Design and Implementation of a Fuzzy Area-Based Image-Scaling Technique

Angelos Amanatiadis, Student Member, IEEE, Ioannis Andreadis, and Konstantinos Konstantinidis

Abstract—In this paper, we propose the design and implementation of an interpolation scheme for performing image scaling by utilizing a dynamic mask combined with a sophisticated neighborhood averaging fuzzy algorithm. The functions that contribute to the final interpolated image are the areas of the input pixels, overlapped by a dynamic mask, and the difference in intensity between the input pixels. Fuzzy if-then rules for these two functions are presented to carry out the interpolation task. Simulation results have shown a fine high-frequency response and a low interpolation error, in comparison with other widely used algorithms. The interpolation can be applied to both gray-scale and color images for any scaling factor. The proposed hardware structure is implemented in a field-programmable gate array (FPGA) chip and is based on a sequence of pipeline stages and parallel processing to minimize computation times. The fuzzy image interpolation implementation combines a fuzzy inference system and an image-interpolation technique in one hardware system. Its main features are the ability to accurately approximate the Gaussian membership functions used by the fuzzy inference system with very few memory requirements and its high-frequency performance of 65 MHz, making it appropriate for real-time imaging applications. The system can magnify gray-scale images of up to 10-bit resolution. The maximum input image size is 1024 × 1024 pixels for a maximum of 800% magnification.

Index Terms—Digital image scaling, fuzzy area-based interpolation, fuzzy zooming controller, real-time image scaling.

I. INTRODUCTION

IMAGE-scaling schemes are becoming more popular due to the wide use of digital imaging devices and are more demanding due to the increasing sensor-capturing capabilities in spatial resolution. Various applications require image zooming, such as digital cameras, electronic publishing, third-generation mobile phones, medical imaging, and image processing. An image-scaling technique performs spatial interpolation since it enlarges the spatial resolution of the input image and completes the missing data from the spatially adjacent data using some type of interpolation operation. Furthermore, it tries to preserve the spectral contents of the input image. Conventional methods, such as the nearest-neighbor or bilinear interpolation [1], exhibit high computational simplicity but face serious blurring problems, particularly in edge regions. Linear approaches are the ones most frequently used since nonlinear methods, such as bicubic interpolation [2] and spline interpolation [3], [4], entail a larger computational burden and involve blurring, despite producing better results. Edge-directed interpolations have been attempted to overcome such shortcomings by applying a variety of operators according to the edge directions [5]. However, only certain angles of edges are identifiable. Vector quantization zooming has been also reported [6] with good results but with the very restrictive scaling factor of two. Neural networks have also been used for image interpolation [7]. These, on the other hand, require a large amount of cells in the networks, rendering them, computationally, too intensive. Fuzzy interpolation approaches have been proposed [8], [9] for 2-D signal resampling, with optimal visual results, but they are lacking in simplicity and require additional processing for edge identification. Area-based interpolation [10] computes each interpolated pixel by a proportional area coverage of an applied filtering window. The method in [10] presents a straightforward and nonfuzzy interpolation. However, the approach requires a great number of time-demanding calculations for the proportional area coverage. The Mitchell and Lanczos [11], [12] interpolation methods are not widely used and are unsuitable in terms of hardware resources required. Fractal zooming [13], [14] has been recently reported with very good visual results, but the computational cost for its coding remains its main disadvantage.

From a hardware point of view, a number of algorithm implementations for image interpolation have been reported in the literature, as have a number of fuzzy controller implementations. However, a fuzzy image interpolation implementation, combining a fuzzy inference system (FIS) and an image interpolation technique in one system, has not been reported in the literature. Moreover, implementations of FISs using Gaussian membership functions are more complex and hardware demanding than those using linear membership functions since triangular or trapezoidal functions are easier to compute as only first-order polynomials are involved. Digital circuits that can compute Gaussian membership functions by using a finite-length series expansion or a polynomial approximation can be used instead [15], [16]. Various hardware implementations for image scaling, which use different interpolation techniques, have been reported in the literature. Field-programmable gate array (FPGA) implementations that reach real-time performance have been developed in [17] and [18]. Additionally, an area-based scaling algorithm is implemented on an FPGA in [19]. A block-level parallel processing architecture for scaling evenly divisible images is presented in [20]. However, none of these interpolation implementations entail an FIS in their process.

Manuscript received July 4, 2007; revised May 5, 2008.

The authors are with the Laboratory of Electronics, Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece (e-mail: aamanat@ee.duth.gr; iandread@ee.duth.gr; konkonst@ee.duth.gr).
In this paper, we propose the design and implementation of an interpolation algorithm for use in image scaling by utilizing a dynamic mask combined with a sophisticated neighborhood averaging fuzzy algorithm [21]. The components considered to result in the final interpolated image are the areas of the input pixels, overlapped by a dynamic mask, and the difference in intensity of the input pixels. Simple fuzzy if–then rules are involved in the inference process of these two components to carry out the interpolation task. Simulation results have shown a fine high-frequency response and a low interpolation error, compared with other widely used algorithms. The interpolation can be applied, for any scaling factor, to both gray-scale and color images. In the latter case, the algorithm is separately applied to each of the three primary components. The hardware prototype was realized on an FPGA, utilizing its apparent flexibility and high performance. Its high-frequency performance makes it appropriate for real-time imaging applications. The system architecture consists of five main functional pipelined units, and parallel processing was used within each unit for minimizing computational times. The presented digital circuit was designed, compiled, and simulated using the Quartus II programmable logic development system of Altera Corporation. The chip chosen is the EP1S20F484C6 of the Stratix family, i.e., a family that combines high digital signal processor performance with low cost. The total number of the system inputs and that of the outputs are 88 and 10 pins, respectively. The typical system clock frequency is 65 MHz, and the internal memory required is 195 kB. The FIS holds 16 rules and utilizes an embedded Gaussian function approximation unit to calculate the Gaussian membership functions used by the proposed scaling algorithm.

The rest of the paper is organized as follows: In Section II, we describe in detail the proposed interpolation and the FIS. In Section III, we present the hardware architecture of the fuzzy interpolation and the implementation of its five basic functional stages. In Section IV, we evaluate the system performance and discuss the typical implementation and timing analysis characteristics. In Section V, we report the simulation results and compare the performance of the proposed algorithm with other widely used interpolation algorithms through quantitative measurements. Finally, we give concluding remarks in Section VI.

II. PROPOSED APPROACH

The aim of the proposed approach is to maximize the smoothness of the interpolated image under the constraint of preserving the edges sharp and continuous. To reduce the computational burden, a scheme using a maximum of four contributing neighboring pixels was chosen. The fuzzy interpolation method utilizes continuous domain filtering, taking into account the proportional area coverage of the filtering window and the differences in intensity of the neighbor pixels. Thus, no additional edge-detection subprocesses are necessary. Three steps are required to describe the overall scaling process. The “zooming-in” process will be explained in detail, whereas the scaling-down process is derived by using the same sequence. First, we construct the output image through undefined pixel intensity values, according to the magnification factor. Subsequently, we determine the area of the dynamic mask whose value depends on the scaling factor. Next, we map each output pixel onto the input image and apply the mask. Based on the dimensions of the mask, the overlapped input pixels could be one, two, or four, as shown in Fig. 1. For each overlapped input pixel, we extract two functions that describe the area coverage from the mask as well as the differences in intensity of the overlapped input pixels. Finally, we apply the fuzzy inference method to these two functions to extract the proportional input pixel weights that will determine the intensity of the output pixel.

A. Construction and Application of the Dynamic Mask

Let \( S(i, j) \) be the input image with \( m \) and \( n \) dimensions and scale factor \( s \). For the scaling-up process, we first construct the output image \( T(k, l) \), where \( k = 1, 2, \ldots, s \times m \), and \( l = 1, 2, \ldots, s \times n \). For a fractional scaling factor, we apply a floor operation to the dimensions of the output image. The intensity values of all pixels in \( T \) are left undefined. The square mask dimensions are defined as follows:

\[
M_{\text{height}} = \frac{1}{s}, \quad M_{\text{length}} = \frac{1}{s}.
\]  

(1)

The mask dimensions are dynamic, depending on the scaling factor. This feature guarantees that after the interpolation, no input area was used twice or more in the scaling process. Next, we map each output pixel onto the input image plane. The coordinates of the center of the mapped output pixel onto the input image are calculated using the following equations to allocate the center of the applied mask in the Cartesian plane [10]:

\[
x = \frac{l + 0.5}{s}, \quad y = \frac{k + 0.5}{s}
\]  

(2)

where \( k = 1, 2, \ldots, s \times m \), and \( l = 1, 2, \ldots, s \times n \), assuming that each pixel has a height and width equal to one. Once the mask is applied, there are basically three possible conditions. The first is that the mask entirely lies within one pixel of the input image. The second is that the projection partially lies in
The differences in intensity functions can be easily determined as

\[ f_{i,j}(x, y) = \left| S(i, j) - S(i-1, j) \right| \quad (7) \]
\[ g_{i,j-1}(x, y) = \left| S(i, j-1) - S(i, j) \right| \quad (8) \]
\[ g_{i,j}(x, y) = \left| S(i, j) - S(i, j-1) \right| \quad (9) \]

where \( S(i, j) \) is the main pixel, as shown in Fig. 1. Equivalent equations are used to represent the \( f(x, y) \) and \( g(x, y) \) functions for the cases of one or two overlapped input pixels by the applied mask.

### C. FIS

To derive the weights that will define the value of the output pixel, we present the area and difference-in-intensity components as two separate inputs to a fuzzy (Mamdani-type) inference system. The system consists of four Gaussian curve membership functions (GCMFs) for the area components, five GCMFs for the difference in intensity of the input pixels, and five GCMFs for the weight outputs, as shown in Fig. 2. The input variables \( f \) and \( g \) denote the percentage mask area coverage for each overlapped pixel and the difference in intensity of the input pixels, respectively. The pixel intensity values range between zero and one after the normalization process. Last, the output variable indicates the proportional weight of each contributing pixel.

The number and shape of the membership functions were selected after an extensive set of preliminary tests. These tests involved simulations with different types and numbers of membership functions, such as triangular, trapezoidal, and Gaussian. The simulation results have also been compared to the straightforward and nonfuzzy approach described in [10].

![Graphical representation of the input and output membership functions.](image)

The tests confirmed the expected tradeoffs between speed, complexity, and qualitative performance. Thus, we have used the Gaussian membership function that presents good quantitative results. The GCMFs were also chosen due to their smoothness and the fact that they are nonzero at all points. Its formula is presented as follows, where \( \sigma \) determines the shape of the curve (curvature), and \( c \) locates the center of the curve:

\[ f(x, \sigma, c) = e^{-\frac{(x-c)^2}{2\sigma^2}}. \quad (10) \]

Although incurring the inference by using Gaussian membership functions renders the process more time demanding, the results were superior to those involving linear membership functions such as trigonal or trapezoidal ones. This means that in the tradeoff between speed and accuracy, we preferred accuracy in terms of better quantitative results.

The number of membership functions was chosen to be only four and five for the area coverage and the difference-in-intensity input variable, respectively, to meet an adequate tradeoff.
where $W$ from the FIS for each equivalent intensity for each contributing overlapped each overlapped input pixel to have the required output weight the area component. This special case is expressed through the so the output is produced through the direct defuzzification of mainly covered by the mask, there is no difference in intensity, from straightforward calculations. For the main pixel, which is premises were attained after extensive comparisons with results of 16 if–then rules, as presented in Table I. The linguistic rule The two inputs are cross connected to the output through a set the proposed structure is already high.

Finally, the output pixel intensity is given by

$$T(k,l) = \sum_{n=0}^{1} \sum_{m=0}^{1} W_{i-n,j-m} \times S(i-n,j-m)$$

(11)

where $W_{i,j}$ represents the final normalized weights extracted from the FIS for each equivalent intensity $S(i,j)$ of the input pixel.

Finally, the output pixel intensity is given by

$$T(k,l) = \sum_{n=0}^{1} \sum_{m=0}^{1} W_{i-n,j-m} \times S(i-n,j-m)$$

On the FIS and the required preprocessing and postprocessing of the inputs and outputs, respectively. The required interfaces and external memory buffer architecture are out of the scope of this paper. However, the demands for external memory are very small since only two line buffers are required for both vertical or horizontal mask filtering. The scaling can be applied to both gray-scale or color images. For color images, the same circuit is used for each different color space component. This can be done serially or in parallel, depending on the hardware resources of the selected chip. The required internal memory in the case of parallel color scaling remains the same, and only the gate count is increased. The internal memory contains the fuzzy set base, consisting of the membership function values associated with the fuzzy sets, the conditional rules to be processed [22], lookup tables, and the constant needed for various calculations. The values of the Gaussian membership functions are determined through an iterative approximation method implemented in hardware; thus, no extra memory is required for the construction of an additional lookup table. The inference is performed using the max–min method, and the center-of-gravity defuzzification method has been utilized to obtain the output. The proposed fuzzy system uses a Mamdani type of inference.

For hardware implementation, a 10-bit fixed-point numerical representation was chosen; therefore, the adequate resolution for all calculations is $2^{-10} = 9.7 \times 10^{-4}$. Thus, the intensities of the input pixels are considered to range from zero to one. The only two values of the hardware system that are in different representations are the scaling factor and the $x$ and $y$ Cartesian coordinates of the center of the mask that are in 7-bit (3 bits for the integer part and 4 bits for the fractional part) and 20-bit (10 bits for the integer part and 10 bits for the fractional part) numerical representations, respectively. The $x$ and $y$ representations also determine the available projection area of the input image, in terms of pixel resolution. Based on these representations, the scaling factor can range from zero to eight, with $2^{-4} = 0.00625$ resolution, and the maximum resolution of the input image can be $1024 \times 1024$ pixels. The architecture can be easily expanded to accommodate larger scaling factors and input resolution images.

As shown in Fig. 3, the system is comprised of five basic functional units: 1) the coverage area and difference in the intensity calculation unit; 2) the fuzzification and knowledge base unit; 3) the inference and rule control unit; 4) the defuzzification unit; and 5) the weighted sum calculation unit. The input data of the system are the intensity values of the four input pixels of the image neighborhood, the $x$ and $y$ coordinates of the center of applied mask, and the scaling factor. The output of the unit is the intensity of the output image pixel.

### III. ARCHITECTURE AND IMPLEMENTATION

The proposed architecture was designed to perform the scaling-up process based on a sequence of pipeline stages for gray-scale images. Parallel processing was used whenever possible to further accelerate the process. The structure is focused on the FIS and the required preprocessing and postprocessing of the inputs and outputs, respectively. The required interfaces and external memory buffer architecture are out of the scope of this paper. However, the demands for external memory are very small since only two line buffers are required for both vertical or horizontal mask filtering. The scaling can be applied to both gray-scale or color images. For color images, the same circuit is used for each different color space component. This can be done serially or in parallel, depending on the hardware resources of the selected chip. The required internal memory in the case of parallel color scaling remains the same, and only the gate count is increased. The internal memory contains the fuzzy set base, consisting of the membership function values associated with the fuzzy sets, the conditional rules to be processed [22], lookup tables, and the constant needed for various calculations. The values of the Gaussian membership functions are determined through an iterative approximation method implemented in hardware; thus, no extra memory is required for the construction of an additional lookup table. The inference is performed using the max–min method, and the center-of-gravity defuzzification method has been utilized to obtain the output. The proposed fuzzy system uses a Mamdani type of inference.

For hardware implementation, a 10-bit fixed-point numerical representation was chosen; therefore, the adequate resolution for all calculations is $2^{-10} = 9.7 \times 10^{-4}$. Thus, the intensities of the input pixels are considered to range from zero to one. The only two values of the hardware system that are in different representations are the scaling factor and the $x$ and $y$ Cartesian coordinates of the center of the mask that are in 7-bit (3 bits for the integer part and 4 bits for the fractional part) and 20-bit (10 bits for the integer part and 10 bits for the fractional part) numerical representations, respectively. The $x$ and $y$ representations also determine the available projection area of the input image, in terms of pixel resolution. Based on these representations, the scaling factor can range from zero to eight, with $2^{-4} = 0.00625$ resolution, and the maximum resolution of the input image can be $1024 \times 1024$ pixels. The architecture can be easily expanded to accommodate larger scaling factors and input resolution images.

As shown in Fig. 3, the system is comprised of five basic functional units: 1) the coverage area and difference in the intensity calculation unit; 2) the fuzzification and knowledge base unit; 3) the inference and rule control unit; 4) the defuzzification unit; and 5) the weighted sum calculation unit. The input data of the system are the intensity values of the four input pixels of the image neighborhood, the $x$ and $y$ coordinates of the center of applied mask, and the scaling factor. The output of the unit is the intensity of the output image pixel.

#### A. Coverage Area and Difference in Intensity Calculation Unit

The inputs of the unit are the intensity values of the four input pixels of the image neighborhood, the $x$ and $y$ coordinates of the center of the applied mask, and the scaling factor $s$. Through a lookup table, the inverse operation for $1/s$ is realized, which represents the vertical and horizontal dimensions of the mask. After a multiplier, the area of the mask $1/s^2$ is available. The

### TABLE I

**Extracted Rules for the Interpolation Task. Two Inputs $f$: Area Coverage and $g$: Difference in Intensity, and the Output $w$: Weight**

<table>
<thead>
<tr>
<th>If $f$ is</th>
<th>and $g$ is</th>
<th>then $w$ is</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>Very low</td>
<td>Very low</td>
</tr>
<tr>
<td>Very low</td>
<td>Low</td>
<td>Very low</td>
</tr>
<tr>
<td>Very low</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Very low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Very low</td>
<td>Very high</td>
<td>Medium</td>
</tr>
<tr>
<td>Low</td>
<td>Very low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>Very high</td>
<td>Medium</td>
</tr>
<tr>
<td>Medium</td>
<td>Very low</td>
<td>Low</td>
</tr>
<tr>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Medium</td>
<td>Very high</td>
<td>High</td>
</tr>
<tr>
<td>High</td>
<td>Very high</td>
<td>High</td>
</tr>
</tbody>
</table>
Fig. 3. Block diagram of the proposed zooming hardware architecture.

first aim of this unit, before making the calculations for the area and difference-in-intensity functions, is to define how many input pixels will contribute and which one of them is the main one. Thus, a shifting operation is also applied to the dimension of the mask to obtain the value $1/2s$, which is then added to both the $x$ and $y$ variables. A floor operation to $x + (1/2s)$ and $y + (1/2s)$ designates the main pixel $(i, j)$ on the Cartesian plane. In hardware, the floor operation for a fixed-point format is realized by replacing the bits of the fractional part of the number with zeros. Three parallel and pipelined subtractors were used for calculating the three differences in intensity of (7)–(9). Since the four are coverage calculations (3)–(6), only four multipliers, four adders, and four subtractors were used, because some sums and subtractions are universal through the calculations. Register arrays have been used to form a pipelined sequence, holding data in the input and output of the first unit and after each combinational function. The four area functions $f$ and the three difference-in-intensity functions $g$ are the 10-bit outputs of this unit.

B. Fuzzification and Knowledge Base Unit

A fuzzification unit is used to translate the crisp values from the inputs into linguistic descriptions in terms of membership functions. This unit consists of modules working in parallel: one for each defined input variable. The inputs of this unit are the area functions $f$ and the difference-in-intensity functions $g$.

The intersection of the crisp input variable values and the fuzzy sets that correspond to the linguistic terms gives the input variable’s degree of membership, called $\mu$, that belongs to a given fuzzy set, with a value from zero to one. The two input variables $f$ and $g$ enter the unit at the same time. A comparator circuit, with a memory index, selects the involved (fired) Gaussian fuzzy sets for each input variable. The memory index, called the active interval selection block [23], has the four offline-defined parameters stored for each involved Gaussian fuzzy set. The construction of the membership functions in Fig. 2 was made in a way such that for each input variable, only two membership functions overlap at a time; thus, only four membership functions will be triggered for each pair of the inputs. Therefore, the problem is simplified to calculating the fuzzy rules that involve the corresponding fuzzy sets. The remaining rules are not contributing, so they will not be calculated, making the system more efficient. This is the reason why we chose to have the knowledge base unit coupled with the fuzzification unit and not inside the inference unit, as in [24]. We designed a comparator circuit to identify the fuzzy sets involved and, consequently, the fuzzy rules involved. The required intervals were easily extracted by successively comparing each input variable value with the starting and ending points of the Gaussian shape membership functions in Fig. 2. The starting and ending points were calculated offline and stored in the system’s memory.

For the Gaussian-shape membership function generator in (10), we preferred to build a digital circuit, based on polynomial approximation [16]. Solutions such as lookup table representations of nonlinear functions with medium or high resolutions are very costly in embedded memory terms. The approximation algorithm used was the centered recursive interpolation (CRI) method [25]. The main advantages of CRI are its simplicity and recursively improved accuracy, owing to an exponential growth of the number of segments of the generated function. Considering the symmetry of the function, calculating both tangents is not necessary, so the CRI approximation algorithm is given as follows:

$$g(x) = (2/\sqrt{e}) - m|x - c_i|; h_1(x) = 1; h_2(x) = 0$$

for $(i = 0; i = q; i++)$

$$\{g'(x) = \min [g(x), h_1(x)]$$

$$h_1(x) = 1/2 (g(x) + h_1(x) - \Delta)$$

$$g(x) = \max [g'(x), h_2(x)]$$

$$h_2(x) = 1/2 (g(x) + h_2(x) + \Delta)$$

$$\Delta = \Delta/4$$

$$g'(x) = \min [g(x), h_1(x)]$$

$$g(x) = \max [g'(x), h_2(x)]$$

where $q$ is the stored number of iterations, the parameter $m = \sqrt{e}/(\sigma/\sqrt{e})$ is the offline-calculated width of the Gaussian function, $c_i$ is the position of the maximum height of each fired membership function extracted from the knowledge base unit, and $g(x)$ is the resulting approximated function. The pipelined design of the Gaussian membership function generator is shown in Fig. 4. Four iterations were used, with $\Delta = 0.121$. The 10-bit fixed-point numerical representation that we used for the inputs is adequate and does not affect the maximum error of the approximation used.

C. Inference and Rule Control Unit

The inference and rule control unit processes the set of rules that were triggered from the knowledge base. For the given pair of $f$ and $g$ input values, only a subset of the rules from the knowledge base contribute to the final result. These are
the active rules. The remaining rules were discarded from the previous unit because the input values have null adherence, along with their membership functions. More specifically, for each combination of the fuzzy sets used for the \( f \) and \( g \) values, one fuzzy rule is fired. The number of all possible combinations is four (2 sets for \( f \) \times 2 sets for \( g \)), which implies that four fuzzy rules are active for each pair of \( f \) and \( g \). For the main pixel, which is mainly covered by the mask, the \( g \) function is never taken into consideration, and the output is produced through direct inference with the area component. The inference operation evaluates the relative adherence of each rule, resulting in a value that represents the importance of the conclusion for the current input condition. Each conclusion is represented by a truncated output membership function, which is performed by the minimum operator, because all the fuzzy rules of our system include premises connected only by AND operators. The 10-bit outputs of the unit are the corresponding degree of the active membership function truncation \( w_{hi} \) associated with conclusion \( i \) and \( ct_i \), which is the position of the maximum height of the active membership function.

**D. Defuzzification Unit**

The defuzzification unit involves determining the output value according to a set of conclusions, which were reached from evaluating the knowledge base’s rules. This conclusion is the aggregation of all the truncated membership functions, which is also known as the center-of-gravity defuzzification method, as follows:

\[
W = \frac{\sum_{i=1}^{n} w_{hi} ct_i}{\sum_{i=1}^{n} w_{hi}} \quad (13)
\]

where \( w_{hi} \) is the corresponding degree of membership functions associated with conclusion \( i \), and \( ct_i \) is the position of the maximum height. From the implementation point of view, the unit performs the two additions \( \sum w_{hi} ct_i \) and \( \sum w_{hi} \) in two parallel pipeline stages. Once all four rules have been processed, the data stored in the defuzzifier’s two adders go to the divider circuit to compute the crisp output value by means of the center-of-gravity defuzzification method. It should be noted that this stage requires an increased number of computations, mainly because of the fixed-point division operation. Thus, we used the speed-optimized block of Quartus II for the 10-bit division. Approximation techniques for implementing the division [26], [27] were not used, as the initial approximation for the division is not feasible due to the nature of the unit’s inputs. The 10-bit output of the unit is the final weight for each contributing input pixel.

**E. Weighted Sum Calculation Unit**

In this stage, the weighted sum, in (11), of all contributing input pixels is performed. The inputs of this unit are the weights \( w \), which are extracted from the defuzzification unit, and the intensities of the contributing pixels \( S \). A first step is required to have our final weights normalized. The pipelined normalization process requires seven adders and one divider. Again, we chose the speed-optimized block for the 10-bit division. Finally, four parallel and pipelined multipliers are used to calculate \( w_i S(i) \), where \( w_i \) represents the final normalized weights extracted from the FIS for each equivalent intensity \( S(i) \) of the input pixel. Three adders, in two pipelined sequence steps, are used for the final summation. The 10-bit output of the unit is the final intensity of the output pixel, ranging from zero to one.

**IV. IMPLEMENTATION ISSUES AND TIMING ANALYSIS**

The proposed hardware structure was designed, compiled, and simulated using the software package Quartus II programmable logic development system of Altera Corporation. The prototype of the system was chosen to be built on an FPGA [28], considering its advantageous characteristics, thereby enabling the modification of the logic design of the system by simply reconfiguring the device. Moreover, a careful selection for the chosen FPGA family was made. Altera’s device was chosen to be hardcopy compatible to have even more reduced cost in case of mass production. The system can scale up gray-scale images of 10-bit resolution with a maximum size of \( 1024 \times 1024 \) pixels to a maximum of 800% magnification.
The hardware system can facilitate input images with larger bit-depth and input resolutions after several hardware modifications. These hardware modifications are efficient and easily applied when taking advantage of the unique characteristic of FPGA reconfigurability. However, the device resources are finite, constraining the expendability to upper limits.

More specifically, for the manipulation of more than 10-bit-depth images, the hardware modifications that should be addressed are the increase of required input and output pins and an overall change to the hardware blocks. The blocks should be adapted to realize the required numerical calculations in terms of bit length. As can be seen in Fig. 5, the finite number of logic elements from the selected device limits the expendability to 11-bit-depth images, since there is a dramatic increase in the logic elements required by the hardware blocks. For larger depth images, a migration to a bigger and more expensive device must be addressed. In contrast to the logic element inefficiency, the required number of embedded memory and pins are adequate, as shown in Fig. 5.

For a larger input image resolution, the hardware modifications required are the increase of input pins which are available and a hardware change only to the coverage area and difference in intensity calculation unit. This means that all the resources would be adequate enough to facilitate larger input images. However, as can be seen in Fig. 6, working on larger sized images means that we violate the real-time constraint of 25 frames/s. The time required for the extraction of this image is $1.048 \times 10^5 \times 15 \times 10^{-9} = 0.015$ s. For a video sequence, the maximum frame rate would be 63 frames/s. The frame rate for different input image resolutions and scaling factors is depicted in Fig. 6. As can be seen, the system can perform a real-time scaling operation for a variety of input resolutions and magnification factors. However, when both the input resolutions and the magnification factors increase, the performance deteriorates, since an increasingly greater number of output pixels has to be computed.

V. SIMULATION RESULTS

To evaluate the performance of the proposed algorithm, we performed several comparison tests. Initially, we demonstrate the visual results, scaling the Lena by first downsampling the original image and then upsampling by the same algorithm to meet the initial dimensions. The results shown in Fig. 7 are a comparison of the proposed method to the standard algorithms of the nearest neighbor, bilinear interpolation, and bicubic interpolation and an area-based interpolation algorithm [19]. Fig. 7(a) represents the ground-truth image.

Although the visual results show good performance, it is well established that visual results fail to provide a decisive indication of the quality of the scaled images. For this reason, several alternative quantitative measurements related to picture quality have been chosen [30], [31]. In particular, we performed an evaluation of the high-frequency response. Contrary to the zero-order methods, which tend to produce jagged and non-continuous edges, and the spline-based approaches, which tend to produce blurry edges, the objective of the proposed method is to enlarge an image while keeping its step edges sharp and continuous. There are various types of edges in an image. However, step edges are visually more distinct than other types of edges. Therefore, the preservation of the sharpness and continuity of the step edges in the enlarging process is a key feature. To evaluate this, we scaled up a computer-generated input image that contains only two different area intensities.
of 225 and 25 by a factor of 1.5. After scaling up the image, a new unwanted area arises that smooths the transition from 225 to 25. A faster transition provides better performance. Fig. 8 illustrates the step-edge response from 225 to 25 for the 90° and 45° angles. Fig. 9 presents in a plot the 90° step-angle response. The proposed technique exhibits the second best response after the nearest-neighbor interpolation technique. However, nearest-neighbor interpolation produces sharp but noncontinuous edges, as shown in Fig. 7(b).

We also evaluated the interpolation root-mean-square error (RMSE) in several images through the aforementioned testing procedure of initially scaling down and then scaling up the image by the same scaling factor to obtain the original size. The RMSE results are presented in Table II and demonstrate an
overall superiority of the proposed method against the interpolations of nearest neighbor, bilinear, bicubic, and an area-based method [19] for all the test images.

VI. CONCLUSION

In this paper, we introduced a novel scaling technique and its hardware implementation. Specifically, we have proposed a new fuzzy approach, which utilizes the areas of the input pixels, overlapped by a dynamic mask, and the differences in intensity of the input pixels. The unified approach presents effective performance in edge regions while preserving the sharpness of the image, without the need for additional processing for edge identification. The interpolation is capable of scaling up or down both gray-scale and color images by any scaling factor. When compared to other scaling techniques, the experimental results of the proposed method prove an overall superiority through quantitative measurements. The hardware implementation of the algorithm, utilizing both an FIS and an image interpolation technique, was realized in an FPGA and is capable of scaling up gray-scale images of 10-bit resolution for a maximum magnification factor of 800%. The maximum input image size can be 1024 × 1024. The architecture is based on a sequence of pipeline stages and parallel processing to minimize computation time. The main features of the processor are the ability to accurately approximate the Gaussian membership functions used by the FIS with very few memory requirements and its high-frequency performance of 65 MHz, making it appropriate for real-time imaging applications.

REFERENCES


Fig. 9. Step-edge response from 225 to 25 pixel intensity after a scaling-up process. A faster transition gives better performance.

<table>
<thead>
<tr>
<th>Test image</th>
<th>Nearest interpolation</th>
<th>Bilinear interpolation</th>
<th>Bicubic interpolation</th>
<th>Area-based interpolation</th>
<th>Proposed interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>0.0448</td>
<td>0.0293</td>
<td>0.0236</td>
<td>0.0422</td>
<td>0.0225</td>
</tr>
<tr>
<td>Cameraman</td>
<td>0.0761</td>
<td>0.0556</td>
<td>0.0484</td>
<td>0.0725</td>
<td>0.0473</td>
</tr>
<tr>
<td>Tire</td>
<td>0.0542</td>
<td>0.0337</td>
<td>0.0263</td>
<td>0.0526</td>
<td>0.0245</td>
</tr>
<tr>
<td>Peppers</td>
<td>0.0296</td>
<td>0.0188</td>
<td>0.0153</td>
<td>0.0208</td>
<td>0.0139</td>
</tr>
<tr>
<td>Football</td>
<td>0.0457</td>
<td>0.0332</td>
<td>0.0304</td>
<td>0.0443</td>
<td>0.0271</td>
</tr>
</tbody>
</table>


Angelos Amanatiadis (S’00) received the Diploma degree from the Democritus University of Thrace, Xanthi, Greece, in 2004. He is currently working toward the Ph.D. degree with the Laboratory of Electronics, Department of Electrical and Computer Engineering, Democritus University of Thrace. His areas of interest include electronic systems design, machine vision, robotics, and real-time FPGA-based systems. He is involved in several national (Greek) and international (European) research projects in the field of real-time machine vision systems.

Mr. Amanatiadis is a member of the Technical Chamber of Greece (TEE).

Joannis Andreadis received the Diploma degree from the Democritus University of Thrace (DUTH), Xanthi, Greece, in 1983 (1978–1983 IKY Scholarship) and the M.Sc. and Ph.D. degrees from the University of Manchester Institute of Science and Technology, Manchester, U.K., in 1985 and 1989, respectively.

He has been with the Laboratory of Electronics, Department of Electrical and Computer Engineering, DUTH, since 1993. His research interests are mainly in electronic systems design, intelligent systems, and machine vision. In these areas, he is the author of more than 170 referred publications in book chapters, international journals, and conference proceedings. He has been an Associate Editor of the *Pattern Recognition Journal* since 1996.

Prof. Andreadis is a Fellow of the Institution of Engineering and Technology (formerly the Institution of Electrical Engineers) and a member of the Technical Chamber of Greece (TEE). He received the best paper award (computer vision and applications) from the 2007 Pacific Rim Symposium on Image and Video Technology.

Konstantinos Konstantinidis received the B.Sc. and M.Sc. degrees from the University of Sheffield, Sheffield, U.K., in 1999 and 2000, respectively. He is currently working toward the Ph.D. degree with the Laboratory of Electronics, Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece.

His research interests include 3-D modeling, robotics, and content-based image retrieval.

Mr. Konstantinidis is a member of the Technical Chamber of Greece (TEE).