A GPU-Based Free-Viewpoint Video System for Surgical Training

Prof. Pierre Boulanger and Kyrylo Shegeda
Department of Computing Science
University of Alberta

March 2014

Surgical Tele-health Training

https://www.bangsurgical.com/Bang-Product-Info

Multiple Perspectives Improve Laparoscopy

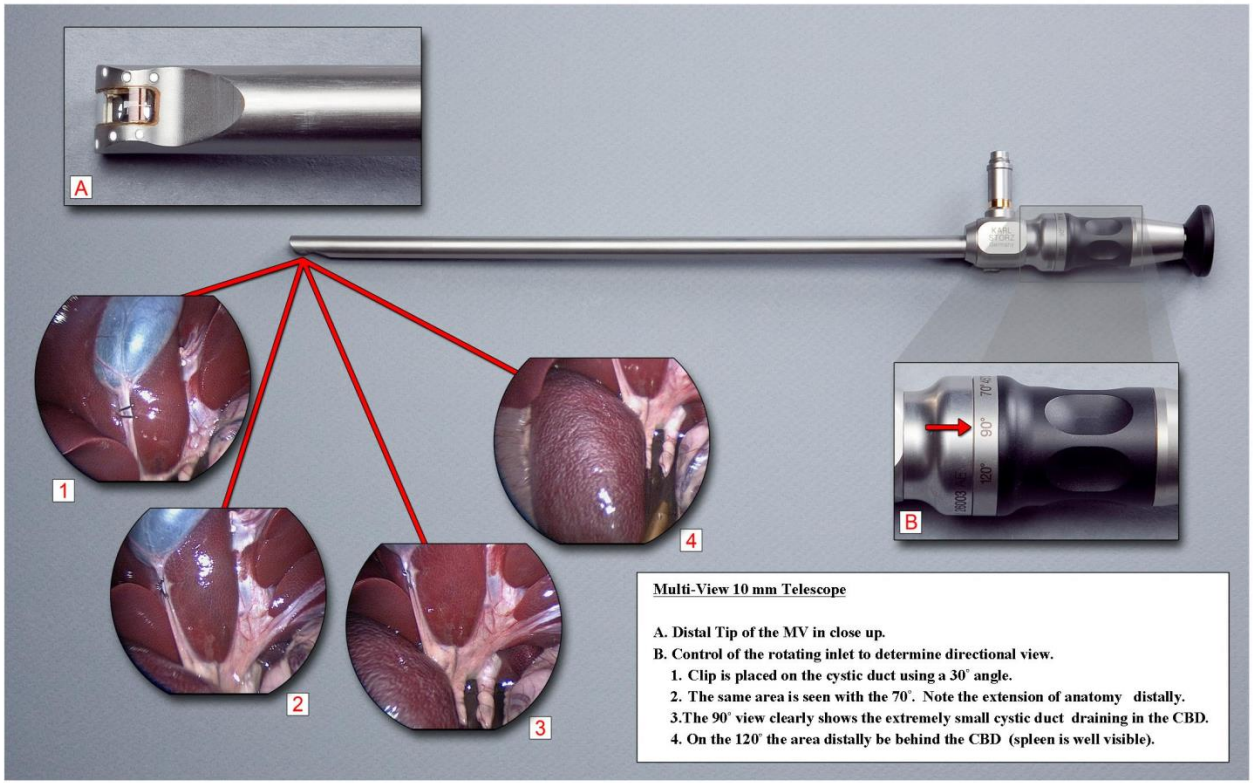
Surgeons given their own view of a laparoscopic task, rather than a shared one, can work more efficiently and accurately, a small new study suggests. Findings from "proof of concept"

experiments appear in the Journal of Laparoendoscopic and Advanced Surgical Techniques

Multi-view Laparoscope

New Multi-View (MV) Rigid Laparoscope

H.J. Soukiasian, M.D., A. Trofimenko, PhD., E.H. Phillips, M.D., FACS, G. Berci, M.D., FACS
Department of Surgery, Cedars-Sinai Medical Center, Los Angeles, CA

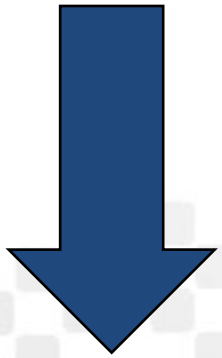


Multi-View 10 mm Telescope

A. Distal Tip of the MV in close up.
B. Control of the rotating inlet to determine directional view.
1. Clip is placed on the cystic duct using a 30° angle.
2. The same area is seen with the 70°. Note the extension of anatomy distally.
3. The 90° view clearly shows the extremely small cystic duct draining in the CBD.
4. On the 120° the area distally be behind the CBD (spleen is well visible).

Next Challenge for Television

- To transmit only partial information
  - (single view) of 3D space



- To transmit all information
  - (all views) of 3D space

FTV (Free-viewpoint TV)

What is Free-viewpoint TV?

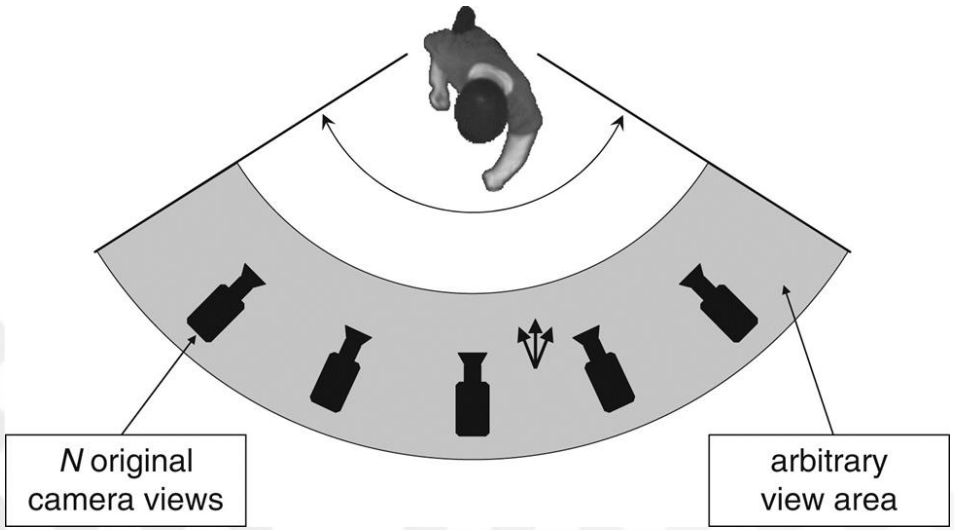
FTV users will be able to freely navigate using a virtual camera viewpoint

The TV scene is captured using a network of synchronised cameras.

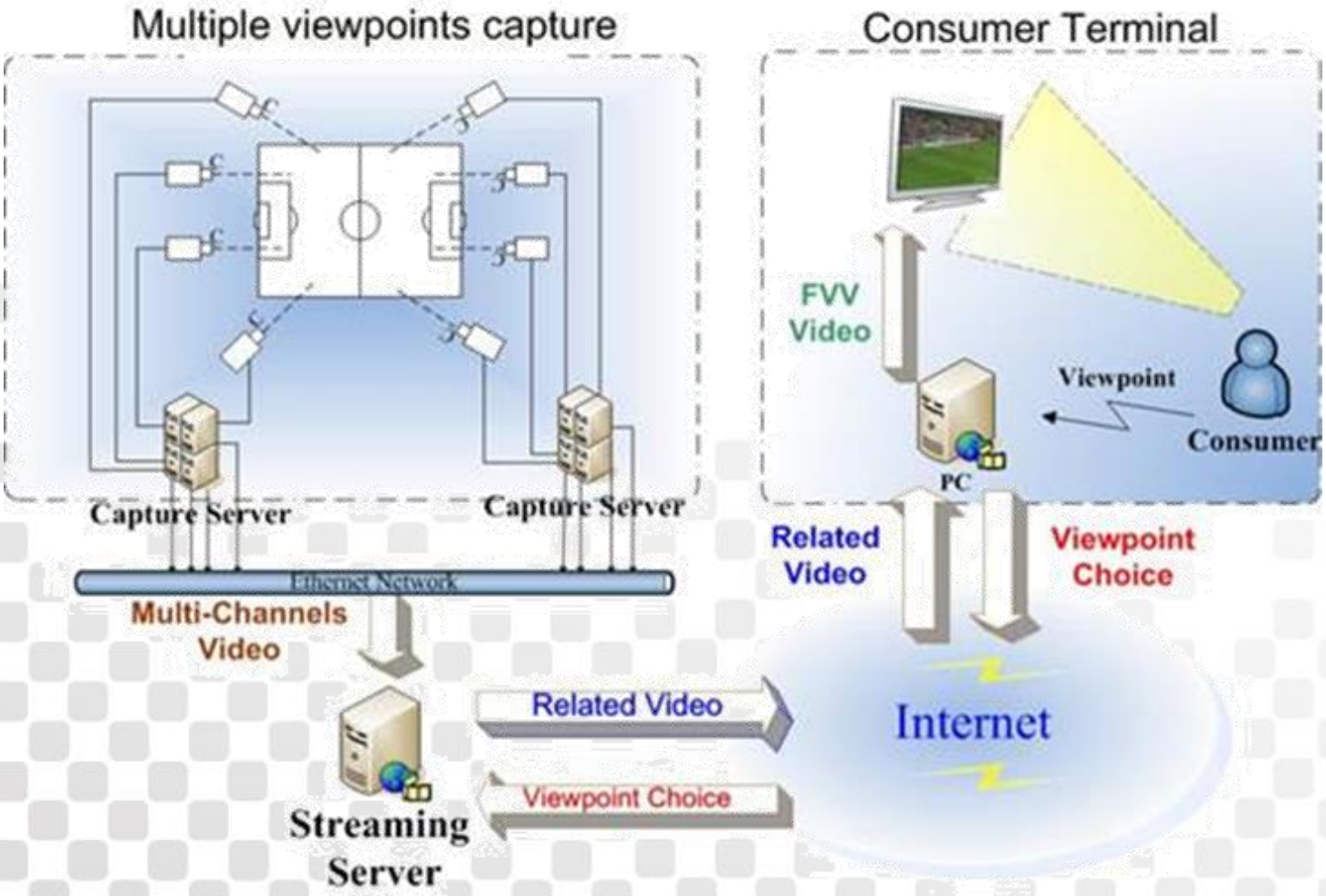
Analysis and fusion of sensor data in order to produce a dynamic 3D-model of the scene.

Transmission of FTV data over IP and graphics rendering on the user's device.

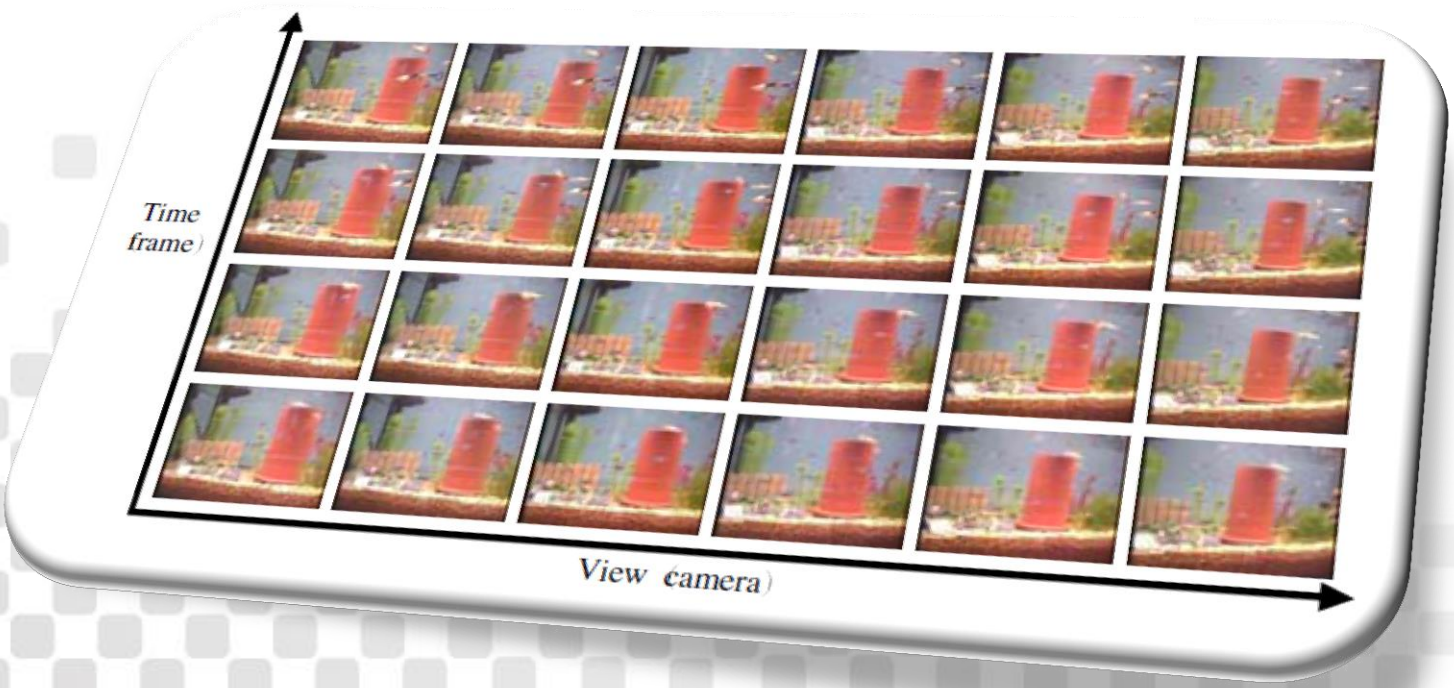
User can navigate through the TV scene using his own "virtual" camera.



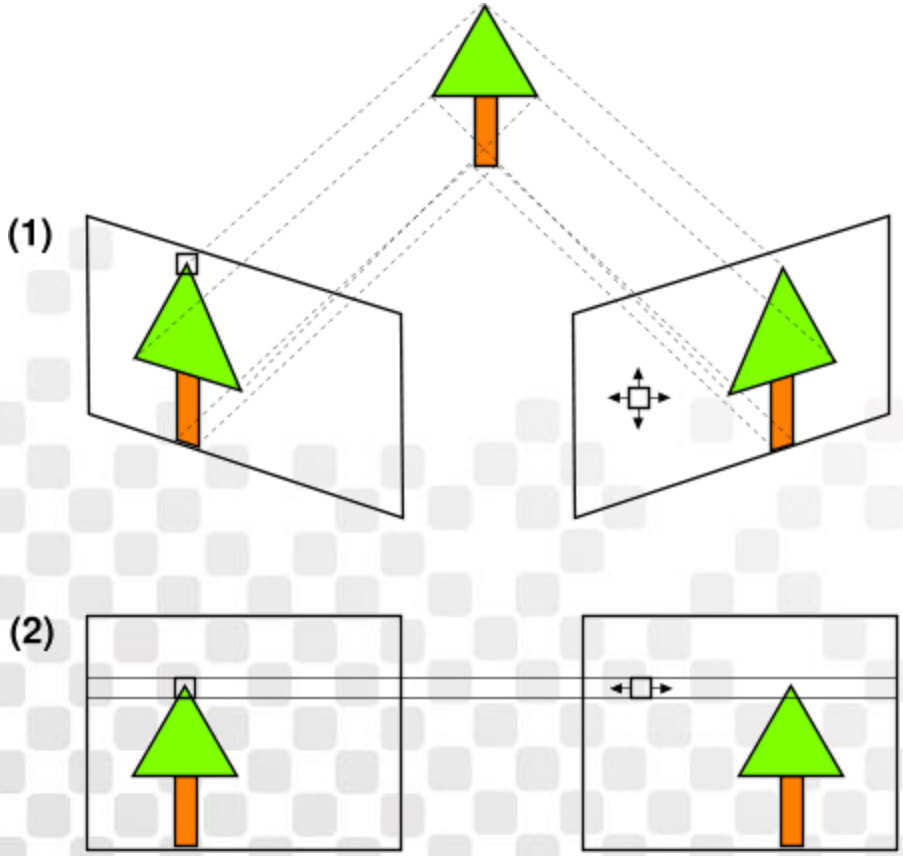
Typical FTV Processing Pipeline



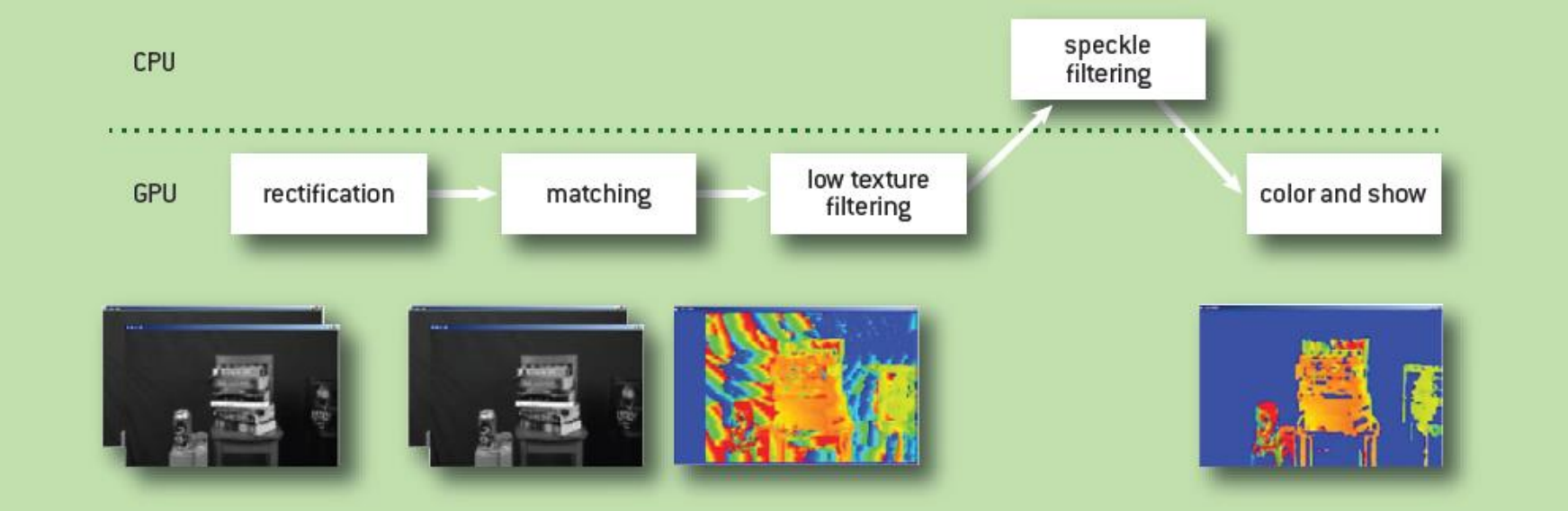
Typical Video Stream



Rectification and Matching

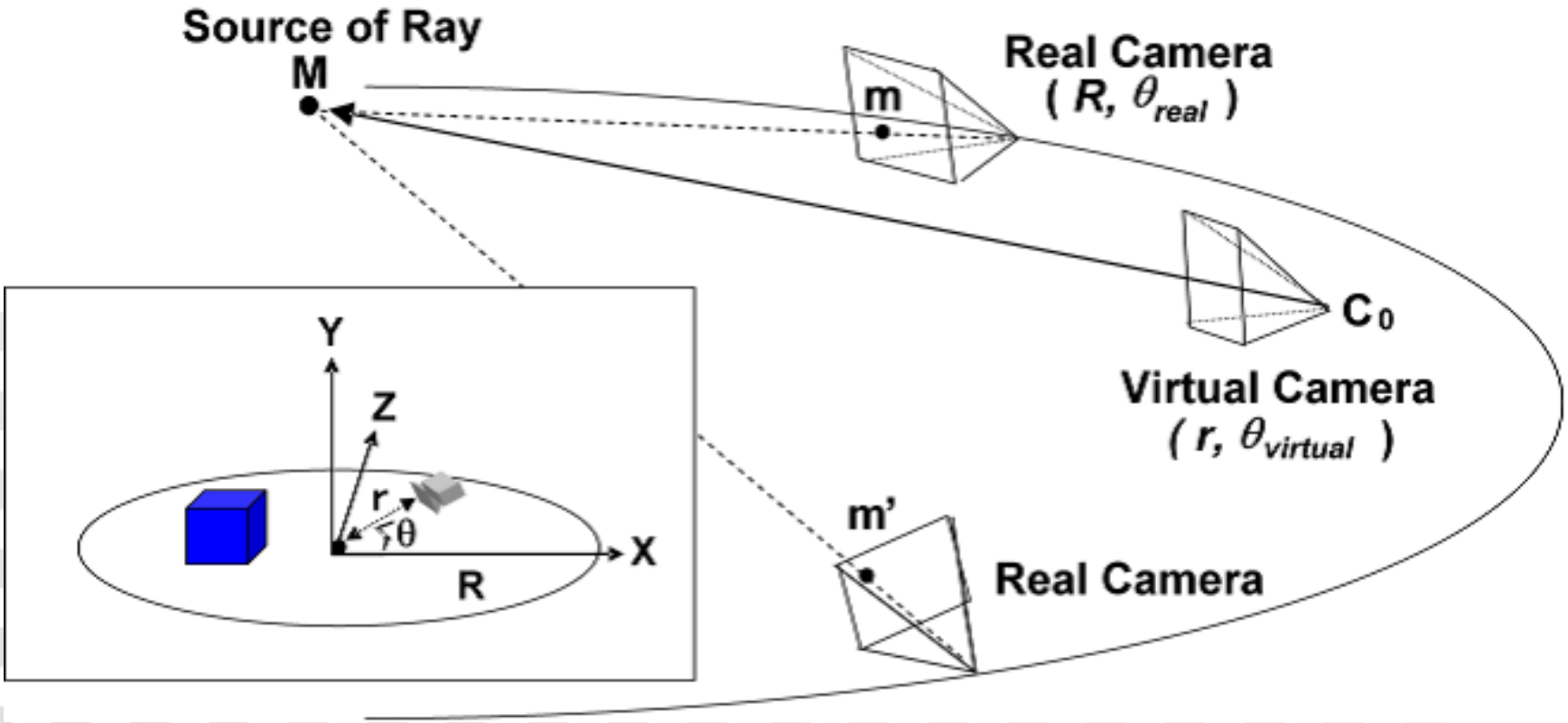


Utilizing GPU Capabilities for Processing HD Video



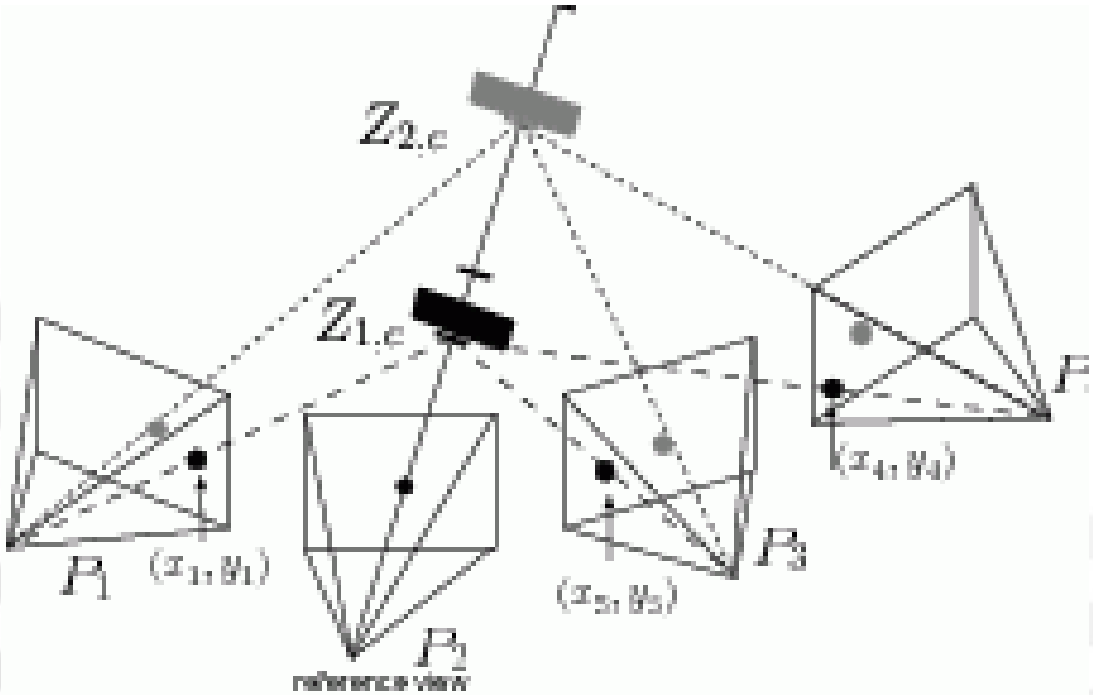
Realtime Computer Vision with OpenCV.
Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, Victor Eruhimov

Ray Analysis Based Method



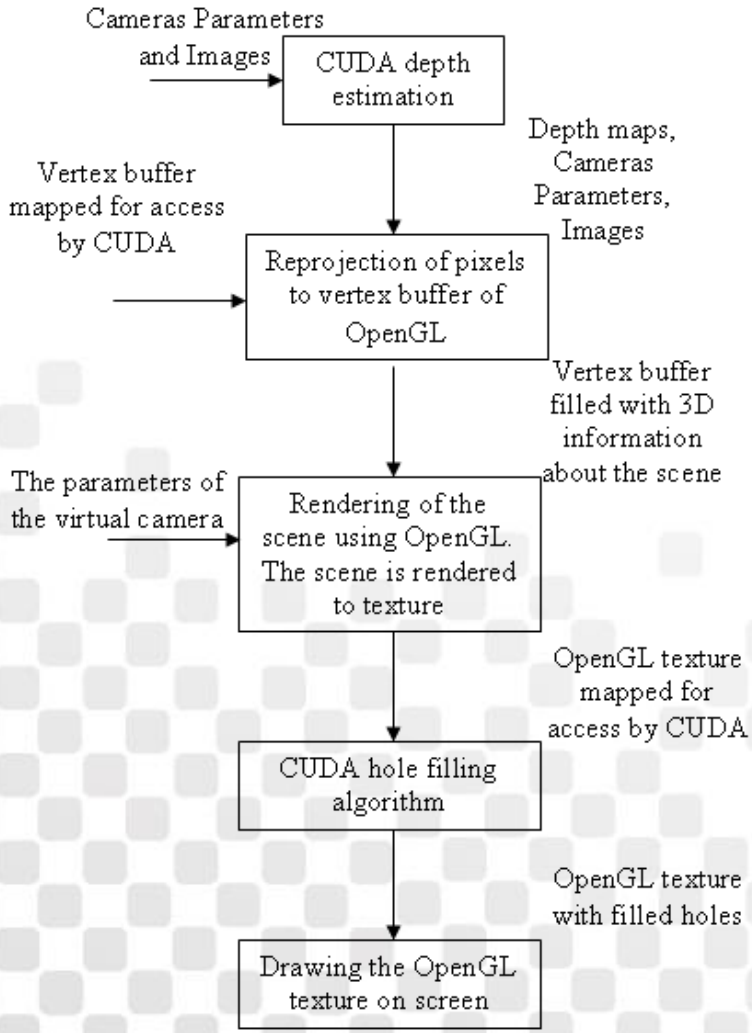
VIEW GENERATION BY RAY-SPACE METHOD
IN CIRCULAR CAMERA SETUP FOR FTV
Takeshi Uemori, Tomohiro Yendo, Toshiaki Fujii and Masayuki Tanimoto

Depth Estimation for Multiple Camera Views



Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video
Yannick Morvan

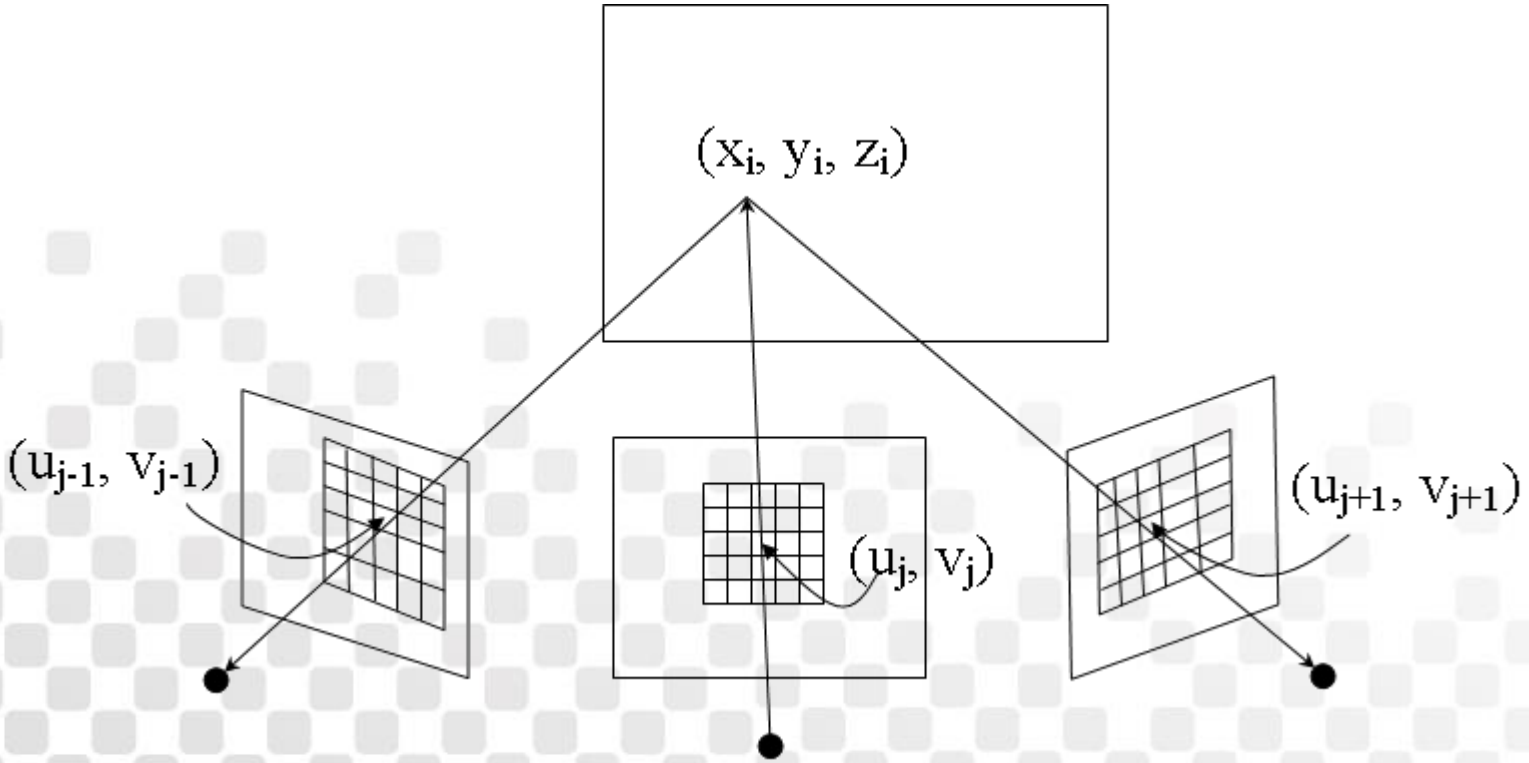
Proposed System



Proposed Framework

- Estimate the depth map for each of the real cameras using projective block matching technique
- Interpolate the depth map and the image of the virtual camera using the information about the depth map of two neighbouring cameras

Projective Block Matching



Sum of Absolute Differences

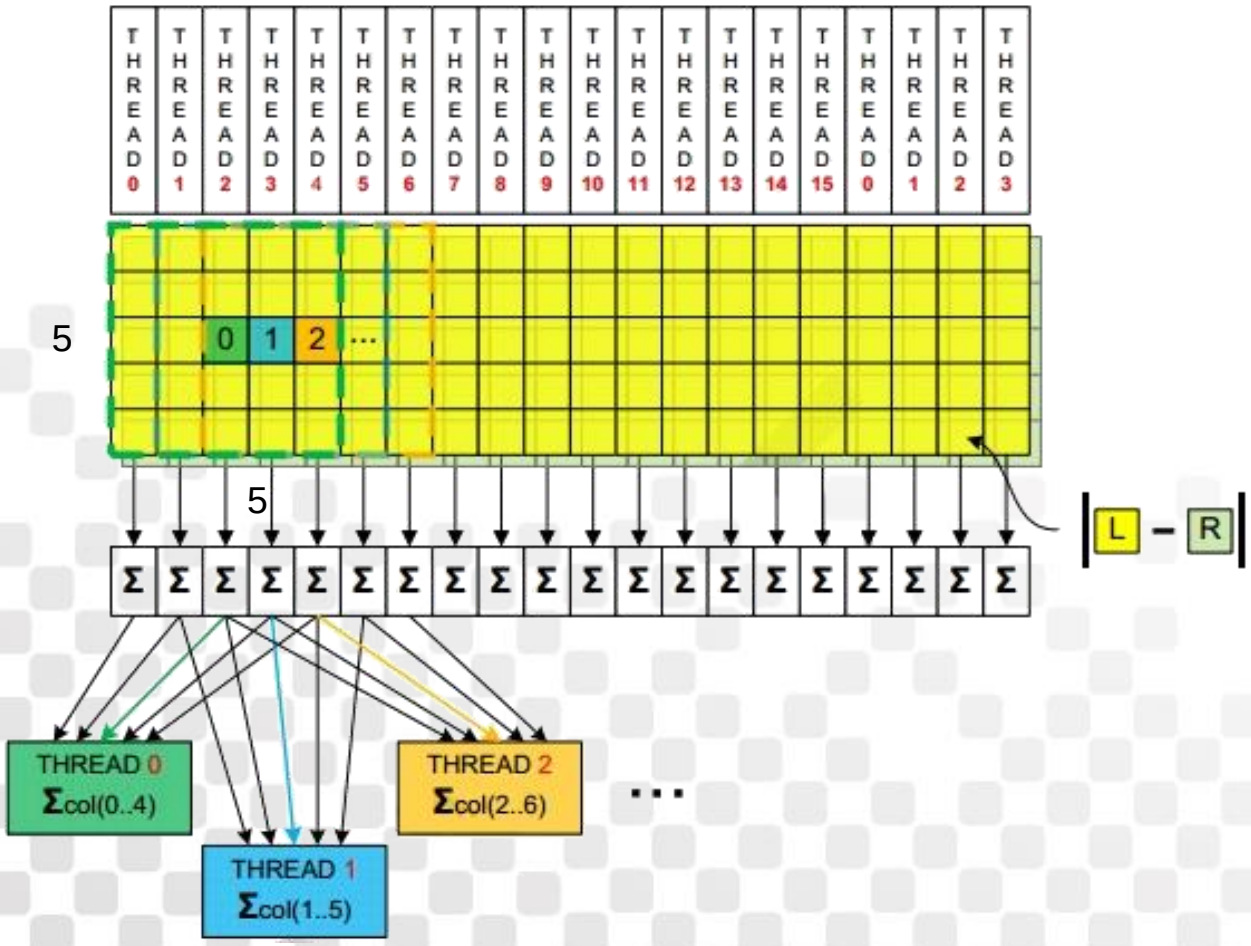
$$SAD = \sum_{k=-M}^{M} \sum_{p=-L}^{L} |\mathbf{I}_{j-1}(\alpha_{j+k}, \beta_{j+p}) - \mathbf{I}_{j}(u_{j+k}, v_{j+p})|$$

- The SAD window was chosen to be the one that gives a better visual quality - 31 by 31 pixels in this particular case
- In the first part of the algorithm, each thread is dedicated to process part of a column of pixels of the camera for which the depth map is being estimated
- Unlike most of the algorithms for depth estimation on a GPU, each thread computes SAD of column of pixels with a block size of (2M + 1) instead of individual pixel

Proposed Interpolation and Occlusion Handling Technique

- Re-project pixels back to global 3D coordinate system from neighbouring cameras
- Project the resulting point cloud onto a reference camera plane
- Fill the resulting holes from the neighbouring pixels

Effective Projective Block Matching Implementation



Reduction in the Number of Operations

- Once the first row of pixels has been processed by a thread block, a rolling window scheme is applied to speed up the calculations
- Instead of repeating the SAD calculation and summation for each of the columns, the SAD function for the pixels in the first row are subtracted from the corresponding one in the accumulation arrays and then the SAD value of the pixels in a new row are added.
- This reduces the amount of calculations that has to be done and more importantly reduces the need to read and write into memory.
- The whole process is repeated for every possible z_i

Reduction in the Number of Operations

- 3 * (2M + 1) reads from texture memory to calculate the whole column of pixels for one thread vs 3 reads from texture memory to substract the SAD of the top row of pixels and add a new row
- 3 * (2M + 1) re-projection operations for each of the threads vs 3 re-projections

Performance Improvement

- Allocate register as array of fixed size (through #define) to store the SAD of the whole column processed by the thread – speedup by a factor of two
- Store the number of depth levels to be processed using preprocessor definition – allows compiler to unroll the loop
- Store cameras' projection matrices in constant memory

View Interpolation without the CPU

- 3D coordinates and color of each of the pixels from neighbouring cameras is stored on the OpenGL 3D vertex buffer.
- Feed the view and projection matrices of the reference plane to OpenGL.
- Render the image on a reference plane into a frame buffer.
- Post-process frame buffer by CUDA filling all the gaps in the image

Advantages-I

- Cameras can be placed around the scene arbitrarily
- All that we need are rotation, translation and intrinsic matrices for each of the cameras using calibration
- No need to rectify and un-rectify images
- The depth map for each camera can be effectively calculated. Depending on the available computational power more views can be employed in it for creating a more consistent map.

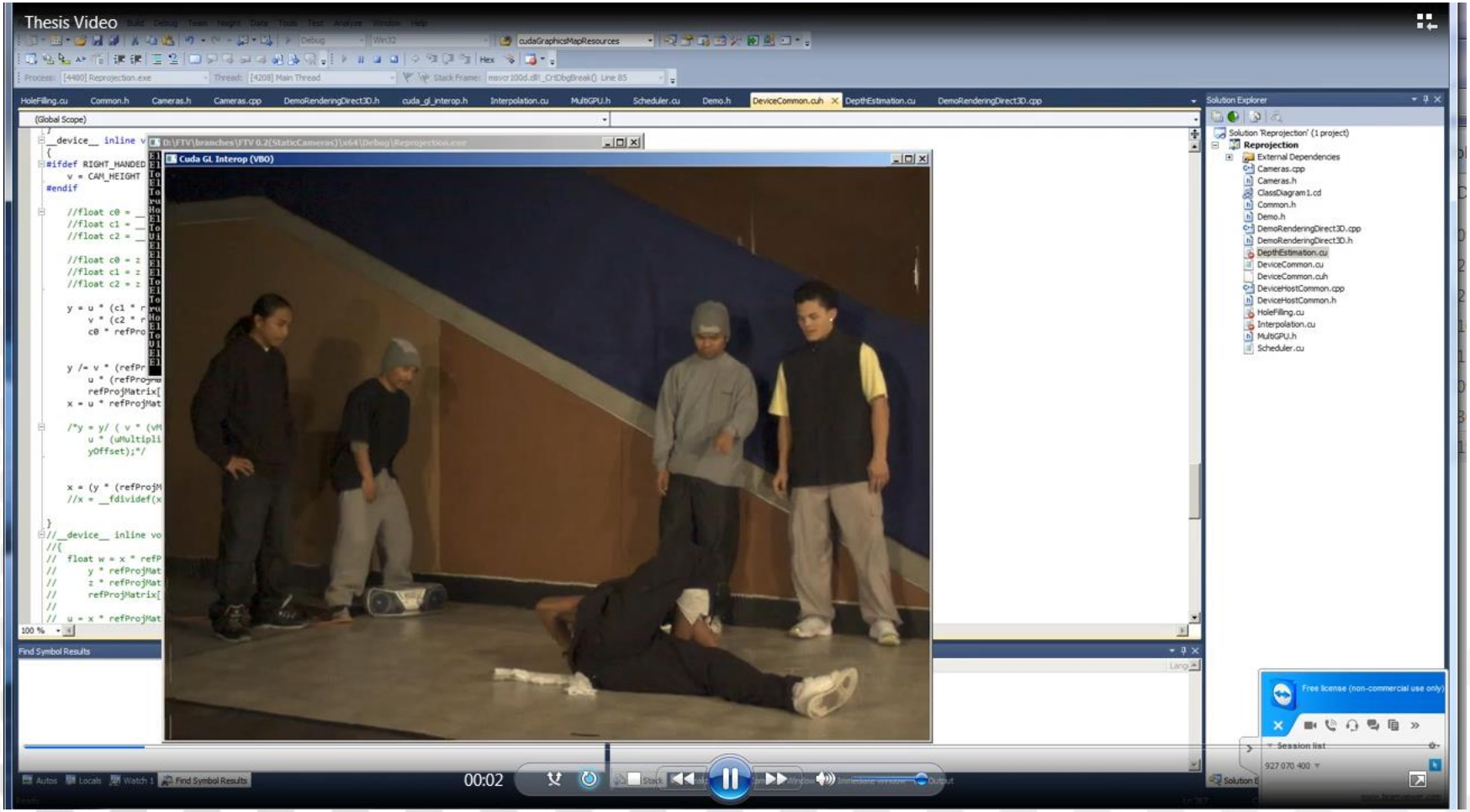
Advantages-II

- The occlusion can be handled more effectively, due to the fact, that virtual viewpoint is created from more than 2 real cameras
- View interpolation part depends only on the depth map for real cameras, thus if it can be calculated more effectively or we don’t need the view to be generated in real time, other depth map estimation techniques can be employed instead.

Testing with Standard Image Database

- To test our framework, we decided to use the sequence “Breakdancing” published by Zitnick from Microsoft research.
- The sequence is 100 frames long with a camera resolution of 1024x768 pixels to simulate HD format.
- In the original paper the depth map was pre-calculated off-line: for each camera the depth map spans 256 depth levels.
- In order to speed up the calculation process, we decided to sweep through every eighth depth plane.
- Thus we sacrificed some of the quality to get the algorithm working in real-time.

Results



Typical Processing Example

Camera 3

New Views with Holes

Camera 5

New Views with Holes Filling

Timing Results

Tested with two Quadro FX 5800 with 30 streaming multiprocessors (SM)

Framework step	Elapsed time (ms)
Depth estimation	16
Projection to 3D	1.15
OpenGL rendering	1.10
Hole filling	2.4
Total	20.65

Discussions

- In our case by using two Quadro FX 5800 we were only able to achieve a 1.3 speedup compared to one GPU.
- This is due to the data transfer on the PCIe 2.0 (16 GBps) which is much slower than the internal bus of a GPU (150 GBps).
- Newer version of the system will allow us to accelerate the transfer rate by a factor two as it will have a PCIe3 bus with a transfer rate of 32 GBps.
- In addition new GPU card like the Keppler has 3072 CUDA cores compared to the Quadro FX 5800 with 240 CUDA cores.
- Another thing is that the number of registers per thread in a new architecture is higher, so we expect it to have a higher occupancy factor.
- **Nevertheless the overall frame rate we achieved was 29.7 frames per second which is very promising.**

Effect of Using a Higher Resolution Depth Map

- To determine the effect of using a higher resolution depth map, we also tested our view generation algorithm with depth maps provided by Zitnick.

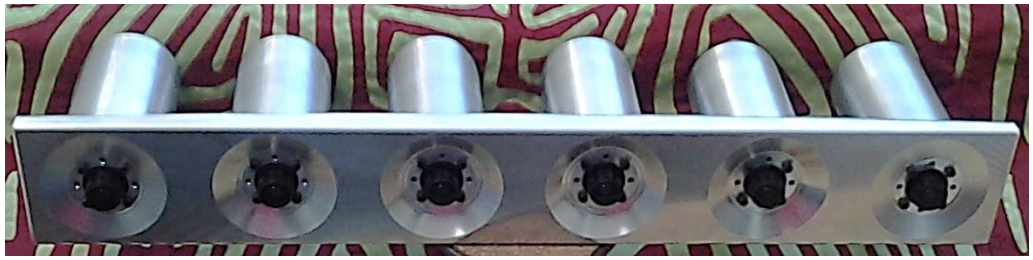
$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{W*H} \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} (L(u,v) - \hat{L}(u,v))^2}$$

Table 3.3: PSNR averaged over 100 frames

Depth map	PSNR
Our algorithm	29
Zitnick	33

Results show that a gain of 4 db in PSNR can be achieved by employing better depth resolution.

OR Compatible Multi-video System



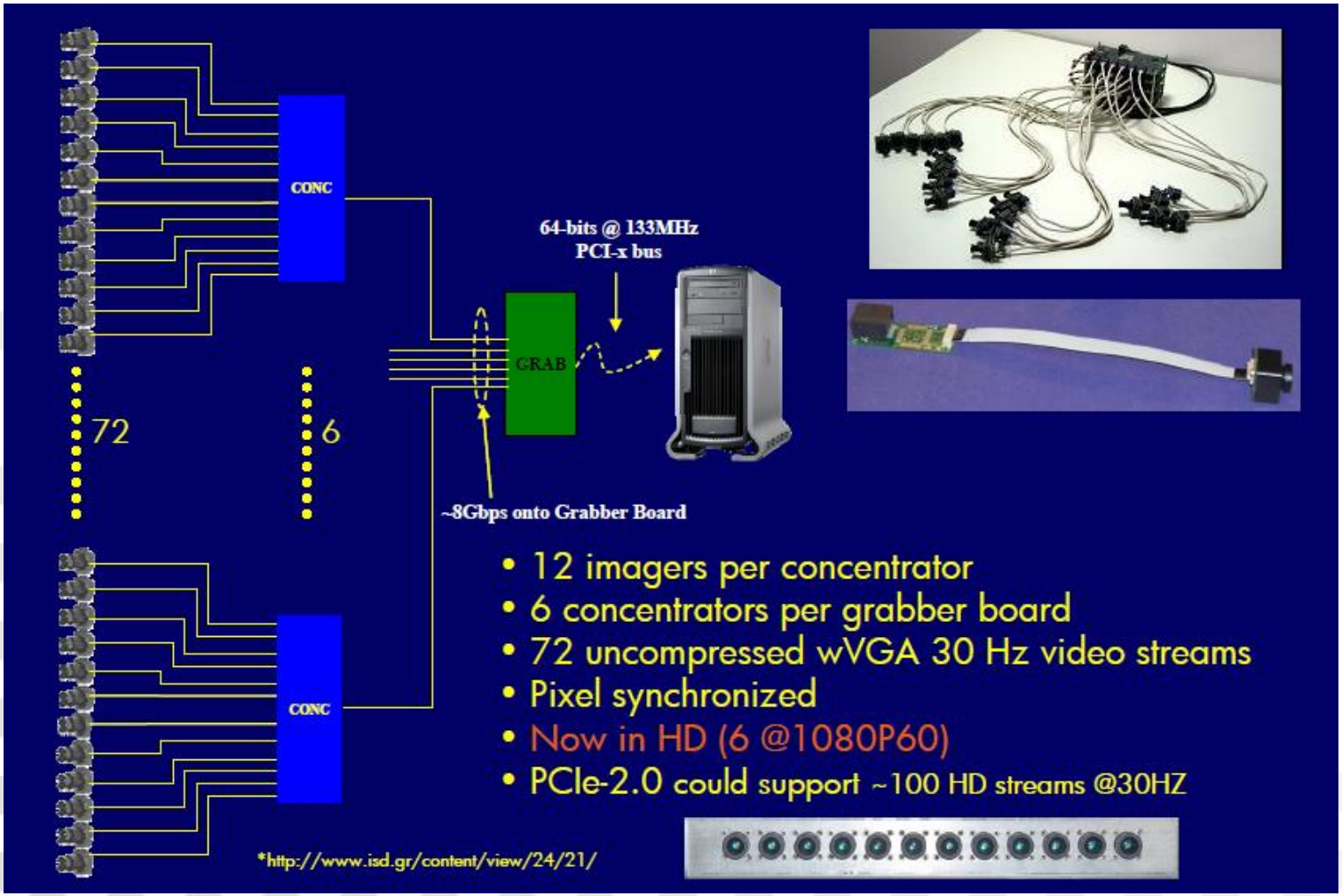
CISCO

ISD S.A.

Integrated Systems Development

http://www.isd.gr

Herodion 72-Camera Architecture



Hardware: Herodion HD system

- Capture and deliver uncompressed 6 HD video frames in real-time
- Hardware synchronized at pixel level
- True HD

Conclusion

- It is now possible to get pixel synchronized true HD picture from multiple cameras at low cost
- Utilizing the capabilities of the GPU allow us to interpolate views in real-time even in HD
- FTV systems is the next generation of media, which gives the user the opportunity to be immersed in the action even over the network
- Next is the OR experiment