Accelerating Image Super-Resolution Regression by a Hybrid Implementation in Mobile Devices

Angelos Amanatiadis, Loukas Bampis, Antonios Gasteratos
School of Engineering, Democritus University of Thrace
12 Vas. Sofias Str, GR-67100, Xanthi, Greece.
E–mail: aamanat@ee.duth.gr

***Abstract*—This paper introduces a new super-resolution algorithm based on machine learning along with a novel hybrid implementation for next generation mobile devices. The proposed super-resolution algorithm entails a multivariate polynomial regression method using only the input image properties for the learning task. Although it is widely believed that machine learning algorithms are not appropriate for real-time implementation, the paper in hand proves that there are indeed specific hypothesis representations that are able to be integrated into real-time mobile applications. With aim to achieve this goal, we take advantage of the increasing GPU employment in modern mobile devices. More precisely, we utilize the mobile GPU as a co-processor in a hybrid pipelined implementation achieving significant performance speedup along with superior quantitative interpolation results.**

I. Introduction

The super-resolution task refers to the process of constructing a High-Resolution (HR) image from a Low Resolution (LR) one, often acquired by inexpensive mobile device imaging sensors. Many applications use this process as their core algorithm for tasks such as enhancing image spatial resolution, synthetic zooming of region of interest, mosaicing and image restoration.

Traditional analytic approaches include Cubic spline interpolation, sharpened Gaussian interpolator functions, wavelet-based methods and fractal interpolation [1], [2], [3]. However, these approaches can often suffer from perceived loss of detail in textured regions mainly because of their incapacity to recover the high-frequency components which where degraded during the low-resolution sampling process. Modern approaches however, have introduced learning based methods including a training phase in the overall process. Freeman et al. [4] treated super-resolution as a learning problem of estimating high-frequency components. This was achieved by learning the fine details which correspond to different image regions seen at low-resolution example images and then use those learned relationships to predict fine details in other images. Machine learning super-resolution, also known as example based super resolution, has been also introduced to numerous recent algorithms with very promising results [5], [6], yet, their main trade-off hides in their demanding learning phase making them inappropriate for real-time applications.

The super resolution machine learning algorithms however, underly a great level of parallelism which can be of very practical use, when explored along with the recent availability of General Purpose Graphic Processing Units (GPGPUs). The computing capability offered by the GPUs can report speedups ranging from several times to hundreds of times depending on the application especially in image processing applications [7]. However, these figures apply only for desktop GPUs since mobile GPUs are often designed with power consumption rather than performance as the primary goal [8]. Current mobile device architectures employ powerful CPUs along with relevant limited GPU resources leading to potential bottlenecks.

In this paper, we present a novel super-resolution algorithm based on multivariate polynomial regression using only the input image properties for the training phase. The parallel characteristics of both the machine learning algorithm and super resolution are addressed to achieve bandwidth performance in mobile device applications. A proposed hybrid implementation is also employed to distribute the computations on both CPU and GPU in a pipelined scheme as to exploit the most of the available computational power of a mobile device.

II. Multivariate Polynomial Regression for Super-Resolution

Our super-resolution approach can be considered as a supervised learning problem using regression analysis as we want to predict real valued output of the HR image from an $M \times N$ pixel LR image. Each horizontal and vertical image line becomes an N or M dimensional vector of training examples, respectively. We apply multivariate polynomial regression for fitting the interpolated values, where the hypothesis is given by the polynomial model $h_\theta(x) = \boldsymbol{\theta}^T \mathbf{x}$ where the matrix $\mathbf{x}$ includes all polynomial terms of x_1 and x_2 up to the third power and $\boldsymbol{\theta} \in \mathbb{R}^{15}$. The terms x_1 and x_2 correspond to the two opposite neighbors of the pixel to be interpolated in vertical or horizontal direction, depending on the image line or column training example, respectively. Selecting the appropriate degree of polynomial for our hypothesis is critical in order to avoid overfitting or underfitting situations. Cubic polynomial has proved to present adequate fitting results and has been also used in many analytic interpolation techniques.

Our regularized cost function to be minimized is the following:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(\mathbf{x}^{(i)}) - \boldsymbol{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \boldsymbol{\theta}_j^2\right] \quad (1)$$

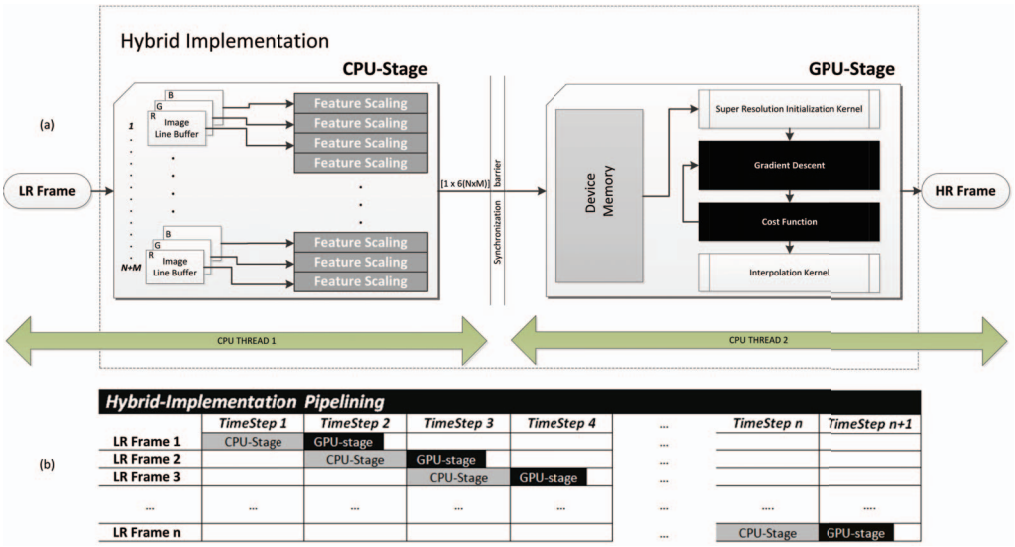

Fig. 1. (a) The Hybrid Implementation Flowchart (b) The Hybrid Implementation Pipelining. By splitting the overall algorithm and assigning different parts to two separate resources enables us to pipeline the procedure. The experimental results have confirmed that for systems armed with powerful CPUs compared to their GPUs abilities, the pipeline approach succeeds a reduction of the overall execution time compared to the GPU-only implementation, more evident in small image sizes.

where m corresponds to the N or M number of training examples, λ the regularization parameter and $n = 15$. The $\boldsymbol{y}$ vector contains the initial $i-th$ pixel value of the LR image across the line or column. For minimizing the cost function, gradient descent updates simultaneously the θ parameters as follows:

$$\theta_j := \theta_j \left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \quad (2)$$

for $j = 1, 2, ..., 14$. Since multiple features in different scales exist, we applied feature scaling in order to help gradient descent to converge much faster, however, mean normalization was not used along with feature scaling, for not introducing extra computational complexity. The learning rate α was also evaluated and set to 0.3 for fast but definite convergence, since it greatly affects the overall speed of our algorithm.

III. CPU-GPU Hybrid Implementation

Since we use a batch learning algorithm both cost estimation function and gradient descent can be implemented in two different kernels with coalesced memory accesses from the device memory. A preceding kernel, creates the HR image from the LR to be filled depending on the resolution factor. The two iterative kernels are executed repeatedly until convergence is achieved. The last kernel inserts the new pixel values to the HR image. In this way, computation performed in different horizontal or vertical lines of the image can be executed in parallel, while the execution of the different kernels is synchronized by barriers. The CPU on the other hand is dedicated for the feature scaling and matrix transpose operations as shown in Fig. 1.

IV. Experimental Results

Experiments were performed on a Nvidia Tegra 3 development platform featuring a Quad-Core ARM Cortex A9 CPU and a Nvidia Quadro 1000M (96 cores) GPU, both with 2GB of RAM. This platform is one of the most representative boards hosting two heterogeneous processors designed for mobile devices such as smartphones and tablets. The average reported time for resolution factor of 2 are 31ms and 107ms for LR images of 256×256 and 640×480 respectively, achieving the overall frame rate of 32.2 and 9.3 frames per second.

V. Conclusion

In this paper we proposed a new multivariate polynomial regression algorithm for super-resolution in mobile devices. For this reason, we utilized the mobile GPU as a co-processor in a hybrid pipelined implementation achieving real-time performance from low inexpensive mobile device imaging sensors.

References

[1] A. Amanatiadis and I. Andreadis, "A survey on evaluation methods for image interpolation," *Measurement Science and Technology*, vol. 20, no. 10, p. 104015, 2009.

[2] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1049–1075, 1999.

[3] A. Amanatiadis and I. Andreadis, "An integrated architecture for adaptive image stabilization in zooming operation," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 600–608, 2008.

[4] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, 2002.

[5] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1127–1133, 2010.

[6] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 349–356.

[7] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[8] T. Akenine-Moller and J. Strom, "Graphics processing units for handhelds," *Proc. IEEE*, vol. 96, no. 5, pp. 779–789, 2008.