# Kronecker Products-based Regularized Image Interpolation Techniques

Blas Trigueros, Ricardo H. Castañeyra, Domingo Rodríguez
Electrical and Computer Engineering Department, University of Puerto Rico, Mayagüez, Puerto Rico
{blas.trigueros, ricardo.castaneyra, domingo}@ece.uprm.edu

## Introduction

In the world of Image Processing the need of render a high-resolution (HR) image from a low-resolution (LR) image is very common. This procedure is well known as image interpolation and is based on approximate the missing pixels information from the surroundings of the low-resolution image. Image interpolation is widely used on the different areas of imaging, computer vision, digital photography, surveillance systems, etc. That is the reason of the diverse algorithms proposed to solve the image interpolation problem such as cubic spline or adaptive spline interpolation techniques. However, most of these methods are heuristics and based on a pixel-by-pixel basis.

In this work, we have implemented on the MATLAB Parallel Computing Toolbox the algorithm proposed by Li Chen and Kim-Hui Yap[1] to accomplished image interpolation. This algorithm is based on *Tikhonov* regularization technique [2] and tries to find a regularized solution to image interpolation by solving a damped least square optimization problem. Furthermore, Kronecker products [3] and singular value decomposition (SVD) [2, 4] are employed in this method in order to reduce the computational cost of the algorithm. By using Parallel MATLAB, the performance of the algorithm can be improved taking advantage of parallel array programming over a multicore computer.

## Problem formulation

The relationship between the LR and the HR images can be modeled in matrix-vector formulation as follows:

$$\boldsymbol{g} = \boldsymbol{DHf} + \boldsymbol{n}, \qquad (1)$$

where **g** represents the LR image observed, **f** represent the HR image to be estimated in our case, and finally **n** is the additive noise vector. Matrix **H** represent the blurring phenomena during the imagine acquisition caused by the lens or imaging systems and **D** is the decimation matrix that represents the decimation process.

Given the observed LR image **g**, the problem of estimating the HR image **f** can be formulated as a least square problem. Using the *Tikhonov* regularization technique, **f** will be the solution to:

$$\mathbf{min}\|\mathbf{DHf} - \mathbf{g}\|_2^2 + \lambda\|\mathbf{Ef}\|_2^2 \qquad (2)$$

but this calculation implies an extreme computational effort due to the high dimensionality of the matrices involved.

The least square problem can be simplified using SVD and assuming that the matrix H can be decomposed into the Kronecker product of two matrices [2, 5]:

$$\boldsymbol{H} = \boldsymbol{H_1} \otimes \boldsymbol{H_2} \qquad (3)$$

Since the decimation matrix **D** can be also decomposed, the product **DH** can be simplified using the properties of Kronecker products:

$$\boldsymbol{D} = \boldsymbol{D_1} \otimes \boldsymbol{D_2} \qquad (4)$$

$$\boldsymbol{DH} = (\boldsymbol{D_1} \otimes \boldsymbol{D_2})(\boldsymbol{H_1} \otimes \boldsymbol{H_2}) = (\boldsymbol{D_1H_1} \otimes \boldsymbol{D_2H_2}) \quad (5)$$

The expression above can be further simplified computing the SVD of $\mathbf{D_1H_1}$ and $\mathbf{D_2H_2}$:

$$\boldsymbol{D_1H_1} = \boldsymbol{U_1}[\Sigma_1 \,|\, \mathbf{0}]\boldsymbol{V_1^T} \qquad (6)$$

$$\boldsymbol{D_2H_2} = \boldsymbol{U_2}[\Sigma_2 \,|\, \mathbf{0}]\boldsymbol{V_2^T} \qquad (7)$$

Using these simplifications, the least square problem can be reformulated as follows:

$$\boldsymbol{min}\left\|\begin{bmatrix}\left([[\Sigma_1|\mathbf{0}]\otimes[\Sigma_2|\mathbf{0}]]\right)\\ \sqrt{\boldsymbol{\lambda}}\boldsymbol{I}\end{bmatrix}\boldsymbol{y} - \begin{bmatrix}\boldsymbol{z}\\ \mathbf{0}\end{bmatrix}\right\|_2^2 \quad (8)$$

Where $\boldsymbol{y} = (\boldsymbol{V_1} \otimes \boldsymbol{V_2})^T\boldsymbol{f}$ $\qquad \boldsymbol{z} = (\boldsymbol{U_1} \otimes \boldsymbol{U_2})^T\boldsymbol{g}$

Resulting on the algorithm to be implemented:

$$\boldsymbol{z} = \left(\boldsymbol{U_2^T}(\boldsymbol{g})^{-s}\boldsymbol{U_1}\right)^{s} \qquad (9)$$

$$\boldsymbol{y} = \left(\begin{bmatrix}\Sigma_1\\ \mathbf{0}\end{bmatrix} \otimes \begin{bmatrix}\Sigma_2\\ \mathbf{0}\end{bmatrix}\right)\left([\Sigma_1^2] \otimes [\Sigma_2^2] + \boldsymbol{\lambda I}\right)^{-1}\boldsymbol{z} \quad (10)$$

$$\boldsymbol{f} = \left(\boldsymbol{V_2}(\boldsymbol{y})^{-s}\boldsymbol{V_1^T}\right)^{s} \qquad (11)$$

If the dimension of the HR image is *dM x dM*, the dimension of the matrices $\boldsymbol{U_1}$, $\boldsymbol{U_2}$, $\boldsymbol{\Sigma_1}$ and $\boldsymbol{\Sigma_2}$ will be *M x M*. Hence, the algorithm reduces significantly the computational cost with respect to the direct solution to (2).

As can be noticed from the algorithm represented above, the only unknown is the variable $\boldsymbol{\lambda}$. This $\boldsymbol{\lambda}$ is the *Tikhonov* regularization parameter and can be found by minimizing the generalized cross-validation (GCV) function [1]:

$$GCV(\boldsymbol{\lambda}) = \frac{\|(\boldsymbol{I} - \boldsymbol{DH}(\boldsymbol{DH})^{+})\boldsymbol{g}\|}{[\boldsymbol{trace}(\boldsymbol{I} - (\boldsymbol{DH})(\boldsymbol{DH})^{+})]^2} \qquad (12)$$

where $(DH)^{+} = ((DH)^T(DH) + \boldsymbol{\lambda I})^{-1}(DH)^T$

A simplified form of the GCV is obtained when Kronecker product and SVD are employed, resulting on:

$$GCV(\boldsymbol{\lambda}) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\left(\frac{\lambda z_{iM+j}}{(\alpha_i\beta_j)^2 + \lambda}\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{M}\left(\frac{\lambda}{(\alpha_i\beta_j)^2 + \lambda}\right)^2} \qquad (13)$$

Where $\boldsymbol{\alpha_i}$ and $\boldsymbol{\beta_j}$ are the *i-th* singular value of $\boldsymbol{\Sigma_1}$ and $\boldsymbol{\Sigma_2}$, respectively, and $\boldsymbol{z_i}$ is the *i-th* entry of the column vector $\boldsymbol{z}$.

## Parallel implementation

In order to improve the performance of the proposed algorithm in terms of time computation, we used the MATLAB Parallel Computing Toolbox for the implementation. This toolbox allows solving computationally and data intensive problems on multicore and computer clusters. Parallelization can be performed by using some parallel constructors such as parallel for-loops, parallel code blocks or distributed arrays. This parallelization can be implemented at a high level with minimum changes of the serial code. The interpolation algorithm has been implemented on the Parallel Toolbox by using distributed arrays to store the large matrices involved. This implementation simulates a parallel environment by partitioning the data across multiple MATLAB sessions.

## Results

In this experiment, the algorithm proposed on [1] was implemented on MATLAB using the Parallel Computing Toolbox to compare the time elapsed on HR image interpolation technique in serial and parallel architectures. The value of the regularization parameter was chosen arbitrary in order to achieve a satisfactory result. The image in Fig. 1(a) was used to perform tests on the algorithm implemented on both MATLAB and parallel MATLAB. In all cases the decimation rate was *d=2* and the blurring low pass filter coefficients were $h = u^T v$, (*u=v=[0.3 0.4 0.3]*). The results can be viewed on Table 1, where there is a notable diminution on the time elapsed to achieve the HR image from a LR image when the HR image is larger than 1024 pixels. Running the algorithm to achieve a 2048x2048 HR image, the difference of serial and parallel implementation were approximately 28 seconds faster when using parallel MATLAB. Also, is notable to mention that when the HR image is 256 or 512, the serial architecture performs faster.

## Conclusion

The implementation of the algorithm based on Tikhonov regularization [1] was implemented satisfactorily on the Parallel Computing Toolbox of MATLAB, resulting on a faster parallel performance when the resolution of the HR image is 1024x1024 or higher. The serial architecture implementation results on faster HR image interpolation when the images are 512x512 or less.

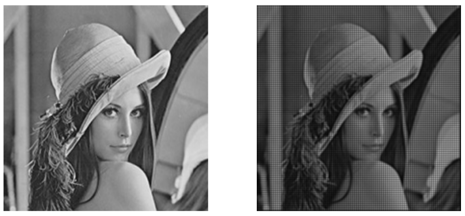
(a) (b)

**Fig1: Interpolation results: (a) LR image, (b) HR reconstructed image.**

**Table 1:**
**Comparison of the algorithm elapsed time on different architectures**

| Implementation Method | Image Size – Lena.jpg | | | |
|---|---|---|---|---|
| | 256x256 | 512x512 | 1024x1024 | 2048x2048 |
| | Time (s) | Time (s) | Time (s) | Time (s) |
| **Matlab (serial)** | 0.084 | 0.64 | 7.78 | 113.11 |
| **Parallel Matlab** | 0.71 | 2.45 | 7.68 | 84.39 |

## References

[1] Li Chen and Kim-Hui Yap, “Regularized Interpolation Using Kronecker Product for Still Images,” *IEEE International Conference on Image Processing*, Vol. 2, pp. 1014-17, Sep 2005.

[2] Julie Kamm and James G. Nagy, “Kronecker product and SVD approximations in image restoration,” *Linear Algebra and its Applications*, Vol. 284, pp. 177-192, Jan 1998.

[3] Charles F. and Van Loan, “The ubiquitous Kronecker product,” *Journal of Computational and Applied Mathematics*, Vol. 123, pp. 85-100, Oct 1999.

[4] Tuan Cao-Huu and Claude Évéquoz, “Singular value decomposition transform with an FFT-based algorithm on the Connection Machine CM5,” *Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 1046-1049, Sep 1995.

[5] S. Rjasanow, “Effective Algorithms With Circulant-Block Matrices”, Linear Algebra and its applications 202, pp. 55-69, Elsevier Science, 1994.