Accelerating Single-Image Super-Resolution Polynomial Regression in Mobile Devices

Angelos Amanatiadis, *Member*, IEEE, Loukas Bampis, and Antonios Gasteratos, *Senior Member*, IEEE

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 two dimensional polynomial regression method using only the input image properties for the learning task. Model selection is applied for defining the optimal degree of polynomial by adopting regularization capability in order to avoid overfitting. 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, the increasing GPU employment in modern mobile devices is exploited. More precisely, by utilizing the mobile GPU as a co-processor in a hybrid pipelined implementation, significant performance speedup along with superior quantitative results can be achieved.*[1].

Index Terms — Polynomial regression, super-resolution, general-purpose GPUs, hybrid implementation.

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]-[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. Furthermore, static function methods are limited to their domain space, and all image properties may not be well projected to the HR image.

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 using 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.

An increasing subset of machine learning methods for super-resolution are those that use regression in order to learn the relationships between the LR and HR images by utilizing high-dimensional feature space. Support vector regression has been applied in the work of Ni and Nguyen [7], where the kernel learning problem was formed as a convex optimization problem. After finding the optimal kernel, the output pixel of the HR image is calculated by applying support vector regression in multiple patches, thus solving the multiple output regression problem as separate single output regressions. Iterative steering kernel regression as proposed in the work of Takeda *et al.* [8] is used for estimating locally adaptive regression functions. An iterative regression/denoising procedure is used to exploit the HR image local properties by estimating radiometric terms in two dimensions for each iteration.

The super-resolution machine learning algorithms are considered highly computationally intensive processes mainly because of the utilization of high-dimensional feature space. However, they underlie 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 [9]. However, these figures apply only for desktop GPUs since mobile GPUs are often designed with power consumption rather than performance as their primary goal [10]. Current mobile device architectures employ powerful CPUs along with relevant limited GPU resources leading to potential bottlenecks.

In this paper, a novel super-resolution algorithm is presented, based on two dimensional polynomial regression using only the input image properties for the training phase. The method is improved by adopting a learning algorithm with regularization capability to avoid overfitting. The parallel characteristics of both the machine learning algorithm and super-resolution are addressed to achieve throughput

[1]A. Amanatiadis is with the School of Engineering, Democritus University of Thrace, GR-67100, Xanthi, Greece (email: aamanat@ee.duth.gr).

L. Bampis is with the School of Engineering, Democritus University of Thrace, GR-67100, Xanthi, Greece (email: loukbabi@ee.duth.gr).

A. Gasteratos is with the School of Engineering, Democritus University of Thrace, GR-67100, Xanthi, Greece (email: agaster@pme.duth.gr).

performance in a mobile device implementation. A proposed hybrid architecture 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 the mobile device [11].

The following section discusses related works on GPU implementations for the super-resolution problem. The proposed polynomial regression method is discussed in Section III. The hybrid implementation in a mobile device is presented in Section IV. Experimental results on both real and synthetic data are presented in Section V, along with their timing performance. Section VI concludes the paper.

II. Related Works

In this section, relevant super-resolution implementations will be presented. It must be noted that while there are many implementations in specific hardware devices like FPGAs [12], [13], DSPs [14] and ASICs [15], the interest in implementations in GPUs seems to be picking up significantly in recent years. However, mobile device implementations for super-resolution using regression are very limited, thus the discussion will be covered in two categories. The first category will include GPU implementations of single function and kernel for super-resolution, whereas the second category will present various super-resolution implementations that entail machine learning regression on GPUs.

A real-time high-quality image upscaling algorithm has been proposed by Giachetti and Asuni [16], based on the iterative smoothing of second-order derivatives. Two filling steps are required and performed by first computing local approximations of the second-order derivatives along the two diagonal directions, using eight-valued neighboring pixels. Interpolated values are modified in the second iterative procedure by trying to minimize an energy function. A graphics processing unit implementation has been proposed for obtaining real-time results. Another high-quality and efficient single-image upscaling technique using patches from extremely localized regions of the input image is proposed in the work of Freedman and Fattal [17]. This work exploits the local scale invariance of natural images where it is pointed out that small patches are very similar to themselves upon small scaling factors. The parallel nature of the proposed algorithm was appropriately utilized allowing a video upscale from 640 × 360 to 1920 × 1080 at 23.9 frames per second. A framework involving two rendering cycles for scaling videos in mobile devices has been recently proposed by Singhal *et al.* [18]. The first cycle is used to obtain the interpolated frame using bilinear interpolation, and the other cycle is used for a sharpening filter. Real-time performance was reported achieving a rate of full 35 frames per second at 560 × 420 resolution.

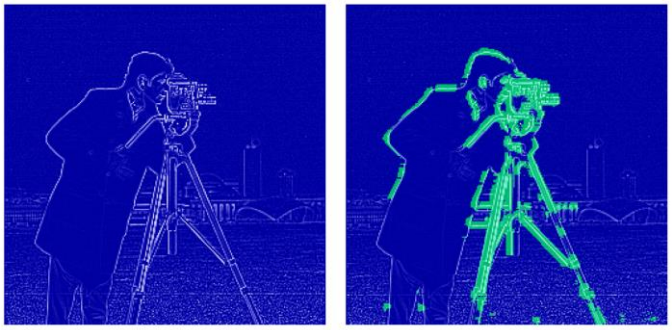
Since all the aforementioned implementations do not include regression in their core algorithms, a second category of related works is presented, where the GPUs are used as computing nodes for calculating super-resolution kernel regression problems. Both local and non-local kernel regression is applied for super-resolution in the work of Wang and Chan [19], where the local polynomial coefficients are obtained by weighted least squares. The proposed adaptive kernel construction and regression were both implemented on GPU, however some modifications were performed, since the computation at each pixel was too complicated to be run entirely in the computing units of the GPU. Hence, some steps were divided into subtasks and assigned to the CPU. This architecture dramatically reduced the processing time in reasonable levels. Finally, a bilateral filtering method was developed by Yang *et al.* [20] for a normalized convolution, in which the weighting for each pixel is determined by the spatial distance from the center pixel as well as its relative difference in intensity. This non-uniform mapping function was learned via support vector machine regression using the feature vectors and the corresponding bilateral filtered values of the training image. The GPU implementation of the method reported a rate of about 473 frames per second on a 1MB grayscale image.

III. Polynomial Regression for Super-Resolution

The proposed method relies in a limited amount of information using only the high frequency patches of the LR image. The HR reconstruction is partitioned according to these patches, decreasing the problem complexity per image patch allowing parallel implementation of the whole process. Unlike example-based super-resolution approaches which require long training phases using natural image prior [21], [22] the proposed method applies training phase only at the LR image patches. From the consumer electronics view, this non time consuming training phase with less computational burden is very important for fast applications, however at the same time the quality of high-frequencies in the HR image must be sufficiently maintained. The estimation of high-frequency details in the current work is faced as a supervised machine learning problem which is resolved using a bivariate polynomial regression. The quality attribute of high-frequencies is defined by the regularization of the regression framework [23], as it will be discussed thoroughly in the next sections. The challenge is even greater when limited amount of training data are available for the training phase since the regression might interpolate poorly on nontraining data. To overcome these challenges a convex combination of regressors based on their cross validated confidence is applied in the current work along with a neighborhood exploitation for selecting the training data points.

A. Preprocessing Based on L^2 and L^∞ Norms

Adopting the general framework of Freeman *et al.* [4], for the super-resolution task, the LR image (X), is first scaled in the desired factor by applying cubic spline interpolation. The HR image (SX) is missing its high-frequency details which are going to be learned and estimated, based on the frequency components of the LR image (X). The regression output image



(a) (b)

Fig. 1. Example of the preprocessing part: a. Laplacian result, b. Post-processed result based on the intersection of thresholded $L^2(\nabla^2 X)$ and $L^\infty(\nabla^2 X)$ patches (α=0.5 and β=0.4).

can subsequently be added to *SX* to produce the final super-resolved image. The cubic spline interpolated image *SX*, suffers from smoothing and blurring in major edges since the cubic B-spline basis is a nonnegative function. Applying regression throughout the whole image region could lead to several drawbacks such as degraded results in textured areas where the contrast is low, as well as increasing the computational complexity of the proposed framework.

A local regression will be applied only to patches of the major edges and their vicinities of the (*X*) where the high spatial frequency components are present. The definition of a major edge is in general distinct from the object contour, where the intensity variations are negligible across the boundaries. The vicinities of the edges were also included since in many super-resolution algorithms oscillation occurs near the strong edges of the HR image in order to compensate the resulting loss of smoothness.

For finding the major edges and their vicinities, a thresholding for each patch is applied to L^2 and L^∞ norms of the Laplacian image *X(i,j)*, where $i \in 1,2,\ldots,M$, $j \in 1,2,\ldots,N$, as follows:

$$\nabla^2 X(i,j) = \sum_{s=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{t=-\frac{n-1}{2}}^{\frac{n-1}{2}} K_\ell(i,j) X(i+s, j+t) \tag{1}$$

$$L^2\left(\nabla^2 X(i,j)\right) = \sum_{s=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{t=-\frac{n-1}{2}}^{\frac{n-1}{2}} K_2(i,j) \nabla^2 X(i+s, j+t) \tag{2}$$

$$L^\infty\left(\nabla^2 X(i,j)\right) = \sum_{s=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{t=-\frac{n-1}{2}}^{\frac{n-1}{2}} K_\infty(i,j) \nabla^2 X(i+s, j+t) \tag{3}$$

where, K_ℓ, K_2, $K_\infty \in \Re^{n \times n}$, are the Laplacian kernel and L^2 and L^∞ norms, respectively. For all the *q* and *l* patches P_{L^2} and P_{L^∞} of images $L^2(\nabla^2 X)$ and $L^\infty(\nabla^2 X)$ respectively, the following subsets are defined as:

$$C_{L^2} \subseteq P_{L^2} : p^q_{L^2} > \alpha \tag{4}$$

$$C_{L^\infty} \subseteq P_{L^\infty} : p^l_{L^\infty} > \beta \tag{5}$$

where α and β are the threshold values. Finally, the LR image patches that the regression will be applied is defined as the intersection of both subsets $C_{L^2} \bigcap C_{L^\infty}$.

Fig. 1(a), depicts the result of the applied Laplacian kernel on the initial 'cameraman' image, whereas Fig. 1(b) shows the post-processed result of the $L^2(\nabla^2 X)$ and $L^\infty(\nabla^2 X)$ norms intersection on the Laplacian image. It can be seen that half of the back of the cameraman with the ground is not detected as major edge as the intensity variations are not significant across that boundary. Since the regression result in such boundaries does not produce visible oscillation of the pixel values [24], local regression will be applied only to the selected edge patches in order not to introduce additional computational burden in the mobile device.

B. Kernel Learning for Polynomial Regression

The proposed super-resolution approach can be considered as a supervised learning problem using regression analysis for the real valued output of the HR image pixels from an $n \times n$ LR image patch. Each LR image patch becomes an $n^2 \times 2$ dimensional matrix of training examples. The *2-D* bivariate polynomial regression is applied for fitting the interpolated values, with the hypothesis given by the polynomial model $h_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^T \mathbf{x}$, where the matrix $\boldsymbol{x}$ includes all the *k* degree polynomial terms x_{i+s} and x_{j+t} and $\boldsymbol{\theta} \in \Re^{\binom{k+2}{2}+1}$. Terms x_{i+s} and x_{j+t} correspond to the image pixels coordinates within the patch with $s \in \left(-\frac{n-1}{2}, \frac{n-1}{2}\right)$ and $t \in \left(-\frac{n-1}{2}, \frac{n-1}{2}\right)$, respectively. Selecting the appropriate degree of polynomial for the hypothesis is critical in order to avoid overfitting or underfitting situations. Other approaches suggest a linear combination of other functions such as Legendre or Hermite polynomials, however this would dramatically increase the computational complexity of the algorithm. High order polynomials have proved to present adequate fitting results and have been also used in many analytic interpolation techniques.

The regularized cost function to be minimized is the following:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{\binom{k+2}{2}} \boldsymbol{\theta}_j^2\right] \tag{6}$$

where *m* corresponds to the number of training examples in each image patch and λ is the regularization parameter. The $\boldsymbol{y}$ vector contains the center pixel value of the LR image patch.

To minimize *J*, its derivatives are set to zero, and the following normal equations are obtained [25]:

$$\boldsymbol{x}^T \boldsymbol{x} \theta = \boldsymbol{x}^T \vec{y} \tag{7}$$

where, $\vec{y}$ is the n^2-dimensional vector containing all the target values from the training set, and $\boldsymbol{x}$ is the n^2 matrix that contains the training example input values. The value of $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$ is given in closed form by the following equation:

$$\theta = \left(\boldsymbol{x}^T \boldsymbol{x} \right)^{-1} \boldsymbol{x}^T \vec{y} \tag{8}$$

For minimizing the cost function, normal equations were chosen instead of gradient descent since they enable solving small linear systems very quickly when the training set is limited and bounded. Furthermore, the required for the normal equations calculation of the pseudo-inverse matrix using singular value decomposition is suitable for parallel processing, which will be exploited with the use of the GPU in the next section.

The rank of the chosen polynomial exerts a critical effect on how the regression will fit to the training data. More precisely, when a high order polynomial is selected overfitting might be introduced, resulting into a high variance or noisy supperresolution results. A low order polynomial however, might introduce blurring effects due to the high bias introduced by the regression process [26]. This model selection problem is improved by the regularization capability of (6). For each training patch a single polynomial is selected based on the cross validation criterion. The cross validation criterion between several tested regularization parameters λ can indicate poor performance in both high variance or high bias cases by evaluating the hypothesis from cross validation examples as:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left(h_\theta \left(x_{cv}^{(i)} \right) - y_{cv}^{(i)} \right)^2 \tag{9}$$

where m_{cv} is the number of cross validation dataset and $\boldsymbol{y}_{cv}$ and $\boldsymbol{x}_{cv}$ are the cross validation subsets of $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively.

C. Convex Combination of Polynomials

After the regression part, for each center pixel of the input patches a single polynomial has been chosen. However, since the input patches are overlapping with their neighbors, neighboring pixels may be fitted with different polynomials since they have been trained with partially different local training examples. These different local training examples contain different partial spatial information of the input image. Based on the selection of the training examples for each image patch, and due to the stochastic random selection, relatively high training errors might be seen when the training candidates are not selected uniformly in the spatial domain leading to a non optimal fitting as shown in Fig. 2(a). To avoid this drawback, the neighborhood context of each input pixel is exploited. Thus, the final estimation for each HR pixel is calculated as a convex combination of polynomial candidates

$$SX(x,y) = \sum_{i=1,\ldots,l} w_i \left(\frac{x}{s}, \frac{y}{s} \right) h_i \left(\frac{x}{s}, \frac{y}{s} \right), \tag{10}$$

where

$$w_i(x,y) = \exp\left(-\frac{J_{test}^{(i)}([x],[y])}{\sigma_C} \right) \Big/ \sum_{j=1,\ldots,l} \exp\left(-\frac{J_{test}^{(i)}([x],[y])}{\sigma_C} \right) \tag{11}$$

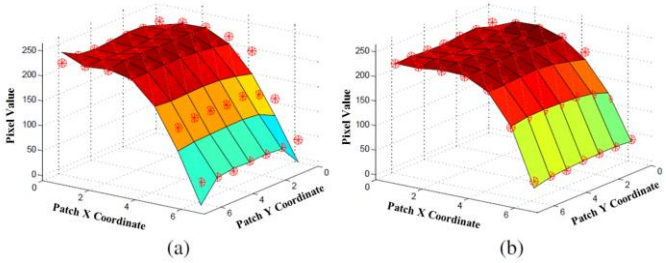


Fig. 2. 2-D fitting example in image patch: a. Without convex combination of polynomials, b. After convex combination of polynomials (n=7, σ_c=500).

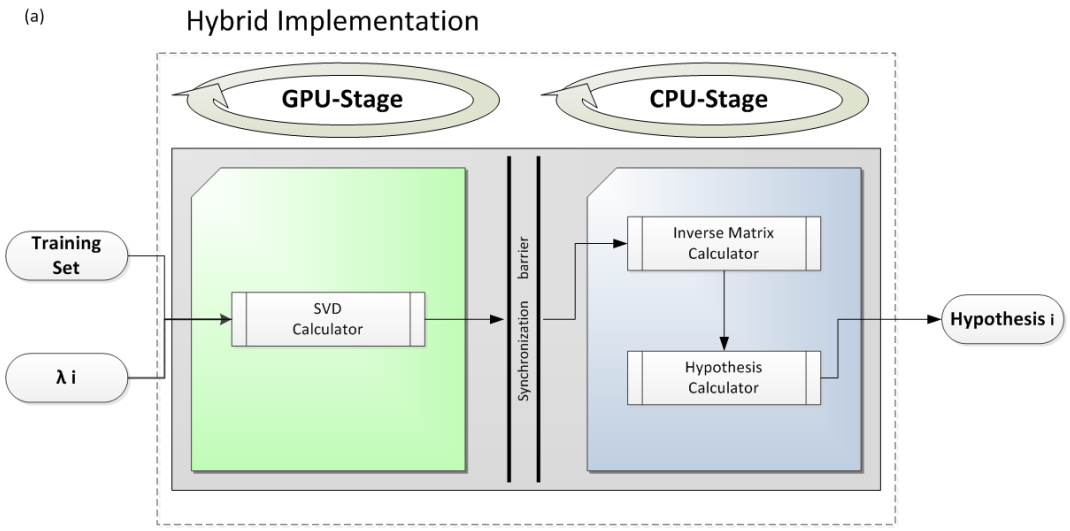
and $\boldsymbol{\sigma_c}$ being a weight parameter, s the scaling factor, and $\left\{ J_{test}^1(x,y), \ldots, J_{test}^l(x,y) \right\}$ the generalization error for each test set of the $l \leq n^2$ polynomial candidates. As it can be seen in Fig. 2(b), the use of the convex combination of polynomial candidates can efficiently single out the outliers, since the candidates contain diverse spatial information of the input image patches providing a better fitting.

Tuning of the hyperparameters is based on the trade-off between computational complexity and super-resolution quality. Increasing the parameters α and β result in fewer selected patches decreasing the runtime complexity but also decreasing the quality of the super-resolved image. A similar tendency is detected when decreasing the parameter n since smaller image patches lower the computational burden but at the same time deteriorate the quantitative results. Furthermore, all hyperparameters are independent of the selected scaling factor as they are applied on the input image, thus any decimal scaling factor can be applied in the proposed method. As it will be discussed in the experimental results section, the optimization of the parameters were performed based on rough estimations [5] and then a fixed set of candidates were fully validated.

IV. Hybrid Implementation in Mobile Device

The proposed hybrid implementation is based on accurate pipelined calculation assignment to both available processing units of the mobile device with respect to their capabilities, trying to accelerate the overall timing execution performance. CPUs in mobile devices exist in the scene for many years and they have been optimized in terms of memory accesses, thread handling, and performing complex calculations on a single stream. Recently, mobile devices have also integrated GPUs in the overall architecture, which are perfectly suited for performing relatively more complex and independent calculations on very large datasets, because of their massive parallelization while offering significant power efficiency [27].

By commissioning the CPU with dependent tasks on small datasets, like multiplications of small matrices and data transferring, and by assigning to the GPU calculations over large datasets, better acceleration timing results can be achieved. The rest of this section is focused on how these different processes of the proposed super-resolution algorithm were assigned to each processing unit, with their equivalent



(b)

Hybrid Implementation Pipelining

	TimeStep 1	TimeStep 2	TimeStep 3	TimeStep 4	...	TimeStep n	TimeStep n+1
λ1	GPU-stage	CPU-stage			...		
λ2		GPU-stage	CPU-stage		...		
λ3			GPU-stage	CPU-stage	...		
λ4				GPU-stage	...		
⋮	⋮	⋮	⋮	⋮	...		
					...	CPU-stage	
λn					...	GPU-stage	CPU-stage

Fig. 3. (a) The Hybrid Implementation Flowchart (b) The Hybrid Implementation Pipelining. By splitting the overall algorithm and assigning different parts to two separate resources enables the pipelining of the procedure. The experimental results have confirmed that for systems armed with powerful CPUs compared to their GPUs abilities, the proposed pipeline approach succeeds a reduction of the overall execution time compared to the CPU-only or GPU-only implementation.

implementations. Discussion of the most computationally intensive parts of the proposed method, such as the input image major edge extraction, the polynomial hypothesis calculation for each edge patch and the convex combination ofpolynomial candidates is also given in the following subsections.

A. Preprocessing

The preprocessing stage encloses the calculation of the Laplacian of the image and the application of L^2 and L^∞ norms. Those implementations are assigned to the GPU, as they are referred to the whole image, giving the opportunity for great parallelization. The calculation of the image Laplacian is translated into applying the convolution kernel described in (1) to the whole input image. Since every pixel in the input image has to be accessed multiple times, the image data are stored in the device texture memory. Texture memory is a cashed, read-only device memory able to improve performance and reduce memory traffic when readings have certain access patterns. One GPU thread per input pixel is able to access all neighboring values efficiently, regardless the memory storing pattern of the image data [28], thus accelerating the application of the Laplacian kernel.

With the Laplacian image stored, the L^2 and L^∞ norms are computed as follows. The L^2 norm of each p input image patch is calculated as:

$$L^2(p)=\sqrt{\sum_{i=1}^{s}\sum_{j=1}^{t}\left|\nabla^2 p(i,j)\right|^2} \qquad (12)$$

The challenge here is to compute the sum efficiently taking advantage of the fact that every summation region overlaps with its neighbors [29]. Thus, $s\times t$ threads accumulate the same $\nabla^2 p(\lceil s/2\rceil,\lceil t/2\rceil)$ value to the whole corresponding $[s,t]$ neighborhood of the L^2 norm output. In order to prevent multiple threads from accumulating simultaneously values in the same memory address, a sliding windows based approach is adopted. The amount of windows applied on an image is M/s in the horizontal order and N/t in the vertical and their dimensions are s by t, in respect with the number of threads consist them. Performing $s\times t$ iterations of the above mentioned algorithm by sliding all windows by one (s times in vertical order and t in horizontal) and synchronizing the threads every time, the overall accumulation is accomplished without any memory access congestion.

A similar approach is adopted for calculating the L^∞ norm:

$$L^\infty(p)=\max\left(\left|\nabla^2 p(i,j)\right|\right) \qquad (13)$$

where $i\in 1,2,3,\ldots,s$ and $j\in 1,2,3,\ldots,t$. In this operation instead of accumulating the $\nabla^2 p(i,j)$ values, the values are compared in order to find the maximum.

B. Polynomial Regression

The most computationally intensive part of the super-resolution regression analysis is the calculation of the necessary for the normal equations pseudo-inverse matrix using the Singular Value Decomposition (SVD). In particular, the pseudo-inverse matrix has to be evaluated for each λ of the chosen range in order to avoid overfitting. Therefore, for every λ the GPU is assigned with the necessary calculation of U, S and V matrices of the SVD. The CPU on the other hand, is responsible for evaluating the pseudo-inverse matrix and calculating the hypothesis [30]. Since both tasks are performed for multiple λ, the opportunity for a pipeline based system arises [11]. For each λ_i in the list, while the GPU calculates a new set of the U_i, S_i and V_i matrices, the CPU calculates the hypothesis for λ_{i-1} having as input the set of the respective U_{i-1}, S_{i-1} and V_{i-1} matrices. Using this method, which is depicted in Fig. 3, a great acceleration is achieved by exploiting the parallel computational power of the GPU and the simultaneous computations on the CPU, which otherwise the last would stay idle. Furthermore, calculation assignment in both CPU and GPU was evaluated against their respective average execution times, since the approximately same execution times would lead to an optimal performance.

C. Combination of Polynomials

The final stage of the proposed implementation incorporates the weighted summation of the polynomial hypotheses from the overlapping patches. This task is a suitable candidate for the GPU as it contains numerous but simple calculations. For every evaluated hypothesis, the assigned number of GPU threads are the same as the number of the hypothesis coefficients. Each one of these threads is responsible for checking whether or not the neighbor pixels corresponds to an edge (that is, they own a hypothesis) and then, deciding to accumulate the corresponding weighted coefficient, resulting to the final hypothesis.

V. Experimental Results

The experimental results section includes evaluations for both quantitative and timing comparisons utilizing the diverse content image dataset of Fig. 4. The under comparison algorithms include a non-parametric regression algorithm [8] based on steering kernel regression (SKR), a two rendering cycle (2RC) scaling algorithm which was also implemented in a mobile device [18], two example based methods utilizing soft information (SIEB) [31] and structure analysis (SAEB) [32], respectively, and finally a super-resolution algorithm with non-local kernel regression (NLKR) [19], where the local polynomial coefficients are obtained by weighted least squares. The selected state-of-the-art algorithms were chosen mainly because of their reported GPU implementations and the regression characteristics that they include.

Fig. 5 and Fig. 6 show the six different HR images of the noisy 'Parrot' and 'Child' input image for a scaling factor of 3, respectively. The proposed method presents an optimal balance between blurring (bias) and noise (variance). 2RC and SIEB methods suffer from larger smoothness whereas NLKR, SAEB and SKR create smooth displacements that cannot capture the discontinuity between strong edges.

Fig. 4. The image dataset of diverse content used for performance evaluation.

Despite the fact the proposed visual results show a good performance, it is well established that visual results fail to provide decisive indication of the quality of the scaled images, thus a quantitative evaluation was also performed.

A. Quantitative Evaluation

For quantitative performance evaluation, HR images obtained from the aforementioned state-of-the-art methods were compared in terms of peak signal-to-noise ratio (PSNR), and structural similarity (SSIM) [33], [34], [35], for assessing the quality of reconstruction. The evaluation procedure firstly builds the LR images from the reference ground truth images. This step includes a blurring and then a downsampling of the reference image by bicubic interpolation. Several works apply nearest neighbor interpolation for the subsampling step, however bicubic interpolation is naturally unbiased when generating low-resolution images. Finally, the HR images are obtained by using the different under evaluation techniques for upscaling the LR images to the initial corresponding sizes. Two of the methods [8], [19], required a slightly different reference images in terms of size in order to compensate the slightly different zoom factors and translation created by their algorithms. In the following tests, such issues were resolved by symmetrically replicating the pixels across image boundaries.

For reducing the computational cost in color images, the images are firstly transformed to the *YIQ* color space and the proposed regression is applied only to the *Y* channel since both *I* and *Q* channels include low frequency information. Finally, the bicubic interpolation of these two chromatic channels is combined with the *Y* channel to form the super-resolved image.

The HR reconstruction results of PSNR and SSIM for a scaling factor of 2 for the image dataset of Fig. 4, are presented in TABLE I and TABLE II, respectively. The selected parameters used in the presented results are $\alpha=0.5$ and $\beta=0.4$ for the L^2 and L^∞ patch thresholds, respectively, $n=7$ for the patch size, $k=4$ for the polynomial degree and $\sigma_C=500$ for the weight parameter.

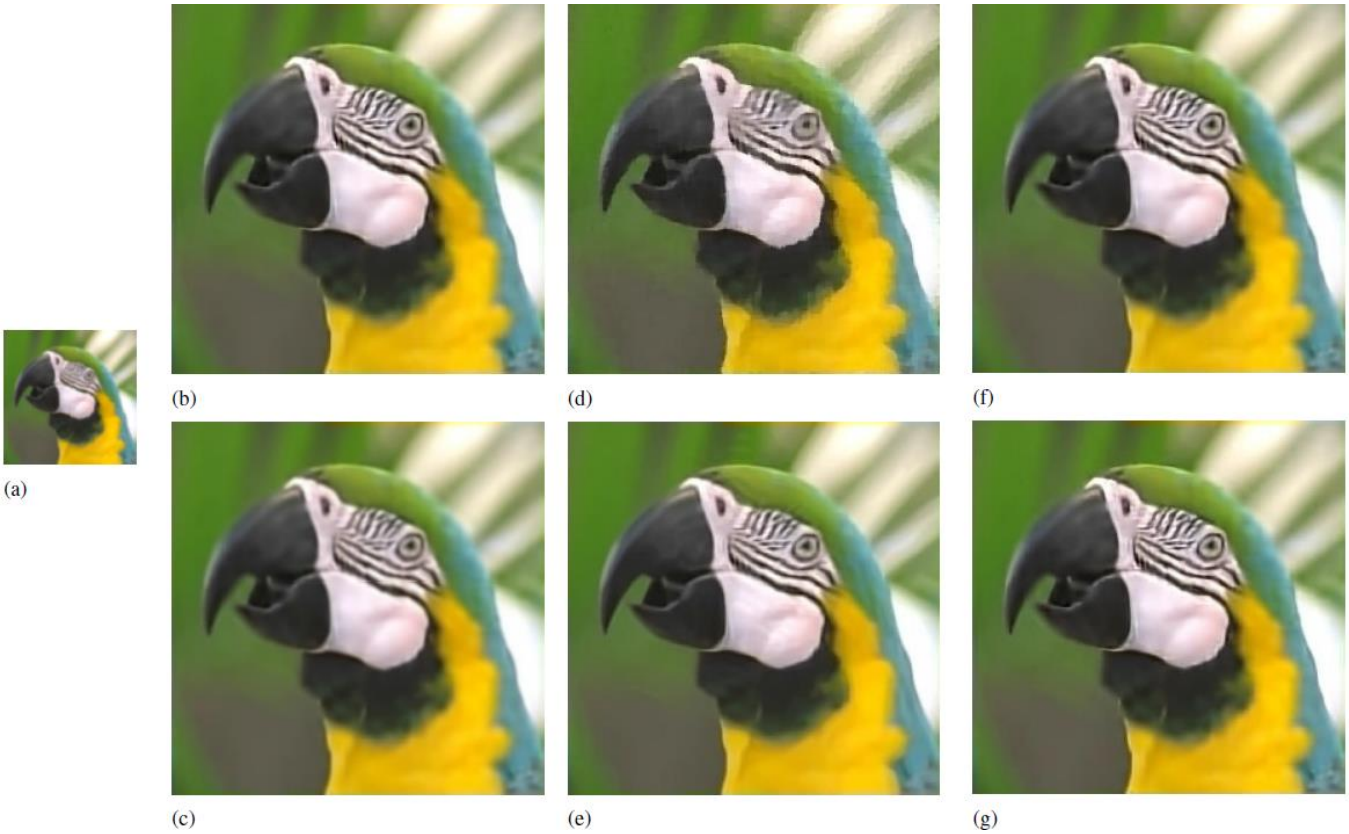


Fig. 5. Super-resolution results for the noisy `Parrot' image for 3×SF (standard deviation of Gaussian noise is 5): (a) downscaled image; (b) result using SKR method; (c) result using 2RC method; (d) result using NLKR method; (e) result using SIEB method; (f) result using SAEB method; (g) result using the proposed method.

Fig. 6. Super-resolution results for the noiseless `Child' image for 3×SF: (a) downscaled image; (b) result using SKR method; (c) result using 2RC method; (d) result using NLKR method; (e) result using SIEB method; (f) result using SAEB method; (g) result using the proposed method.

TABLE I

PSNR (*dB*) FOR DIFFERENT IMAGES FOR 2 × SR RATIO

Images	SKR	2RC	SIEB	SAEB	NLKR	Proposed
Baboon	24.307	22.430	24.913	24.523	23.101	24.927
Cameraman	28.337	26.826	28.560	28.204	27.778	28.652
Chip	33.032	29.838	33.412	32.329	33.322	33.813
Child	32.088	29.810	31.982	30.711	30.180	32.312
Lena	32.633	29.273	32.661	31.993	31.718	32.738
Monarch	25.914	23.911	26.642	26.341	25.523	26.412
Parrot	32.895	29.722	33.120	31.803	31.238	33.210
Parthenon	26.623	25.874	27.205	26.966	26.825	27.305
Peppers	30.663	29.375	30.213	29.890	30.091	30.053
Zebra	30.465	28.414	32.189	32.210	31.899	32.964

TABLE II

SSIM FOR DIFFERENT IMAGES FOR 2 × SR RATIO

Images	SKR	2RC	SIEB	SAEB	NLKR	Proposed
Baboon	0.9810	0.9637	0.9840	0.9872	0.9728	0.9887
Cameraman	0.9013	0.7355	0.8832	0.8810	0.8133	0.9063
Chip	0.9689	0.9520	0.9711	0.9634	0.9712	0.9723
Child	0.8893	0.8132	0.8501	0.8499	0.8480	0.8903
Lena	0.8520	0.8144	0.8444	0.8480	0.8613	0.8710
Monarch	0.8891	0.8485	0.9020	0.8714	0.8713	0.8961
Parrot	0.9338	0.9163	0.9380	0.9205	0.9067	0.9514
Parthenon	0.7303	0.6754	0.7480	0.7431	0.7443	0.7631
Peppers	0.9802	0.9867	0.9820	0.9773	0.9892	0.9856
Zebra	0.9769	0.8813	0.9768	0.9703	0.9635	0.9777

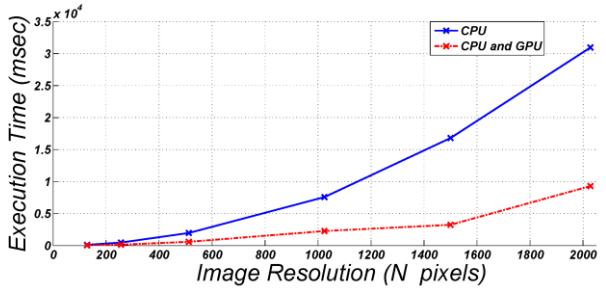


Fig. 7. Required execution time at different image resolutions. Image size is *N*×*N*.

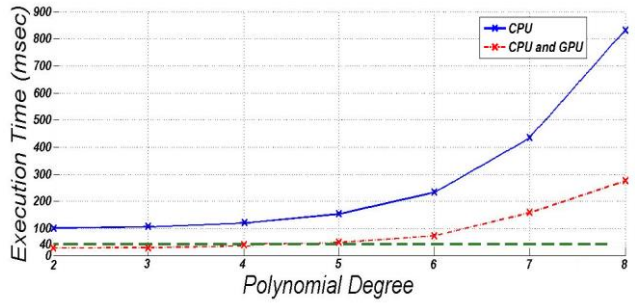


Fig. 8. Required execution time for different polynomial orders. The straight dashed line indicates the real-time performance of 25 frames per second.

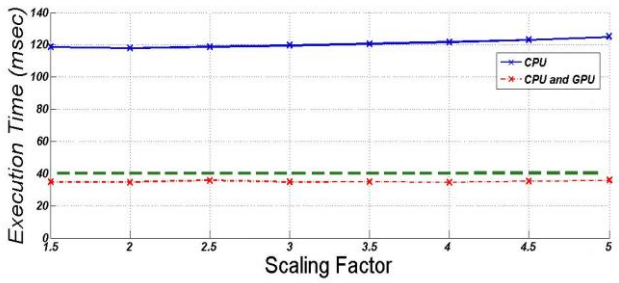


Fig. 9. Required execution time for different scaling factors. The straight dashed line indicates the real-time performance of 25 frames per second.

B. Timing Performance

Timing experiments were performed on a development platform featuring a 1.3 GHz quad-core CPU and a GPU with a total of 96 cores running at 0.7GHz, 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 influence of the input image resolution to the performing time is depicted in Fig. 7. The experiments took place over the same image in many resolution instances in order to preserve a fare ratio of defined patches. It can be noted that the proposed hybrid implementation keeps a lower execution time than the CPU implementation, maintaining a relatively stable demotion factor. Acceleration ratio is increased with the increase of the image resolution since when image resolution is low, the computation load of GPU thread is not very high and the frequent thread context switch demands a lot of system running time. However, when the image resolution is high, the computation load needed by each thread is sufficient and the cost of context switch is reduced.

Fig. 8 demonstrates the effect of the chosen polynomial degree to the execution time. The calculations occurred over an image of *256×256* resolution, for a scaling factor of *2*. As it can be seen for a polynomial degree of *4*, real-time executions can be succeeded. However, by decreasing the grade of the hypothesis, real-time performance can be achieved also in larger images. Fig. 9, illustrates the dependence of the execution time over various scaling factors. The relative limited timing variance shown in the diagram can be sufficiently explained by the fact that the most computationally intensive part of the hybrid implementation is the calculation of the hypothesis which occurs on the input image, which is independent on the resolution of the super-resolved image.

VI. CONCLUSION

In this paper, a novel super-resolution algorithm based on two dimensional polynomial regression is presented. The method relies on a single image learning algorithm with regularization capability for preventing overfitting. The parallel characteristics of both the regression analysis and normal equations are addressed to accelerate throughput performance in modern mobile devices. A proposed hybrid architecture is also employed to assign specific computations on both CPU and GPU in a pipelined scheme as to exploit the most of the available computational power of the mobile device. Experimental results show a fine balance between image quality and speed performance.

REFERENCES

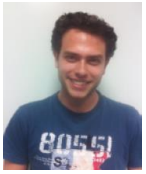
[1] A. Amanatiadis and I. Andreadis, "A survey on evaluation methods for image interpolation," *Measurement Science and Technology*, vol. 20, no. 10, p. 104015, 2009.

[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 superresolution," *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 International Conference on Computer Vision*, pp. 349–356, Sept. 2009.
[7] K. S. Ni and T. Q. Nguyen, "Image superresolution using support vector regression," *IEEE Trans. Image Process*., vol. 16, no. 6, pp. 1596–1610, 2007.
[8] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process*., vol. 16, no. 2, pp. 349–366, 2007.
[9] 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.
[10] T. Akenine-Moller and J. Strom, "Graphics processing units for handhelds," *Proc. IEEE*, vol. 96, no. 5, pp. 779–789, 2008.
[11] A. Amanatiadis, L. Bampis, and A. Gasteratos, "Accelerating image super-resolution regression by a hybrid implementation in mobile devices," *in Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, USA, pp. 335–336, Jan. 2014.
[12] A. Amanatiadis, I. Andreadis, and K. Konstantinidis, "Design and implementation of a fuzzy area-based image-scaling technique," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 8, pp. 1504–1513, 2008.
[13] T. Szydzik, G. M. Callico, and A. Nunez, "Efficient fpga implementation of a high-quality super-resolution algorithm with real-time performance," *IEEE Trans. Consum. Electron*., vol. 57, no. 2, pp. 664–672, 2011.
[14] S. Lopez, G. M. Callico, F. Tobajas, J. F. Lopez, and R. Sarmiento, "A novel real-time dsp-based video super-resolution system," *IEEE Trans. Consum. Electron*., vol. 55, no. 4, pp. 2264–2270, 2009.
[15] S. Marsi, S. Carrato, and G. Ramponi, "Vlsi implementation of a nonlinear image interpolation filter," *IEEE Trans. Consum. Electron*., vol. 42, no. 3, pp. 721–728, 1996.
[16] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *IEEE Trans. Image Process*., vol. 20, no. 10, pp. 2760–2768, 2011.
[17] G. Freedman and R. Fattal, "Image and video upscaling from local selfexamples," *ACM Transactions on Graphics*, vol. 30, no. 2, pp. 12:1– 12:11, 2011.
[18] N. Singhal, I. K. Park, and S. Cho, "Implementation and optimization of image processing algorithms on handheld gpu," *in Proc. IEEE International Conference on Image Processing*, pp. 4481–4484, Sept. 2010.
[19] C. Wang and S. Chan, "A new bandwidth adaptive non-local kernel regression algorithm for image/video restoration and its gpu realization," *in Proc. IEEE International Symposium on Circuits and Systems*, pp. 1388–1391, May 2013.
[20] Q. Yang, S. Wang, and N. Ahuja, "Svm for edge-preserving filtering," *in Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1775–1782, June 2010.
[21] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process*., vol. 19, no. 11, pp. 2861–2873, 2010.
[22] K. Zhang, X. Gao, D. Tao, and X. Li, "Multi-scale dictionary for single image super-resolution," *in Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1114–1121, June 2012.
[23] N. K. Bose and N. A. Ahuja, "Superresolution and noise filtering using moving least squares," *IEEE Trans. Image Process*., vol. 15, no. 8, pp. 2239–2248, 2006.
[24] K. I. Kim and Y. Kwon, "Example-based learning for single-image super-resolution," *Pattern Recognition*, 2008, pp. 456–465.
[25] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," *in Proc. NIPS*, vol. 6, pp. 281–288, Dec. 2006.
[26] H. He, K. He, and G. Zou, "A lorentzian stochastic estimation for video super resolution with lorentzian gradient constraint," *IEEE Trans. Consum. Electron*., vol. 58, no. 4, pp. 1294–1300, 2012.
[27] Y.-C. Wang and K.-T. Cheng, "Energy-optimized mapping of application to smartphone platforma case study of mobile face recognition," *in Proc. IEEE Computer Vision and Pattern Recognition Workshop*, pp. 84–89, June 2011.
[28] D. Ruijters and P. Thevenaz, "Gpu prefilter for accurate cubic b-spline interpolation," *The Computer Journal*, vol. 55, no. 1, pp. 15–20, 2012.
[29] F. Zhou, W. Yang, and Q. Liao, "Single image super-resolution using incoherent sub-dictionaries learning," *IEEE Trans. Consum. Electron*., vol. 58, no. 3, pp. 891–897, 2012.
[30] L.-J. Kau and C.-S. Chen, "Speeding up the runtime performance for lossless image coding on gpus with cuda," *in Proc. IEEE International Symposium on Circuits and Systems*, pp. 2868–2871, May 2013.
[31] Z. Xiong, D. Xu, X. Sun and F. Wu, "Example-based super-resolution with soft information and decision," *IEEE Trans. Multimedia*, vol 15, no. 6, pp. 1458-1465, Oct. 2013.
[32] C. Kim, K. Choi and J.-B. Ra, "Example-based super-resolution via structure analysis of patches," *IEEE Signal Process. Lett.*, vol. 20, no. 4, pp. 407-410, April 2013.
[33] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process*., vol. 13, no. 4, pp. 600–612, 2004.
[34] A. Amanatiadis and I. Andreadis, "Performance evaluation techniques for image scaling algorithms," *in Proc. IEEE International Workshop on Imaging Systems and Techniques*, pp. 114–118, Sept. 2008.
[35] X. Gao, W. Lu, D. Tao, and X. Li, "Image quality assessment based on multiscale geometric analysis," *IEEE Trans. Image Process*., vol. 18, no. 7, pp. 1409–1423, 2009.

BIOGRAPHIES

A. Amanatiadis (S'00-GS'04-M'09) received the Diploma and the Ph.D. (with Honors) from the Department of Electrical and Computer Engineering, DUTH, Greece, 2004 and 2009, respectively. He has published more than 40 scientific papers and he is co-author of three book chapters. He was twice guest editor in the Imaging Systems and Techniques Special Issue in Measurement Science and Technology Journal, IOP and awarded as one of Transactions 'Outstanding Reviewers' in appreciation of outstanding service to the IEEE Instrumentation and Measurement Society. His areas of interests include electronic systems design, machine vision, robotics and real-time embedded systems. He is a member of the IEEE, EuCogIII and the Technical Chamber of Greece (TEE).



L. Bampis received the Diploma from the Department of Electrical and Computer Engineering, DUTH, Greece, in 2013. He is currently pursuing his Ph.D. degree in the field of machine vision and embedded systems. His areas of interests include real-time image processing using hardware accelerators and parallel processing.

A. Gasteratos (M'99-SM'11) is an Assoc. Prof. at Democritus University of Thrace (DUTH). He holds a B.Eng. and a PhD in Electrical and Computer Engineering, DUTH, (1994 and 1999, respectively). During the last 10 years he has been principal investigator to 5 EC, 2 ESA and 3 national funded projects related mostly to machine vision, robotics and security themes. He has published over 140 papers in peer reviewed journals and international conferences. He is Assoc. Editor at the International Journal of Optomechatronics and the Human-Centric Computing and Information Sciences. He is also a reviewer for projects supported by the EC and of many (over 40 different) international journals, mostly in the field of Computer Vision and Robotics. Antonios Gasteratos has been a member of the programme committee in numerous international conferences and chairman and co-chairman in several international conferences and workshops.