CATEGORY: **VIDEO & IMAGE PROCESSING - VI04**

POSTER **P5162** | CONTACT NAME **Antonio Sanz Montemayor: antonio.sanz@urjc.es**

GPU TECHNOLOGY CONFERENCE

Tegra K1 Imaging Performance Study: Local Binary Patterns (LBP)

David Concha, Jesús Sánchez-Oro, Raúl Cabido, Juan J. Pantrigo, Antonio S. Montemayor
{david.concha, jesus.sanchezoro, raul.cabido, juanjose.pantrigo, antonio.sanz}@urjc.es
http://www.gavab.etsii.urjc.es/capo/

Universidad Rey Juan Carlos
CAPO Computación de Altas Prestaciones y Optimización
GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

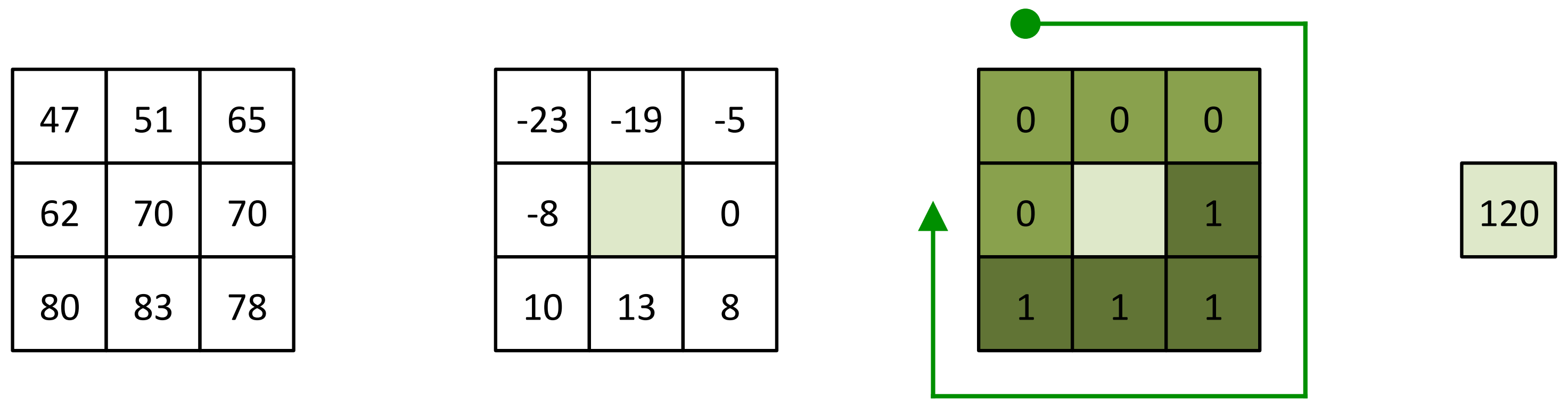
Introduction

In this work we test the computational performance of the recent Tegra K1 (TK1) mobile platform, its ARM processor as well as its CUDA-capable GPU of 192 CUDA cores compared to a high-performance workstation (Intel Xeon E3 1270v3 3.5 GHz with NVIDIA GTX 780 GPU of 2304 CUDA cores).

For this purpose, we start with a Local Binary Patterns algorithm. This algorithm is very popular as well as very interesting in terms of general applicability in the image processing and computer vision fields. In addition, one of the most important TK1 target is the automotive imaging field (pedestrian detection, sign recognition, motion extraction, etc.).

Local Binary Patterns

Local Binary Patterns (LBP) is a method for computing texture information from an image. It was presented by Ojala et al. (1994) but many extensions and applications have been devised. The strength of the method relies on its simplicity as well as great results in such a variety of different applications (face detection, motion detection, feature descriptor, background extraction, LBP +HOG combination for people detection, etc.).



$$01111000 = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 120$$

Given an intensity image as input, the LBP method outputs a texture-data image (LBP image). In its basic form the LBP algorithm labels a considered pixel of an image performing a concatenation of 8 binary values around the center pixel (3x3 neighborhood). These binary values are extracted thresholding each of the eight surrounding pixels using the intensity value of the center as a threshold. This operation is performed over the entire image in a sliding window approach.

$$LBP(img,c) = \sum_{p=7}^{0} s(img[c] - img[p]) \cdot 2^p \qquad s(x) = \begin{Bmatrix} 1 & if\ x \geq 0 \\ 0 & otherwise \end{Bmatrix}$$

Original

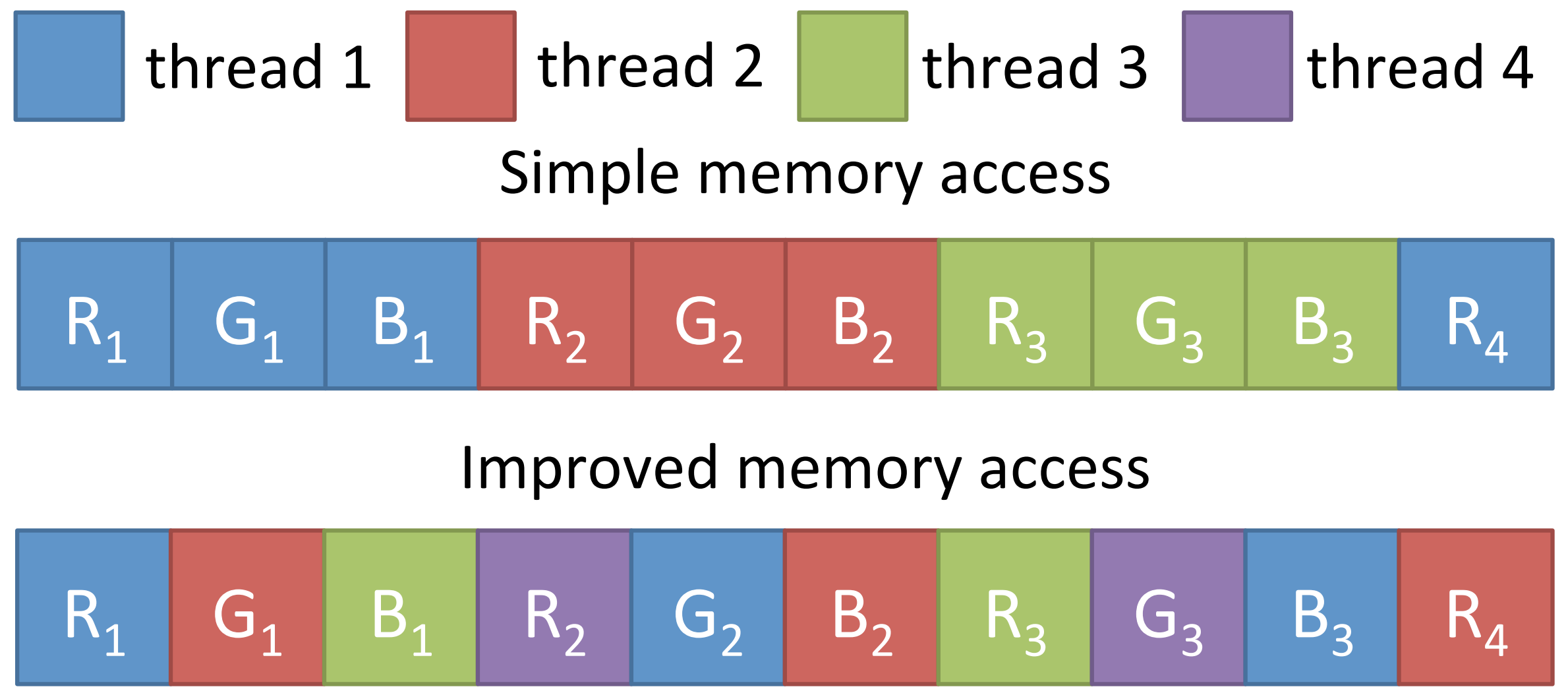
LBP

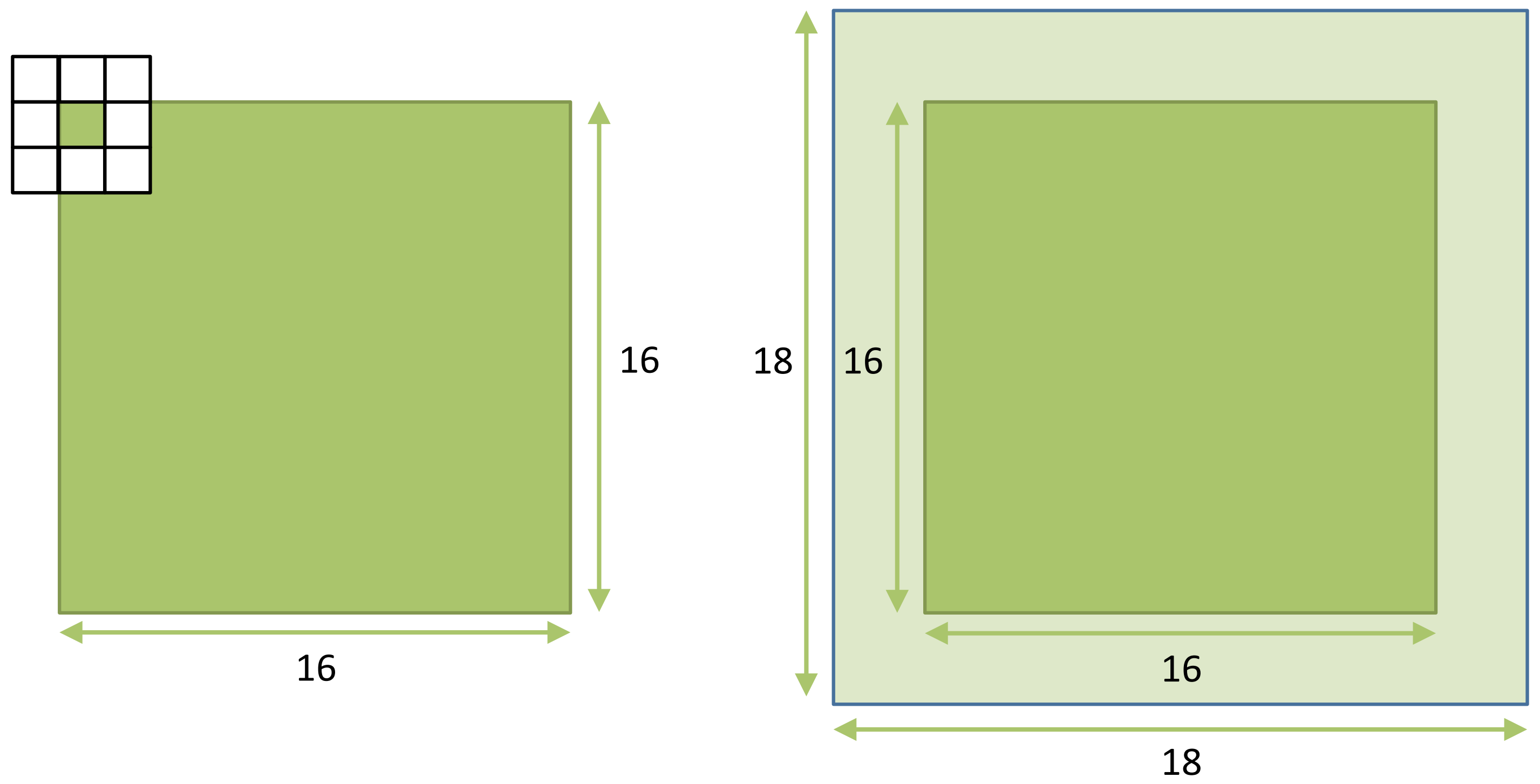
[1] T. Ojala, M. Pietikäinen, and D. Harwood (1994), *Performance evaluation of texture measures with classification based on Kullback discrimination of distributions*, Procs. of the 12th IAPR Intl. Conf. on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585

Implementation

The LBP implementation is very parallel friendly because each pixel's output value is evaluated independently. Moreover, memory accesses in the adjacent 8-neighborhood eases caching. The CPU version is straightforward, even a multithreaded solution using OpenMP, while the GPU implementation takes advantage of the improved performance of device shared memory.

In the GPU case, each thread does not access to the three RGB values for each pixel, but each thread accesses to a different channel from each pixel.



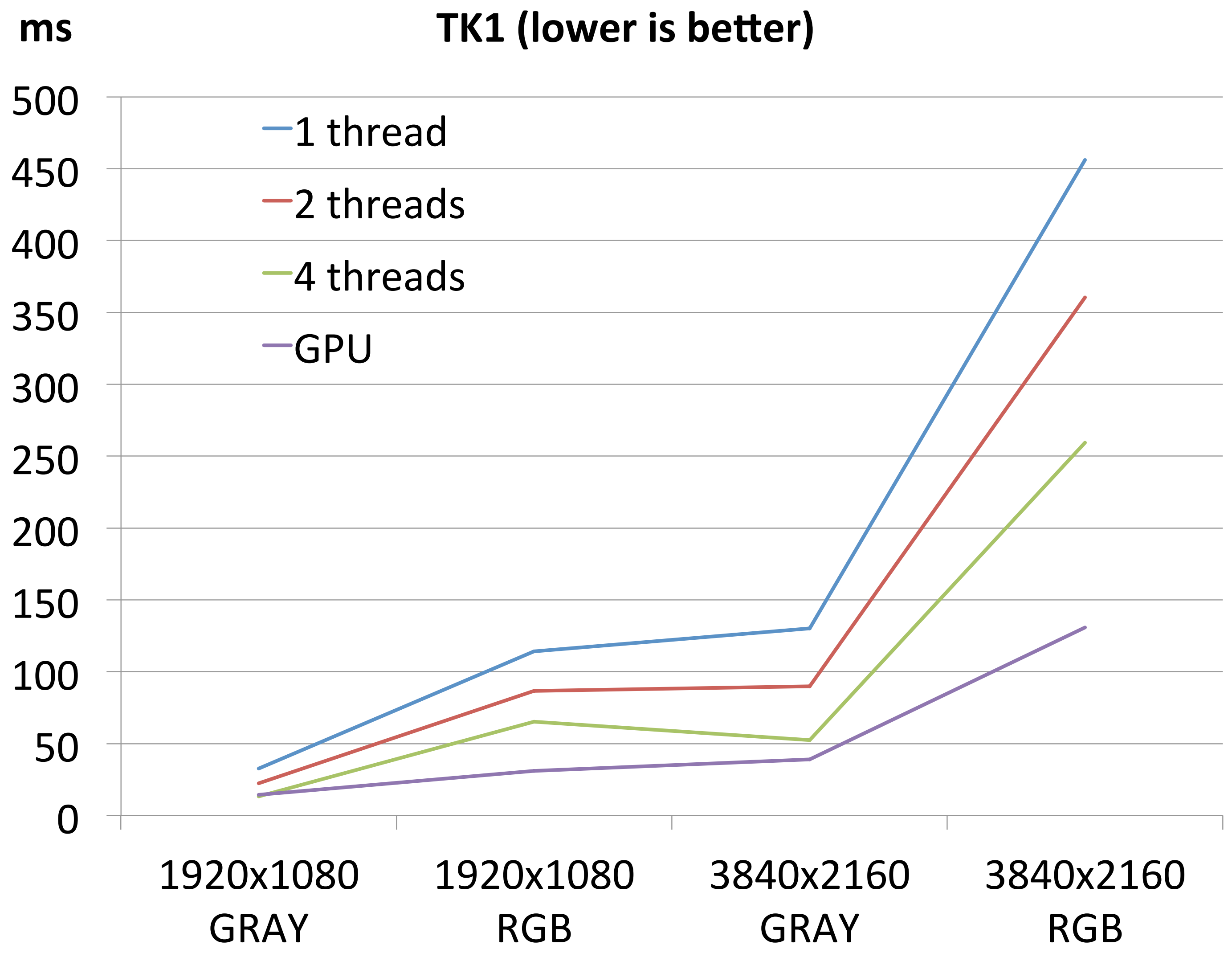


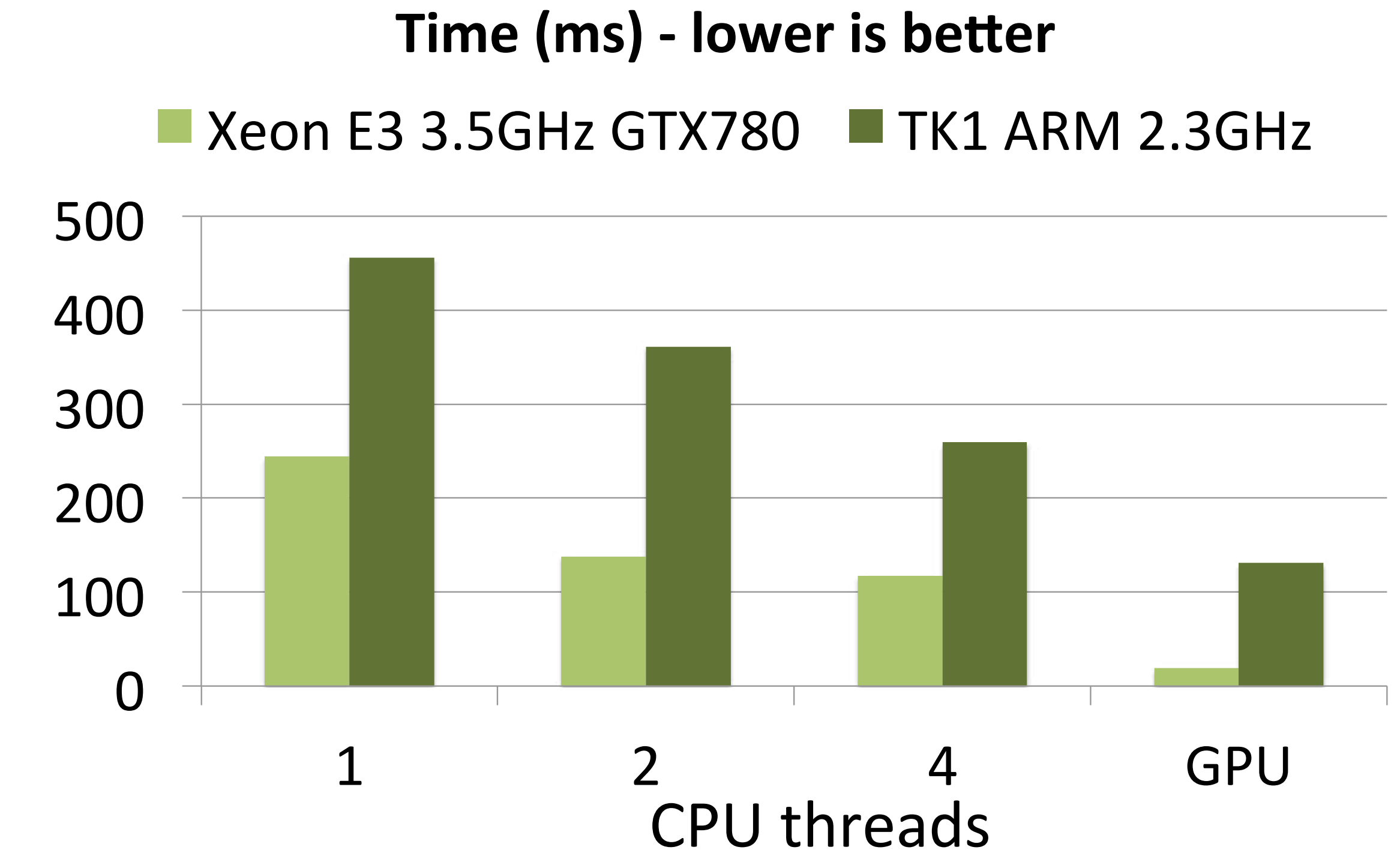
(a) No shared memory (b) Shared memory

For instance, for a 16x16 region, the LBP algorithm reads 8 neighbor pixels for each output pixel (a), performing 16 x 16 x 8 = 2048 global memory reads. Using the device shared memory (b), we can avoid multiple reads of the same neighbor pixel. Therefore, we only need to perform 18 x 18 = 324 global memory reads and 16 x 8 = 2048 shared memory reads that becomes much faster. This improvement is able to reduce the computing time for a 4K image and three channels (RGB) from 191 to 130 ms. Notice that this improvement is obtained for a low arithmetic intensity problem.

Experimental results

We firstly analyze the performance of the TK1 CPU and GPU when considering different image sizes (HD and 4K) and number of channels (GRAY and RGB). The GPU speedup increases with the image size.





In the most demanding configuration, we can clearly observe how TK1 GPU is even faster than a 2-threaded desktop CPU, and the 4-threaded Tegra solution is almost as fast as a single threaded CPU. Notice that we are using a high performance workstation (Xeon E3 3.5 GHz) with a high end GPU.

The GPU of TK1 is able to obtain competitive results compared to a modern workstation in this particular example. We can observe that even the TK1 CPU results are surprisingly good.

This research was partially supported by the Spanish Ministry of Education and Science CICYT TIN2011-28151 and TIN2014-57633