# Multiplierless Mumford and Shah functional implementation

Maurizio Martina, and Guido Masera

The authors are with CERCOM (Center for Multimedia Radio Communications) - Dipartimento di Elettronica - Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy. maurizio.martina@polito.it, guido.masera@polito.it. This work is supported by the NEWCOM++ NOE. Contact author: Maurizio Martina, CERCOM - Dipartimento di Elettronica - Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy. Tel. +39 011 564 4205, maurizio.martina@polito.it

# Multiplierless Mumford and Shah functional implementation

**Abstract**

This paper proposes the implementation of the Mumford and Shah functional without using complex operations as multiplications and divisions. Our goal is to show that the achieved results in terms of performance/complexity trade-off are well suited for video applications of the Mumford and Shah functional, such as motion estimation based on segmentation techniques. To this purpose two implementations, with and without multiplications, have been developed and ported on a DSP board able to get frames from a camera and to play out the results on a standard VGA monitor: reported results show a relative speed-up of a factor 3 for the multiplierless version with no visual quality degradation.

**Index Terms**

Image Segmentation, Mumford and Shah functional, Performance Evaluation, DSP

## I. Introduction

During the last years several image and video standards (e.g. JPEG 2000, MPEG4, H.264, ...) have been proposed to increase the quality and the availability of multimedia services. One of the major novelties of the most recent video standards concerns the motion estimation. Many works propose the use of image segmentation instead of rectangular blocks of pixels to recognize and track object motion (e.g. [16], [25]). Among the proposed approaches an interesting technique to achieve both image segmentation and regularization is the Mumford and Shah functional [23]. In [6], [7] and [8] a novel interpretation of the optic flow has led to an extension of the Mumford and Shah functional to motion segmentation. This approach named Motion Competition is intended to join motion estimation and segmentation to derive a variational approach for the segmentation of the image domain into regions of homogeneous motion.

As far as the Mumford and Shah functional implementation is concerned, several works deal with iterative algorithms [4], [18], [17], [9], [32], [31] and [15]. Furthermore, since the Mumford and Shah functional is very computationally intensive, different solutions have been proposed to tackle its complexity [4], [17], [9], [32], [31]. In particular, as the computation of the Mumford and Shah functional is based on iterative algorithms, the greatest part of the proposed studies focused on reducing the number of required iterations.

The multigrid approach is an interesting technique to speed-up iterative methods for solving elliptic problems. The basic idea behind multigrid techniques is to find first a coarse solution to the problem, solving it on a coarse mesh and then employ this solution to refine the original problem on a fine mesh. This approach has been successfully employed for the Mumford and Shah functional in [9] where a noteworthy speed-up is achieved reducing the number of iterations.

Other solutions are based on the Steepest Descent (e.g. [4] and [31]) to reduce the number of iterations. Besides the Preconditioned Conjugate Gradient method is a very effective technique for solving sparse linear equation systems as $Ax = b$. It is based on preconditioning the so called *A-orthogonality* that improves the convergence to the solution employing *A-orthogonal* search directions $d_{(i)}$ ($d_{(i)}^T A d_{(j)} = 0$). In the last years its use to speed-up the Mumford and Shah functional implementation has been addressed [15], [14] showing interesting results.

To speed-up the convergence of the Mumford and Shah functional it is crucial to consider not only the number of iterations, but also the operations required on pixels to perform segmentation and regularization. In fact the Mumford and Shah functional requires several multiplications and two divisions, that cannot be reduced to simple shift operations. The aim of this work is to reduce the "arithmetic" complexity of the Mumford and Shah functional, independently of the technique employed to reduce the number of iterations, by performing the data processing in the logarithmic domain. This change of domain reduces multiplications and divisions into simple additions and subtractions [22]. However, this complexity reduction comes at the expense of some approximations, that are investigated in this work. The aforementioned techniques, such as multigrid, Steepest Descent, Preconditioned Conjugate Gradient, can be jointly employed with the arithmetic optimizations proposed in this work to achieve larger speed-up.

A simple numerical iterative solution based on non-linear Gauss-Seidel method [18] is employed to show the speed-up granted by the proposed method. Moreover, to prove the effectiveness of the proposed method, both a standard implementation with multiplications and divisions and the proposed multiplierless approach were ported on a DSP board for image and video processing, the Texas Instruments Image Developers Kit (IDK) [28]. Performance results obtained for both the implementations are shown in section V. Section II is devoted to briefly summarize the theoretical framework, whereas sections III and IV deal with the proposed approach. Finally in section VI some conclusions are drawn.

## II. Theoretical Framework

The Mumford and Shah approach is based on a mathematical model that considers the segmentation problem as a partition of the image domain $\Omega \subset \mathbf{R}^2$ in open subsets $\Omega_i$. Given an image, whose intensity function is defined as $g : \Omega \to \mathbf{R}$, the Mumford and Shah functional [23] aims to find a smooth approximation $u$ of $g$ in each sub-domain $\Omega_i$:

$$E(u, K) = \int_{\Omega} (u - g)^2 dx + \alpha \int_{\Omega \setminus K} |\nabla u|^2 \, dx + \beta \parallel K \parallel \tag{1}$$

where $K$ is the set of $\Omega_i$ boundaries and $\parallel K \parallel$ denotes $K$ set length, $\alpha$ is a parameter related to the scale [31], $\beta$ is a parameter related to the contrast [31], $x = (x_1, x_2)$ are the image coordinates, $dx$ is the Lebesgue measure in the plane and

$$|\nabla u|^2 = \left| \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} \right|^2 . \tag{2}$$

It has been demonstrated that the Mumford and Shah functional admits a solution (e.g. [12], [10], [3], [11]); however a big advance in solving it came from the $\Gamma$-convergence concept. In [1] Ambrosio and Tortorelli demonstrated

that the functional described in (1) can be approximated, in the $\Gamma$-convergence sense by

$$E_\varepsilon(u,z) = \int_\Omega (u-g)^2 + \alpha z^2 \left|\nabla u\right|^2 + \beta \left[\varepsilon \left|\nabla z\right|^2 + \frac{(1-z)^2}{4\varepsilon}\right] dx \tag{3}$$

where the function $z$ yields an approximate description of the set of curves $K$ and

$$|\nabla z|^2 = \left|\frac{\partial z}{\partial x_1} + \frac{\partial z}{\partial x_2}\right|^2. \tag{4}$$

Moreover, they demonstrated that the family of functionals $E_\varepsilon(u,z)$ converges to the functional $E(u,K)$ in the sense of $\Gamma$-convergence as $\varepsilon \to 0$. Thus, the solution obtained by minimizing $E_\varepsilon(u,z)$ tends to the solution obtained by minimizing $E(u,K)$. As (3) is an elliptic problem its minimizers satisfy the Euler-Lagrange differential equation both for $u$ and $z$. So that we need to:

1) develop $u$ and $z$ Euler-Lagrange equations;
2) discretize $g$, $u$ and $z$ on a uniform grid of $Rows \times Cols$ nodes with mesh size $h$ - thus they become arrays $g_{i,j}, u_{i,j}, z_{i,j}$.

As a consequence, we obtain a nonlinear system of equations for the $2 \cdot Rows \cdot Cols$ unknown values $u_{i,j}, z_{i,j}$. Employing an iterative algorithm, as the nonlinear Gauss-Seidel method [18] with $h = 1$, we obtain

$$u_{i,j} = \frac{\alpha \tilde{u}_{i,j} + g_{i,j}}{1 + \alpha\,(\mathbf{z^2})}, \qquad z_{i,j} = \frac{4\hat{z}_{i,j} + p^2}{16 + 4\frac{\alpha p}{\beta}\left|\nabla u\right|^2_{i,j} + p^2} \tag{5}$$

where

$$\mathbf{z^2} = z^2_{i+1,j} + z^2_{i-1,j} + z^2_{i,j+1} + z^2_{i,j-1} \tag{6}$$

$$\tilde{u}_{i,j} = z^2_{i+1,j}u_{i+1,j} + z^2_{i-1,j}u_{i-1,j} + z^2_{i,j+1}u_{i,j+1} + z^2_{i,j-1}u_{i,j-1} \tag{7}$$

$$\hat{z}_{i,j} = z_{i+1,j} + z_{i-1,j} + z_{i,j+1} + z_{i,j-1} \tag{8}$$

$$p = \frac{1}{\varepsilon} \tag{9}$$

From (5) it can be observed that noteworthy computational complexity is required to compute $u_{i,j}$ and $z_{i,j}$ as they require several multiplications and two divisions. For a complete formal derivation of (5) the reader can refer to [18].

## III. Multiplierless formulation

In order to avoid the use of multiplications and divisions, (5) can be ported into the logarithmic domain. This porting is also routinely adopted in several channel decoding algorithms such as turbo [26] and LDPC decoders [33]. However, to really exploit the advantage of working into the logarithmic domain we ought to reduce as much as possible the number of conversions from the linear to the logarithmic domain and vice-versa from the logarithmic to the linear one. In channel decoding applications the whole processing can be kept in the logarithmic domain and there is no need for a final conversion to the linear domain. In the evaluation of the Mumford and Shah functional the need for converting back processing outputs to the linear domain depends on the applications: for example this

conversion is necessary in restoration applications [4] but not in edge detection [30]. Let's apply the logarithm to the left and to right sides of (5).

$$\begin{aligned}
\ln u_{i,j} &= \ln N_{u_{i,j}} - \ln D_{u_{i,j}} && (10)\\
&= \ln\left(\alpha\tilde{u}_{i,j} + g_{i,j}\right) - \ln\left[1 + \alpha\left(z_{i+1,j}^2 + z_{i-1,j}^2 + z_{i,j+1}^2 + z_{i,j-1}^2\right)\right] \\
\ln z_{i,j} &= \ln N_{z_{i,j}} - \ln D_{z_{i,j}} && (11)\\
&= \ln\left(4\hat{z}_{i,j} + p^2\right) - \ln\left(16 + 4\frac{\alpha p}{\beta}\left|\nabla u\right|_{i,j}^2 + p^2\right)
\end{aligned}$$

where

$$4\left|\nabla u\right|_{i,j}^2 = \left(u_{i+1,j} - u_{i-1,j}\right)^2 + \left(u_{i,j+1} - u_{i,j-1}\right)^2 \quad (12)$$

For the sake of simplicity we can analyze separately the four terms $\ln N_{u_{i,j}}$, $\ln D_{u_{i,j}}$, $\ln N_{z_{i,j}}$ and $\ln D_{z_{i,j}}$.

$$\begin{aligned}
\ln N_{u_{i,j}} &= \ln\left(\alpha\tilde{u}_{i,j} + g_{i,j}\right) = \ln\left(e^{\ln\alpha + \ln\tilde{u}_{i,j}} + e^{\ln g_{i,j}}\right) && (13)\\
\ln D_{u_{i,j}} &= \ln\left[1 + \alpha\left(z_{i+1,j}^2 + z_{i-1,j}^2 + z_{i,j+1}^2 + z_{i,j-1}^2\right)\right] && (14)\\
&= \ln\left[e^0 + e^{\ln\alpha + 2\ln z_{i+1,j}} + \ldots + e^{\ln\alpha + 2\ln z_{i,j-1}}\right] \\
\ln N_{z_{i,j}} &= \ln\left(4\hat{z}_{i,j} + p^2\right) = \ln\left(e^{\ln 4 + \ln\hat{z}_{i,j}} + e^{2\ln p}\right) && (15)\\
\ln D_{z_{i,j}} &= \ln\left(16 + 4\frac{\alpha p}{\beta}\left|\nabla u\right|_{i,j}^2 + p^2\right) && (16)\\
&= \ln\left(e^{\ln 16} + e^{\ln\alpha + \ln p - \ln\beta + \ln 4\left|\nabla u\right|_{i,j}^2} + e^{2\ln p}\right)
\end{aligned}$$

where

$$\begin{aligned}
\ln\tilde{u}_{i,j} &= \ln\left(z_{i+1,j}^2 u_{i+1,j} + z_{i-1,j}^2 u_{i-1,j} + z_{i,j+1}^2 u_{i,j+1} + z_{i,j-1}^2 u_{i,j-1}\right) && (17)\\
&= \ln\left(e^{2\ln z_{i+1,j} + \ln u_{i+1,j}} + \ldots + e^{2\ln z_{i,j-1} + \ln u_{i,j-1}}\right) \\
\ln\hat{z}_{i,j} &= \ln\left(z_{i+1,j} + z_{i-1,j} + z_{i,j+1} + z_{i,j-1}\right) && (18)\\
&= \ln\left(e^{\ln z_{i+1,j}} + \ldots + e^{\ln z_{i,j-1}}\right) \\
\ln 4\left|\nabla u\right|_{i,j}^2 &= \ln\left(e^{2\ln\left|u_{i+1,j} - u_{i-1,j}\right|} + e^{2\ln\left|u_{i,j+1} - u_{i,j-1}\right|}\right) && (19)
\end{aligned}$$

and

$$\begin{aligned}
\ln\left|u_{i+1,j} - u_{i-1,j}\right| &= \ln\left|e^{\ln u_{i+1,j}} - e^{\ln u_{i-1,j}}\right| && (20)\\
\ln\left|u_{i,j+1} - u_{i,j-1}\right| &= \ln\left|e^{\ln u_{i,j+1}} - e^{\ln u_{i,j-1}}\right| && (21)
\end{aligned}$$

As it can be observed we can implement the Mumford and Shah functional building an algorithm that

- evaluates the logarithms of $g_{i,j}$ and $u_{i,j}$, $z_{i,j}$ initial values
- performs operations like $\ln\left|e^{\delta_1} \pm e^{\delta_2}\right|$ for a certain number of iterations

It is noticeable that the values for $\alpha$, $\beta$ and $p$ ought to be expressed in the logarithmic domain too. Furthermore, depending on the application, results (i.e. $u$ and $z$) could be converted back into the linear domain. The strength of

this solution stems from computing the logarithms only on $g_{i,j}$ and $u_{i,j}$, $z_{i,j}$ initial values, then the computation is performed in the logarithmic domain till the final results are achieved.

As the logarithm goes to infinity when the pixel value is zero, we ought to avoid zero values. Since each pixel is represented on 8-bits in $[0, 1]$ the smallest non–zero value is $2^{-8}$. As a consequence, zero can be approximated as a value $MS\_EPS < 2^{-8}$. For a fixed point implementation a simple solution is choosing $MS\_EPS = 2^{-9}$, that means having at least 9 bits to represent the fractional part of a sample.

## IV. The $\max^*$ and $\max^-$ functions

As it is well known in the field of channel codes (e.g. turbo codes) [26], [24] the following equality holds true

$$\ln\left(e^{\delta_1} + e^{\delta_2}\right) = \max\left(\delta_1, \delta_2\right) + \ln\left(1 + e^{-|\delta_2 - \delta_1|}\right) \tag{22}$$

In particular for turbo codes [2] it has been proved that (22) can be simply implemented building a look–up–table (LUT) storing the term $\ln\left(1 + e^{-|\delta_2 - \delta_1|}\right)$. The LUT entries and the values stored into the LUT depend on the precision required by the system, as detailed in the following paragraphs. Said $r$ the number of bits required to appreciate the term $|\delta_2 - \delta_1|$ we can find the number of entries for the LUT by solving the following inequality for $m$:

$$\ln\left(1 + e^{-\frac{m}{2^r}}\right) \leq 2^{-(r+1)} \tag{23}$$

Similarly holds true

$$\ln\left|e^{\delta_1} - e^{\delta_2}\right| = \max\left(\delta_1, \delta_2\right) + \ln\left(1 - e^{-|\delta_2 - \delta_1|}\right) \tag{24}$$

Using the same approach we can approximate $\ln\left(1 - e^{-|\delta_2 - \delta_1|}\right)$ solving

$$\ln\left|1 - e^{-\frac{n}{2^r}}\right| \leq 2^{-(r+1)} \tag{25}$$

It is noticeable that the term $\ln\left(1 + e^{-|\delta_2 - \delta_1|}\right)$ takes values into a limited range, in fact

$$|\delta_2 - \delta_1| \in [0, \infty) \rightarrow \ln\left(1 + e^{-|\delta_2 - \delta_1|}\right) \in (0, \ln 2]. \tag{26}$$

So that given a certain accuracy (e.g. $2^{-r}$) the logarithm can be easily approximated. On the other hand the term $\ln\left(1 - e^{-|\delta_2 - \delta_1|}\right)$ takes values into an unlimited range, in fact

$$|\delta_2 - \delta_1| \in [0, \infty) \rightarrow \ln\left(1 - e^{-|\delta_2 - \delta_1|}\right) \in (-\infty, 0). \tag{27}$$

In order to map this term in a small LUT we ought not only to select a certain accuracy (e.g. $2^{-r}$), but also to limit the co-domain. Since the function approaches $-\infty$ as $|\delta_2 - \delta_1| \rightarrow 0$, we can force $|\delta_2 - \delta_1| \in [l, \infty)$. To obtain a hardware friendly value, we impose that $l$ is a power of two: $l = 2^{-EPS}$. As for samples, we approximate the zero with $l = MS\_EPS$, namely we select $EPS = 9$ so that the we approximate $-\infty$ with $\ln 2^{-9}$.

In the literature the function obtained approximating (22) by means of a LUT is usually referred to as $\max^*$. In the following we will refer to the function obtained approximating (24) with a LUT as $\max^-$. Employing the

$\max^*$ and the $\max^-$ operators we can reformulate the multiplierless solution as:

$$\ln N_{u_{i,j}} = \overset{*}{\max}(\ln \alpha + \ln \tilde{u}_{i,j}, \ln g_{i,j}) \quad (28)$$

$$\ln D_{u_{i,j}} = \overset{*}{\max}(\overset{*}{\max}(\overset{*}{\max}(\overset{*}{\max}(0, q_{i+1,j}), q_{i-1,j}), q_{i,j+1}), q_{i,j-1}) \quad (29)$$

$$\ln N_{z_{i,j}} = \overset{*}{\max}(\ln 4 + \ln \hat{z}_{i,j}, 2 \ln p) \quad (30)$$

$$\ln D_{z_{i,j}} = \overset{*}{\max}(\overset{*}{\max}(\ln 16, \ln \alpha + \ln p - \ln \beta + \ln 4 \, |\nabla u|^2_{i,j}), 2 \ln p) \quad (31)$$

$$\ln \tilde{u}_{i,j} = \overset{*}{\max}(\overset{*}{\max}(\overset{*}{\max}(s_{i+1,j}, s_{i-1,j}), s_{i,j+1}), s_{i,j-1}) \quad (32)$$

$$\ln \hat{z}_{i,j} = \overset{*}{\max}(\overset{*}{\max}(\overset{*}{\max}(\ln z_{i+1,j}, \ln z_{i-1,j}), \ln z_{i,j+1}), \ln z_{i,j-1}) \quad (33)$$

$$\ln 4 \, |\nabla u|^2_{i,j} = \overset{*}{\max}(2 \, \bar{\max}(\ln u_{i+1,j}, \ln u_{i-1,j}), 2 \, \bar{\max}(\ln u_{i,j+1}, \ln u_{i,j-1})) \quad (34)$$

with

$$q_{i,j} = \ln \alpha + 2 \ln z_{i,j} \quad (35)$$

$$s_{i,j} = 2 \ln z_{i,j} + \ln u_{i,j} \quad (36)$$

$$\overset{*}{\max}(\delta_1, \delta_2) = \max(\delta_1, \delta_2) + \ln \left(1 + e^{-|\delta_2 - \delta_1|}\right) \quad (37)$$

$$\bar{\max}(\delta_1, \delta_2) = \max(\delta_1, \delta_2) + \ln \left(1 - e^{-|\delta_2 - \delta_1|}\right) \quad (38)$$

Experimental results show that $r$ can be kept small accepting a certain quality loss with respect to a standard solution that implements (5). As suggested in [20] a fixed point implementation of (5) can grant near floating point performance employing 16 bits to represent fractional values; in the following samples of code this will be referred to as `1 << 16`. As an example, Fig. 1 and 2 show the quality obtained on $u$ (regularized image) and $z$ (segmented image) for a fixed point implementation with 16 bits for fractional values, $r = 6$ and $r = 8$ respectively. As it can be observed, comparing the original Mumford and Shah results ($u^{(org)}$ and $z^{(org)}$) with the proposed multiplierless version ($u^{(r)}$ and $z^{(r)}$) the quality loss for $r = 8$ is negligible: we obtain a Mean Square Error (MSE) of $6.12 \times 10^{-6}$, corresponding to a Peak Signal to Noise Ratio (PSNR) of about 52 dB, for $u$; similarly we have MSE=$4.94 \times 10^{-5}$ (PSNR=43 dB) for $z$. However, from (23) and (25) we find $m = n = 2559$. On the other hand, we obtain $m = n = 511$ with $r = 6$ by slightly reducing the quality of the result: MSE=$1.61 \times 10^{-4}$ (PSNR=38 dB) and MSE=$7.13 \times 10^{-4}$ (PSNR=32 dB) for $u$ and $z$ respectively. It is worth pointing out that the proposed methodology does not make any hypothesis on the base employed for the logarithm $B_{log}$. In order to make the proposed methodology more hardware oriented we prefer to choose $B_{log} = 2$. In fact, as detailed in [22], hardware implementation of multiplications and divisions gets significant simplifications by using binary logarithms. So that the results shown in this paper have been obtained selecting $B_{log} = 2$.

[Fig. 1 about here.]

[Fig. 2 about here.]

It is worth pointing out that the proposed logarithmic, multiplierless Mumford and Shah functional implementation

can be applied independently of the $\alpha$, $\beta$ and $p$ values. To this purpose three sets of parameter values are considered to show the impact of $r$ on $u_{i,j}$ and $z_{i,j}$. The parameter values selected are compatible with the ones suggested in [5], [4] and [21]. As a typical case ($A$) in the following we will show results obtained with $\alpha = 9$, $\beta = 0.0039$, and $p = 20$ with 20 iterations. Moreover, we consider the best and worst cases with respect to the ranges studied in [21], namely $\alpha \in [1, 64]$, $\beta \in [1/256, 1/2]$ and $p \in [2, 32]$. As it can be inferred from (5), one case is $\alpha = 64$, $\beta = 1/256$ and $p = 32$ ($B$), the other is $\alpha = 1$, $\beta = 1/2$ and $p = 2$ ($C$). For both of them the number of iterations is 20 - as for the case $A$. In order to better quantify the performance of the proposed approach, simulations on several QCIF ($144 \times 176$ pixels) frames, namely "foreman", "carphone", "claire", "grand-mother", "miss-america", "mother-and-daughter" and "salesman" have been performed. For each frame, floating point results ($u^{(org)}$,$z^{(org)}$), obtained from a standard Mumford and Shah fuctional implementation, have been compared against the fixed point ones obtained varying $r$ from 3 to 16 bits ($u^{(r)}$,$z^{(r)}$) in the proposed multiplierless implementation. In Fig. 3 the LUT entries ($m$ or $n$) required to build the LUTs that approximate the $\max^*$ and $\max^-$ operators are shown; the $m$ (or $n$) values required by $r = 6$ and $r = 8$ are highlighted with the diamond and the star symbols respectively. The procedure to find $m$ (or $n$) and the values to fill the LUTs is given at the end of the paragraph. As it can be inferred from Fig. 1, 2 and 3 an acceptable trade-off between complexity and accuracy is achieved by selecting $r = 6$. In Fig. 4 the distribution of the absolute error, namely the distribution of $u^{(org)} - u^{(r)}$ and $z^{(org)} - z^{(r)}$, obtained by comparing $u^{(org)}$, $z^{(org)}$ with $u^{(r)}$, $z^{(r)}$ ($r = 6$ and $r = 8$) in the test-cases $A$, $B$ and $C$ is shown. Finally in Fig. 5 four curves show the behavior of the error mean value $\mu(u_{i,j}^{(org)} - u_{i,j}^{(r)})$, $\mu(z_{i,j}^{(org)} - z_{i,j}^{(r)})$ and variance $\sigma^2(u_{i,j}^{(org)} - u_{i,j}^{(r)})$, $\mu(z_{i,j}^{(org)} - z_{i,j}^{(r)})$ as a function of $r$, for the test-cases $A$, $B$ and $C$. To further highlight the performance of the proposed multiplierless implementation in Fig. 6 we show the Dice similarity coefficient [13] as a function of $r$ for the segmented images ($z$) obtained with the proposed multiplierless solution (with and without logarithmic to linear conversion) compared with the standard Mumford and Shah functional implementation. As the Dice similarity coefficient is one of the intensity-based overlap indices for binary images, we first binarized $z^{(org)}$ and $z^{(r)}$ (with and without log-lin conversion), then according to [27] we obtained

$$\mathrm{Dice}(z^{(org)}, z^{(r)}) = \frac{2a}{2a + b + c} \tag{39}$$

where $a$ is the number of corresponding pixels of value '1' in both binary images, $b$ is the number of pixels taking value '1' only in $z^{(org)}$ and $c$ is the number of pixels taking value '1' only in $z^{(r)}$. Stemming from (39) we can infer that the Dice similarity coefficient is 1 if objects in the two binary images overlap perfectly, whereas it is 0 if there is no overlap. Even if "similarity" is application dependent, according to [34] a Dice similarity coefficient value greater than 0.7 indicates excellent agreement. Considering the results presented in Fig. 6 we can infer that both multiplierless implementations (with and without log-lin conversion) of $z^{(r)}$ show excellent agreement with $z^{(org)}$ for all the seven test images in the three test cases $A$, $B$ and $C$.

Finally, the procedure to set-up these two LUTs can be run off-line as:

```
#define MS_EPS pow(2,-9) // zero approximated as 2^-9
#define r 6              // precision bits
```

```
m = 0;                  // start m search
while ( log2(1 + exp2(-m/(1 << r))) > pow(2,-(r+1)) )
  m++;
m--;
err_Tab1 = malloc(sizeof(int)*m); // allocating the max* LUT
for (i=0; i<m; i++)               // filling the max* LUT
  err_Tab1[i] = floor(log2(1+exp2(-(float)i/(1 << r)))*(1 << 16) + 0.5);

  // if n=0 we use MS_EPS
n = 1;                  // start n search
while ( fabs(log2(1 - exp2(-n/(1 << r)))) > pow(2,-(r+1)) )
  n++;
n--;
err_Tab2 = malloc(sizeof(int)*n); // allocating the max- LUT
err_Tab2[0] = floor(log2(MS_EPS)*(1 << 16) + 0.5); // using MS_EPS
for (i=1; i<n; i++)               // filling the max- LUT
  err_Tab2[i] = floor(log2(1-exp2(-(float)i/(1 << r)))*(1 << 16) + 0.5);
```

[Fig. 3 about here.]

[Fig. 4 about here.]

[Fig. 5 about here.]

[Fig. 6 about here.]

## V. DSP IMPLEMENTATION AND EXPERIMENTAL RESULTS

To have real data concerning the speed-up granted by the proposed multiplierless implementation with respect to the direct implementation of (5), two fixed point C models have been ported on a Texas Instruments DSP board [19]. The Texas Instruments Image Developers Kit [28] is based on a board equipped with a TMS320C6711 DSP running at 150 MHz with 16 MBytes of SDRAM and a daughter-card able to capture frames and to display results on a standard monitor through a VGA connector. In order to obtain experimental results, the following environment has been set-up: each frame captured by the camera is filtered and subsampled to obtain a QCIF. Each pixel of the original image $g_{i,j}$ is represented on 8 bits in the range $[0, 1]$; $u_{i,j}$ is initialized with the original pixels $g_{i,j}$ whereas $z_{i,j}$ is initialized as $z_{i,j} = 1$ with $\{i, j\} \in \{[0, Rows - 1], [0, Cols - 1]\}$. The first operation concerns $g_{i,j}$, $u_{i,j}$ and $z_{i,j}$ initialization. Since each captured frame is made of 8 bits pixels ($g\_char_{i,j}$) we force $g_{i,j}$, $u_{i,j}$ and $z_{i,j}$ to be `int` values (32 bits) in [0,1] with 16 bits for the fractional part. This choice is more than sufficient to represent $MS\_EPS$ as defined in section III.

```
for (i=0; i<R; i++)
{
  for (j=0; j<C; j++)
  {
    g[i][j] = (unsigned int)floor( ((float)g_char[i][j]/255) *
                                   (float)(1 << 16) + 0.5);
    u[i][j] = g[i][j];
```

```
    z[i][j] = 1 << 16; // 16 fractional bits
  }
}
```

As far as the multiplierless implementation is concerned, after this operation we should map the linear values to their logarithmic counterparts. Exploiting $g\_char_{i,j}$ 8 bits representation, we use a 256 entries LUT that performs in a single step the initialization and the logarithmic conversion; moreover we can easily set $z_{i,j} = 0$:

```
len = R*C;
for (i=0; i<R; i++)
{
  for (j=0; j<C; j++)
  {
    g_log[i][j] = log_tab[g_char[i][j]];
    u_log[i][j] = log_tab[g_char[i][j]];
  }
}
memset(z_log,0,sizeof(int)*len);
```

where the `log_tab` can be computed off-line as:

```
#define MS_EPS pow(2,-9) // zero approximated as 2^-9
#define LEV 255          // maximum pixel value

  // if 0 we use MS_EPS
log_tab[0] = floor(log2(MS_EPS)*(1 << 16) + 0.5);
  // filling the lin2log table
for (i=1; i<=LEV; i++)
  log_tab[i] = floor(log2(i/LEV)*(1 << 16) + 0.5);
```

The $\max^*$ and $\max^-$ operators, that are the basic blocks of the proposed solution, can be easily and effectively implemented as two macros:

```
#define r 6
#define m 511
#define n 511

#define max(x,y) ( ((x)>(y)) ? (x) : (y) )
#define maxxp(a,b) (((abs(a-b)>>(16-r)) < m)? \
                    (max(a,b) + err_Tab1[abs(a-b)>>(16-r)]):(max(a,b)))
#define maxxm(a,b) (((abs(a-b)>>(16-r)) < n)? \
                    (max(a,b) + err_Tab2[abs(a-b)>>(16-r)]):(max(a,b)))
```

Said $K$ the number of iterations ($K = 20$ in this experiment), both the programs perform $K - 1$ iterations in the logarithmic domain and during the last iteration they properly format the results to be displayed.

In the following the renormalization step that occurs during the last iteration is shown, for the standard implementation:

```
u_char[i][j] = (unsigned char) floor(255*((float)u[i][j]/(1 << 16)));
z_char[i][j] = (unsigned char) floor(255*((float)z[i][j]/(1 << 16)));
```

and for the multiplierless one:

```
tmp = pow(2,((float)u_log[i][j]/(1 << 16)));
u_char[i][j] = (unsigned char) floor(255*tmp);
tmp = pow(2,((float)z_log[i][j]/(1 << 16)));
z_char[i][j] = (unsigned char) floor(255*tmp);
```

Experimental results obtained running the standard implementation of (5) and the proposed multiplierless solution on the IDK platform are shown in Fig. 7, where the original frame ($g$) and its regularized ($u$) and segmented ($z$) versions are depicted. As it can be observed the proposed multiplierless solution, that, as previously stated, employs 6 bit precision tables to approximate $\max^*$ and the $\max^-$ operators, achieves nearly the same quality of its multiplication/division based counterpart. To further reduce the complexity of the proposed solution another implementation, that does not convert logarithmic values to linear ones, has been developed. Since the fixed point $u_{i,j}$ and $z_{i,j}$ logarithmic values are in $[\log_2 MS\_EPS, 0] \cdot 2^{16}$, namely $[-9 \cdot 2^{16}, 0]$, changing the sign and dividing by $2^{12}$ (12 right shifts), we obtain values in $[0, 144]$. Simulations show that 11 right shifts are enough to limit $u_{i,j}$ and $z_{i,j}$ logarithmic values in $[0, 255]$ (8 bits); as a consequence these logarithmic values can be displayed:

```
u_log_char[i][j] = (unsigned char) (-(u_log[i][j] >> 11));
z_log_char[i][j] = (unsigned char) (-(z_log[i][j] >> 11));
```

[Fig. 7 about here.]

In Fig. 8 visual examples of the multiplierless solution without log-lin conversion are shown. As it can be inferred from the previous paragraphs the standard and multiplierless implementations require a different amount of data memory. Three $Rows \times Cols$ bytes memory buffers are common to both standard and multiplierless implementations to display $g$, $u$ and $z$ values on the IDK output VGA ($144 \times 176 = 24.75$ kB). Three $Rows \times Cols$ local buffers are required for fixed point computation. Each buffer contains $Rows \times Cols$ fixed point (`int`) values ($144 \times 176 \times 4 = 99$ kB). The multiplierless implementation requires three further arrays: one for the logarithmic conversion and two for the $\max^*$ and $\max^-$ correction tables. The logarithmic conversion table, required to translate the 8 bits wide original pixels into the corresponding logarithmic values, is a 256 `int` value array ($256 \times 4 = 1$ kB). The error correction tables, made of the $m$ and $n$ `int` values, contain the data obtained solving (23) and (25) ($m = 511$ and $n = 511$ thus $511 \times 4 \simeq 2$ kB each). Thus the total memory required by the standard implementation is $3 \times 24.75 + 3 \times 99 = 371.25$ kB, whereas the multiplierless one requires $371.25 + 1 + 2 \times 2 = 376.25$ kB. It is worth pointing out that the memory increase required by the multiplierless implementation very limited (about 1.35%). Furthermore, as suggested in [29], all the frame buffers are stored into the main board SDRAM heap, as a consequence, the binary files size does not take into account the 371.25 kB data memory. On the other hand, since the logarithmic table and the correction tables require only about 3 kB they are constant automatic variables on the stack; as a consequence, they are part of the multiplierless implementation binary file. Since the binary files size for the three implementations is about 200 kB, we can conclude that the multiplierless solution overhead is negligible. From the complexity point of view, the three implementations run times have been measured. The

measured times concern the frame grabbing, the filtering and subsampling procedure to reduce the frame to QCIF size, the Mumford and Shah functional and the displaying. In table I the measured run times to perform the Mumford and Shah functional on a frame with the aforementioned models are summarized. As it can be observed in table I the multiplierless implementation runs 3 times faster than the standard one, with an execution time reduction of the 66%. Furthermore, if no logarithmic to linear conversion is performed, the execution time can be reduced of the 68%. Since results shown in table I have been obtained with 20 iterations on QCIF frames, we can infer that the standard implementation requires about 24 $\mu s$ per sample per iteration. On the other hand the multiplierless requires about 8 $\mu s$ per sample per iteration whereas the multiplierless without log-lin conversion requires about 7.4 $\mu s$ per sample per iteration.

[TABLE 1 about here.]

[Fig. 8 about here.]

## VI. Conclusion

In this work a multiplierless implementation of the Mumford and Shah functional has been presented. The impact on results quality introduced by the proposed solution has been analyzed through several simulations and a negligible visual quality degradation has been verified. Moreover the proposed solution has been developed as a C program, ported on a DPS board and compared with a standard solution. Run-time results show that the proposed implementation can reduce from 66% to 68% the execution time with respect to a standard implementation with no visual quality degradation.

## References


[1] L. Ambrosio and V. M. Tortorelli, "Approximation of functionals depending on jumps by elliptic functionals via Γ-convergence," *Comm. Pure Appl. Math.*, vol. 43, pp. 999–1036, 1990.

[2] S. Benedetto and G. Montorsi, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal of Selected Areas in Communications*, vol. 19, no. 5, pp. 871–882, may 2001.

[3] A. Bonnet, "On the regularity of edges in image segmentation," *Ann. Inst. H. Poincaré Anal. Non Linéaire*, vol. 13, pp. 485–528, 1996.

[4] A. Brook, R. Kimmel, and N. A. Sochen, "Variational restoration and edge detection for color images," *Journal of Mathematical Imaging and Vision*, vol. 18, no. 3, pp. 247–268, 2003.

[5] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, feb. 2001.

[6] D. Cremers, "A variational framework for image segmentation combining motion estimation and shape regularization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 53–58.

[7] D. Cremers, "A multiphase level set framework for motion segmentation," in *International Conference on Scale Space Theories in Computer Vision*, 2003, pp. 659–614.

[8] D. Cremers and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion segmentation," *International Journal of Computer Vision*, vol. 62, no. 3, pp. 249–265, 2005.

[9] D. Cremers, F. Tishhäuser, J. Weickert, and C. Shnörr, "Diffusion snakes: introducing statistical shape knowledge into the Mumford and Shah functional," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 295–313, 2002.

[10] G. Dal Maso, J. M. Morel, and S. Solimini, "A variational method in image segmentation: existence and approximation results," *Acta Math.*, vol. 168, no. 1-2, pp. 89–151, 1992.

[11] G. David, "$C^1$ arcs for the minimizers of the Mumford-Shah fuctional," *SIAM J. Appl. Math*, vol. 56, pp. 783–888, 1996.

[12] E. De Giorgi, M. Carriero, and A. Leaci, "Existence theorem for a minimum problem with free discontinuity," *Arch. Rational. Mech. Anal*, vol. 108, pp. 195–218, 1989.

[13] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[14] F. Gibou and R. Fedkiw, "Fast hybrid k-means level set algorithm for segmentation," in *International Conference on Statistics and Mathematics*, 2002.

[15] F. Gibou, D. Levy, C. Cardenas, P. Liu, and A. Boyer, "Partial differential equation based segmentation for radiotherapy treatment planning," *Mathematical Biosciences and Engineering*, vol. 2, pp. 209–226, 2005.

[16] S. Jehan-Besson, M.Barlaud, and G. Aubert, "Video object segmentation using eulerian region-based active contours," in *International Conference in Computer Vision*, 2001, pp. 353–361.

[17] G. Koepfler, C. Lopez, and J. M. Morel, "A multiscale algorithm for image segmentation by variational method," *SIAM J. Numer. Anal.*, vol. 31, no. 1, pp. 282–299, feb. 1994.

[18] R. March, "Visual reconstruction with discontinuities using variational methods," *Image and Vision Computing*, vol. 10, pp. 30–38, 1992.

[19] M. Martina, "Standard and multiplierless Mumford and Shah DSP implementations," downloadable at www.vlsilab.polito.it/∼martina.

[20] M. Martina and G. Masera, "Mumford and Shah functional: Finite precision analysis and software implementation," in *IEEE International Symposium on Signal Processing and Information Technology*, 2004.

[21] M. Martina and G. Masera, "Mumford and Shah functional: VLSI analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 487–494, Mar. 2005.

[22] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. 11, no. 4, pp. 512–517, Aug 1962.

[23] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Comm. Pure Appl. Math.*, vol. 42, pp. 577–685, 1989.

[24] S. Papaharalabos, P. Takis-Mathiopoulos, G. Masera, and M. Martina, "On optimal and near-optimal turbo decoding using generalized max* operator," *IEEE Communication Letters*, vol. 13, no. 7, pp. 522–524, Jul 2009.

[25] M. Ristivojevic and J. Konrad, "Joint space-time motion-based video segmentation and occlusion detection using multiphase level sets," in *Symposium on Electronic Imaging, Visual Communications and Image Processing*, 2004, pp. 1–12.

[26] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *IEEE International Conference on Communications*, 1995, pp. 1009–1013.

[27] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: A new image similarity index," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2385–2401, Nov 2009.

[28] TI, "http://focus.ti.com/lit/ug/spru494a/spru494a.pdf."

[29] TI, *SPRU499: TMS320C6000 Imaging Developers Kit (IDK) Video Device Driver Users Guide*.

[30] W. Vanzella, F. A. Pellegrino, and V. Torre, "Edge detection revisited," *IEEE Trans. on Systems, Man, and Cybernatics-Part B: Cybernetics*, vol. 34, no. 3, pp. 1500–1518, jun 2004.

[31] W. Vanzella, F. A. Pellegrino, and V. Torre, "Self-adaptive regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 804–809, jun. 2004.

[32] T. P. Vogl, J. W. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biological Cybrnetics*, vol. 59, pp. 257–263, 1988.

[33] T. Zhang, Z. Wang, and K. K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *IEEE International Symposium on Circuits and Systems*, 2001, pp. 202–205.

[34] A. Zijdenbos, B. Dawant, R. Margolin, and A. Palmer, "Morphometric analysis of white matter lesions in MR images: Method and validation,," *IEEE Transactions of Medical Imaging*, vol. 13, no. 4, pp. 716–724, Apr 1994.

## LIST OF FIGURES

(a) $u_{i,j}^{(org)}$ original

(b) $z_{i,j}^{(org)}$ original

(c) $u_{i,j}^{(r=8)}$ with $r = 8$

(d) $z_{i,j}^{(r=8)}$ with $r = 8$

(e) $u_{i,j}^{(org)} - u_{i,j}^{(r=8)}$

(f) $z_{i,j}^{(org)} - z_{i,j}^{(r=8)}$

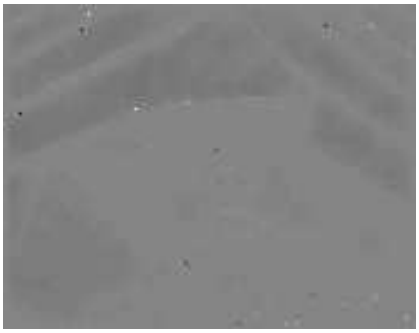
Fig. 1. Visual examples of the $\max^*$ and $\max^-$ approximation for a frame of the "foreman" sequence, test case $A$: (a) $u_{i,j}$ original, (b) $z_{i,j}$ original, (c) $u_{i,j}$ with $r = 8$, (d) $z_{i,j}$ with $r = 8$, (e) and (f) differences between original and $r = 8$ implementations

(a) $u_{i,j}^{(org)}$ original

(b) $z_{i,j}^{(org)}$ original

(c) $u_{i,j}^{(r=6)}$ with $r = 6$

(d) $z_{i,j}^{(r=6)}$ with $r = 6$



(e) $u_{i,j}^{(org)} - u_{i,j}^{(r=6)}$

(f) $z_{i,j}^{(org)} - z_{i,j}^{(r=6)}$

Fig. 2. Visual examples of the $\max^*$ and $\max^-$ approximation for a frame of the "foreman" sequence, test case $A$: (a) $u_{i,j}$ original, (b) $z_{i,j}$ original, (c) $u_{i,j}$ with $r = 6$, (d) $z_{i,j}$ with $r = 6$, (e) and (f) differences between original and $r = 6$ implementations

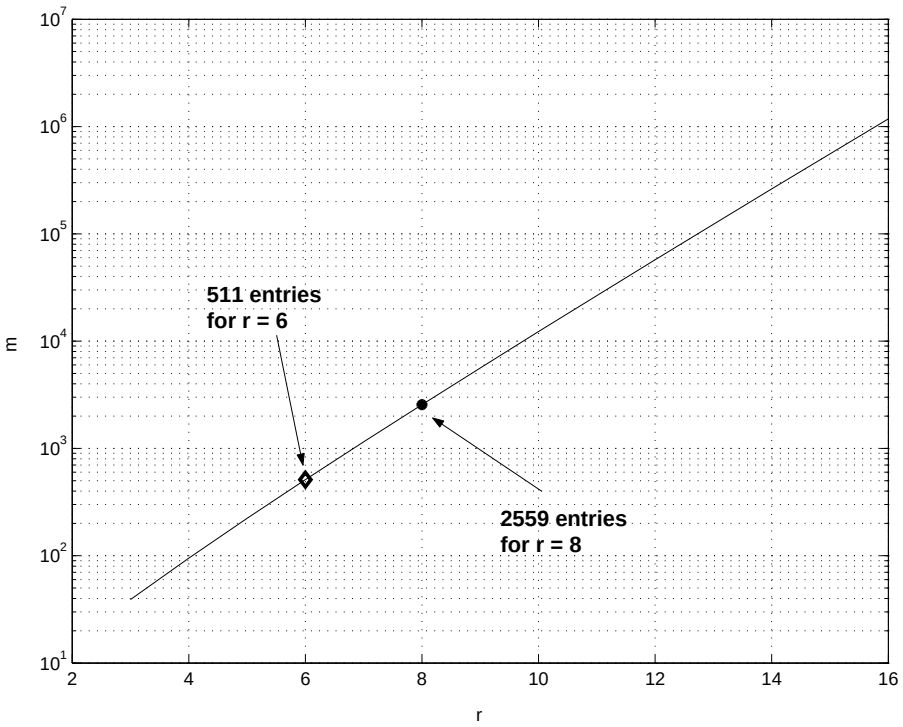
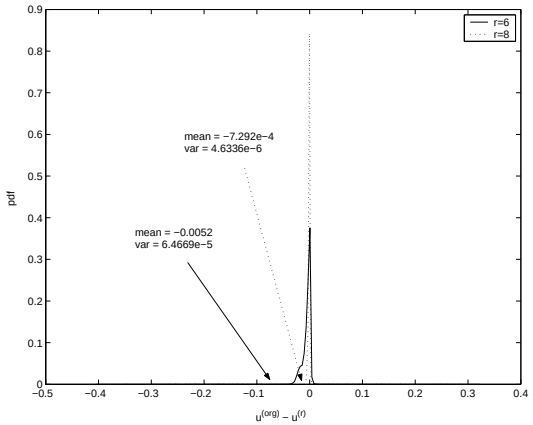
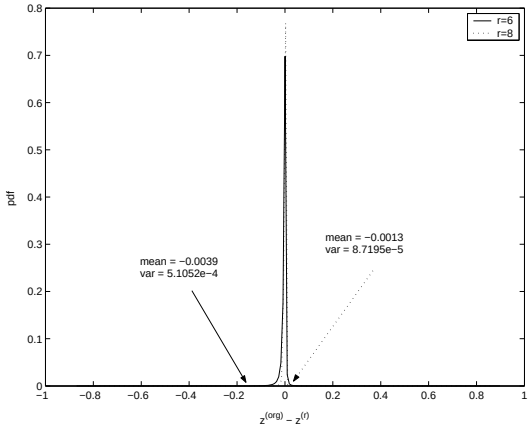


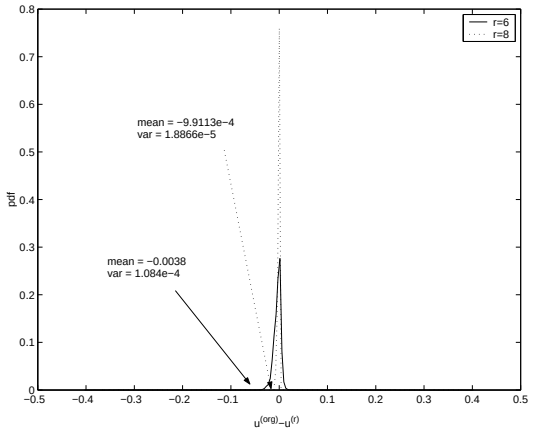

Fig. 3. LUT entries ($m$ or $n$) Versus cell width ($r$) required to approximate the $\max^*$ and $\max^-$ operators



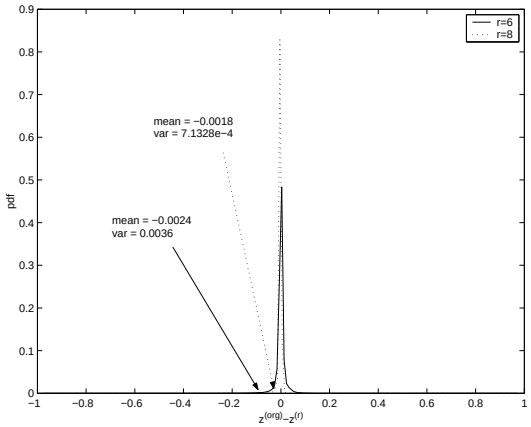

(a) $u_{i,j}^{(org)} - u_{i,j}^{(r)}$ distribution $(A)$



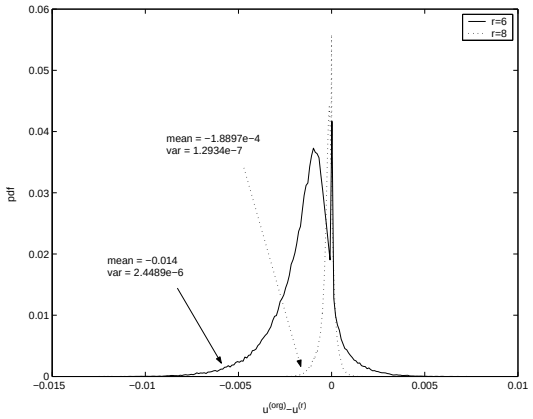

(b) $z_{i,j}^{(org)} - z_{i,j}^{(r)}$ distribution $(A)$



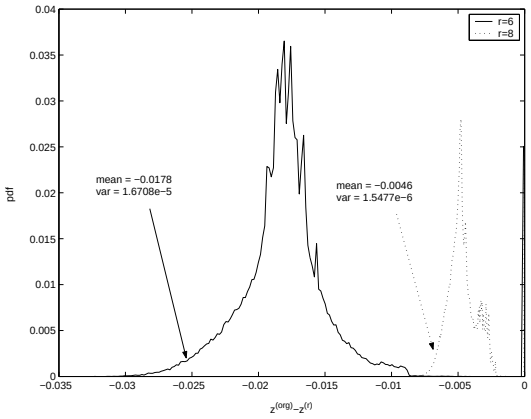

(c) $u_{i,j}^{(org)} - u_{i,j}^{(r)}$ distribution $(B)$




(d) $z_{i,j}^{(org)} - z_{i,j}^{(r)}$ distribution $(B)$

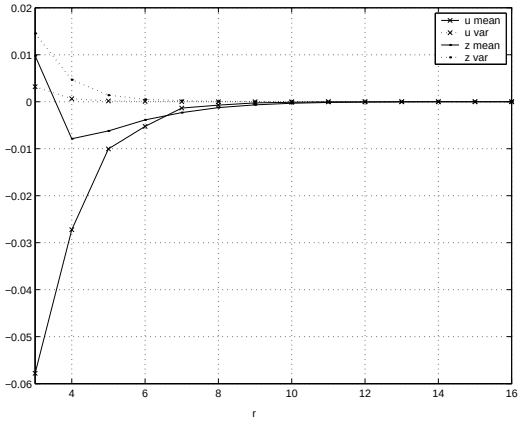



(e) $u_{i,j}^{(org)} - u_{i,j}^{(r)}$ distribution $(C)$

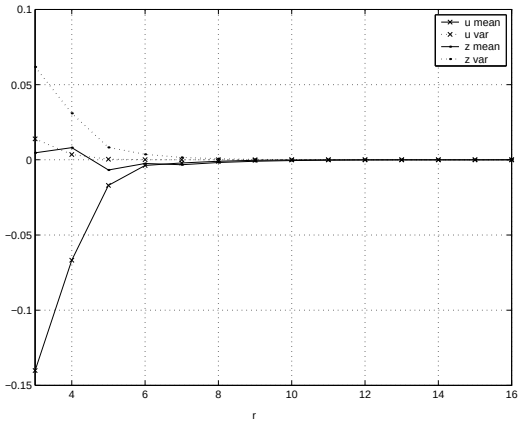



(f) $z_{i,j}^{(org)} - z_{i,j}^{(r)}$ distribution $(C)$

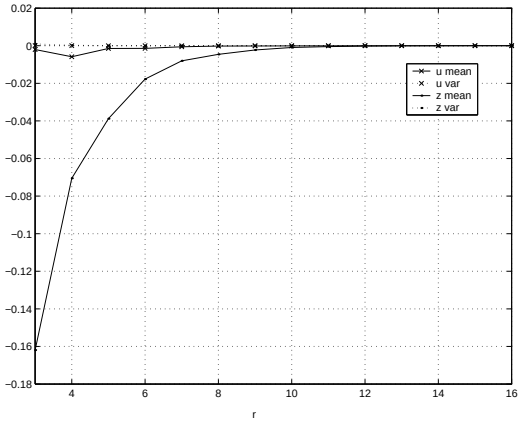
Fig. 4. Distributions of the absolute errors $u_{i,j}^{(org)} - u_{i,j}^{(r=8)}$ and $z_{i,j}^{(org)} - z_{i,j}^{(r=8)}$ in the three test cases $A$, $B$ and $C$ with $r = 6$ and $r = 8$




(a) absolute error mean and variance ($A$)

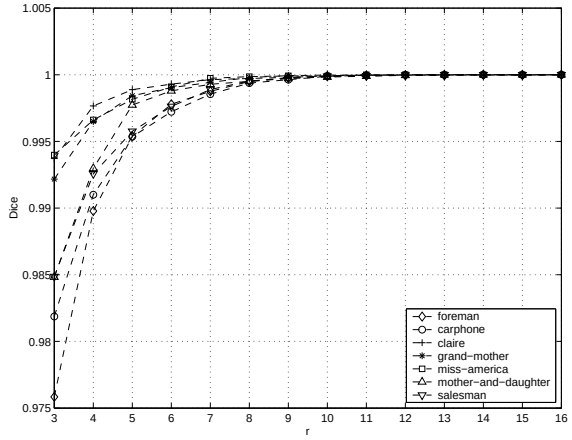



(b) absolute error mean and variance ($B$)

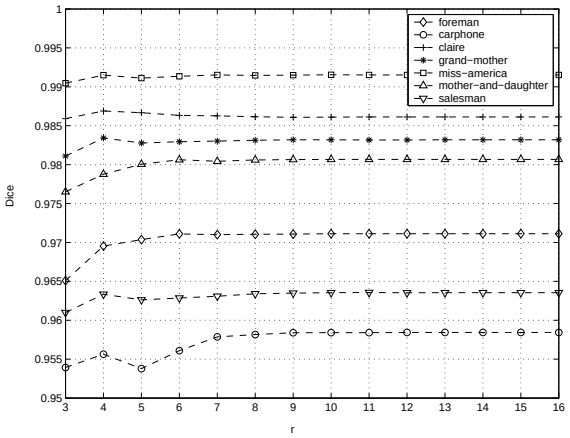



(c) absolute error mean and variance ($C$)

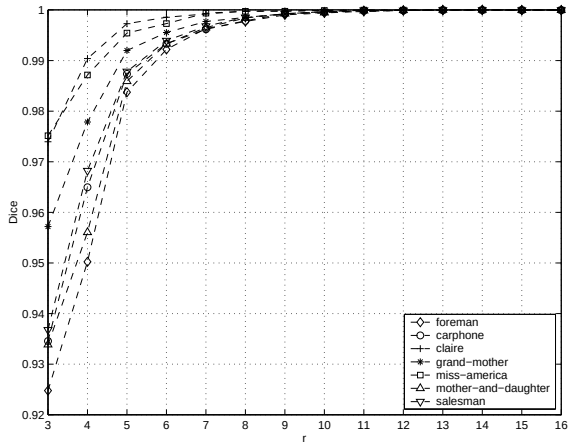
Fig. 5. $u_{i,j}^{(org)} - u_{i,j}^{(r)}$ and $z_{i,j}^{(org)} - z_{i,j}^{(r)}$ mean value and variance as functions of $r$ for the three test cases ($A$), ($B$) and ($C$)



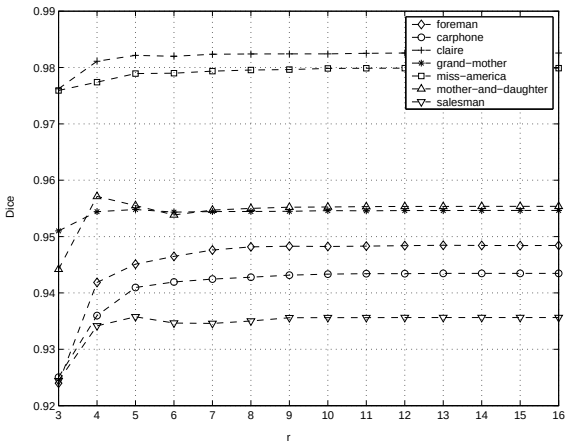

(a) $z^{(r)}$ with log-lin conversion ($A$)



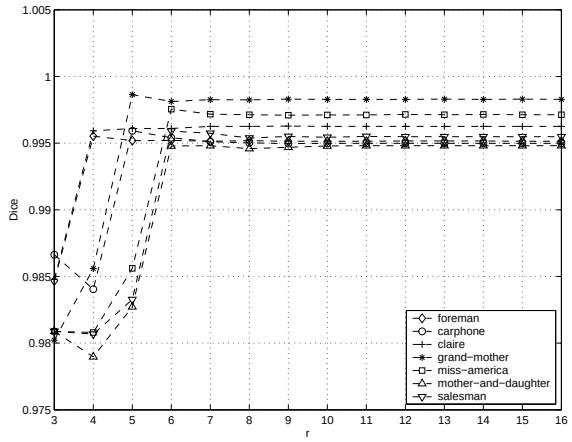

(b) $z^{(r)}$ without log-lin conversion ($A$)



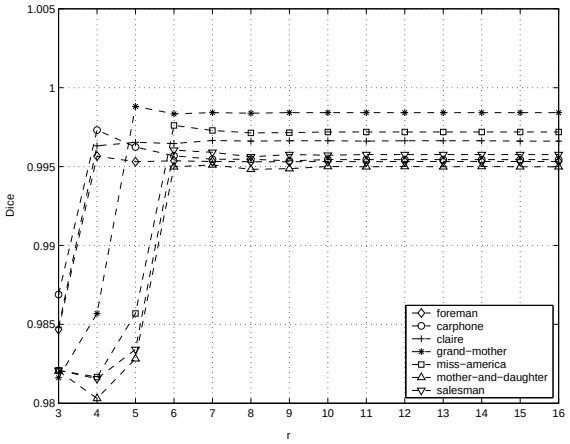

(c) $z^{(r)}$ with log-lin conversion ($B$)




(d) $z^{(r)}$ without log-lin conversion ($B$)

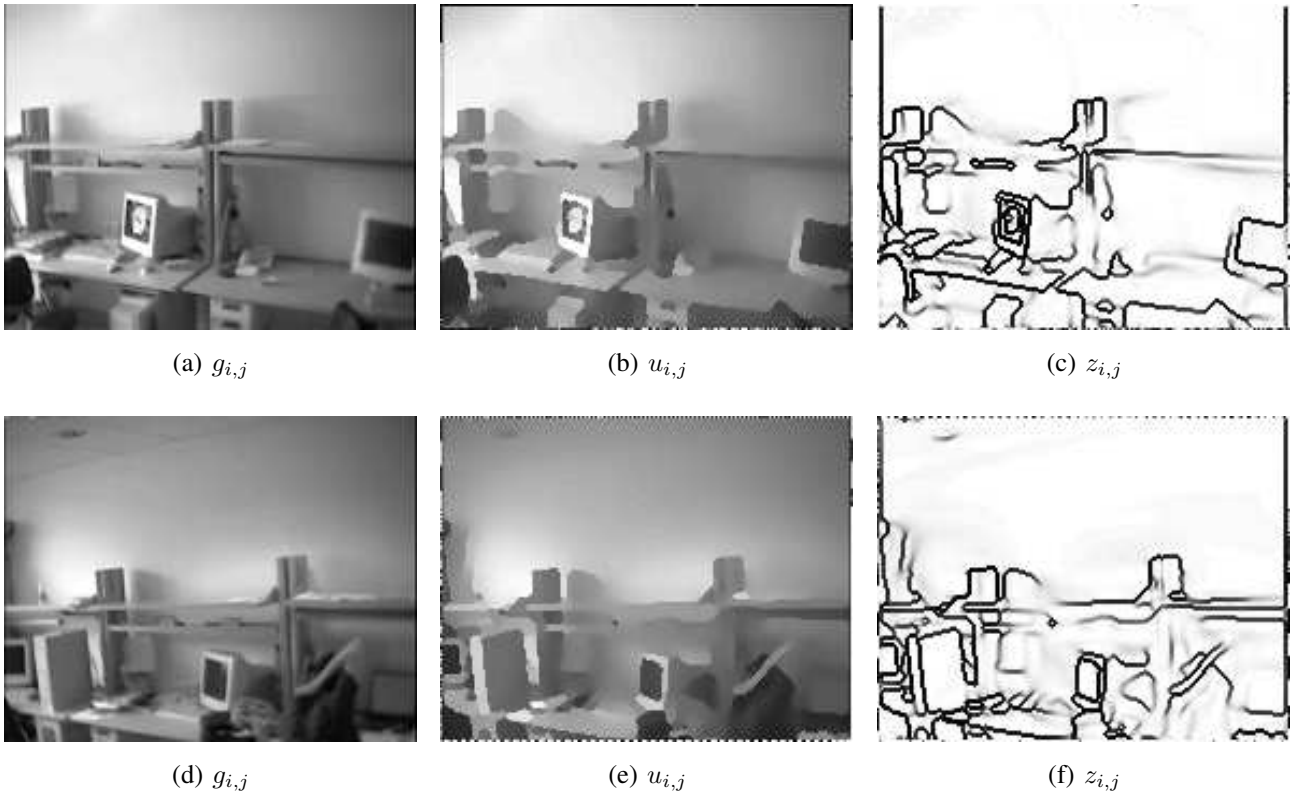



(e) $z^{(r)}$ with log-lin conversion ($C$)




(f) $z^{(r)}$ without log-lin conversion ($C$)

Fig. 6. Segmentation similarity between the standard implementation ($z^{(org)}$) and the multiplierless one ($z^{(r)}$) with (a), (c), (d) and without log-lin conversion (b), (d), (f): Dice similarity coefficient as a function of $r$ for the seven test images in the test cases $A$, $B$ and $C$



(a) $g_{i,j}$ (b) $u_{i,j}$ (c) $z_{i,j}$

(d) $g_{i,j}$ (e) $u_{i,j}$ (f) $z_{i,j}$

Fig. 7. Visual examples of the data displayed by the IDK board for the multiplierless implementation (a), (b) and (c), and for the standard implementation (d), (e) and (f)

(a) $g_{i,j}$ (b) $\ln(u_{i,j})$ (c) $\ln(z_{i,j})$

Fig. 8. Visual examples of the data displayed by the IDK board without log-lin conversion

## LIST OF TABLES

TABLE I
**Measured run time on the IDK board of the implemented Mumford and Shah functional**

| | **Standard (Fig. 7)** | **Multiplierless (Fig. 7)** | **Mulplierless no log-lin (Fig. 8)** |
|---|---|---|---|
| run time | 12 s | 4 s | 3.75 s |
| run time reduction | - | 66% | 68% |