CREATE VIRTUAL CAMERA

- Perform the video through a virtual perspective camera view, which is generated by the panoramic frame.
First select a point P in panorama image, then the 3D point P project to image point q can be written by the following:

\[ \lambda_q = [K|0_3] \begin{bmatrix} R & 0 \\ 0_3 & 1 \end{bmatrix} \begin{bmatrix} 0^T_3 \\ -C \end{bmatrix} P \]

- \( R \): 3D distortion matrix (3x3)
- \( K \): The camera intrinsic matrix from focal length

And the ray \( s \) from the cylinder center can be represented by:

\[ s = \lambda R^{-1} K^{-1} p \]
Then we can build the virtual camera view by finding each pixel’s corresponding position on the cylindrical texture.

\[
T_x = \left( \frac{W_p}{FOV} \right) \left\{ \arctan \left( \frac{-s(1)}{s(3)} \right) \right\} + \frac{W_p}{2}
\]

\[
T_y = \left( \frac{1}{2} - \frac{s(2)}{\sqrt{s(1)^2+s(3)^2}} \right) H_p
\]

\[
W_p, H_p, FOV \text{ are the width, height and field of view of the panoramic texture respectively.}
\]

\[
(T_x, T_y) \text{ are the coordinates of the unrolled cylindrical texture.}
\]

There’s gonna be some sub-pixel circumstances, so interpolation is needed.
CLIENT-SIDE IMPLEMENTATION

• Lots of complex calculation involves in the algorithm

• Port the program to GPU:
  • Decoding video on the CPU
  • Calculation and fetching operation are performed on the GPU

• Nvidia CUDA supports direct OpenGL texture rendering
  • After fetching operations, the output written to the bound texture buffer on the GPU
  • Saving the transfer overhead from device to the host
CLIENT-SIDE COMPARISON

- Performed on a machine with an Intel i7-2600 CPU and an Nvidia GeForce GTX 460 GPU.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Average</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without GPU</td>
<td>255.7</td>
<td>35.8</td>
</tr>
<tr>
<td>With GPU</td>
<td>10.1</td>
<td>3.2</td>
</tr>
</tbody>
</table>

Table 1: Execution time per frame (ms).
CLIENT-SIDE
ZOOM AND PANNING OPERATION

• Panning: Modify the rotation angle of x-axis $\theta_x$.
• Tilting: Modify the rotation angle of y-axis $\theta_y$.
• Zoom: Modify the focal length $f$. 
• When separated with the real network, the client is still able to perform close to the 3-second smooth playout. The varies depend on the bandwidth available to client.
**EXPERIMENT**

**INTERPOLATION**

<table>
<thead>
<tr>
<th>Nearest neighbor</th>
<th>bilinear</th>
<th>bicubic</th>
</tr>
</thead>
<tbody>
<tr>
<td>2916 µs</td>
<td>2840 µs</td>
<td>3242 µs</td>
</tr>
</tbody>
</table>
The resolution has finite impact on performance, since the whole process has moved to GPU.
CONCLUSION

- Based on video stream of some cameras, processed and stitched into a panorama video, we are able to support any free-view angle from the camera array, i.e., a virtual camera.
- The frame of the virtual camera can be generated in less than 10 ms on a computer with a standard GPU. It can then easily be scaled to support many concurrent users.
THX FOR LISTENING