The embeddability of lane detection algorithms on heterogeneous architectures

Romain Saussard, Boubker Bouzid, Marius Vasiliu, Roger Reynaud

THE EMBEDDABILITY OF LANE DETECTION ALGORITHMS ON HETEROGENEOUS ARCHITECTURES

Romain Saussard [⋆†] *Boubker Bouzid* [⋆] *Marius Vasiliu* [†] *Roger Reynaud* [†]

[⋆] Renault S.A.S.
[†] Institut d'Électronique Fondamentale - Université Paris Sud
{romain.saussard, boubker.bouzid}@renault.com
{marius.vasiliu, roger.reynaud}@u-psud.fr

ABSTRACT

Lane detection plays a crucial role for Advanced Driver Assitance System (ADAS) or autonomous driving applications. Literature shows a lot of lane detection algorithms can work in real time with good results. However, they require much computer processing and cannot be embedded in a vehicle ECU without deep software optimizations. In this paper, we discuss the embeddability of lane detection algorithms by comparing state-of-the-art algorithms in terms of functional performance and computational timing. We identify what essential parts of lane detection are time consuming, and show these parts can be computed in real time on embedded systems.

***Index Terms*—** ADAS, Lane Detection, Embedded Processing

1. INTRODUCTION

On-vehicle lane detection system is an important component for ADAS application. In fact it can be use for warning systems (e.g. alerting the driver in case of dangerous situation: Lane Departure Warning) or active systems (Lane Keeping Assist or Lane Centering Assist).

Lane detection has been deeply studied this past twenty years [1, 2, 3, 4]. However, embedded lane detection is not much studied despite the fact car manufacturers and suppliers are interested in this kind of works. Some automotive providers already offer embedded solutions for lane detection. They produce intelligent cameras which perform embedded real time lane detection, the algorithm is computed by FPGA/ASIC in camera. However, state-of-the-art algorithms show better results.

Semiconductor companies are moving into the market of embedded systems for ADAS with heterogeneous architectures. Heterogeneous architectures embed several processing units with different capabilities on the same System on Chip (SoC), often with massively parallel computing unit. We can cite the Tegra K1 SoC of Nvidia (which embed ARM, GPU and an ISP), the TDA2x SoC [5] of Texas Instrument (ARM, DSP and EVE vectorial processor). This type of SoC offers a good solution for embedding image processing algorithms in the automotive field because it can handle high performance computing with low-power consumption.

In this paper, we discuss the embeddability of lane detection algorithms on heterogeneous architectures. First we will study the different approaches for lane detection, their results and identify their bottlenecks. Then we will discuss how they can be adapted for embedded processing and what performances can be expected with the different approaches.

2. LANE DETECTION ALGORITHMS

The aim of lane detection algorithms is to find positions of lanes on the roads and to characterize them. Based on [6], a lane detection algorithm can be separated into four parts:

- *Image cleaning:* This step enhances the input image in order to get better results with features extraction. It can be an algorithm dealing with abrupt illumination change (e.g. when vehicle goes under a tunnel), color space conversion, ROIs extraction, etc.
- *Features extraction:* Low level features are extracted from the image. Algorithms used for this step can be gradient computation, color segmentation, thresholding, Inverse Perspective Mapping (IPM). The IPM (also called bird-eyes view) consists in applying a homography on the image in order to obtain parallel lines.
- *Lane model fitting:* A geometric model of lane (2D or 3D) is used on extracted features in order to find potential lanes. Hough transform (straight lines or curves) is often used in this step.
- *Temporal integration:* Results of previous frames are used in order to detect lane in current frame. It can be used to reduce the search area in the image and the computational cost. Temporal integration is not always used, and often implies data fusion with other sensors, e.g. IMU, GPS, etc.

In spite of some datasets can be found for pedestrian detection [7], to our knowledge, no free dataset with ground truth can be found for lane detection algorithms. That is why it is hard to compare results of state-of-the-art lane detection algorithms.

However, some papers give results and performance of their algorithm. Thus in [8], where both IPM and Hough transform are used, algorithm has been tested on KITTI database [9] (1392×512 color images) with an average of 68.18 ms processing time per frame. The average error between marked points and lines located with GPS + map is 15 cm.

The algorithm in [10] performs an IPM transformation with integration on Y axis for features extraction and uses RANSAC for lane fitting. It achieves 50 frames per second rate with 640×480 input images. Their provided correct detection rate is 90.89% in urban conditions.

In [11], the lane fitting is based on active contours with initialization performed by Canny & Hough lines detection. It shows good results, 95% of correct lane detection, but poor performance, only 2 frames per seconds with a resolution of 240×256. We can notice it has been tested on old CPU, executing the algorithm on a recent one may show better performance.

Most state-of-the-art algorithms use IPM and/or Hough transformation. These two kernels are essential to have good results for lane detection, but they have a high computational cost. That is why we choose to focus on these two kernels, to implement and optimize them on an embedded SoC.

3. HETEROGENEOUS ARCHITECTURES

Semiconductors companies such as Nvidia, Texas Instrument and Freescale propose heterogeneous architectures to meet automotive industry needs for embedding image processing algorithms for ADAS. These architectures handle high performance computing with massively parallel computing units (e.g. GPU or vectorial processor) and low-power consumption. In this paper, we propose a general approach which can fit with any of these SoCs, and we show some preliminary results obtained on Nvidia Tegra K1 heterogeneous SoC.

3.1. Nvidia Tegra K1 SoC

The Nvidia Tegra K1 SoC is composed of a quad-core ARM Cortex A15 CPU (1.5 GHz clock rate) providing ARMv7 instruction set and an out-of-order speculative issue 3-way superscalar execution pipeline, each core has a NEON and FPU unit [12].

The Kepler GPU of the K1 is composed of one streaming multiprocessors of 192 cores [13], accessible with CUDA [14] and the new standard OpenVX [15]. The parallelization with CUDA is qualified as SIMT (Single Instruction Multiple Threads). The global memory of the K1 is the same for GPU and ARM, both can potentially access to the same data. Memory area of each processing unit is handled by the OS.

3.2. K1 Specific Features

The K1 also has several hardware processing units like Image Signal Processor (ISP) and Image Processing Accelerator providing fast and specific image processing algorithms such as debayering, noise reduction, lens correction etc. These units are very fast but the user can only control a limited set of parameters and the chaining order of kernels is restricted.

In addition of the basic instructions, ARM and Kepler GPU have specific features which can be used to accelerate image processing algorithms. Thus, ARM processor provides SIMD instructions with NEON units [16]. In order to handle the issues of reading and writing accesses of different threads, CUDA provides atomic instructions based on hardware design. This feature is very useful for histogram construction and the voting process of Hough transform.

The Nvidia GPU texture memory provides couple of advantages for reading images data. When accessing to non-integer pixel coordinates (e.g. Im[i-0.5][j-0.5]), texture units handle linear interpolation, this is cost-free because computed by hardware design. Moreover texture units handle access to pixels outside the image (e.g. Im[-1][-1]), also cost-free. This feature can highly decrease the IPM computational timing.

4. KERNEL MAPPING AND EMBEDDABILITY

According to section 2, IPM and Hough transform are two kernels used by state-of-the-art algorithms with best results, but have a high computational cost. Embedding a given algorithm on a heterogeneous architecture is a difficult task because one can not easily find how to allocate kernels on the different processing units (kernel mapping) [17].

4.1. Kernel Mapping Optimization

Let P be the vector of the processing units of a given heterogeneous architecture, K be the vector of kernels of a given algorithm, the matrix M constitutes the mapping of K on P , given by $P = M.K$. Let φ be the dependency matrix (dependencies between kernels K), $\tau(M)$ be the execution function (returning the execution time for each kernel), $\delta(M)$ be the transfer function (returning the transfer delay needed for each kernel), $\eta(M)$ be the occupancy function (returning the occupancy for each computing unit) and $f(P, K, \varphi, \tau(M), \delta(M), \eta(M))$ be the cost function (returning the global execution time of the algorithm). The aim of kernel mapping optimization is to find M minimizing f :

$$\arg\min_{M} \left[f\left(P, K, \varphi, \tau(M), \delta(M), \eta(M)\right)\right]$$

Parameters of function f can be measured or predicted for different M. Measure needs all kernels implementation on all

processing units (very time consuming), but prediction needs kernels and architectures analysis (very efficient but needs deep SW / HW knowledge). Actually, our approach mixed both techniques, starting with some specific representative kernels implementations, and finishing with performance prediction analysis to achieve the best kernel mapping [18].

4.2. Gradient Computation

Gradient computation is often performed with convolution operators (e.g. Sobel filter). First, it can easily be accelerated by using separable filter. Moreover convolution operators are easily parallelizable because the result of one pixel is not dependent on other pixels result. Thus it can be accelerated with GPU or by using NEON instructions and multicore on CPU ARM.

According to our results, an algorithm performing horizontal and vertical Sobel filter is 14 times faster by using 4 cores and $8 \times int16$ NEON vectors than using only one core. Our implementation of this algorithm is 20 times faster on the GPU of K1 than the implementation on ARM with one core. However, using GPU implies extras delays for transferring data between CPU and GPU (about 2ms for a 2MB image).

4.3. Inverse Perspective Mapping (IPM)

Many state-of-the-art lane detection algorithms use IPM, it results an image with vertical and parallel lines which reduces the complexity of lane marking. Camera intrinsic (focal length and optical center) and camera extrinsic (pitch angle, yaw angle and height above the ground) parameters are needed to construct the homography matrix [19]. Pitch angle and yaw angle may not be constant (e.g. vehicle vibrations), that is why camera is often associated with an IMU [8].

A pixel I_{IPM} of the bird-eyes view image with coordinate (u, v) corresponds to the pixel I_{in} with coordinate (x, y) of the input image, so $I_{IPM}(u, v) = I_{in}(x, y)$. Coordinates are linked by: $\begin{pmatrix} s.x & s.y & s \end{pmatrix}^T = H.\begin{pmatrix} u & v & 1 \end{pmatrix}^T$, with H the homography matrix. However, x and y will be non-integer values, so interpolation should be performed in order to compute $I_{in}(x, y)$. Despite the complexity of the IPM is linear, this transformation is associated with some computational cost and is often apply on a ROI.

Linear interpolation can be handled very quickly with texture memory of the GPU which highly reduces the computing time. Indeed only the matrix product has to be computed for each pixel, so it can be parallelized.

Our IPM implementation on GPU provides 5 ms on a 1920×1080 HD image (with a transfer delay 8.5 ms). These results on HD image are here to illustrate reachable performance, real implementation uses a ROI of a smaller image.

Table 1. Best performances for tested algorithms, execution time is given in pixels per second. As Hough transform performance depends on the number of input points, which can vary with a fixed image size, the value is given in points per second.

Algorithm	*Performance*	*Computing Unit*
Horizontal & vertical gradient	450 Mp/s	ARM
IPM transformation	415 Mp/s	GPU
Hough transform	960 kPts/s	GPU
Simple lane detection	200 Mp/s	ARM + GPU

4.4. Hough Transform

The computational complexity of a two-dimensional Hough transform is given by: $O(n.s)$, where n is the number of input points and s the number of iterations needed. Based on [20], Hough transform computation can be accelerated on GPU. In fact reducing the number of input points and parallelizing each iteration decreases the computational timing.

We implemented a simple algorithm which performs a Sobel filter, a thresholding, a morphological filter and apply the Hough transform. According to our results, the best mapping is to embed Hough transform on GPU, and other kernels on ARM. Our algorithm has been tested with a 1920×1080 HD image, about 0.2% of total pixels are used for Hough transform. The ARM processes in 5.4 ms, and the Hough transform on GPU in 4.5 ms. Thus, by adding the memory delay and latency for GPU kernel execution, the full lane detection algorithm is executed in 10.4 ms. Results for tested algorithms are given in table 1.

Our next hardware target for studying ADAS algorithms embeddability is the TDA2x platform [5]. Gradient will probably be computed by the DSP, and IPM and Hough transform by the EVE vectorial processors.

4.5. Conclusion

The embeddability of image processing algorithms is a big issue for automotive industry, it is not a simple task to predict if a robust known algorithm can be embedded on a given SoC. In this paper we discussed the embeddability of lane detection algorithms on a heterogeneous architecture, the K1 SoC. It has been shown that costly operations such as IPM or Hough transform can be optimized and executed in real time on an embedded SoC. However, doing such studies for multiple algorithms and SoCs represents a lot of work. The most effective solution for dealing with embeddability of image processing algorithms is performance prediction. Thus, we are working on performance prediction for algorithms embedded on different SoCs, in order to help car manufacturers and suppliers to choose what algorithm – SoC association presents the best performance and efficiency.

5. REFERENCES

[1] M. Bertozzi and A. Broggi, "Gold: A parallel real-time stereo vision system for generic obstacle and lane detection," *Image Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 62–81, 1998.

[2] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 20–37, 2006.

[3] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 16–26, 2008.

[4] S. Sivaraman and M. M. Trivedi, "Integrated lane and vehicle detection, localization, and tracking: A synergistic approach," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 2, pp. 906–917, 2013.

[5] J. Sankaran and N. Zoran, "TDA2X, a SoC optimized for advanced driver assistance systems," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2204–2208.

[6] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 886–893.

[8] W. Lu, S. A. R. Florez, E. Seignez, and R. Reynaud, "An improved approach for vision-based lane marking detection and tracking," in *2013 International Conference on Electrical, Control and Automation Engineering*. DEStech Publications, Inc., 2014, pp. 382–386.

[9] G. Andreas, L. Philip, S. Christoph, and U. Raquel, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[10] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 7–12.

[11] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image and Vision computing*, vol. 22, no. 4, pp. 269–280, 2004.

[12] T. Lanier, "Exploring the design of the cortex-A15 processor," *URL: http://www.arm.com/files/pdf/at-exploring_the_design_of_the_cortex-a15.pdf*, 2011.

[13] NVIDIA, "Nvidia kepler GK110 architecture whitepaper," 2012.

[14] NVIDIA, "Cuda C programming guide," 2014.

[15] E. Rainey, J. Villarreal, G. Dedeoglu, K. Pulli, T. Lepley, and F. Brill, "Addressing system-level optimization with OpenVX graphs," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 658–663.

[16] G. Mitra, B. Johnston, A. P. Rendell, E. McCreath, and J. Zhou, "Use of SIMD vector operations to accelerate application code performance on low-powered ARM and Intel platforms," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 1107–1116.

[17] H. Zhou and C. Liu, "Task mapping in heterogeneous embedded systems for fast completion time," in *Embedded Software (EMSOFT), 2014 International Conference on*. IEEE, 2014, pp. 1–10.

[18] R. Saussard, B. Bouzid, R. Reynaud, and M. Vasiliu, "Predicting ADAS algorithms performances on K1 architecture," in *URL: http://on-demand-gtc.gputechconf.com*. NVIDIA GTC, 2015.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2003.

[20] G. van den Braak, C. Nugteren, B. Mesman, and H. Corporaal, "Fast hough transform on GPUs: Exploration of algorithm trade-offs," in *Advances Concepts for Intelligent Vision Systems*, pp. 611–622. Springer, 2011.