Complex Automotive Applications

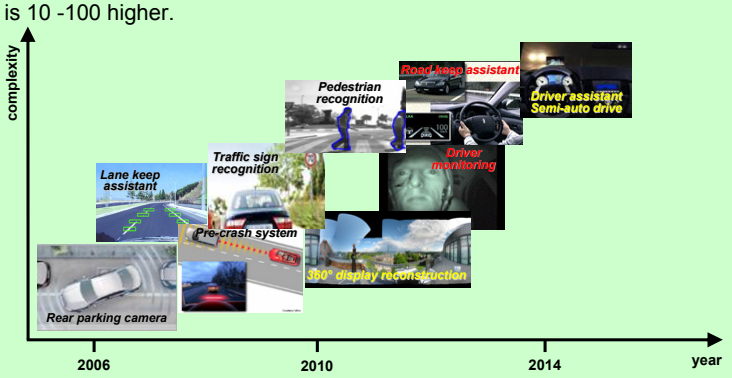
Marius Vasiliu, Michèle Guiffès
AXIS / IEF, Univ. Paris Sud, Orsay 91405, FRANCE

I. Which place for GPU in Embedded Automotive ?

The number of embedded automotive applications grows exponentially, while the sensor price gets lower for same or better performances.

Currently, automotive smart equipment manufactures use FPGA, µC, or DSP dedicated systems for each function (parking detection, night vision, collision detection etc.). Each sub-system needs between 5 to 20 W and the global wattage can exceed 100 W.

Our works demonstrates that complex automotive applications can successfully be implemented on portable laptops or equivalent embedded architectures based on NVidia GPU processors. This "centralized" architecture needs an equivalent electrical power as classical distributed multi-sensor multi-system (60-120W), while the available computing power is 10 -100 higher.

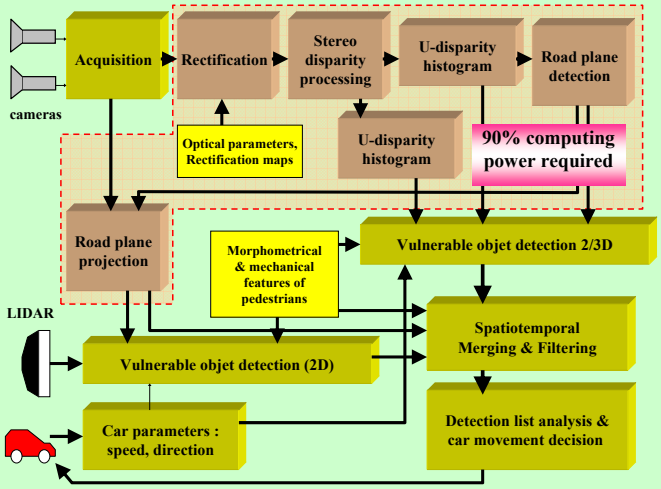


NVidia GPU/CUDA architecture can easily switch between several software dedicated "bricks", according to the car state : parking, city traffic or highway driving.

II. Driver assistance in pedestrian detection / avoidance

Typical implementation example taken from LOVe project :

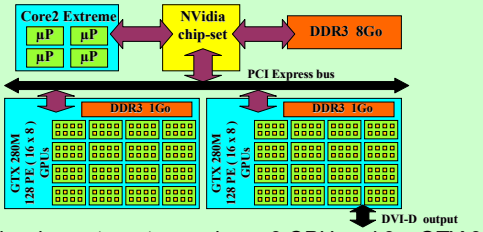
- multi-sensor system intended for pedestrian detection and avoidance, on 40-50 m depth scene in front of the car (Renault prototype)
- LIDAR refresh rate : ≅90ms (10-11 Hz), 4 beam layers
- camera frame rate : ≅33ms (25-30 Hz), 640x480 pixel resolution
- fast (<100ms) and reliable (>99%) results needed



Typical processing flow and bricks slicing : pink on GPU, yellow on CPU

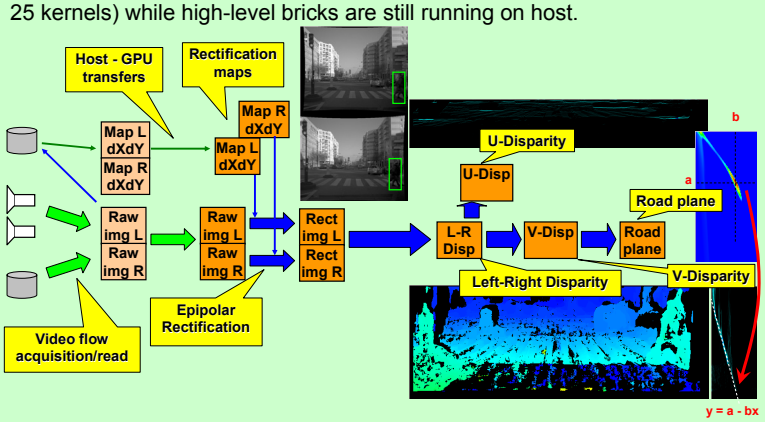
III. Hardware implementation

Several hardware targets were analyzed and tested: "pure" CPU, CPU + TI 6437 DaVinci DSP, CPU + Nec ImapCar, and CPU + NVidia GPU. By far, the best results are provided by the GPU solution. Moreover, the GPU was also easier to program than the other coprocessors.



Best hardware target : quad-core2 CPU and 2 x GTX 280M

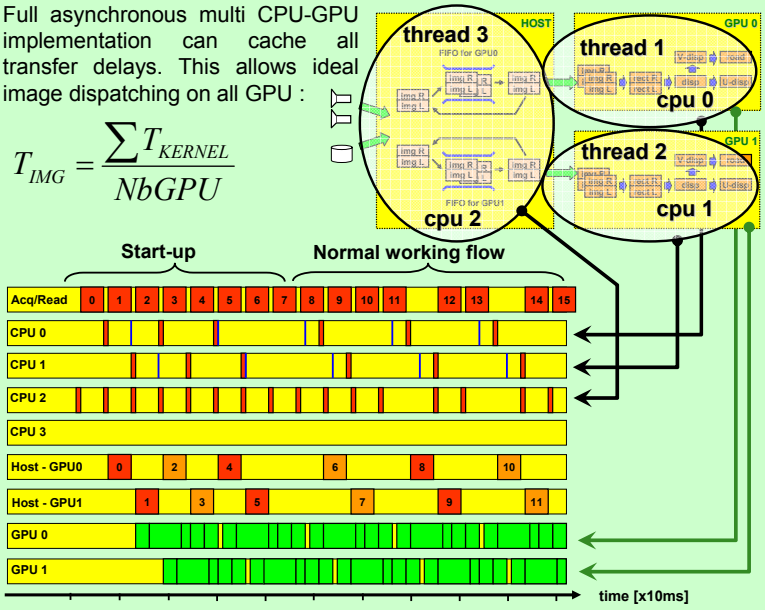
Low-level, high-intensive computing bricks are ported in Cuda/C++ (about 25 kernels) while high-level bricks are still running on host.



Data flow and memory allocation on host - GPU systems (x1 GPU)

Full asynchronous multi CPU-GPU implementation can cache all transfer delays. This allows ideal image dispatching on all GPU :

$$T_{IMG} = \frac{\sum T_{KERNEL}}{NbGPU}$$



Asynchronous activity on each sub-system (bus, 4xCPU, 2xGPU etc.)

IV. Results

CPU	Coprocessor	CPU charge	Max Fps rate
CoreDuo x4 (mono-thread)	none	25%	0.5 - 4.5
CoreDuo x4 (full-threaded)	none	100%	2 - 18
CoreDuo x2	ImapCAR	30%	<10
CoreDuo x4	DSP DaVinci	25%	5 - 14
CoreDuo x2 (synch)	QuadroFX 3600M	8%	35
CoreDuo x4 (synch)	GTX 285	6%	45 - 52
CoreExtreme x4 (synch)	GTX 280M x 2	12%	50 - 56
CoreExtreme x4 (asynch)	GTX 280M x 2	3%	76

V. References

[1] M. Gouiffès, A. Patri, and M. Vasiliu "Robust obstacles detection and tracking using disparity for car driving assistance", *Proc. SPIE* Vol. 7539, 75390H (Jan. 18, 2010)