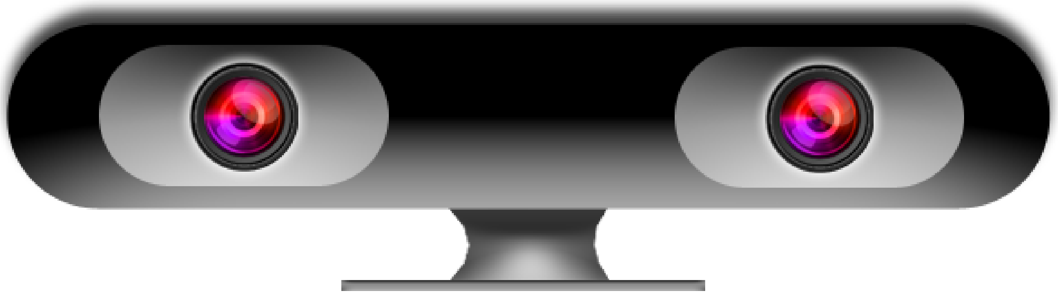
Real-time Dense Passive Stereo Vision:

A Case Study in Optimizing Computer Vision Applications Using OpenCL™ on ARM®



Gian Marco Iodice, ARM Ltd

Media Processing Group, Cambridge (UK)

gianmarco.iodice@arm.com

11th May 2015, Santa Clara, CA USA – Embedded Vision Summit - «Enabling Computer Vision on ARM»

Agenda

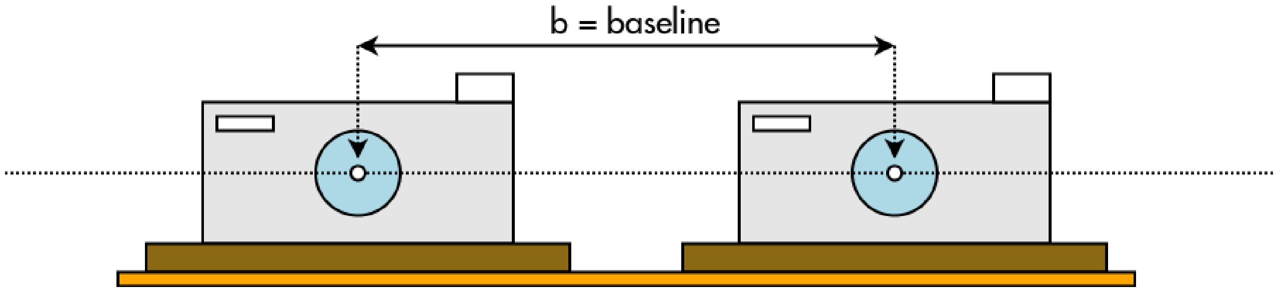
- **Stereo Vision Overview**
- **Implementation**
- **OpenCL Optimizations**
- **Conclusion and future works**

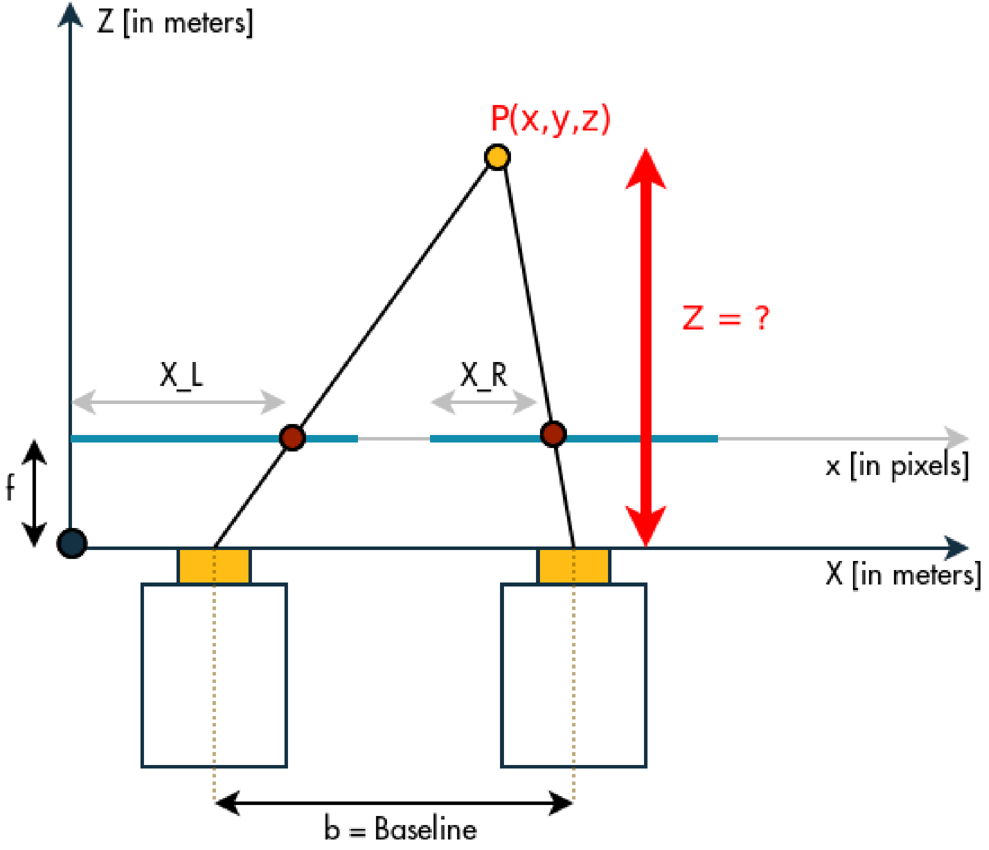
ARM

Stereo Vision Overview

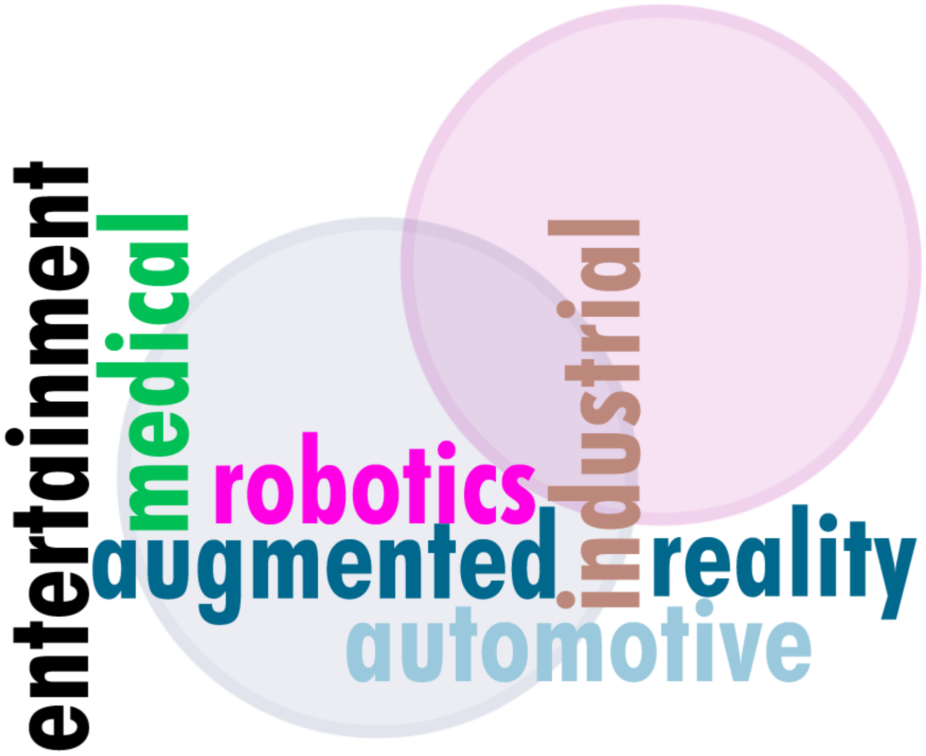
What is Dense Passive Stereo Vision?

- ***Stereo Vision*** is a visual sensing technique aimed at inferring **depth** by comparing two views of the same scene.



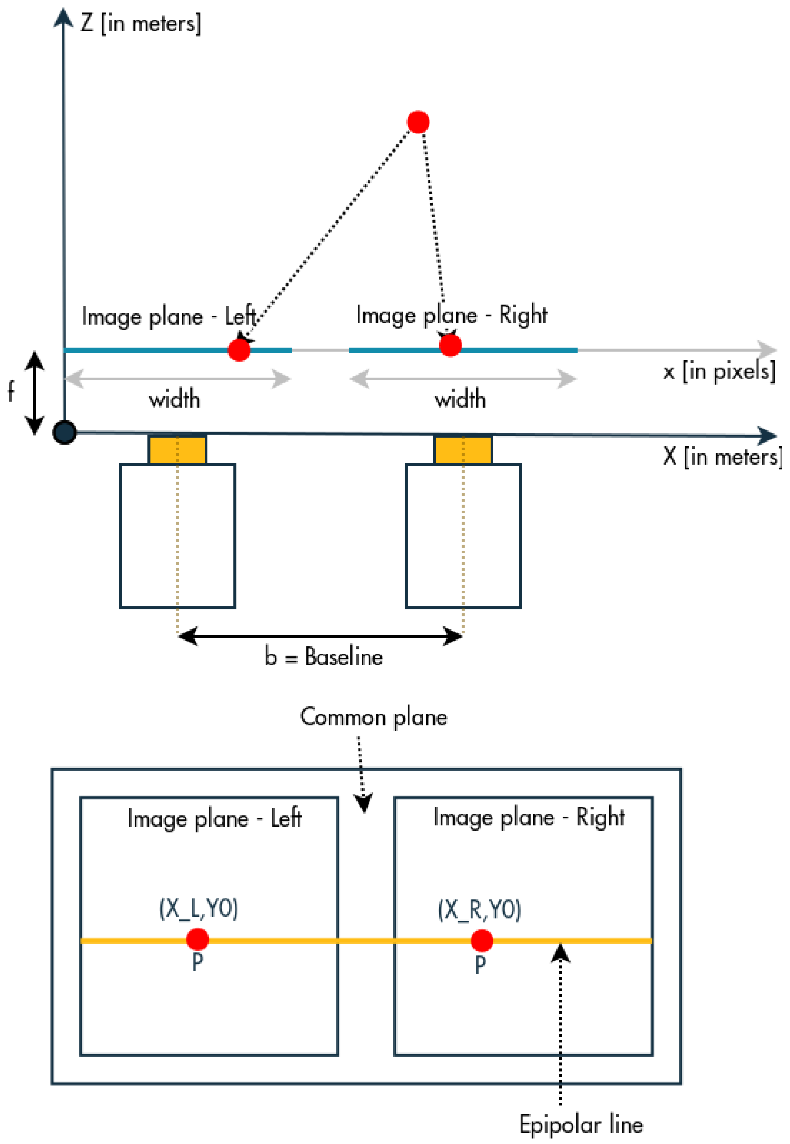


Fields of application



How does it work? (1)

- **Assuming…**
  - **Cameras optically identical**
    - same image sensor
    - same focal length
  - **Cameras horizontally aligned**
  - **Images rectified**
    - no lens distortion
  - **Images captured at the same instant**

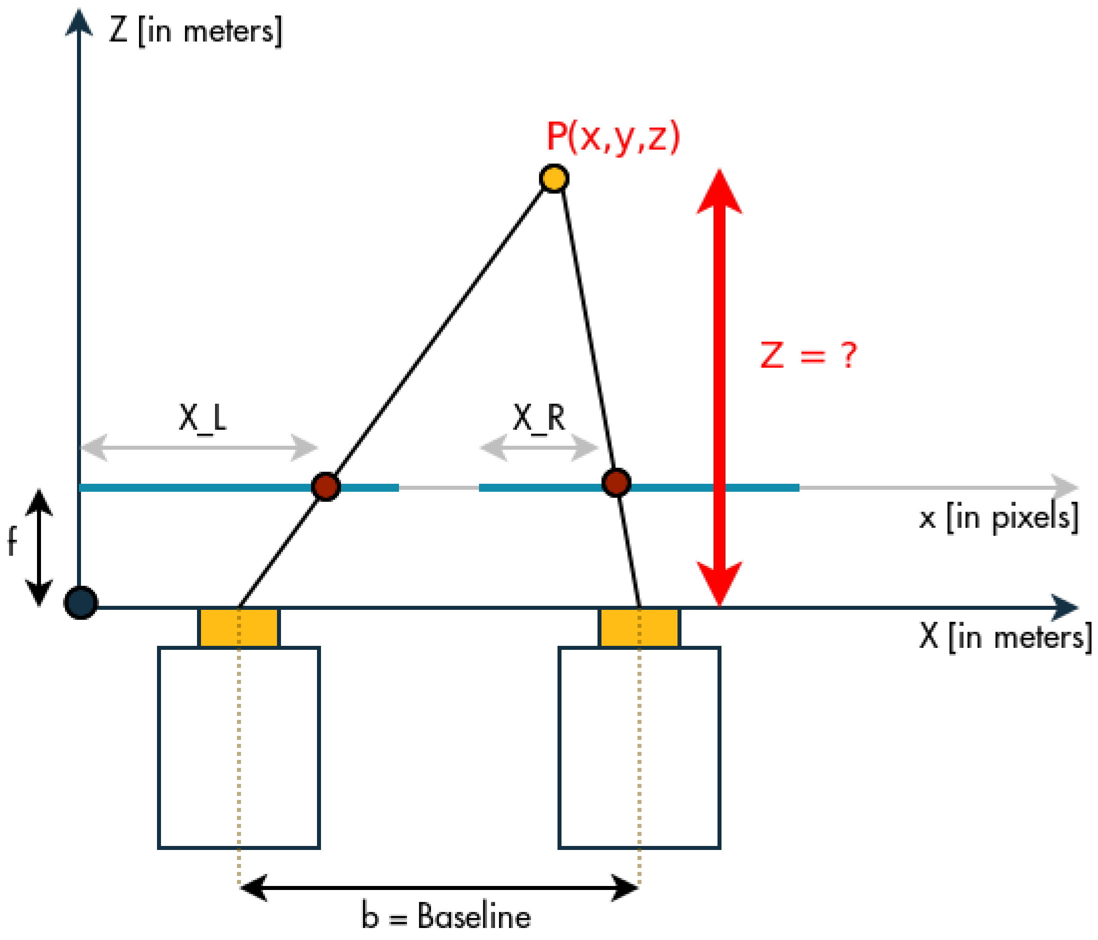


we can talk about…. Horizontal Epipolar Line Constraint

- **Disparity:** it is the difference in x coordinates (d = xL - xR) of the corresponding pixel in the left and right images

How does it work? (2)

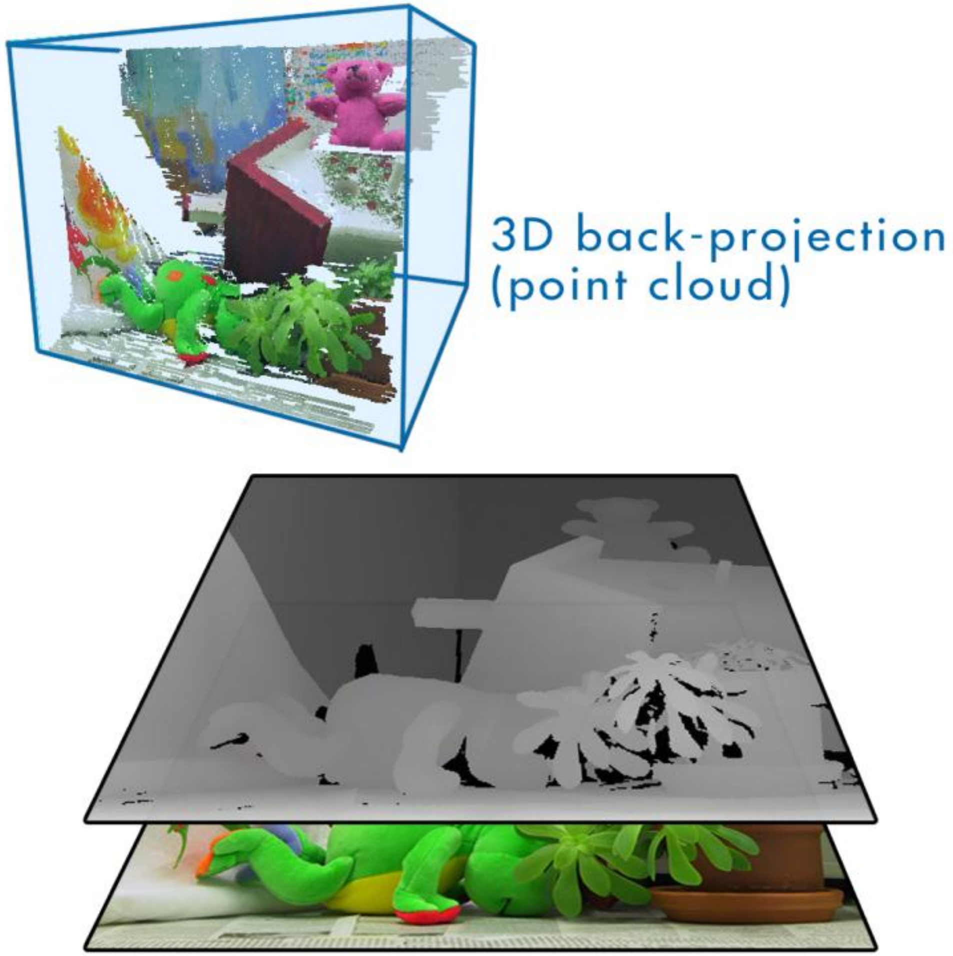
Depth from disparity via triangulation

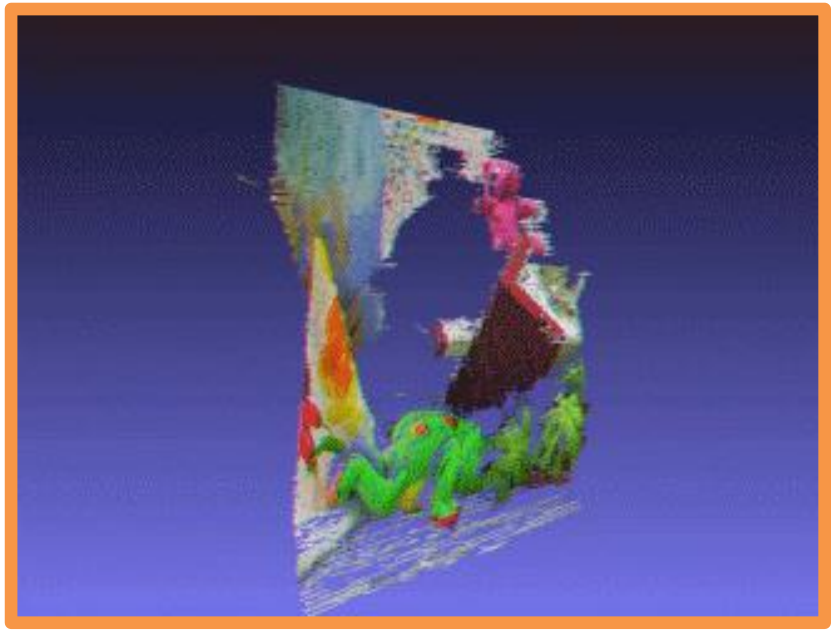


$$Z = \frac{b \cdot f}{d \cdot pxs_{ize}}$$

- **Z:** distance (in meters) between the cameras and point P
- **b:** baseline
- **f:** focal length
- **px_size:** size of the pixel on the image sensor
- **d:** disparity

Disparity Map and Point Cloud

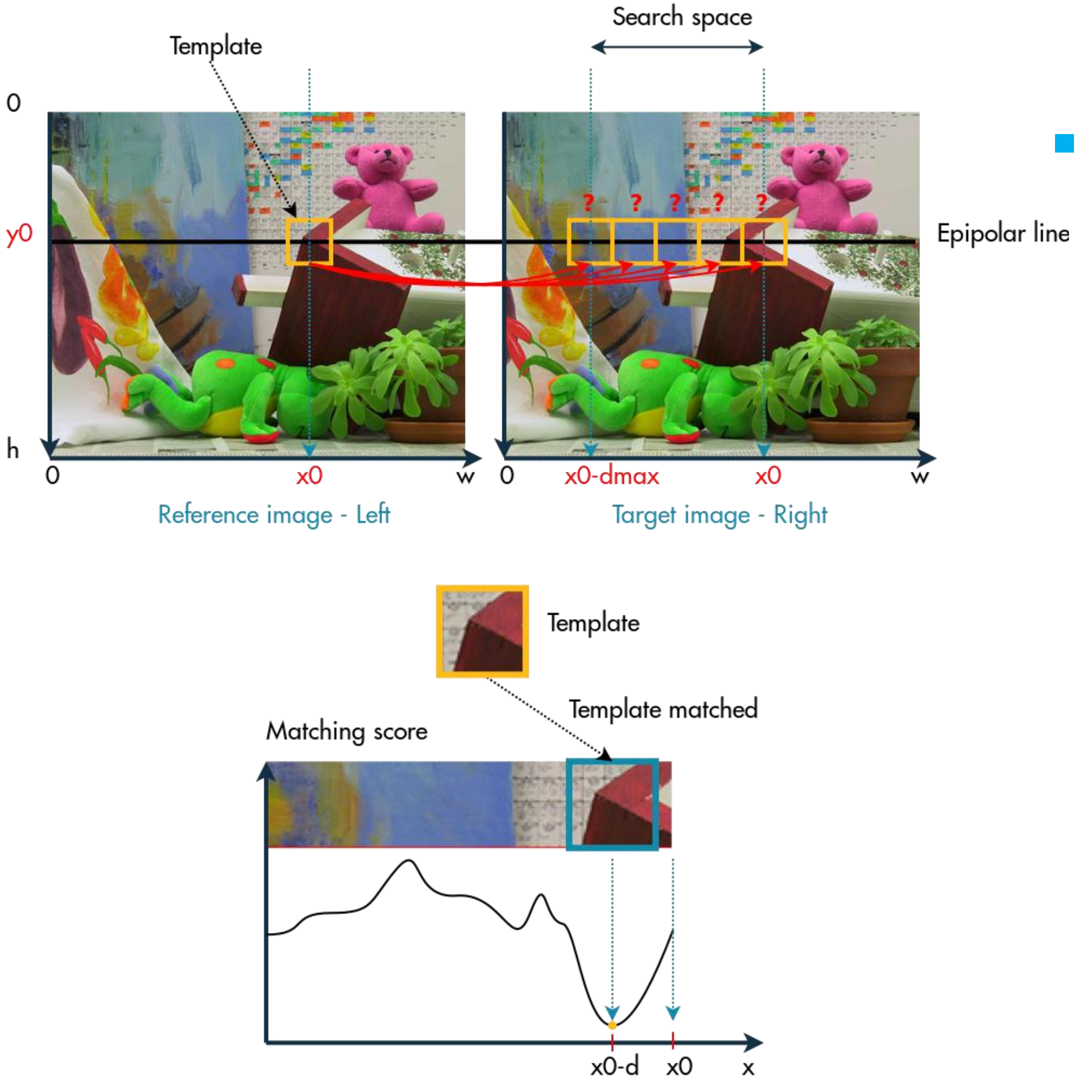




Point Cloud rendered with MeshLab

Correspondence Problem

Template matching



- **For each pixel in the left image:**
  - Extract **NxN** block around it (Reference Template).
  - Compare the reference template with all blocks in the search space of right image using a **similarity measure** (i.e. SAD, SSD, **SHD**,…).
  - The disparity of each pixel is simply selected by the WTA strategy (Winner-Takes-All). Best match wins.

Implementation

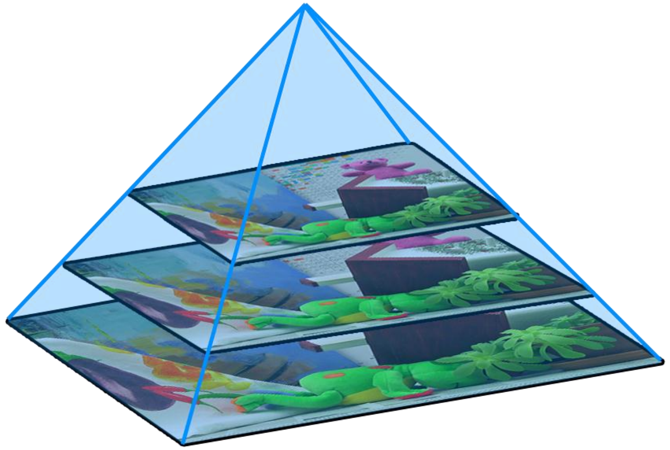
ARM

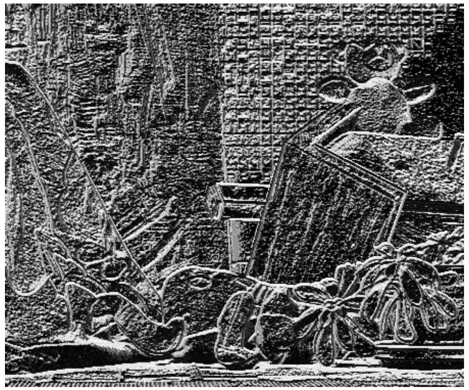
Recipe

- **Grayscale images**

- **Multi-Resolution strategy** (*aka* **coarse-to-fine** strategy)

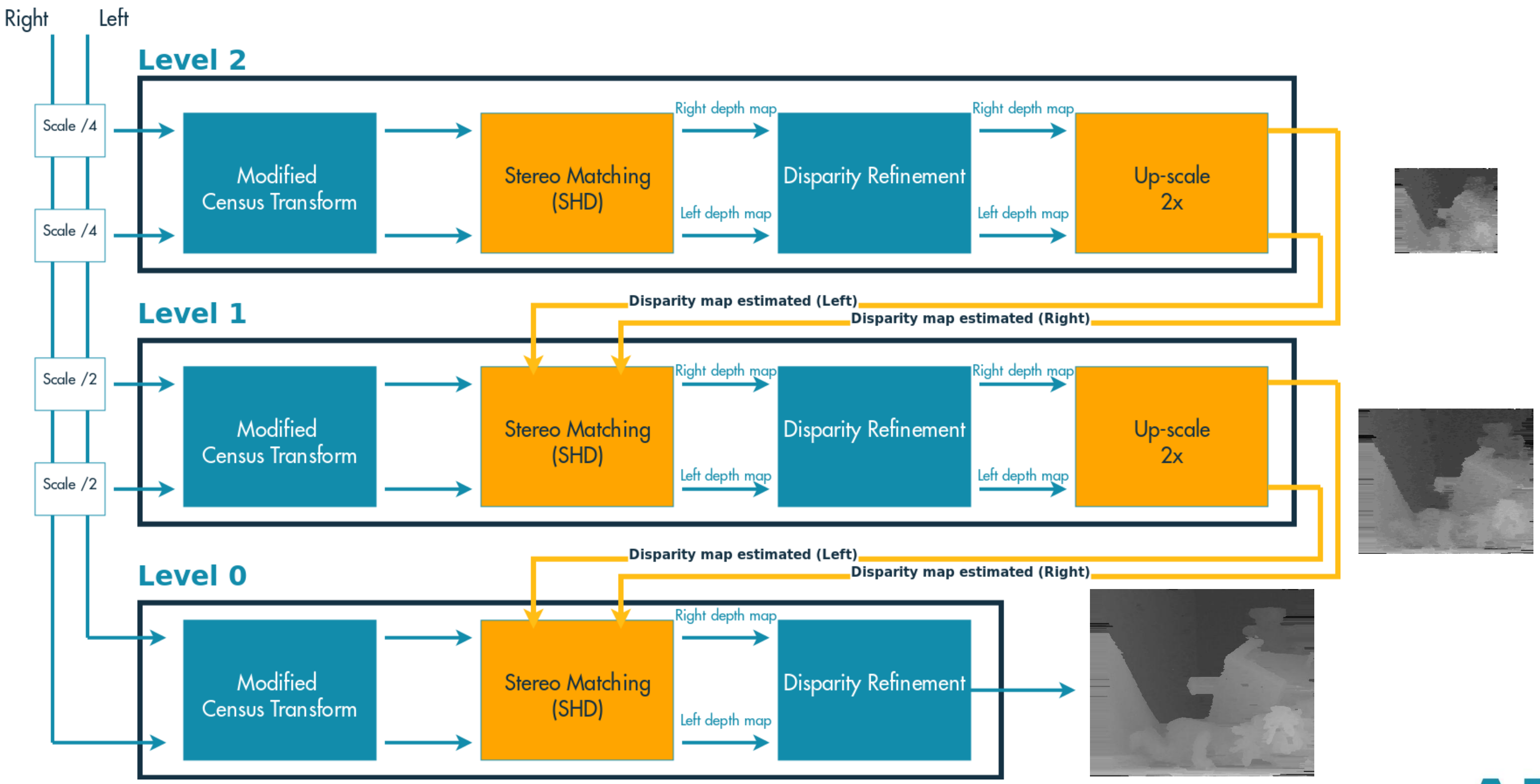
- **Modified Census Transform (MCT) 9x9 and 7x7**
  - 10 bytes per pixel for MCT 9x9
  - 6 bytes per pixel for MCT 7x7





ARM

Pipeline


Right
Left
Level 2
Scale /4
Scale /4
Modified Census Transform
Stereo Matching (SHD)
Right depth map
Left depth map
Disparity Refinement
Right depth map
Left depth map
Up-scale 2x
Disparity map estimated (Left)
Disparity map estimated (Right)
Level 1
Scale /2
Scale /2
Modified Census Transform
Stereo Matching (SHD)
Right depth map
Left depth map
Disparity Refinement
Right depth map
Left depth map
Up-scale 2x
Disparity map estimated (Left)
Disparity map estimated (Right)
Level 0
Modified Census Transform
Stereo Matching (SHD)
Right depth map
Left depth map
Disparity Refinement

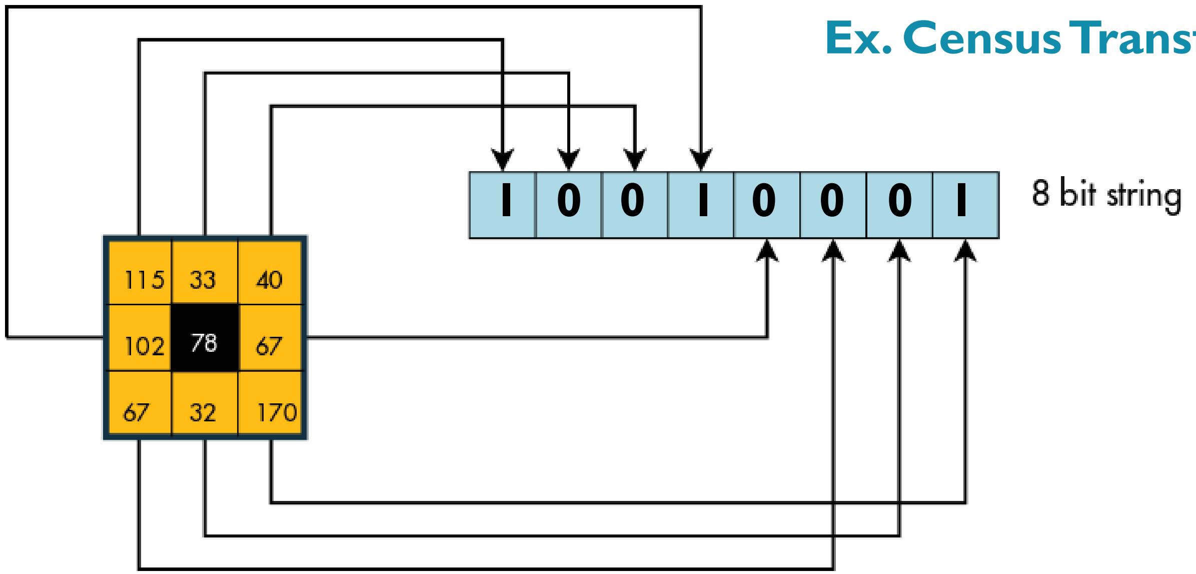
Census Transform – Ramin Zabih Et al., 1994

- It is a non-parametric local image transform which result does not depend on camera gain and light condition.

- It replaces each pixel by a N-bit string which summarizes the local spatial structure.

- For each neighboring pixel (except the center one) it is associated one bit of that N bit string.
  - Each bit is set if **the corresponding neighboring** pixel value **is greater than the center pixel value**

Ex. Census Transform 3x3



115 > 78?	1
33 > 78?	0
40 > 78?	0
102 > 78?	1
67 > 78?	0
67 > 78?	0
32 > 78?	0
170 > 78?	1

Modified Census Transform – Bernhard Froba Et al., 2004

- Extension of the work did by Bernhard Froba Et al. in 2004

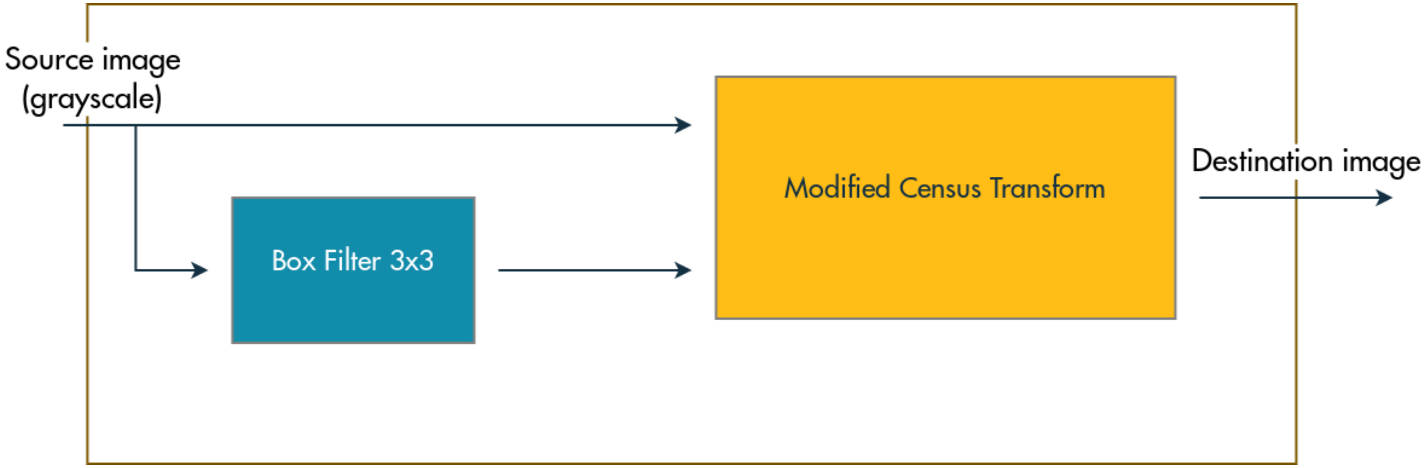
- Instead comparing the neighboring pixels with the center pixel, **it compares the values of the neighboring pixel with the mean intensity value of the local window 3x3** centered on it

Ex. Modified Census Transform 7x7

7 x 7

115	33	40	102	67	170	67
118	61	40	12	47	12	67
118	63	42	14	47	19	82
110	66	40	10	52	17	83
115	68	41	10	54	18	82
111	72	43	8	57	17	81
110	70	40	10	59	21	80

(42 + 14 + 47 + 40 + 10 + 52 + 41 + 10 + 54) / 9 = 34.44



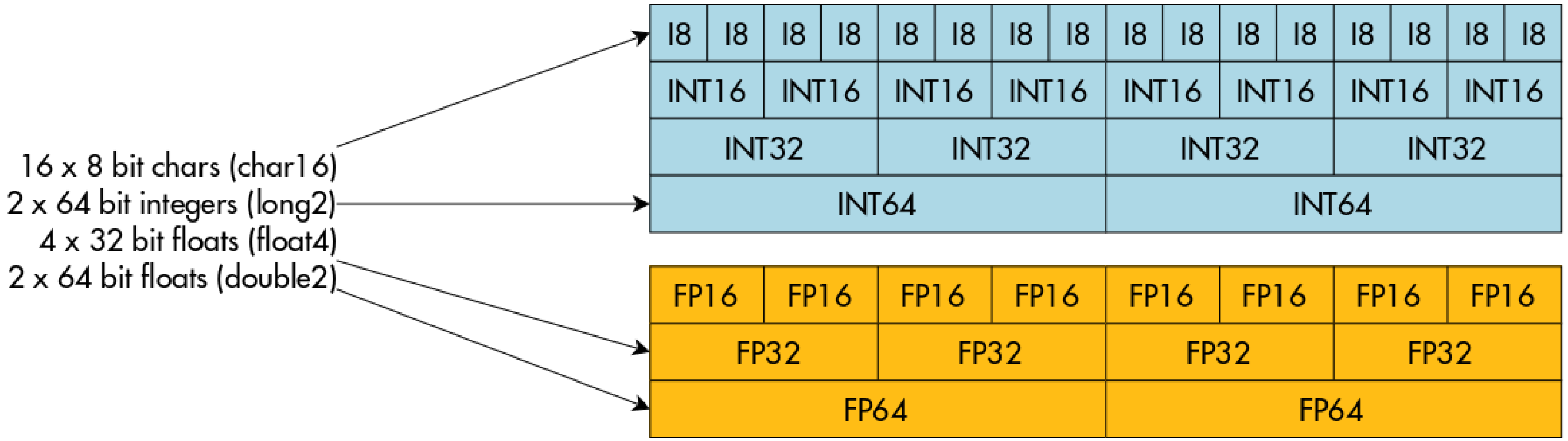
ARM

OpenCL Optimizations

GPU compute on Mali™

- **Full profile OpenCL v1.1 in hardware for Mali-T600 / T700 GPUs**
  - Backward compatibility support for OpenCL v1.0.
  - Image types supported in HW and driver.
  - printf implemented as an extension to v1.1 driver.

- **Mali-T600 and T700 series GPUs have a SIMD instructions**
  - Mali-T600 / T700 can natively support all CL data types.
  - Images data support.
  - Integers and floating point are supported equally quickly.

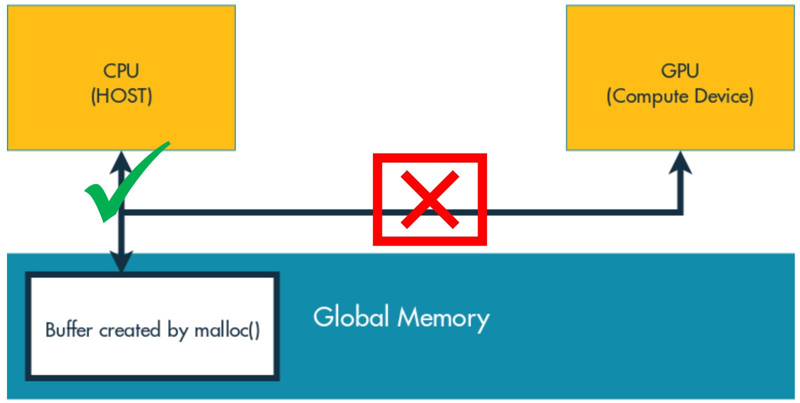


ARM

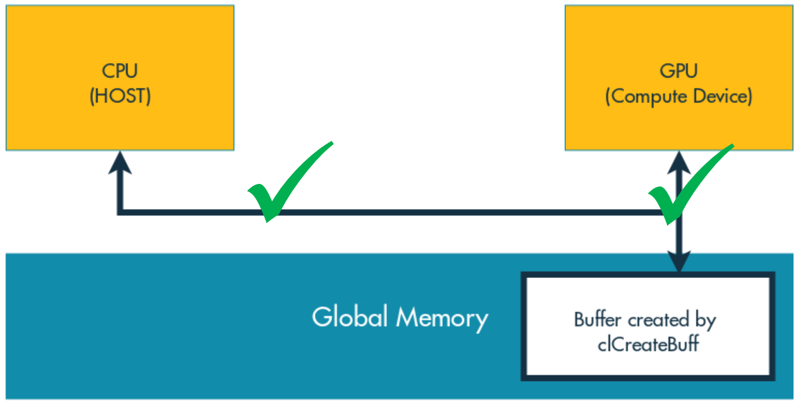
General advices (1)

- **All CL memory buffers are allocated in global memory that is physically accessible by both CPU and GPU cores**
  - However, only memory that is allocated by **clCreateBuffer()** is mapped into both the CPU and GPU virtual memory spaces.
  - Memory allocated using **malloc()**, etc, is only mapped onto the CPU.
  - So calling **clCreateBuffer()** with **CL_MEM_USE_HOST_PTR** and passing in a user created buffer requires the driver to create a new buffer and copy the data (identical to **CL_MEM_COPY_HOST_PTR**).

Buffers created by **malloc()** are not mapped into the GPU memory space



clCreateBuffer(CL_MEM_ALLOC_HOST_PTR) creates buffer visible by both GPU and CPU



ARM

General advices (2)

- **Try to use as much as possible vector instructions**
  - Avoid writing kernels that operate on single bytes or scalar values.
  - It can allow to execute less threads
  - It can allow to reduce the # of load/store instructions

- **Use cl built-in functions (in short *cl BIF*) when possible**
  - **Math cl BIFL:** cos(), sin(), atan(), log, pow,…
  - **Geometric cl BIFL:** dot(), normalize(),…

- **Use correct data types for your specific problem**
  - e.g. uint16?, ushort16?, uchar16?..

Further details and general advices at malideveloper.arm.com where you can find tutorials, videos and developer guides:

- **OpenCL** SDK tutorial
- **RenderScript™** tutorial

Mali DEVELOPER CENTER

ARM

Further optimizations…

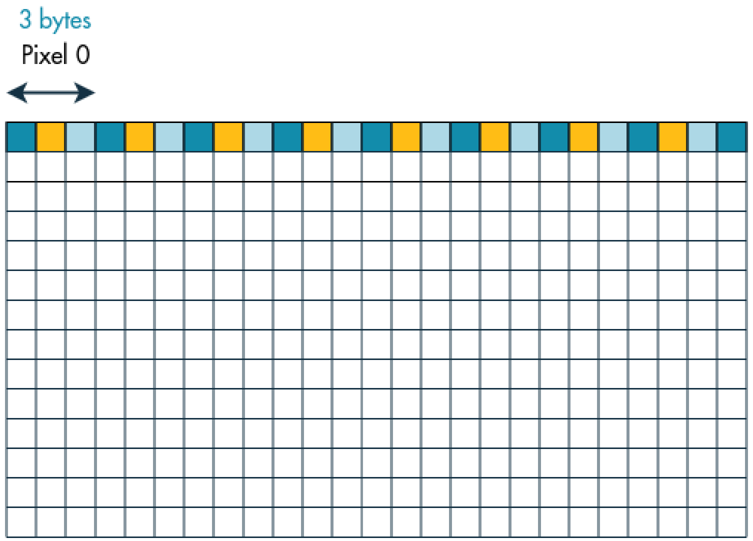
- **Data layout**
  - *Modified Census Transform: case study*
- **No serialization of CPU and GPU workloads**
  - *Stereo vision pipeline*
- **Parallel tasks with a single kernel**
  - *Stereo vision pipeline*
- **Complex arithmetic expressions instead of look-up tables**
  - *Popcount: case study*
- **Avoiding branches with Padding Bytes and cl BIF**
  - *Fill Occluded Nearest Lower Pixel: case study*

Data layout (1)

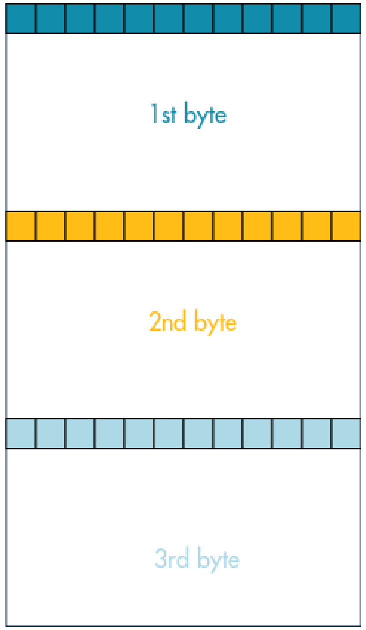
Modified Census Transform: case study

- How we store data <u>has a significant impact</u> on the performance of single kernel and the whole pipeline.
  - Interleaved data generally requires more load/store instructions

- Sometimes it makes other stages easily vectorizable…



Interleaved

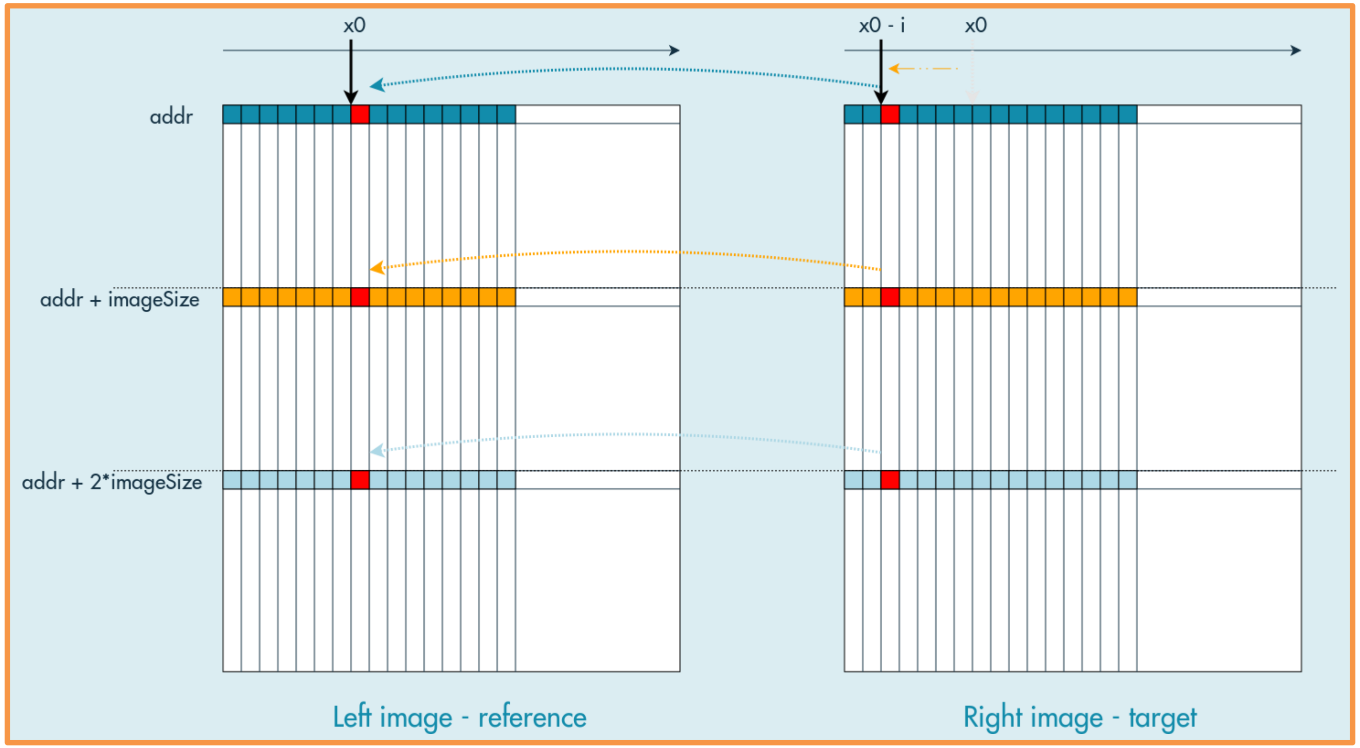


Planar

Data layout (2)

Modified Census Transform: case study

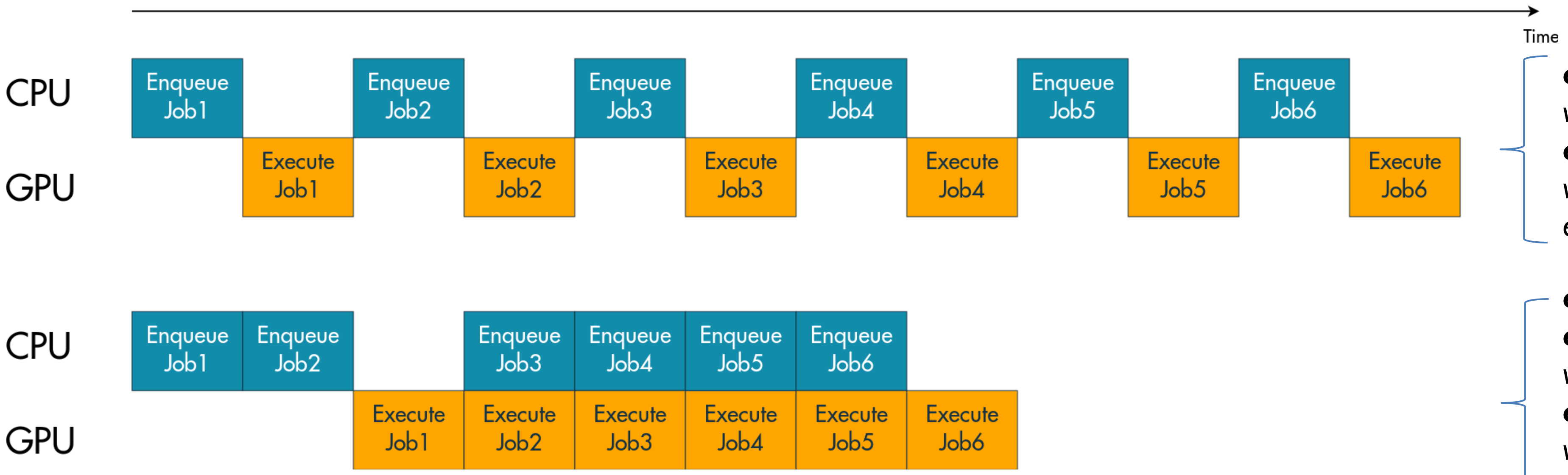
- Using planar data layout:
  - Performance of **Modified Census Transform** are improved by a factor **1.4x** due by:
    - Reduced # of store operations
    - Reduced # of arithmetic instructions for the swizzling
  - It makes the **stereo matching stage** **easily vectorizable**




```
uchar16 ref0 = vload16(addrLeft);
uchar16 ref1 = vload16(addrLeft + offset2ndImg);
uchar16 ref2 = vload16(addrLeft + 2*offset2ndImg);
uchar16 cost;

...

for(i = 0; i < maxDisparity; i++)
        target0 = vload16(addrRight + i);
        target1 = vload16(addrRight + offset2ndImg + i);
        target2 = vload16(addrRight + 2*offset2ndImg + i);

        cost = shd(ref0, target0);
        cost += shd(ref1, target1);
        cost += shd(ref2, target2);

                    ...
endfor
```

No serialization of CPU and GPU workloads (1)

- Avoid the **serialization of CPU and GPU workloads** in order to hide the driver overhead
  - Keep the GPU busy while you're enqueuing the kernels
  - Particular important when there are several CL kernels to enqueue



enqueue Frame 0
wait for Frame 0 to complete…
enqueue Frame 1
wait for Frame 1 to complete…
etc.

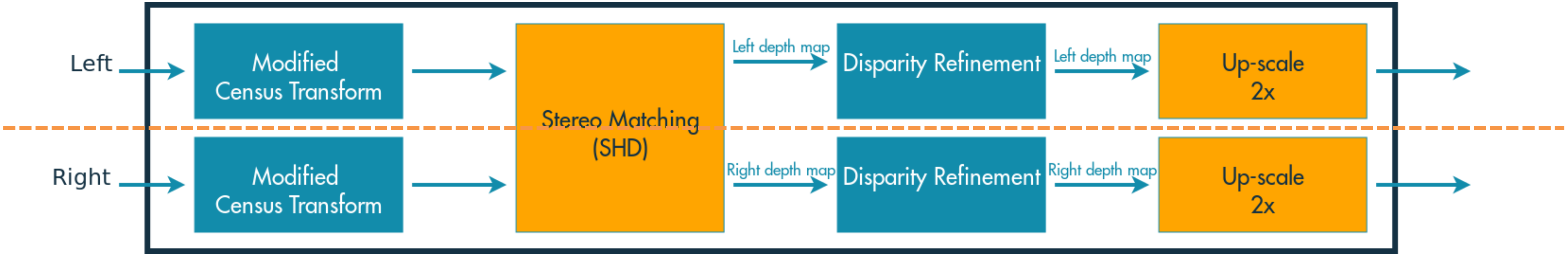
enqueue Frame 0
enqueue Frame 1
wait for Frame 0 to complete…
enqueue Frame 2
wait for Frame 1 to complete…
etc.

ARM

Parallel tasks with a single kernel (1)

- Some kernels could be executed in parallel

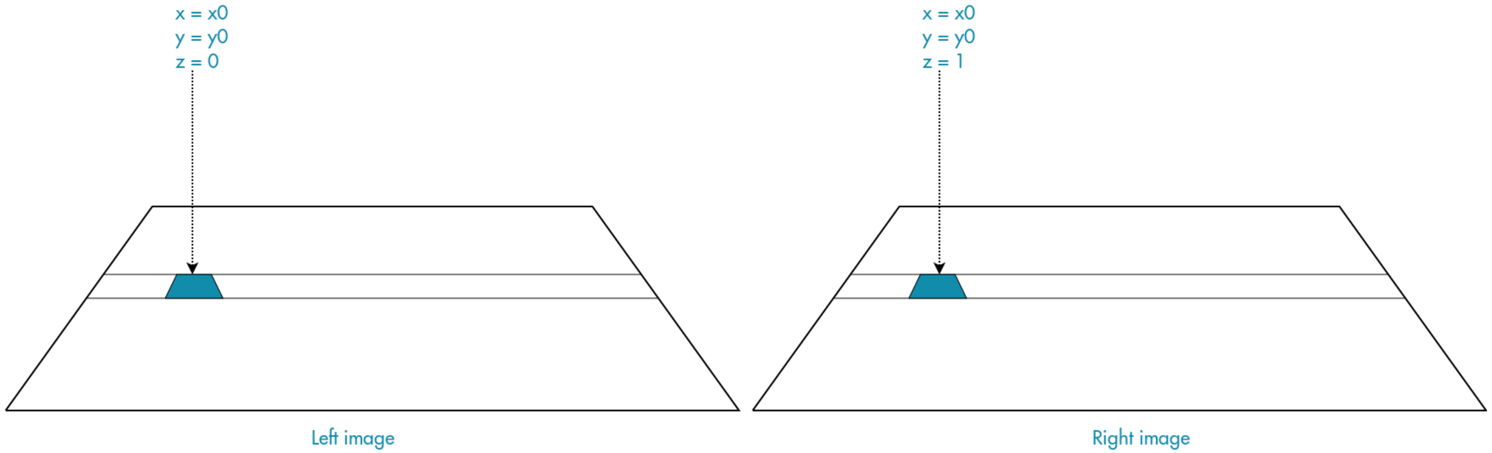
- The algorithm has **2 independent flows**: each one for computing respectively the left and right disparity map.



Parallel tasks with a single kernel (2)

- Assuming that both left and right images have same resolutions and same kernels parameters, we can use a **single kernel** and the **3rd dimension of gws** (*global work-group size*) for accessing the right element

```
const int addr = x + y * stride + z * offset2ndImage;          // Address
```

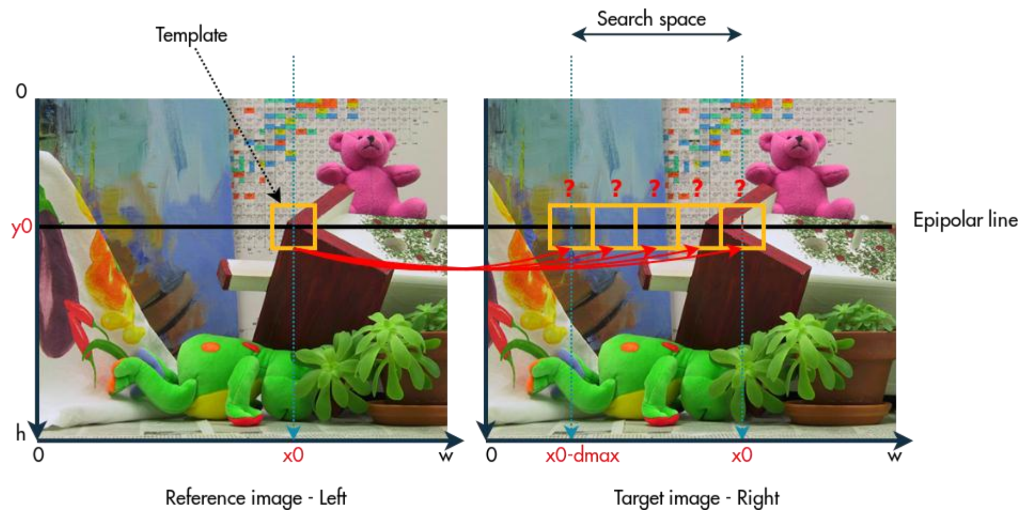


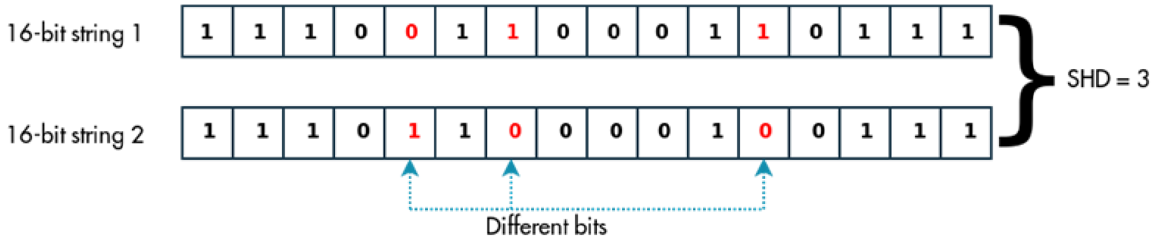
- It allows to reach the **maximum GPU utilization at lower resolution** where otherwise few threads would be dispatched per kernel

Complex expressions instead of look-up tables (I)

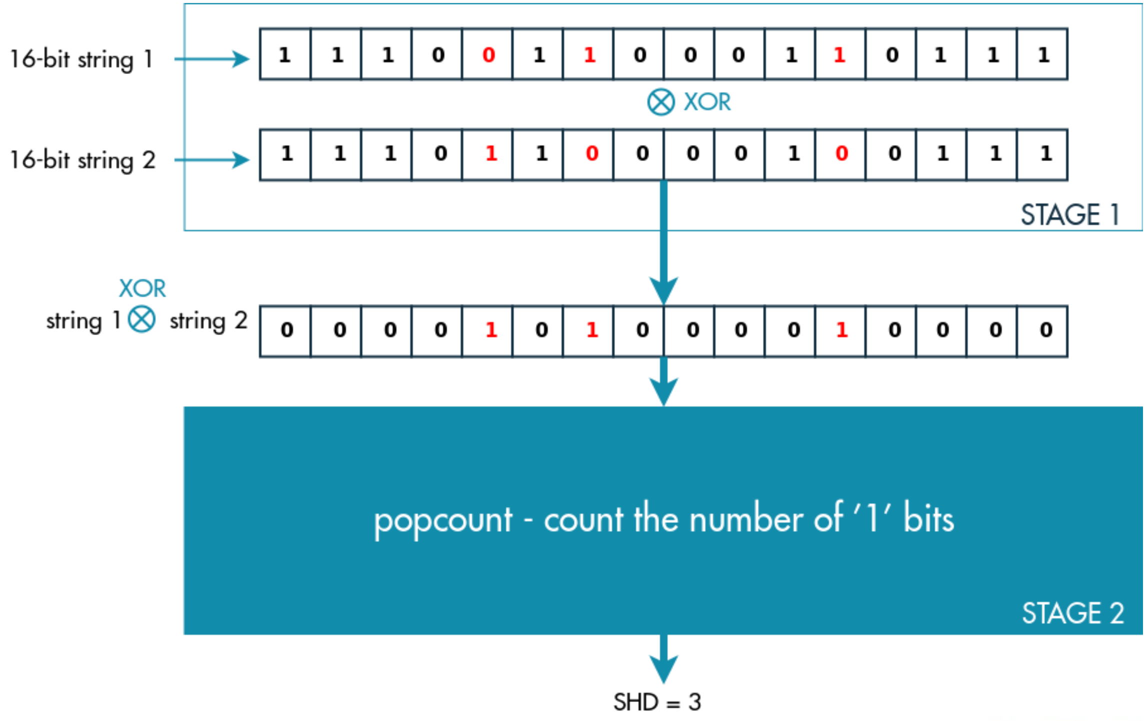
Popcount: case study

- The **similarity measure** used by the *Stereo Matching* stage is the **Sum of Hamming Distance (SHD)**.





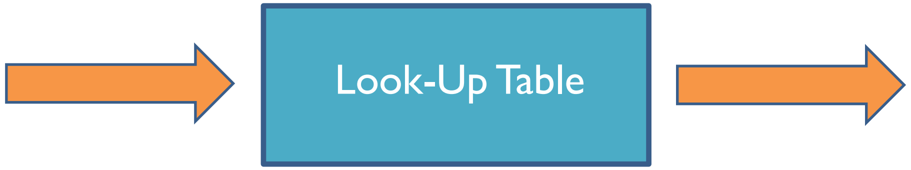
xor + popcount.



Complex expressions instead of look-up tables (2)

Popcount: case study

- **Look-up table**
  - Only scalar memory access
  - Few arithmetic instructions



- **Arithmetic parallel algorithm (*Divide et Impera*)**
  - No memory access
  - The # of arithmetic instructions are much more but…this approach is **~3x** faster than the look-up table using vector operations

Scalar

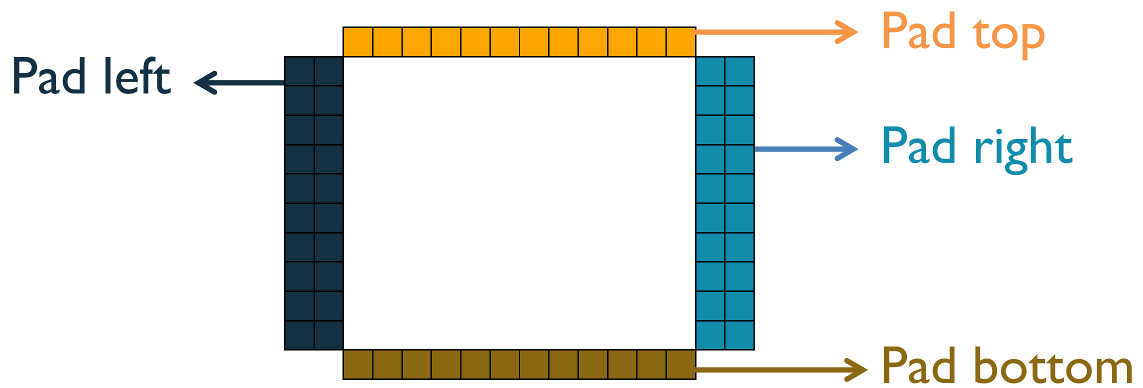
```
input cost;
cost = (cost & (uchar)0x55) + (cost >> 1 & (uchar)0x55);
cost = (cost & (uchar)0x33) + (cost >> 2 & (uchar)0x33);
cost = (cost >> 4  + cost ) & (uchar)0x0f;
return cost;
```

Vector

```
input cost;
cost = (cost & (uchar16)0x55) + (cost >> 1 & (uchar16)0x55);
cost = (cost & (uchar16)0x33) + (cost >> 2 & (uchar16)0x33);
cost = (cost >> 4  + cost ) & (uchar16)0x0f;
return cost
```

Avoiding Branches with Padding Bytes and cl BIF (1)

- *"An algorithm with many conditionals is likely not to be optimal"* so try to **avoid as much as possible loops and if/else conditions**:
  - **Padding bytes:** can be used for avoiding the boundary check.

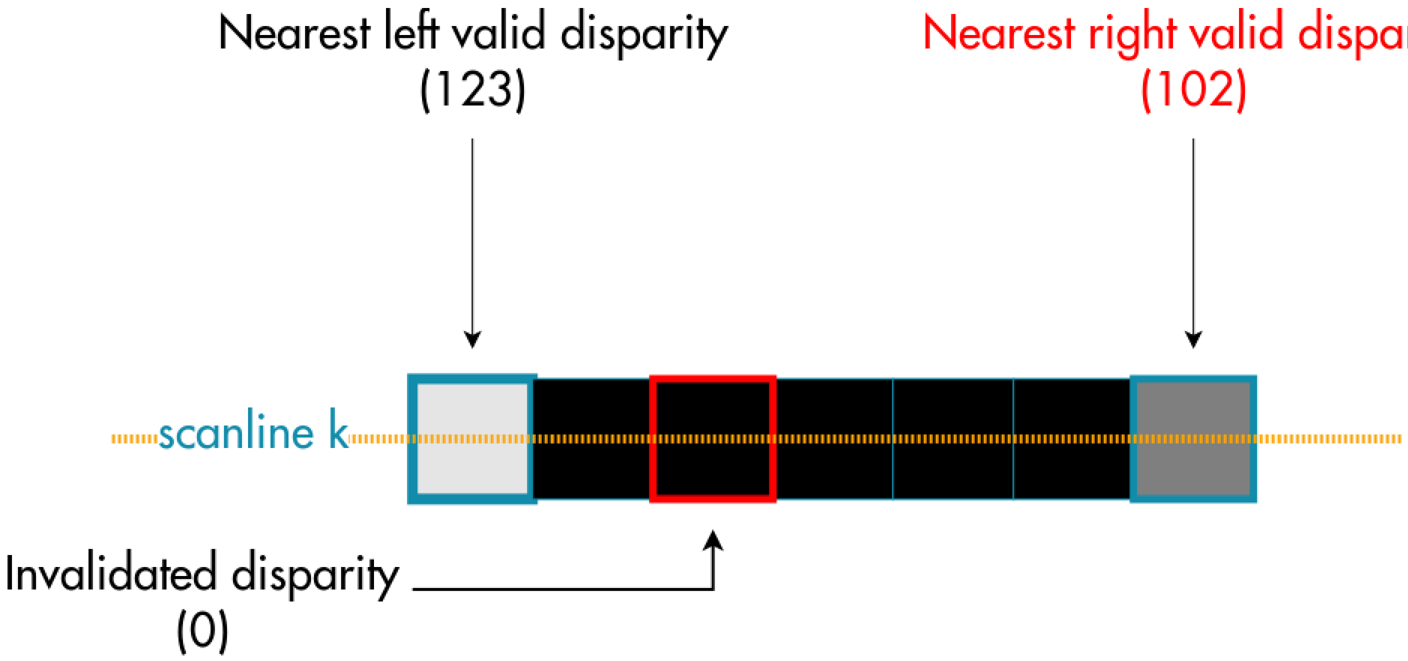


  - **cl BIF:** OpenCL provides **relational built-in functions** that can be used for avoiding branches
    - **select**(a, b, condition): *condition? b : a*
    - **all**(x): *It returns 1 if MSB in all components of x are set*
    - **clamp**(x, a, b): *Clamp x in the interval defined [a, b]*
    - **any**(x): *It returns 1 if any component of x is set*
    - ….

Avoiding Branches with Padding Bytes and cl BIF (2)

Fill occluded pixel nearest lower: case study

- In the *disparity refinement stage*, the invalidated disparity is replaced with the **nearest valid lower disparity** on the same scanline.



Avoiding Branches with Padding Bytes and cl BIF (3)

Fill occluded pixel nearest lower: case study

51x faster

```
while(boundary_condition) {
    if(dispLeft == 0)
        dispLeft = *(ptrDispSrc + k - i);

    if(dispRight == 0)
        dispRight = *(ptrDispSrc + k + i);

    if(dispLeft !=0 && dispRight != 0)
        break;

    i++;
    // Boundary condition
    ....
}
// Select the lower disparity
dstDisp = dispLeft < dispRight? dispLeft : dispRight;
```

```
while(!check && boundary_condition) {
    loadLeft = vload16(dispSrc - i);
    loadRight = vload16(dispSrc + i);

    dispLeft = select(dispLeft, loadLeft, dispLeft == 0);
    dispRight = select(dispRight, loadRight, dispRight == 0);

    check = all( dispLeft!=0 && dispRight != 0);
    i++;

    // Boundary condition
    ....
}
// Select the lower disparity
dstDisp = select(dispRight, dispLeft, dispLeft < dispReft);
```

Conclusion and future works

Results (1)

- The implemented algorithm:
  - is **easy to parameterize**
  - is **configurable** in terms of **disparity range**
  - computes disparity for **occluded pixels**
  - offers **good reliability** throughout a wide variety of scene and illumination conditions.

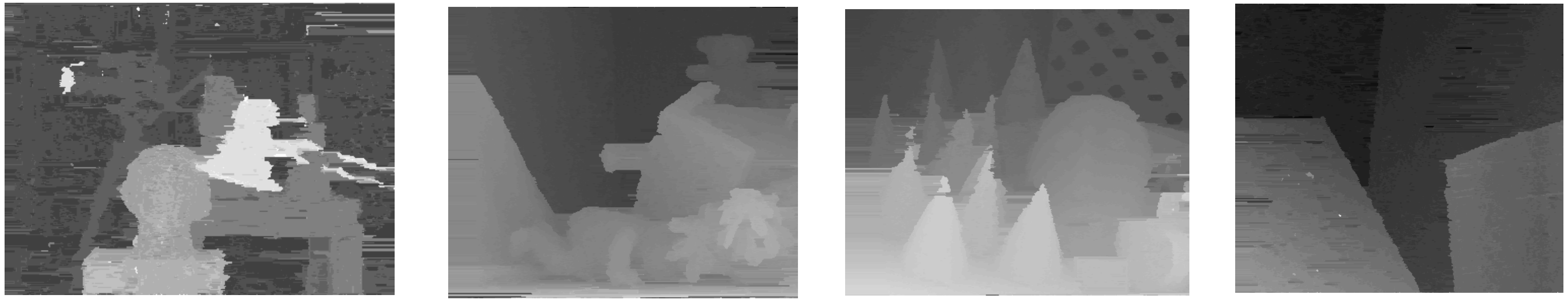
- The system was speed up on development platform featuring an **ARM Mali GPU:**
  - **~120 fps** with **60 disparity levels** at **320x240**
  - **~52 fps** with **60 disparity levels** at **640x480**

- Moreover good performance are obtained as well without using of coarse-to-fine strategy.
  - **~49 fps** with **60 disparity levels** at **320x240**

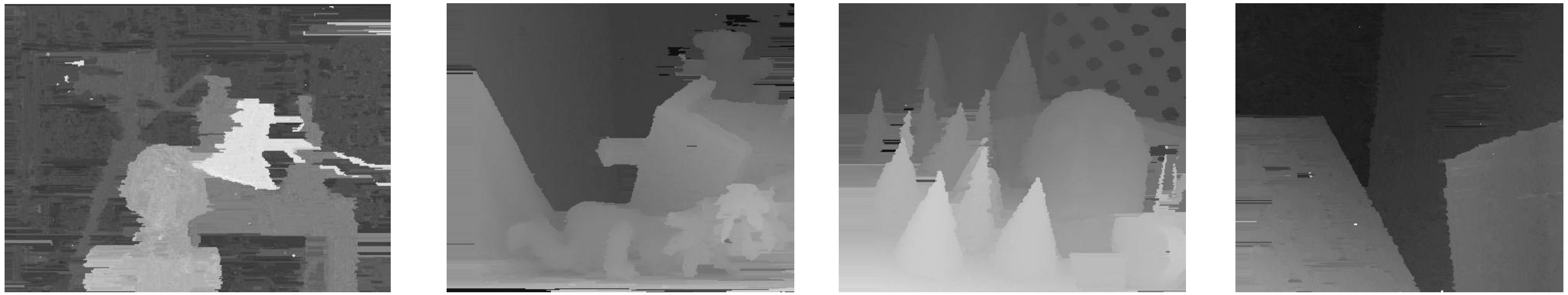
ARM

Results (2)

Dataset from
vision.middlebury.edu/stereo/



Coarse-to-fine



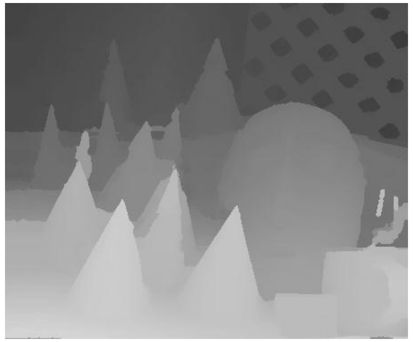
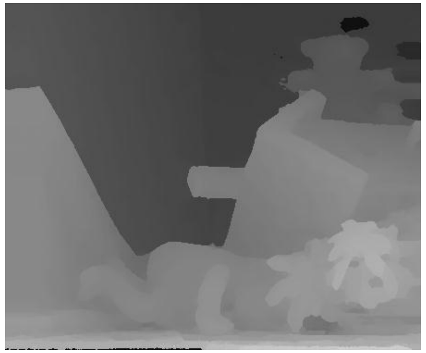
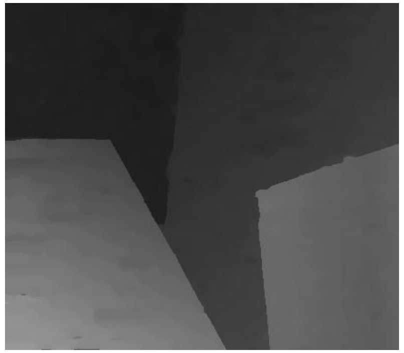
NO Coarse-to-fine

ARM

Future works

- **Use of Sparse Modified Census Transform 7x7**
  - It allows to reduce the # of load/store and arithmetic instructions
  - More erroneous disparity

- **Improve accuracy of Disparity Refinement stage**
  - Median Filter
  - Weighted Median Filter
  - Sub-Pixel estimation

ARM

Final considerations…

- Results reached by GPU compute on ARM Mali are definitely **promising for stereo vision applications** demonstrating the feasibility to achieve real-time performance on Mobile ARM GPU

- **Small changes** in OpenCL code **can lead** to reach **big performance**
  - e.g. data layout, correct data type,…

- Well optimized data layout, types, etc can help reduce the size of kernels (*KISS* approach)
  - It may reduce the number of registers each kernel needs allowing more work items to run on the GPU at the same time (*e.g. stereo matching stage*)

Before finishing…

- This project was developed with a joint cooperation between **ARM Ltd - Media Processing Group, Cambridge – UK** and the Dept. of Information Engineering of the the **University of Pisa - Italy.**

- **Gian Marco Iodice**, ARM Ltd – Media Processing Group, Cambridge (UK)
- **Anthony Barbier**, ARM Ltd – Media Processing Group, Cambridge (UK)
- **Prof. Roberto Saletti**, University of Pisa – Dept. of Information Engineering (Italy)

Question time

ARM

References

- malideveloper.arm.com

- vision.middlebury.edu/stereo/

- D. Scharstein and R. Szeliski. «*A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*», International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002. Microsoft Research Technical Report MSR-TR-2001-81, November 2001.

- E. GUDIS, O. VAN DER WAL, S. KUTHIRUMMAL AND S. CHAI, 2012 «Multi-Resolution Real-Time Dense Stereo Vision Processing in FPGA». In *International Symposium on Field-Programmable Custom Computing Machine.*

- R. ZABIH AND J. WOODFILL, 1994. «*Non-parametric Local Transforms for Computing Visual Correspondence*». In *Proceedings of European Conference on Computer Vision*, vol.2

Thanks