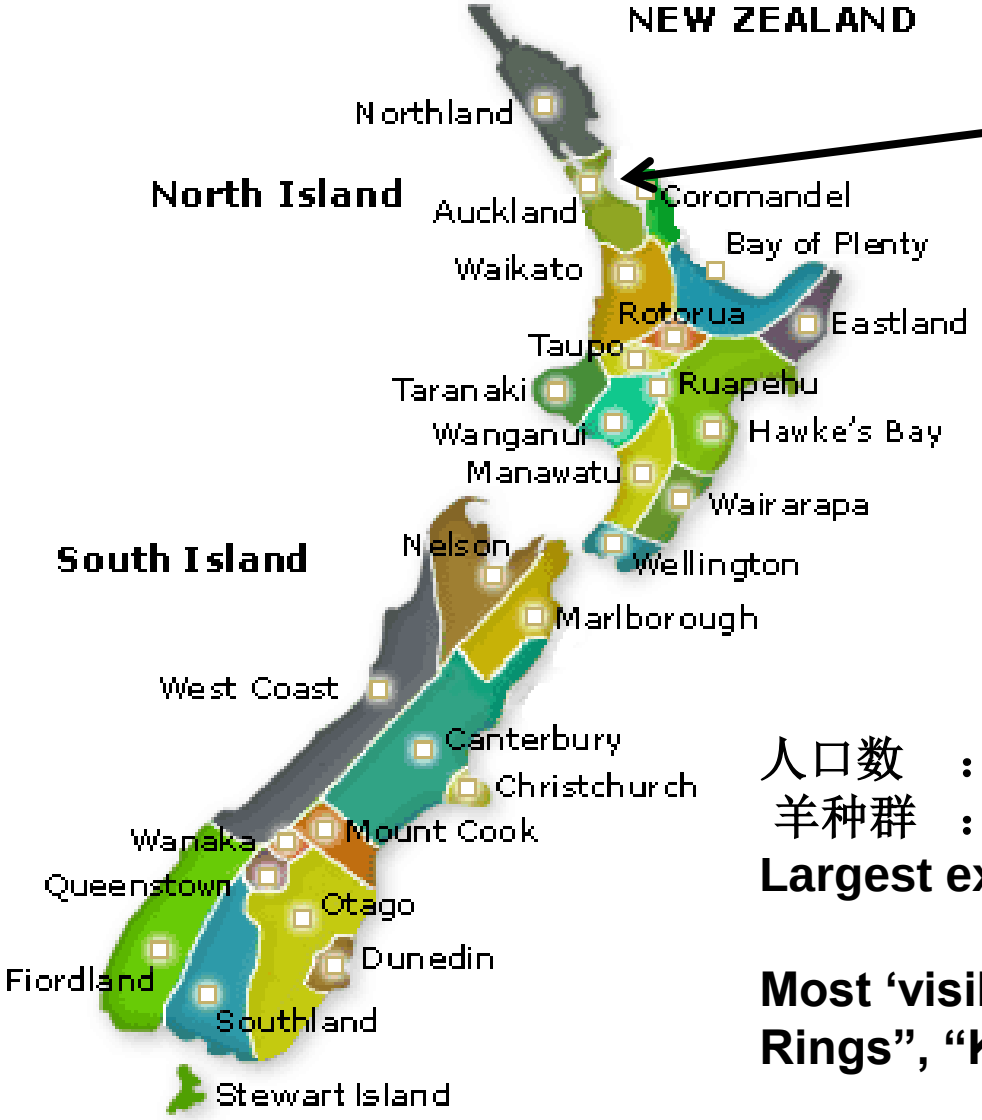
High Resolution Real-time Stereophotogrammetry
高分辨率实时 摄影测量

Khurram Jawed,
John Morris,
Tariq Khan
and
Georgy Gimel'farb

Computer Science/
Electrical and Computer Engineering
The University of Auckland
New Zealand

Iolanthe rounds Channel Island in the Auckland-Tauranga Race, Easter, 2007

New Zealand



Auckland: 1 million people

人口数　：4百万
羊种群　：4千万

Largest exports: Dairy products – milk and cheese

Most 'visible' exports: "Lord of the Rings", "King Kong" and "Avatar"

Auckland has only 1 million people –

bit of a 'country town' – compared to most world cities?

What to do?

Go sailing, of course!!

Grand Goal

- **Emulate human vision**
  - **Capture a 3D model of the environment in real time**

  but

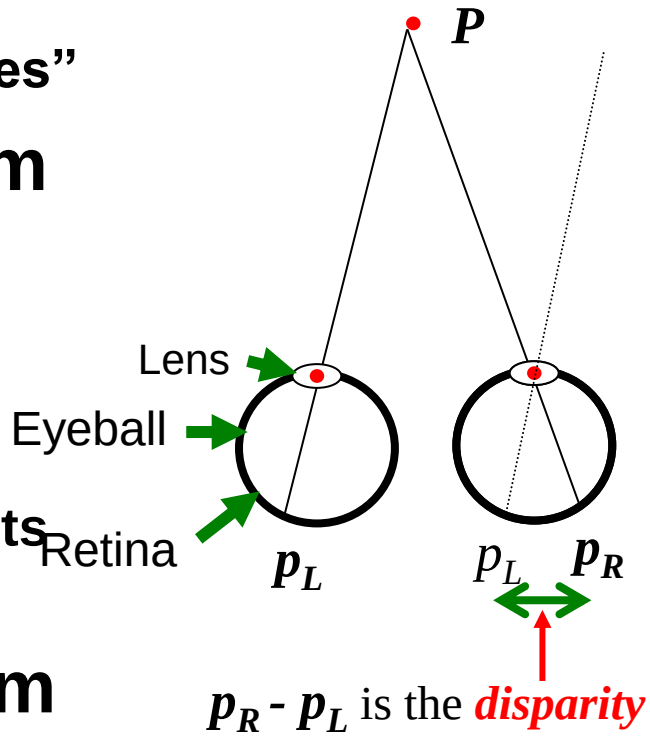
    - **Human eye-brain combination is amazing!**
    - **Our eyes have > 10^8 rods and cones**
    - **Brain is massively parallel (~10^{11}) neurons**
    - **Running very slowly (several kHz)**
    - **Highly robust and fault tolerant**
- **Practicalities**
  - **Single (even multiple core) CPU can't get close**
  - **Use Reconfigurable Hardware (FPGA's)**
- **Our target: Process**
  1 Mpixel images with 1% depth accuracy at 30 fps

Applications – A sample

- Collision avoidance, Autonomous navigation
- Surveillance, security
  - People tracking, counting, …
- Recognition via 3D models
  - Recognition of humans, other objects
- Process monitoring
  - Tracking objects
    - Recognizing orientation, alignment
    - Fruit on conveyer belts
    - Luggage of different sizes
- Marker-less motion capture
  - Movies: capture actor's movements before adding special effects
    - Gollum in 'Lord of the Rings', 'King Kong', 'Avatar', …

Disparity

- **Single image has no depth**
  - **Humans infer depth from scene "clues"**
- **Stereo takes two images from different view points**
  - **left and right image**
- **Disparity**
  - **Displacement of corresponding points from one image to the other**
- **Depth trivially calculated from disparity**



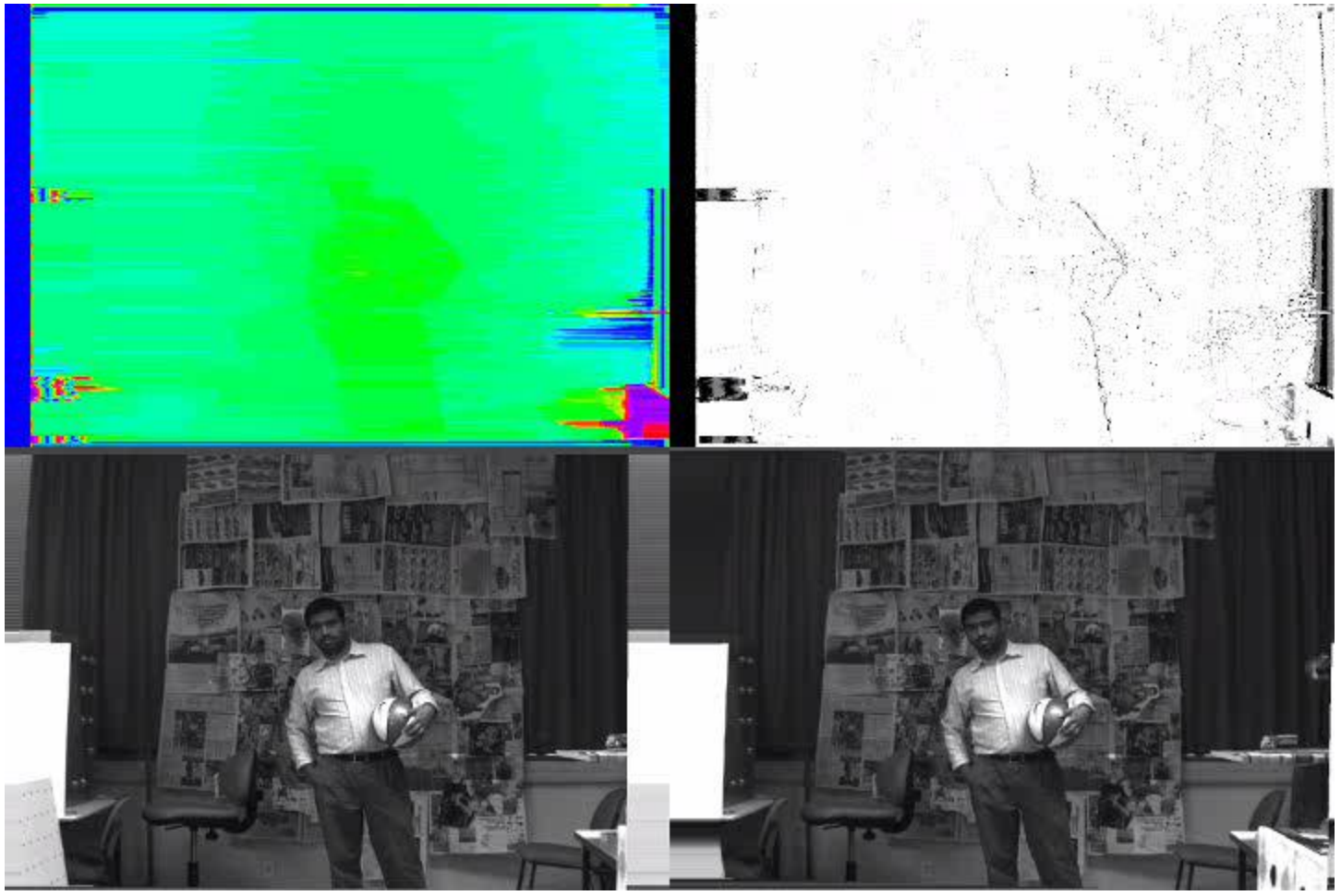
$$\text{Depth, } z = \frac{bf}{pd}$$

b= baseline distance, f = focal length,
p = pixel width, *d* = disparity

Real-time 3D movies

Disparity (Depth) – False Colour

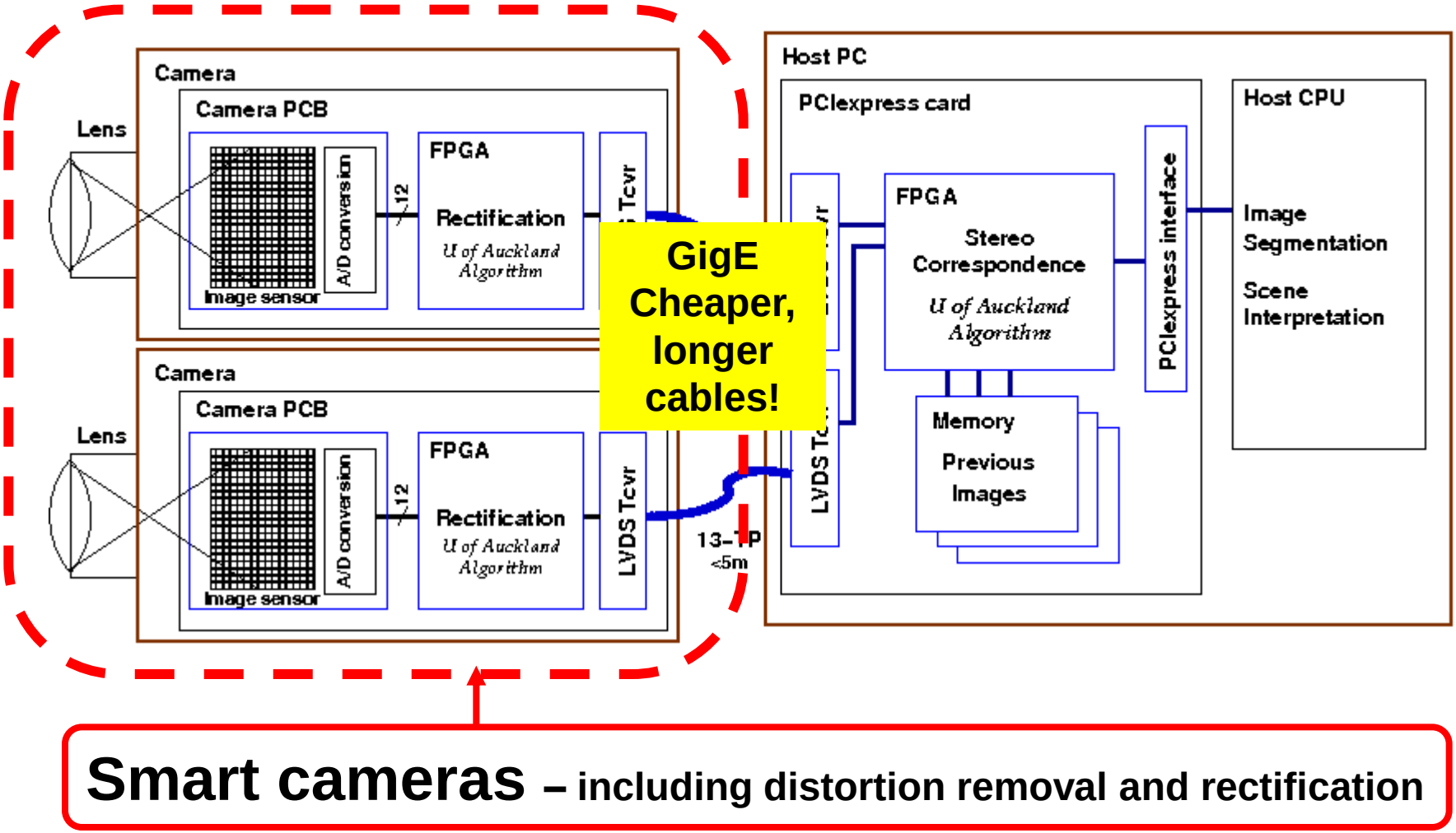
Occlusions



Left Camera (Rectified)

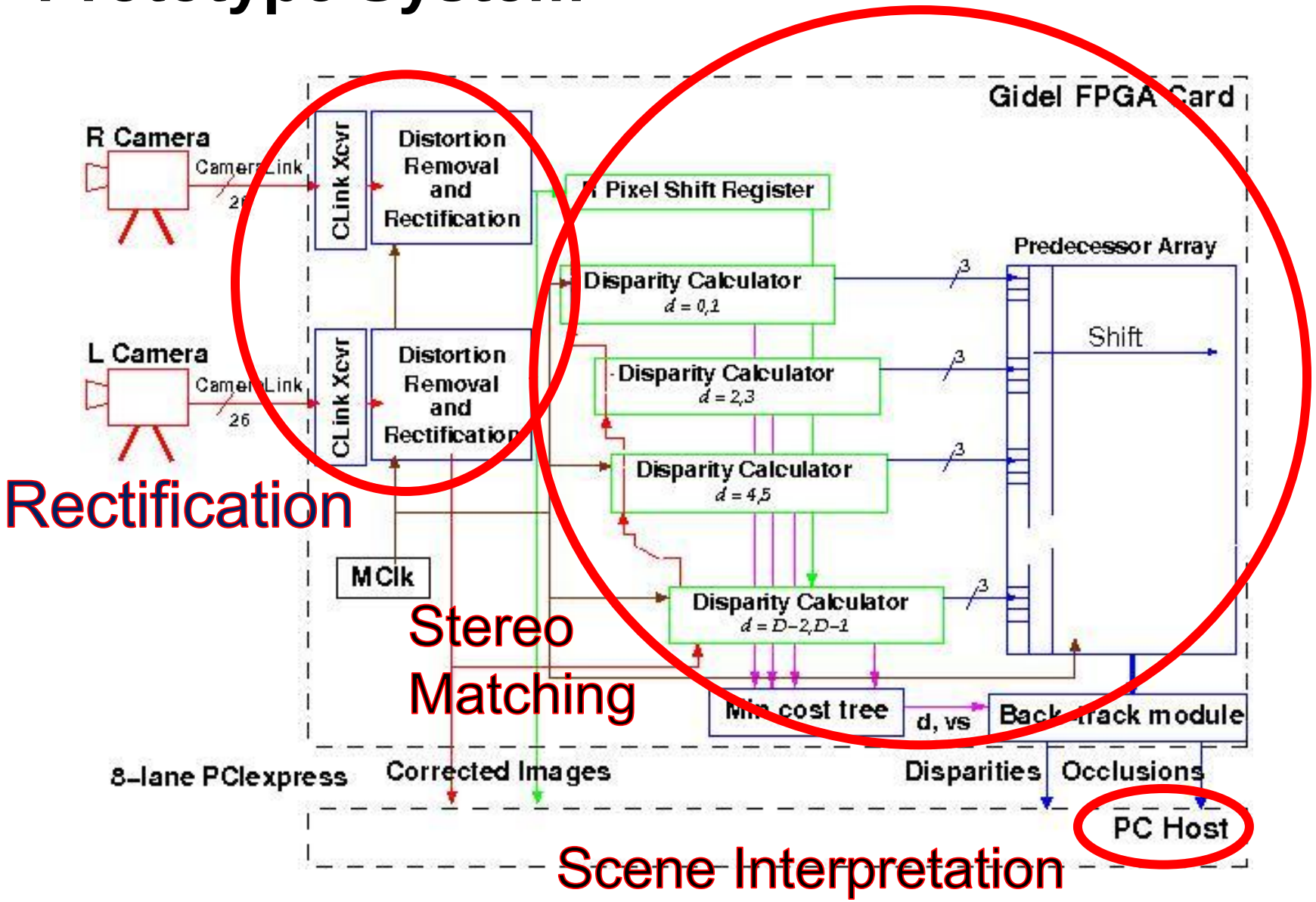
Right Camera (Rectified)

Final System



Smart cameras – including distortion removal and rectification

Prototype System



Rectification

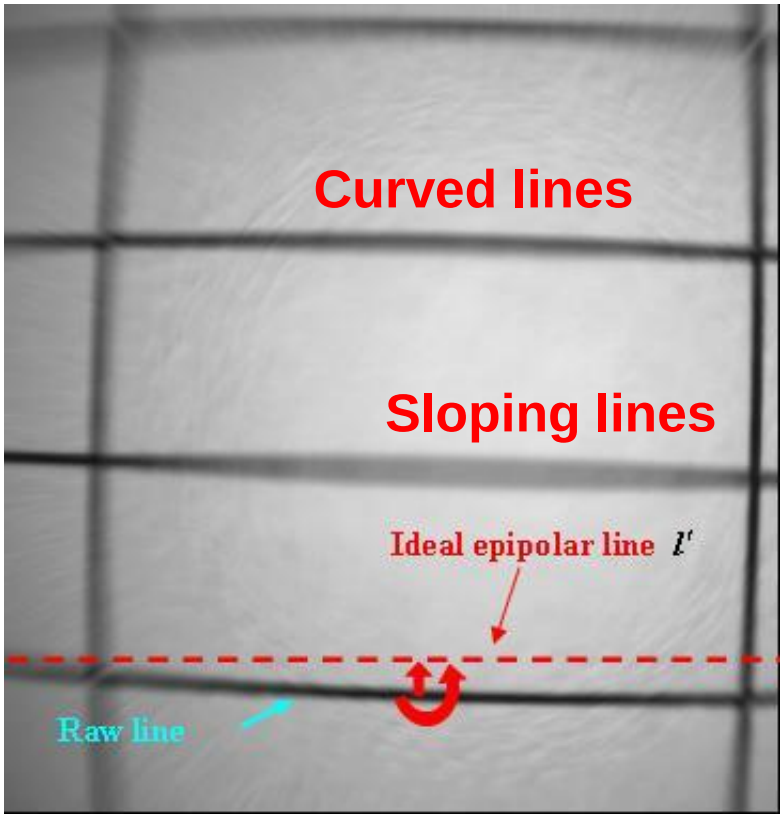
- Removal of effects of lens distortion

and

- Camera misalignment

Real Cameras, Real lenses

- **Real lenses turn straight lines into curves**
- **Stereo needs two cameras**
  - **Perfect alignment difficult (or expensive) to achieve**



Real camera image

Extreme case – very cheap lens!!

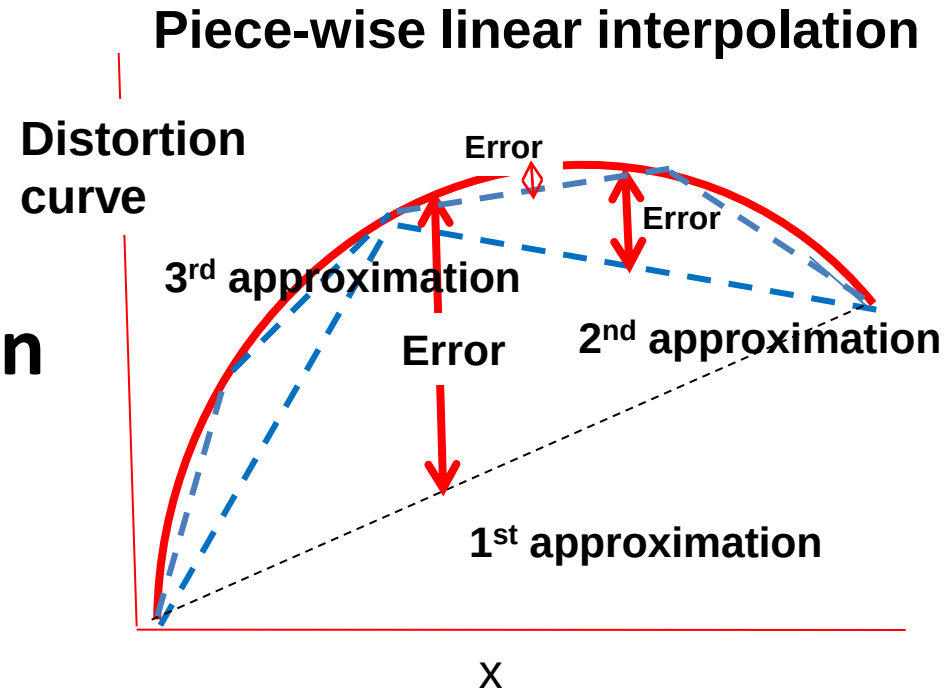
Real Time Rectification

- Naïve approach
  - Use a Look Up Table
  - One LUT entry per pixel
  - Simple, fast

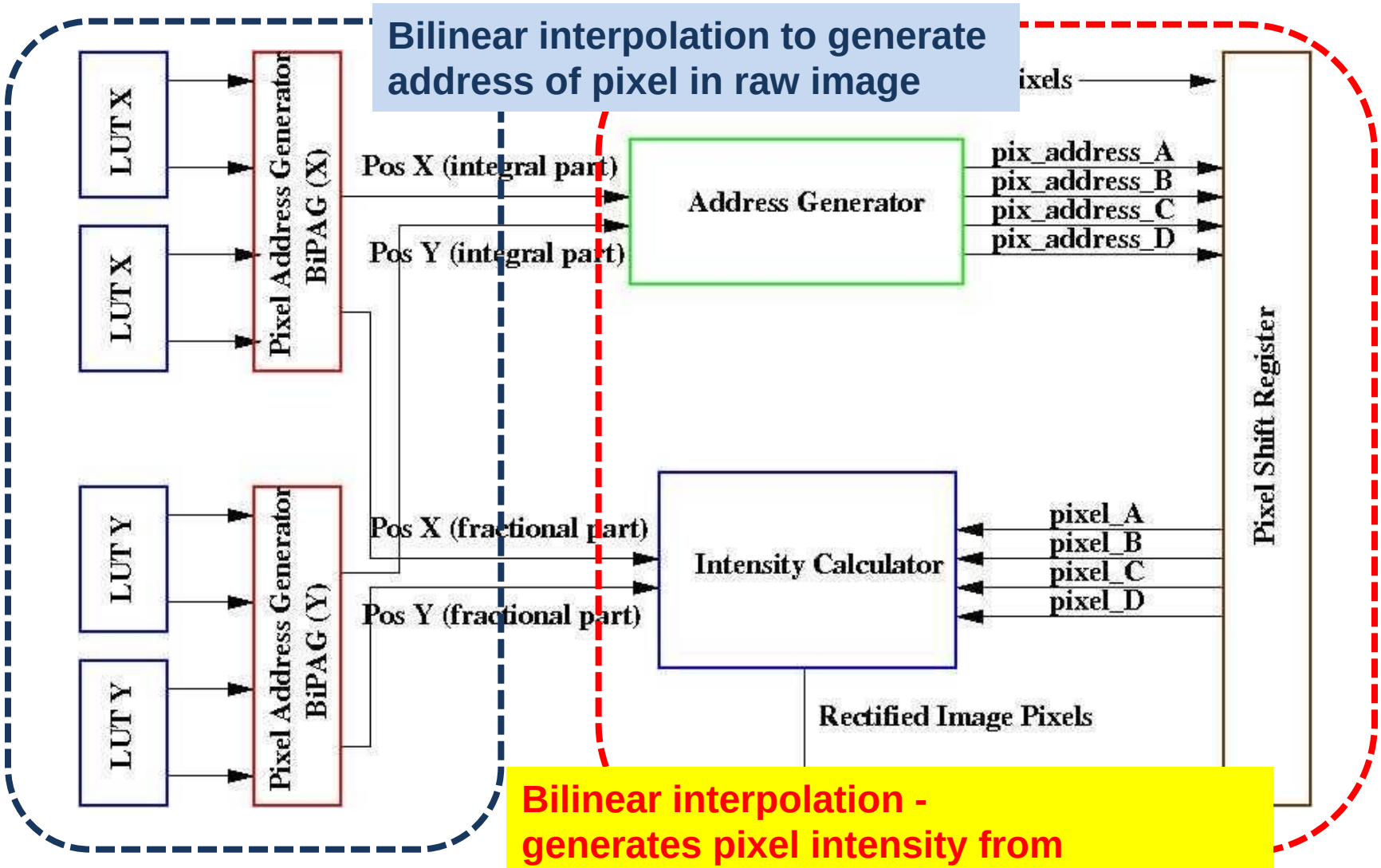
but

- Insufficient memory in current FPGAs
- Distortion functions are smooth

⇨Use reduced LUT

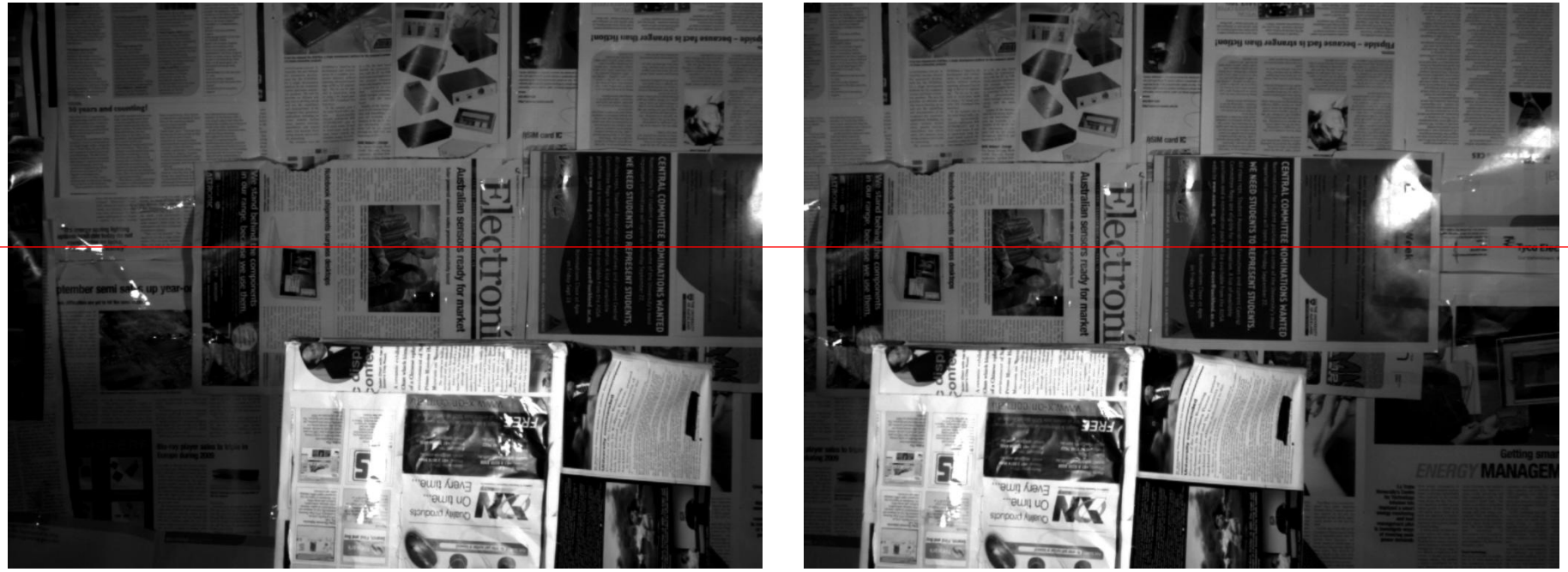


Real Time Rectification



Rectificatio

Raw image pair



Stereo Pair

Rectified image pair

Correspondence

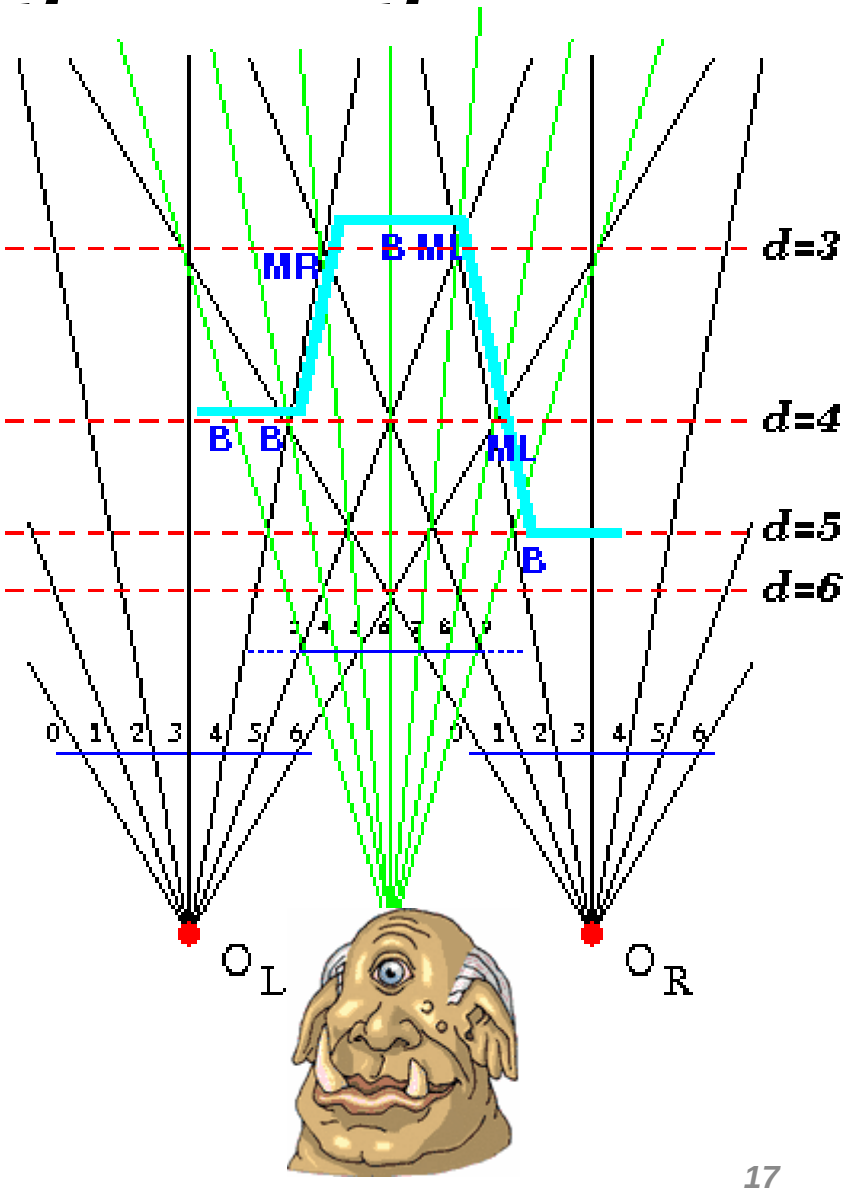
or

Stereo Matching

Determining corresponding points in each image

Symmetric Dynamic Programming Stereo

- Basic aim of stereo
  - Construct a 3D map of the scene
- SDPS
  - Don't try to reconstruct the left (or right) image
  - Turn yourself into Cyclops
  - .. and reconstruct a Cyclopean view
    - Optical centre half-way along baseline
- Key advantage:
  - Label pixels with their visibility states: MR, B, ML
- Disparity changes are associated with visibility state changes



Dynamic Programming Algorithm

For each scan line

For each pixel

Compute optimum cost for each profile

ending at (disparity, visibility state)

Store predecessor

Back-track to generate optimum profile

Cost functions

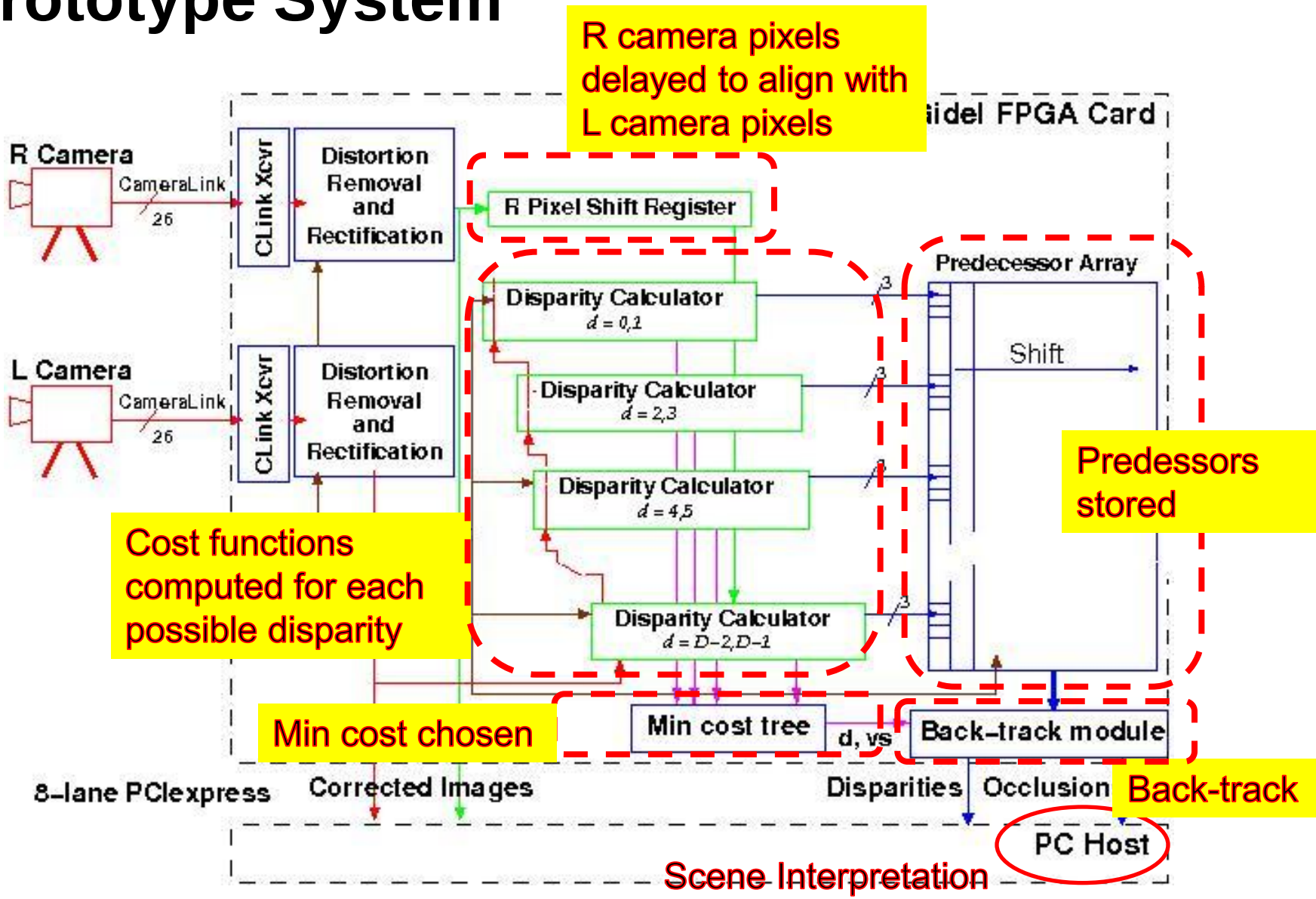
Signal differences $dI(x,d) = |g^{L}_{\frac{x+d}{2}} - g^{R}_{\frac{x-d}{2}}|$

$$c_{x,d,B} = dI(x,d) + \min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR})$$

$$c_{x,d,ML} = occ\_term + \min(c_{x-1,d-1,ML}, c_{x-2,d,B}, c_{x-2,d,MR})$$

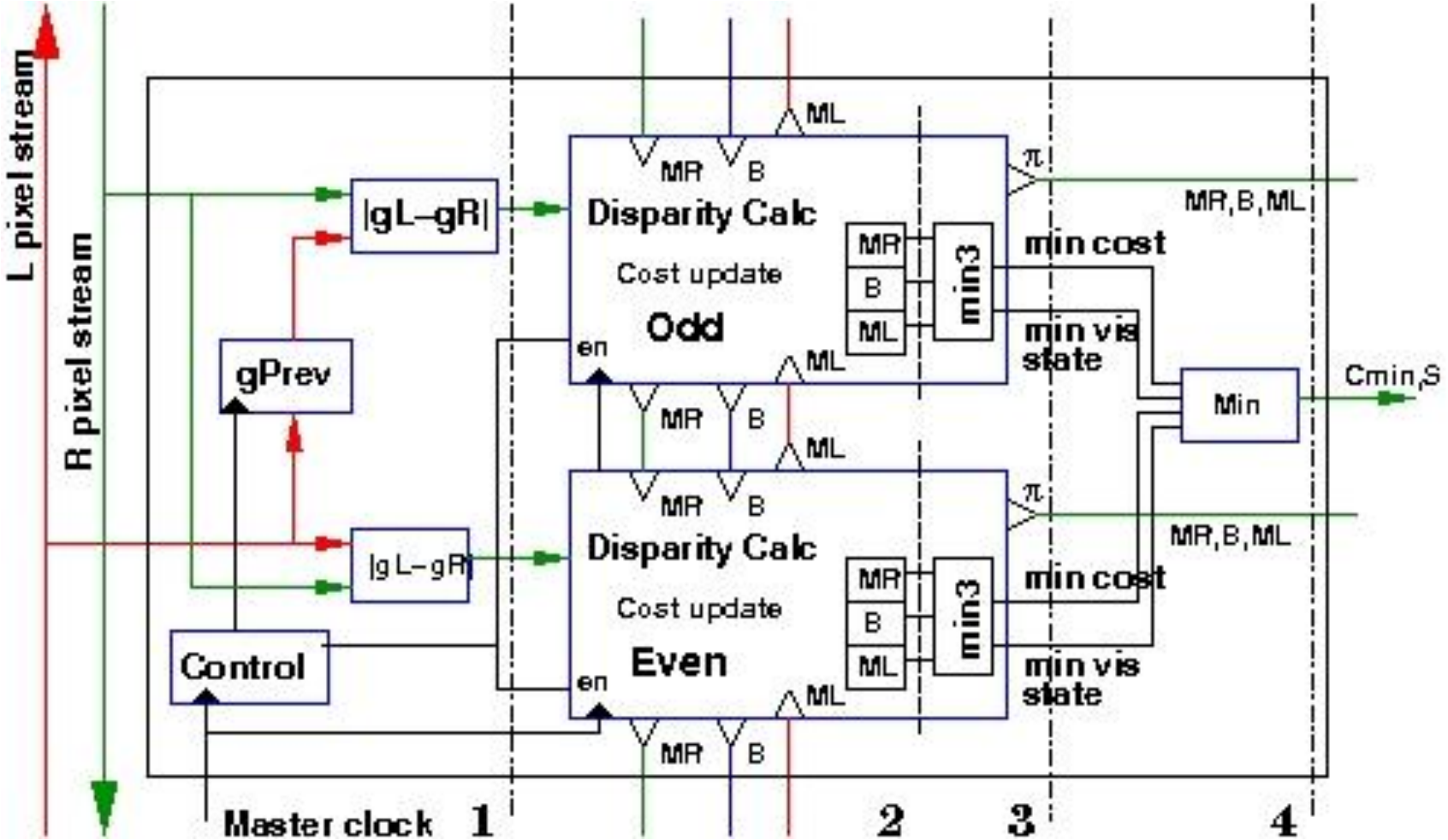
$$c_{x,d,MR} = occ\_term + \min(c_{x-1,d+1,MR}, c_{x-1,d+1,B})$$

Prototype System



Disparity Calculator block

Small, compact, fast circuit **Replicated for each possible disparity**



Pipelined for throughput

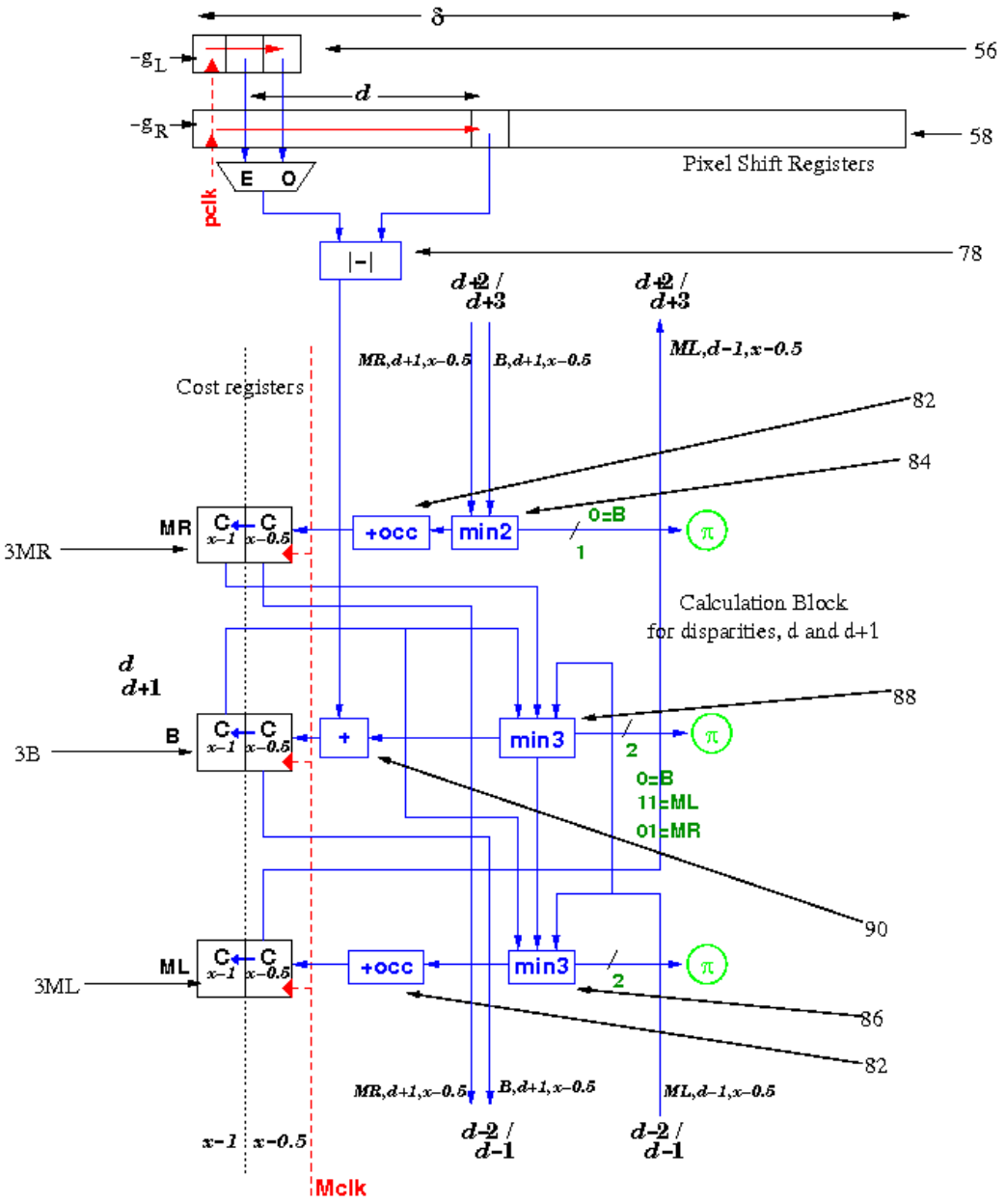
SDPS

- Disparity Calculator Block
- Small
- 2 x min3
- 1 x min2
- 1 x abs difference
- 3 x adders
- Mostly 8-14 bits

- Since it's small, it can be replicated many times

⇒ Large disparity range

⇒ High depth resolution



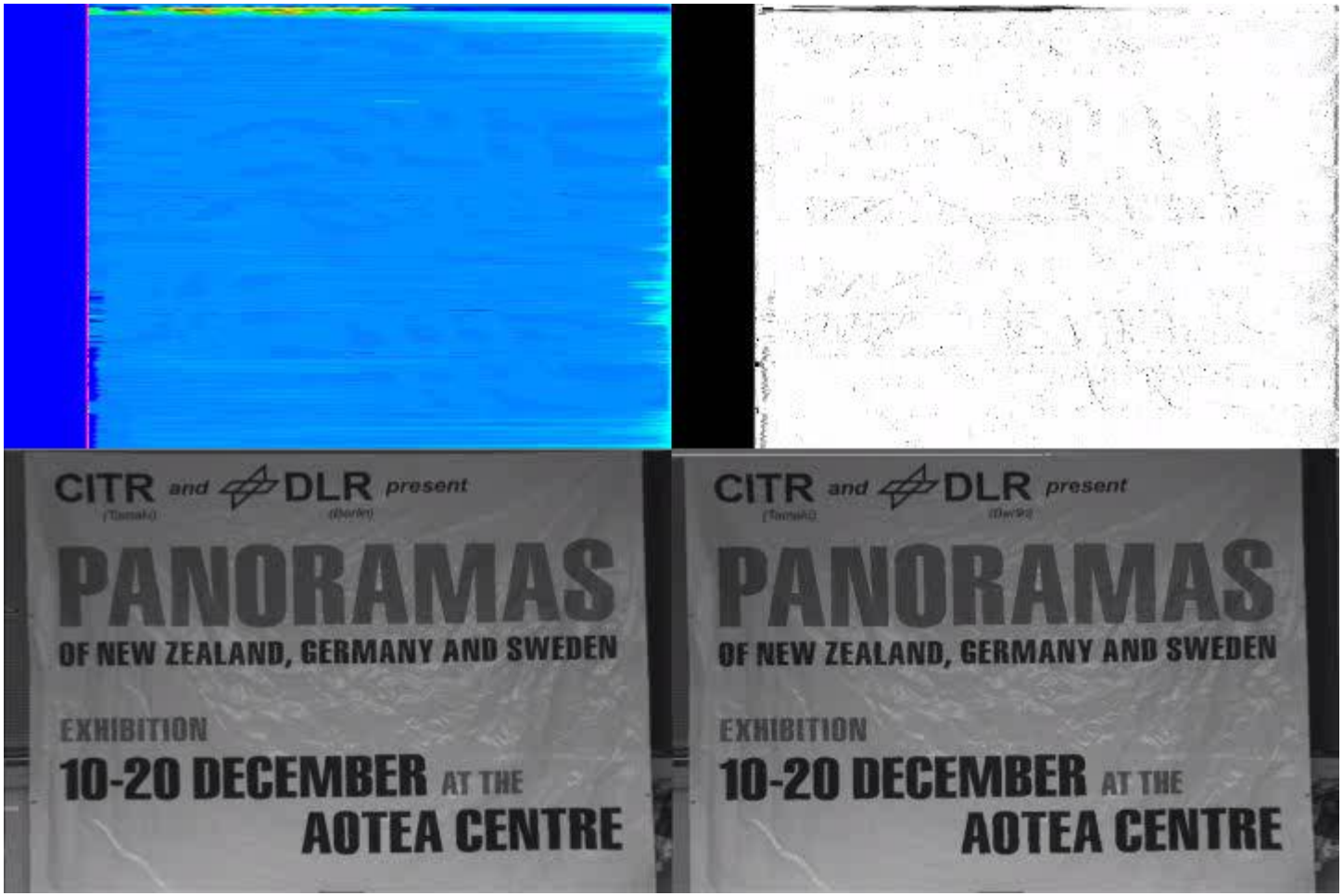
Resource Utilization
Full system *including rectification*

Stratix III EP3SL150

Pixel (bits)	Scan line (pixels)	Scan line buffer	Δ pixels	Logic Utilization		ALUT	Registers	Memory (10^6 bits)
					out of	113600	113600	5.6
8	1024	64	40	23%		17840	12204	3.5
8	512	128	40	22%		17736	12189	3.4
8	1024	64	64	28 %		24056	14315	3.6
8	1024	64	100	39%		17966	17966	3.8
10	1024	8	40	22%		17736	12350	1.9
10	1024	8	128	48%		41284	20645	2.3
12	1024	8	128	48%		42215	21194	2.5
12	1024	8	256	89%		83364	371406	3.2

Needs larger FPGA

Summary


CITR and DLR present
PANORAMAS
OF NEW ZEALAND, GERMANY AND SWEDEN
EXHIBITION
10-20 DECEMBER AT THE
AOTEA CENTRE
CITR and DLR present
PANORAMAS
OF NEW ZEALAND, GERMANY AND SWEDEN
EXHIBITION
10-20 DECEMBER AT THE
AOTEA CENTRE

Graphics Processing Unit (GPU)

Will it perform better?

Graphics Processing Unit (GPU)

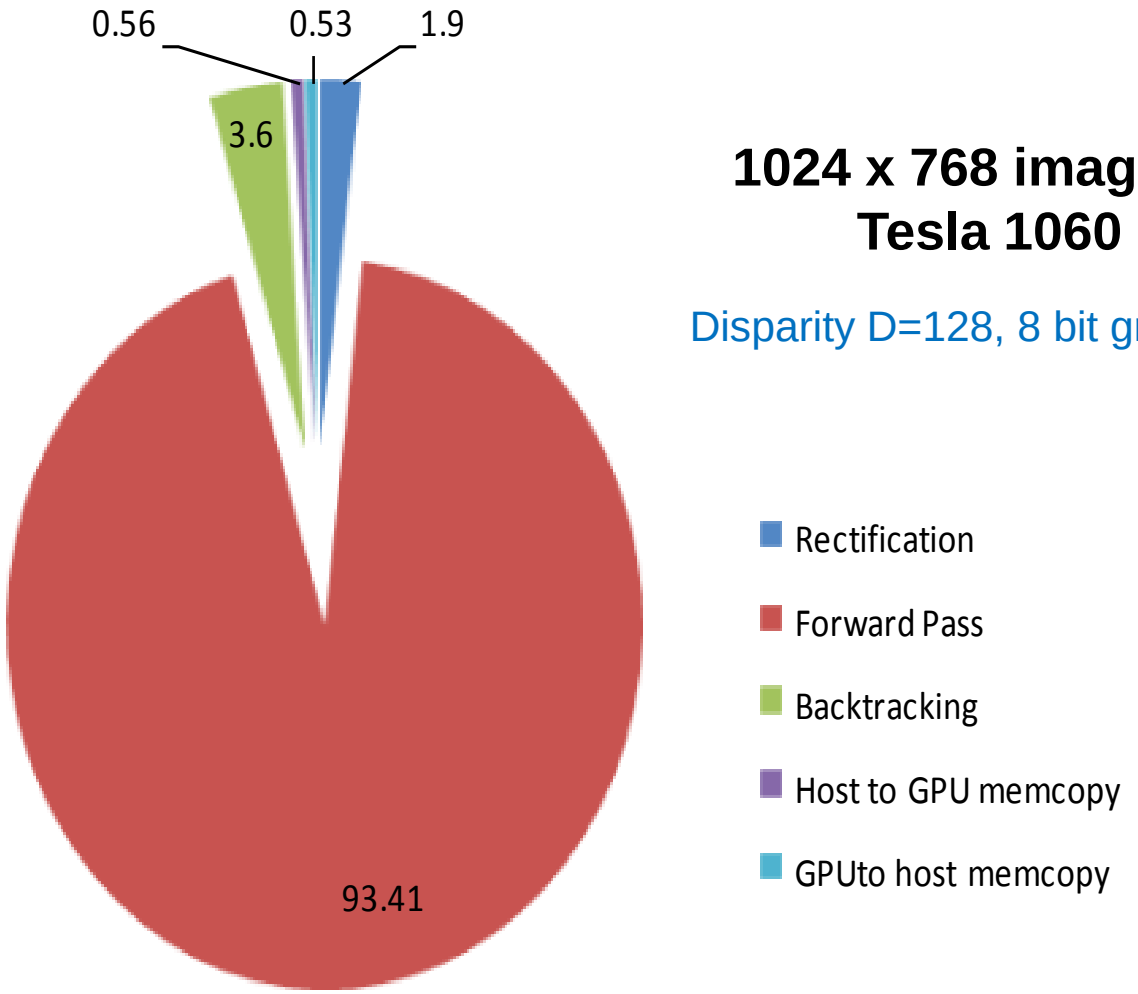
- **Modern GPU**
  - **Cheap**
  - **One in every PC**
  - **~200 ALUs in SIMD architecture**
  - **Very high raw computation power!!**
- **Optimized for rendering images**
- **Can the computation power be harnessed for other applications?**
- **We implemented the same stereo algorithm (Gimel'farb's SDPS) on a high-end GPU**
  - **nVidia Tesla**

Results: Execution time

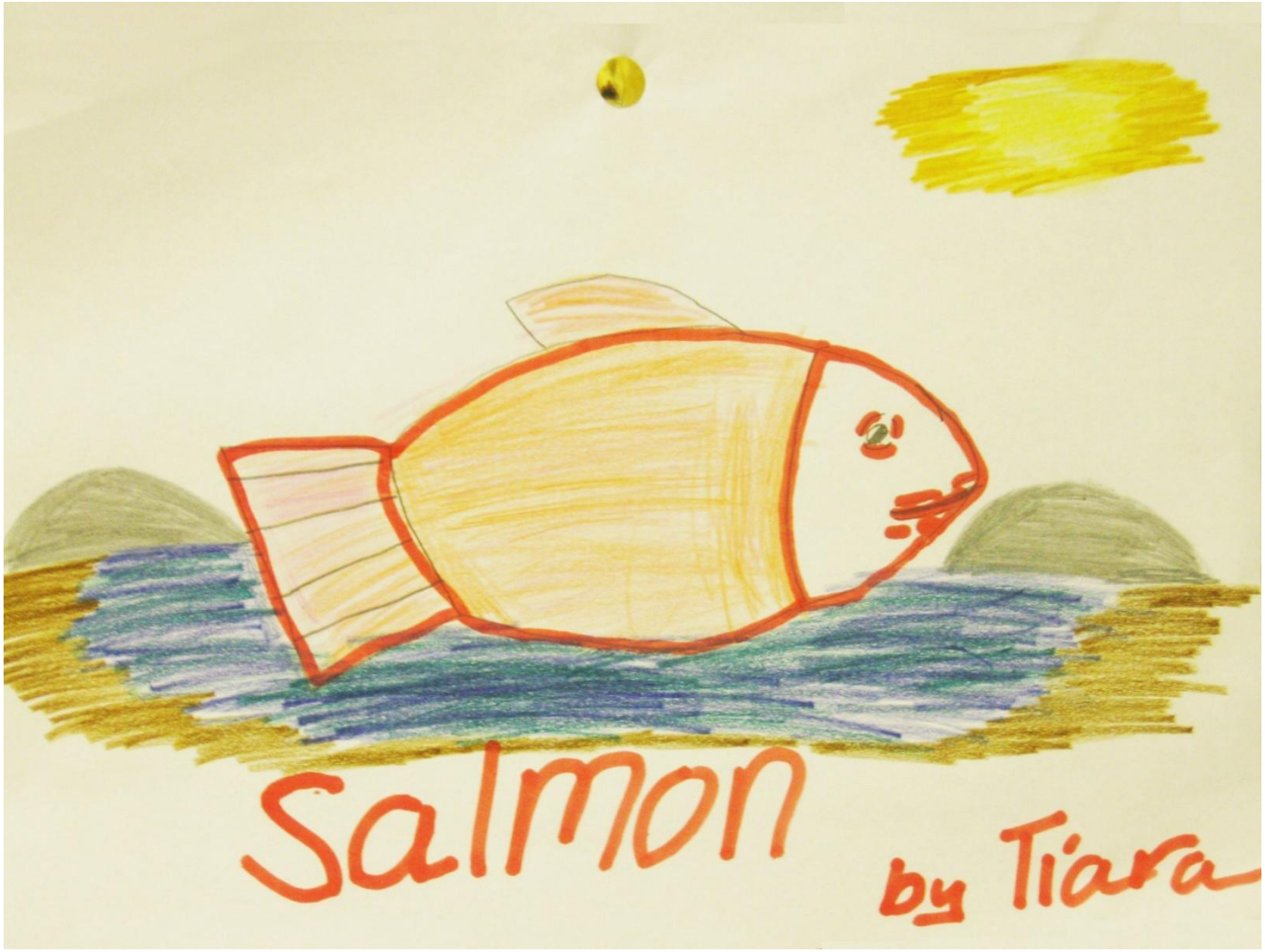
Resolution	Max disparity	GPU Nvidia GeForce GTX 280	FPGA Altera Stratix III	CPU 1 Intel Xeon 2.33GHz - 64 bit	CPU 2 Intel extreme 3.16 GHz
1024x768	128	20 fps 1.76 x 10^9 disp/sec	30 fps 2.6 x 10^9 disp/sec	0.18 fps 0.016 x 10^9 disp/sec	0.22 fps 0.019 x 10^9 disp/sec
640x480	128	55 fps 1.74 x 10^9 disp/sec	80 fps* 2.5 x 10^9 disp/sec	0.52 fps 0.016 x 10^9 disp/sec	0.57 fps 0.018 x 10^9 disp/sec
320x240	128	230 fps 1.4 x 10^9 disp/sec	400 fps* 2.4 x 10^9 disp/sec	2.7 fps 0.016 x 10^9 disp/sec	2.9 fps 0.017 x 10^9 disp/sec
256x256	128	310 fps 1.3 x 10^9 disp/sec	450 fps* 2.2 x 10^9 disp/sec	3.7 fps 0.016 x 10^9 disp/sec	5 fps 0.019 x 10^9 disp/sec
2048x1536	256	2.86 fps 2.02 x 10^9 disp/sec	Not feasible on this FPGA	0.0022 fps 0.016 x 10^9 disp/sec	0.0027fps 0.019 x 10^9 disp/sec

- All results are for 8 bit grey images.
- Starred results for the FPGA implementation estimated by de-rating 1024 x 768 pixel results for the increased overhead of re-trace times and blank lines in typical cameras
- CPU results are using single core

Results: Execution time breakdown



Salmon: Precise 3D Contours in Real Time

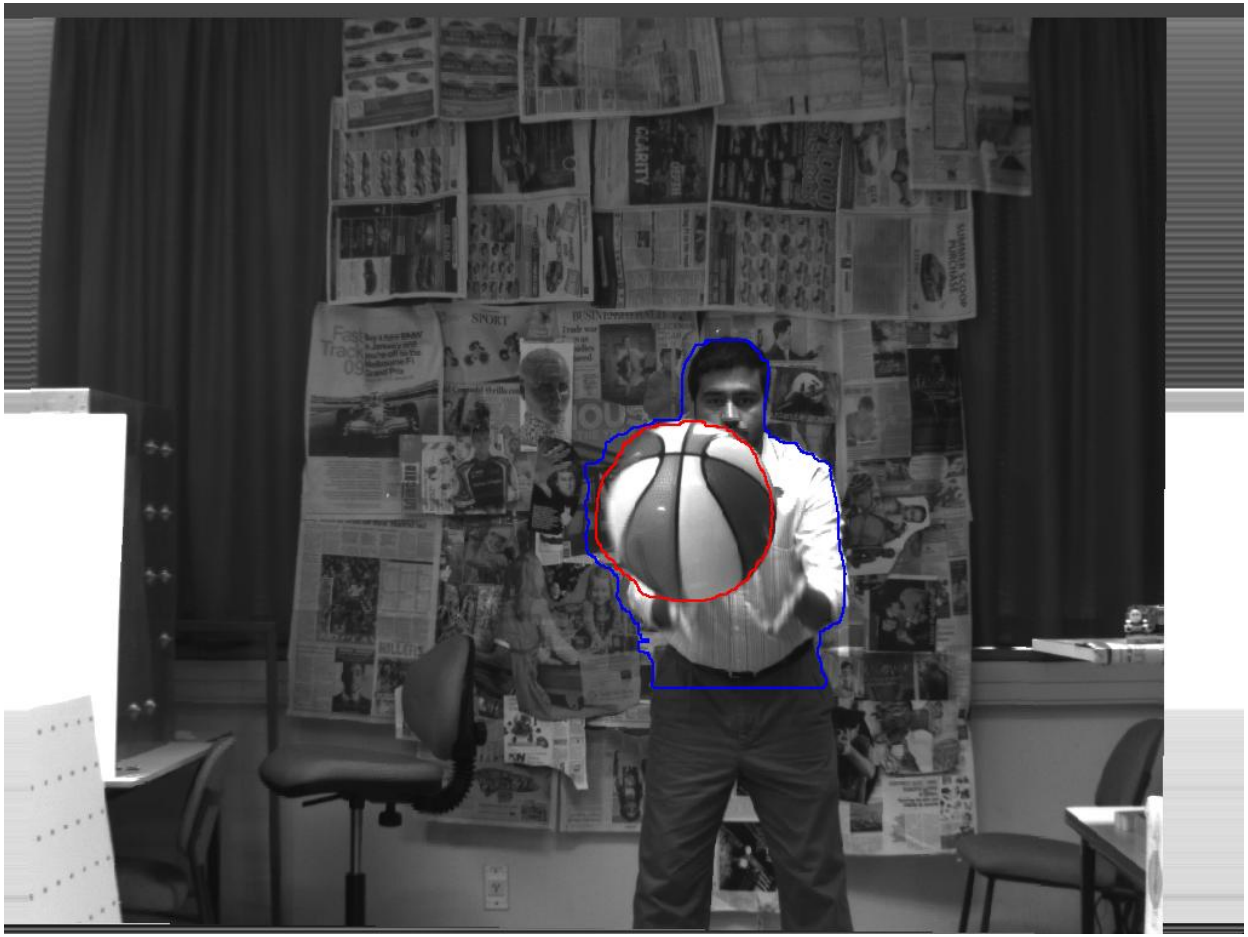


Scene interpretation

Step 1 - Find Contours

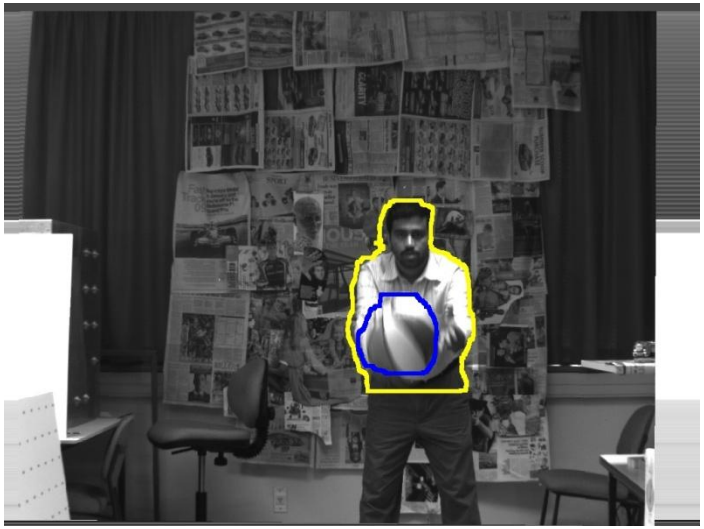
‘Salmon’ algorithm

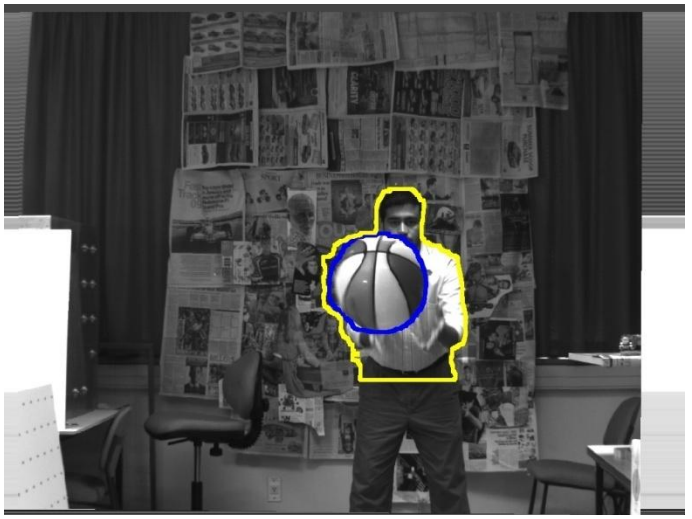
Tracks a contour edge using disparity and contour maps
Runs in real time generating about 20 contours

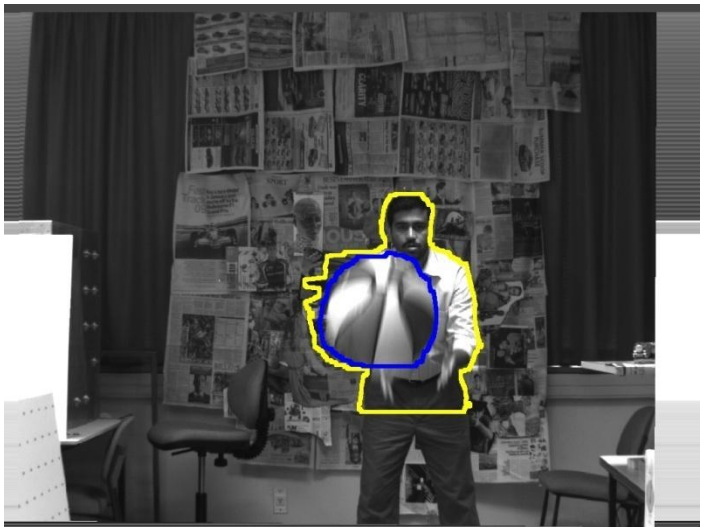


d = 14

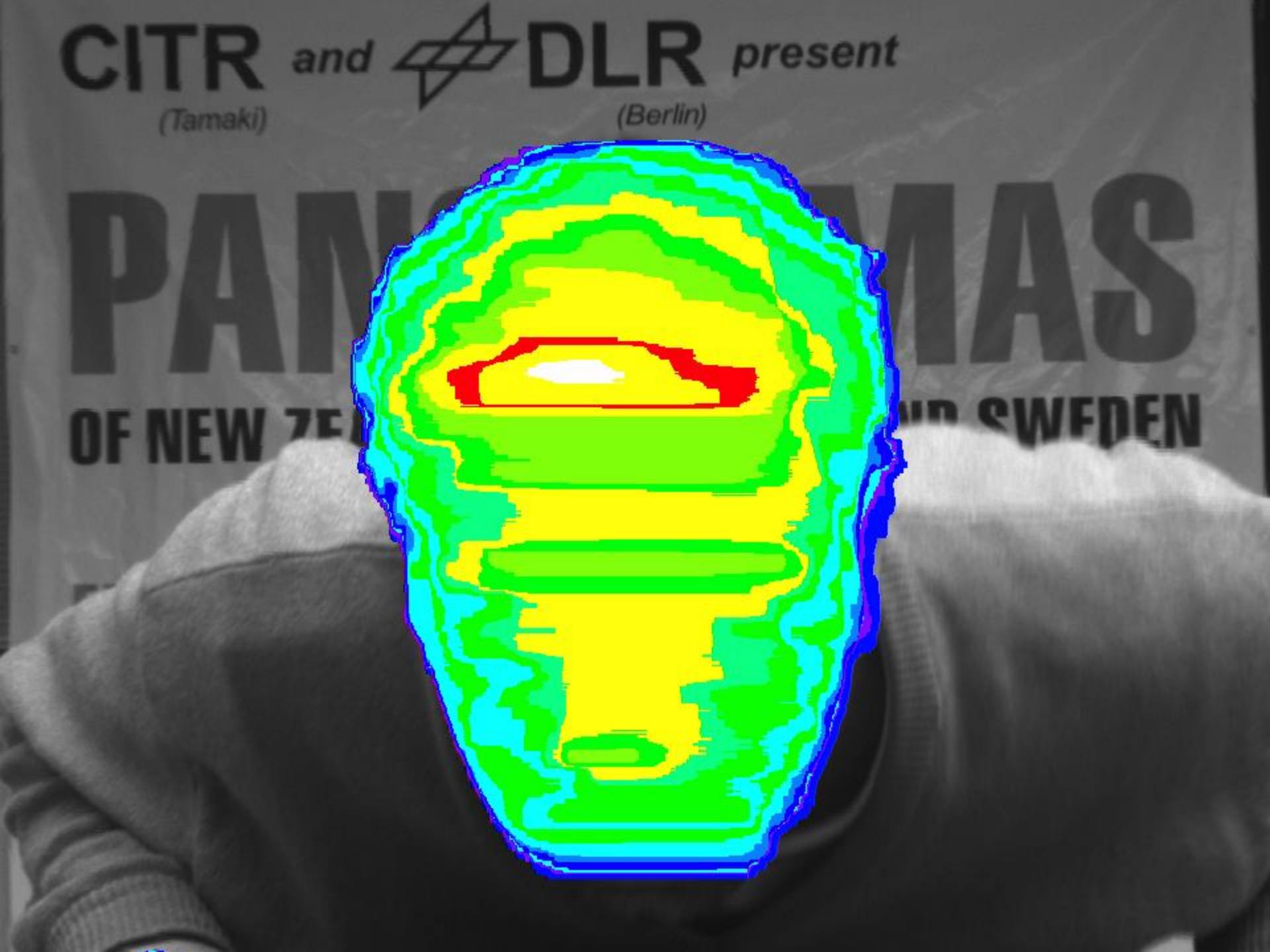
d = 21








CITR and DLR present
(Tamaki)
(Berlin)
OF NEW
SWEDEN


CITR
(Tamaki)
and
DLR
(Berlin)
present
OF NEW
SWEDEN


CITR and DLR present
(Tamaki)
(Berlin)
OF NEW
SWEDEN

Conclusion

- Low latency - few (~ 32) scan lines
  - Good optics and mechanical configuration
- Dynamic programming ⇨compact circuitry
- Depth *and* occlusion map produced
- Host processor can use all its cycles f
  scene interpretation
- Calculates 3×10^9 disparities
- GPU computes about the sa
  - With higher latency and overhe
- Salmon generates contour maps in real time

Best hardware performance known!!

Ongoing work

- **Object detection and recognition is easier**
  - **We have more information (3D vs 2D)!**
- **Object tracking**
  - **Also easier!**
- **Improving matching quality**
  - **SDPS produces many artefacts**
  - **But only uses 50% of medium-sized FPGA**

  ∴ **Space for more advanced circuitry**

If you are interested in any of these areas
You are welcome to visit our laboratory!!

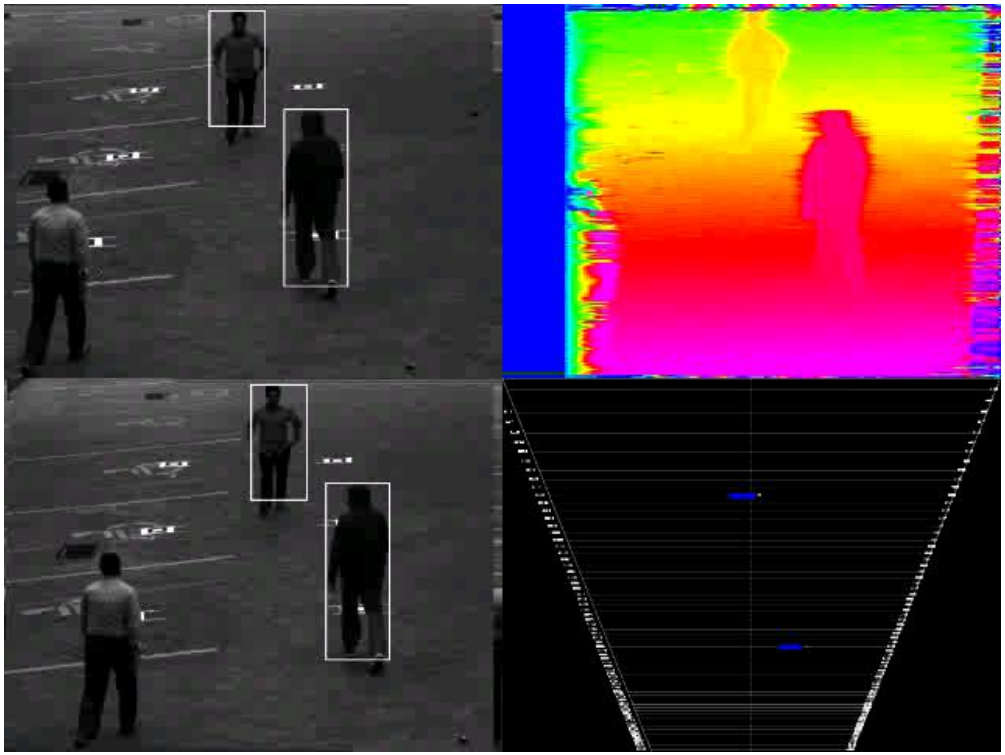
People Tracking

- **Surveillance**
  - **Detection of criminals**
- **Pedestrian traffic**
  - **Building design**
  - **Traffic statistics**
- **Pedestrian detection**
  - **Safer cars**

People Tracking Sequences

- Our lab has a database of people tracking sequences with
  - Left and right camera images
  - Disparity (depth) maps
  - Occlusion maps
- Varying complexity
  - Single people

    ⇩
  - Crowds
- We are interested in collaboration to use the 3D information for improved tracking (or any other application!)



j.morris@auckland.ac.nz


如果冬天 在北京,
有夏天在Auckland!
别忘记ACCV’2010
在New Zealand南
岛!

1M pixel images,
30 frames/sec,
disparity range 128 ➔1% depth accuracy