14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

# IMAGE INTERPOLATION USING AN ADAPTIVE INVERTIBLE APPROACH

*Olivier Crave, Gemma Piella, and Béatrice Pesquet-Popescu*

ENST, Signal and Image Proc. Dept.
46, rue Barrault, 75634 Paris, FRANCE
phone: + (33) 1 45 81 73 27, fax: + (33) 1 45 81 71 44, email: {crave,piella,pesquet}@tsi.enst.fr
web: www.tsi.enst.fr

## ABSTRACT

In this paper we present a new image interpolation algorithm based on the adaptive update lifting scheme described in [1]. This scheme allows us to build adaptive wavelets that take into account the characteristics of the underlying signal. Inspired by this technique, we propose an image resizing scheme which has the ability to adapt itself to discontinuities like edges and assures a perfect reconstruction going from low to high resolution and then back to low resolution. Such a feature is highly desirable, for example, for forth and back conversion between the two existing High Definition Television formats, in order to preserve the integrity of the original image in a chain of successive transformations. The proposed algorithm adaptively updates one polyphase component of the original image and then computes the rest of the components of the output image by means of a gradient-driven interpolation method.

## 1. INTRODUCTION

Algorithms that preserve the sharpness of edges have already been studied in the past ([2]-[13]). In [2], [3], the interpolation techniques are based on a local analysis of the spatial structure to compute adaptive values for the interpolation coefficients in order to enhance the quality around the edges. In [4], the authors use variational methods to formulate the interpolation as the constrained minimization of a functional. In [5], they estimate the local covariance from the low resolution image and use this information to control interpolation at high resolution. In [6], they generate a high resolution edge map to direct the interpolation. In [7], they take into account the discontinuities and the sharp luminance variations while interpolating. In [8], they use a PDE-based constrained smoothing to reconstruct through iterations the geometric properties of the low resolution image. In [9], they interpolate from the pixel level data-dependent triangulation of the lower resolution image. However, in most cases, the low resolution image cannot be recovered from the interpolated one, because during the interpolation process, the information contained in the original image is not entirely preserved.

Our goal is to implement an algorithm capable of performing changes in resolution. It should be able to resample an image by a fractional factor $P/Q$. In particular, we want to apply it for up and down conversion of the two existing High Definition Television (HDTV) image formats ($1280 \times 720$ and $1920 \times 1080$). From $1280 \times 720$ to $1920 \times 1080$, we need an interpolation of $3/2$ in each direction. From $1920 \times 1080$ to $1280 \times 720$, we need a decimation of $2/3$ in each direction. We want to design a resampling process such that the output image is subjectively pleasant to the viewer (preserve as much as possible the image quality without introducing blocking artifacts and excessive smoothing). In addition, we impose the condition that when going from the $1280 \times 720$ format to the $1920 \times 1080$ one, and then going back to $1280 \times 720$, the original image is retrieved. This means that during the interpolation process there is no loss of information.

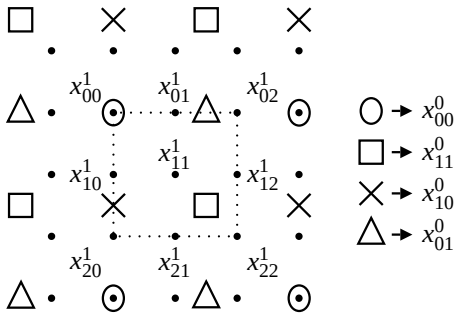


Figure 1: Polyphase components of the original image $x^0$ and of the interpolated image $x^1$.

The interpolation algorithm designed to this end is based on the adaptive update lifting scheme described in [1]. This scheme allows us to build adaptive wavelets that take into account the characteristics of the underlying signal like in this original work of Gerek and Çetin [14]. Inspired by this technique, we propose an image resizing scheme which has the ability to adapt itself to discontinuities like edges. Moreover, it assures perfect reconstruction when going from low to high resolution and then back to low resolution.

The paper is organised as follows: in the next Section, we present the content-adaptive interpolation and reverse operation algorithms of our invertible approach. In Section 3, we provide some simulation results and comparisons and we conclude in Section 4.

## 2. ADAPTIVE APPROACH

Let $x^0$ be the original image. We want to interpolate $x^0$ by a rational factor $3/2 \times 3/2$ to obtain $x^1$. Let us decompose $x^0$ into its four polyphase components: $x^0_{00}$, $x^0_{01}$, $x^0_{10}$, $x^0_{11}$. Note that using a similar notation, $x^1$ will have nine components $x^1_{00}$, $x^1_{01}$, $x^1_{02}$, $x^1_{10}$, $x^1_{12}$, $x^1_{20}$, $x^1_{22}$ as shown in Fig. 1. Here, the black dots represent the grid of the output image $x^1$, while the rest of the markers represent the grid of the input image $x^0$.

The method proposed in this paper will be described in the form of two content-adaptive algorithms: an interpolation algorithm which transforms $x^0$ in $x^1$ and a downsampling al-

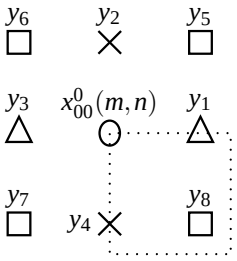


Figure 2: Neighborhood of $x$.

gorithm which makes the inverse operation.

### 2.1 Adaptive interpolation

The interpolation algorithm adaptively updates one polyphase component of the original image and then computes the rest of the components of the output image by means of a gradient-driven method. It works in five successive steps as described below.

**1.** We first compute $x^1_{00}$ by updating $x^0_{00}$ in an adaptive way (see [1] for a detailed description). The basic idea is to choose the update filter depending on a local gradient. In this way, only homogeneous regions are smoothed while discontinuities are preserved.

Let us consider a $3\times 3$ neighborhood around sample $x = x^0_{00}(m,n)$, and label its neighbours by $y_1,\ldots,y_8$, as it is shown in Fig. 2. We obtain $x^1_{00}$ as

$$x' = x^1_{00}(m,n) = \alpha_d x + \sum_{j=1}^{8} \beta_{j,d} y_j, \qquad (1)$$

where $\alpha_d$ and $\beta_{j,d}$ are some predefined weights such that $\alpha_d + \sum_{j=1}^{8} \beta_{j,d} = 1$ and $d$, the decision parameter, depends on a local gradient vector $v \in \mathbb{R}^8$ with components $v_j = x - y_j$, $j \in \{1,\ldots,8\}$. We give some examples of how to compute $d$ and the weights in Section 3.

With this update step, we have obtained one polyphase component of the output image $x^1$. We compute the rest of the polyphase components by a gradient-driven interpolation as it is described in the following steps.

**2.** From $x^1_{00}$, The gradient function $g^0_a$ is as follows:

$$g^0_a = (1-\Delta G_x)\cdot(1-\Delta G_y)$$

where $\Delta G_x, \Delta G_y$ are the normalized gradients of $x^1_{00}$ in the horizontal and vertical directions, respectively. This gradient function is interpolated using a bilinear method and a resizing rate of 2 into a new gradient function $g^1_a$ to get estimated values of the gradient function for all the pixels of the original image. The function $g^1_a$ is decomposed into its four polyphase components $g_{a,00}$, $g_{a,10}$, $g_{a,01}$, $g_{a,11}$.

This step aims to detect possible sharp variations in the neighborhood of the samples which will be used for interpolation. High values of $g^1_a$ correspond to smooth regions, whereas small values of $g^1_a$ indicate a possible edge. We will use these values to perform the interpolation along edges and not across them.

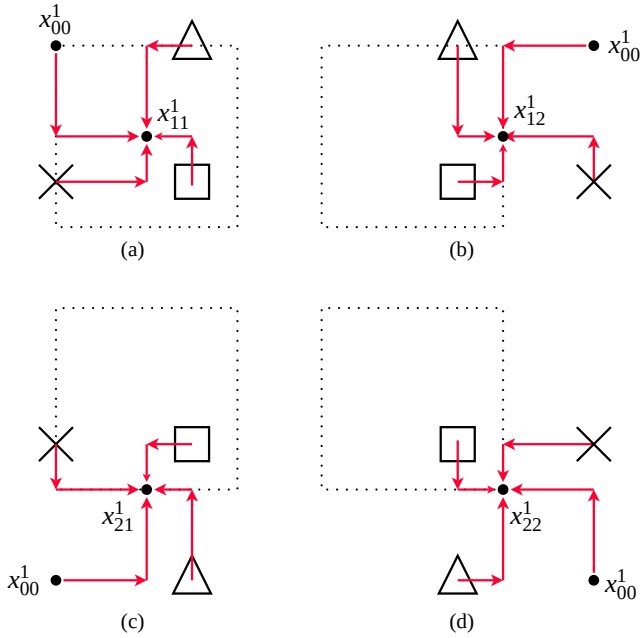


Figure 3: Computation of (a) $x^1_{11}$, (b) $x^1_{12}$, (c) $x^1_{21}$ and (d) $x^1_{22}$.

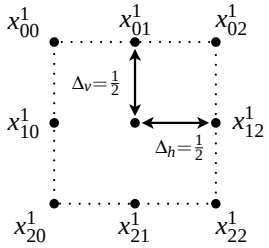


Figure 4: Distance computation.

**3.** Next, we compute $x^1_{11}$. We assume that each sample of $x^1_{11}$ is computed by a combination of the four closest available samples as illustrated in Fig. 3(a). That is,

$$\begin{aligned} x^1_{11}(m,n) = {} & w_{a0}x^1_{00}(m,n) + w_{a1}x^0_{01}(m,n) \\ & + w_{a2}x^0_{11}(m,n) + w_{a3}x^0_{10}(m,n). \end{aligned} \qquad (2)$$

The weights $w_{aj}$ have two components: one related to $g^1_a$ and the other to the spatial "distance" $r_j$ (between the given sample and $x^1_{11}(m,n)$). In particular,

$$r_j = (1-\Delta_{h_j})\cdot(1-\Delta_{v_j}).$$

Here $\Delta_{h_j}$ denotes the horizontal distance and $\Delta_{v_j}$ the vertical one. They have been normalized such that in the interpolated image, the distance between two horizontal (resp. vertical) aligned pixels is $\Delta_h = 1/2$ (resp. $\Delta_v = 1/2$). This is illustrated in Fig. 4.

Each weight $w_{aj}$ is then obtained as the product of $r_j$ by the value of the gradient $g^1_a$ at that position. Hence,

$$w_{a0} = \frac{1}{4}g_{a,00}\ ,\ w_{a1} = \frac{3}{8}g_{a,01}\ ,\ w_{a2} = \frac{9}{16}g_{a,11}\ ,\ w_{a3} = \frac{3}{8}g_{a,10}.$$

By symmetry, we obtain $x^1_{12}, x^1_{21}, x^1_{22}$ as can be seen in Fig. 3(b), 3(c) and 3(d).

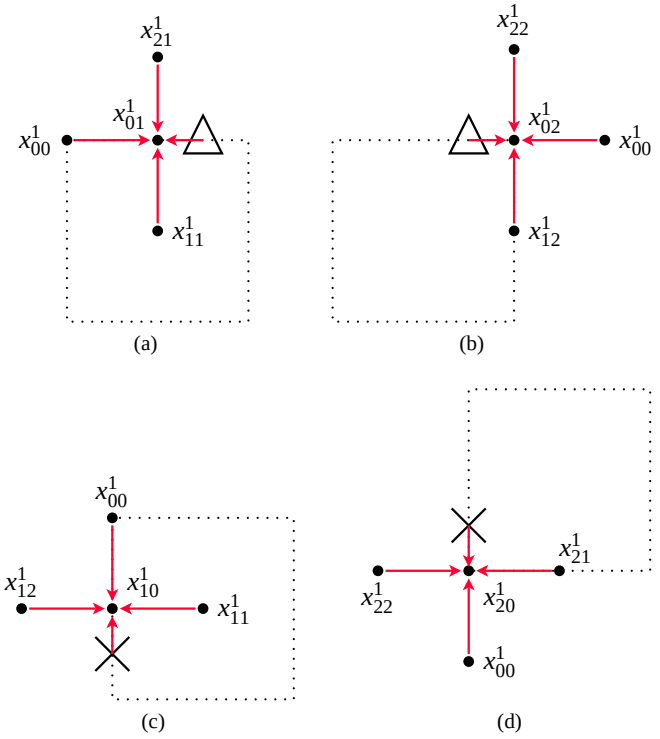


Figure 5: Computation of (a) $x^1_{01}$, (b) $x^1_{02}$, (c) $x^1_{10}$ and (d) $x^1_{20}$.

**4.** From $x^1_{00}$, $x^1_{11}$, $x^1_{12}$, $x^1_{21}$ and $x^1_{22}$ we compute a new gradient function $g^0_b$. It is computed in the same way as $g^0_a$ was obtained from $x^1_{00}$. Function $g^0_b$ is then upsampled using a bilinear interpolation and a resizing rate of $3/2$. We obtain a second gradient function $g^1_b$ of the size of $x^1$. Functions $g^0_b$ and $g^1_b$ are decomposed into their four and nine polyphase components $g^0_{b,00}$, $g^0_{b,10}$, $g^0_{b,01}$, $g^0_{b,11}$ and $g^1_{b,00}$, $g^1_{b,10}$, $g^1_{b,01}$, $g^1_{b,11}$, $g^1_{b,02}$, $g^1_{b,12}$,$g^1_{b,22}$, $g^1_{b,21}$, $g^1_{b,20}$, respectively.

**5.** Finally, we calculate $x^1_{01}$. As before, we assume that each interpolated sample is computed by a combination of the four closest available samples (see Fig. 5(a)):

$$x^1_{01}(m,n) = w_{b0}x^1_{00}(m,n) + w_{b1}x^0_{01}(m,n) + w_{b2}x^1_{2,1}(m-1,n) + w_{b3}x^1_{11}(m,n). \quad (3)$$

The weights $w_{bj}$ are obtained in the same way as $w_{aj}$ were obtained in step 3. Now, the $r_j, j = 0,\ldots,3$ components are $\frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{2}$. Thus,

$$w_{b0} = \frac{1}{2}g^1_{b,00}\ ,\ w_{b1} = \frac{3}{4}g^0_{b,01}\ ,\ w_{b2} = \frac{1}{2}g^1_{b,21}\ ,\ w_{b3} = \frac{1}{2}g^1_{b,11}.$$

By symmetry, we obtain the rest of the polyphase components, $x^1_{02}$, $x^1_{10}$ and $x^1_{20}$ as shown in Fig. 5(b), 5(c) and 5(d).

### 2.2 Adaptive inverse algorithm

The proposed algorithm is invertible, which means that the reconstructed image after downsampling should be identical to the original image, after interpolation and downsampling. The downsampling algorithm will therefore go through all the steps of the interpolation algorithm in reverse order.

**1.** We begin by computing the gradient functions $g^0_b$, $g^1_b$ as it was described in step 4 of the interpolation algorithm.

**2.** By inversing (3), we get

$$x^0_{01}(m,n) = \frac{1}{w_{b1}}\big(x^1_{01}(m,n) - w_{b0}x^1_{00}(m,n) - w_{b2}x^1_{2,1}(m-1,n) - w_{b3}x^1_{11}(m,n)\big).$$

The weights $w_{bj}$ are computed (as in the interpolation algorithm) using the values of $g^0_b$, $g^1_b$ and $r_j$. By symmetry, we get $x^0_{10}$.

**3.** We then compute a gradient function $g^1_a$ identical to the one computed in step 2 of the interpolation algorithm.

**4.** By inversing (2), we obtain

$$x^0_{11}(m,n) = \frac{1}{w_{a2}}\big(x^1_{11}(m,n) - w_{a0}x^1_{00}(m,n) - w_{a1}x^0_{01}(m,n) - w_{a3}x^0_{10}(m,n)\big).$$

The weights $w_{aj}$ are computed using the polyphase components of $g^1_a$ and $r_j$.

**5.** We still need to obtain $x^0_{00}$. Note that in equation (1), the weights depend on the decision parameter $d$, which has been computed from $x^0_{00}, x^0_{01}, x^0_{10}, x^0_{11}$. However, at this point, we do not know $x^0_{00}$ but 'only' its update form $x^1_{00}$. We have shown in [1] that, under some special circumstances, it is possible to recover $d$ from $x^1_{00}$ and $x^0_{01}, x^0_{10}, x^0_{11}$, and hence invert (1):

$$x = x^0_{00}(m,n) = \frac{1}{\alpha_d}\big(x' - \sum_{j=1}^{8}\beta_{j,d}y_j\big). \quad (4)$$

Those circumstances will be briefly defined in the next section, in the case of two adaptive schemes that will serve to illustrate the algorithm.

## 3. SIMULATION RESULTS

We restrict ourselves to the case where $d$ can only take two values. In particular, at the interpolation stage, $d$ is obtained by thresholding the seminorm of the local gradient $v$:

$$d_{\text{interpolation}} = [p(v) > T],$$

where $[P]$ returns 1 if the predicate $P$ is true and 0 if it is false, $p$ is a seminorm and $T$ a given threshold. Analogously, at the decimation stage,

$$d_{\text{decimation}} = [p(v') > T'],$$

where $v' \in \mathbb{R}^8$ is a local gradient vector with components $v'_j = x' - y_j$. From [1], we know necessary and sufficient conditions which guarantee there exists a threshold $T'$ such that $d_{\text{interpolation}} = d_{\text{decimation}}$ for every pixel.

### 3.1 Example 1

We first present an example of applying the adaptive interpolation algorithm with $p$ being a laplacian seminorm. That is,

$$p(v) = |\sum_{j=1}^{4} v_j|.$$

It is a particular case of a weighted gradient seminorm,

$$p(v) = |a^T v|$$

where $a \in \mathbb{R}^{N*}$.

The necessary and sufficient conditions for the threshold criterion to hold in the weighted gradient case is that $\beta_d$ and $a$ are colinear and $|\alpha_0| \leq |\alpha_1|$. The proof is given in [1]. In the case of a laplacian seminorm, where $a = (1,1,1,1,0,0,0,0)^T$, the necessary and sufficient conditions are reduced to $\beta_{d,j} = \beta_d$ for $j = 1,\ldots,4$ and $|1-4\beta_0| \leq |1-4\beta_1|$.

We choose $\beta_0 = 1/8$ and $\beta_1 = 0$. Hence, in smooth regions ($d = 0$), updated sample values $x^1_{00}$ will be obtained by computing the average of $x^1_{00}$ with the average of its eight neighbours. Whereas in regions where the gradient is higher ($d = 1$), there will be no update, i.e., $x^1_{00} = x^0_{00}$.

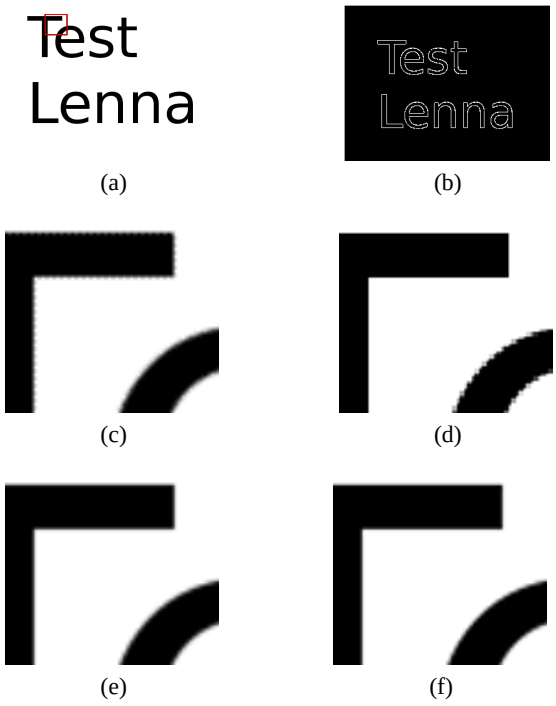


Figure 6: a) Original image. b) Decision map using a threshold $T = 30$. c)-f) Resized images using c) a laplacian seminorm and a threshold $T = 30$, d) a nearest neighbour interpolation, e) a bilinear interpolation, f) a bicubic interpolation.

We take as input the synthetic image shown at Fig. 6 (a). The decision map using a threshold $T = 30$ is shown at Fig. 6 (b). Here the black pixels correspond to $d = 0$ (smooth regions) whereas white pixels correspond to $d = 1$. Fig. 6 (c) shows the interpolated image obtained using our approach. For comparison, Fig. 6 (d)-(f) show the results obtained with some linear techniques, namely, the nearest neighbour, bilinear and bicubic interpolations. The effects of the non-linear interpolation of our method are visible on the edges. They are smoother on the interpolated image obtained with $T = 30$ than on the one obtained with a higher threshold such that $d = 0$ everywhere (i.e., no adaptive update), as can be seen in Fig. 7.


Figure 7: Resized images a) with a non adaptive update, b) with an adaptive update.

### 3.2 Example 2

For our second example we choose the seminorm:

$$p(v) = \Big(\sum_{j=1}^{4} |v_j|^2 + \frac{1}{2}\Big(\sum_{j=5}^{8} |v_j|^2\Big)\Big)^{1/2}. \quad (5)$$

It is a member of the family of quadratic seminorms defined by

$$p(v) = (v^T M v)^{1/2}, \quad v \in \mathbb{R}^N, \quad (6)$$

where $M$ is a symmetric positive semi-definite matrix. In our case, $M$ is a diagonal matrix with strictly positive entries $M_{jj} = \lambda_j$ for all $j$: $M = \mathrm{diag}(1,1,1,1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2})$. In this case, perfect reconstruction is guaranteed [1] if we choose $\beta_d = \mu_d(1,1,1,1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2})^T$. We take $\mu_1 = 0$ (i.e., no update when the gradient is high). If we assume the input signal to be contaminated by additive uncorrelated Gaussian noise, we must take

$$\mu_0 = \frac{\sum_j \lambda_j}{\sum_j \lambda_j^2 + (\sum_j \lambda_j)^2}$$

in order to minimize the noise variance. That gives us $\mu_0 = 6/41$.

Fig. 8 shows the image "house" interpolated using our adaptive method, nearest neighbour, bilinear and bicubic [15] techniques. The interpolated images have been zoomed in order to see the details. The nearest neighbour image is by far the worst of all with a lot of aliasing. The adaptive scheme gives an image close to the one obtained using a bilinear method, while the bicubic one still presents the best quality. However, it is important to notice that the non-adaptive methods do not allow to retrieve the original image after downsampling, as it does using the proposed approach. Fig. 3.2 shows the interpolated "house" image obtained with the adaptive method using a quadratic seminorm ($T = 70$).

Using the proposed method in the non adaptive update case ($d = 0$ over the entire image), the images are very blurry, while if one chooses a lower threshold, the edges tend to be sharper, the maximum being reached when $x^1_{00} = x^0_{00}$ ($d = 1$ everywhere). In that last case, the images still look a little bit blurry, and are of the same quality as images obtained by a bilinear interpolation.

## 4. CONCLUSION

In this paper, we have presented a novel image interpolation algorithm. The proposed algorithm adaptively updates one polyphase component of the original image and then computes the rest of the components of the output image by means of a gradient-driven interpolation. The major characteristic of this method is that it is invertible. The experiments

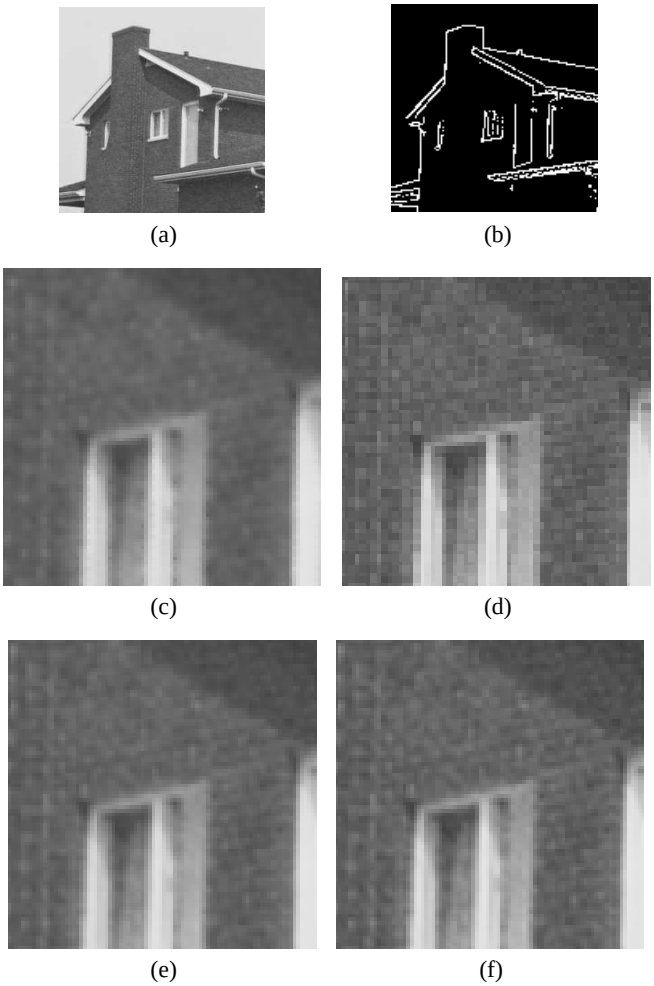

Figure 8: a) Original image. b) Decision map using a threshold $T = 70$. c)-f) Resized images using c) a quadratic seminorm and a threshold $T = 70$, d) a nearest neighbour interpolation, e) a bilinear interpolation, f) a bicubic interpolation.

show that the proposed method is comparable, in quality, to bilinear interpolation.

Figure 9: Interpolated image using our approach with a quadratic seminorm and a threshold $T = 70$.

## REFERENCES


[1] H.J.A.M. Heijmans, B. Pesquet-Popescu and G. Piella, "Building nonredundant adaptive wavelets by update lifting," *Applied and Computational Harmonic Analysis*, no. 18, pp. 252–281, May 2005.

[2] V.R. Algazi, G.E. Ford and R. Potharlanka, "Directional interpolation of images based on visual properties and rank order filtering," *Proc. of the IEEE Int. Conf. Acoustics, Speech, Sig. Process*, pp. 3005-3008, May 1991.

[3] S.W. Lee, J.K. Paik, "Image interpolation using adaptive fast B-spline filtering," *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Process*, pp. 177-179, 1993.

[4] R.R. Schultz and R.L. Stevenson, "A Bayesian Approach to Image Expansion for Improved Definition," *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 233–242, 1994.

[5] X. Li and M. T. Orchard, "New Edge-Directed Interpolation," *IEEE Transactions on Image Processing*, vol. 10, No. 10, pp. 1521-1527, October 2001.

[6] J. Allebach and P.W. Wong, "Edge-directed interpolation," *Proc. IEEE Int. Conf. Image Processing*, vol. 3, pp. 707–710, 1996.

[7] S. Battiato and G. Gallo and F. Stanco, "A Locally-Adaptive Zooming Algorithm for Digital Images," *Elsevier Image Vision and Computing Journal*, vol. 20/11, pp. 805–812, Sept. 2002.

[8] B.S. Morse and D. Schwartzwald, "Isophote-based interpolation," *Proc. IEEE Int. Conf. Image Processing*, vol. 3, pp. 227–231, 1998.

[9] D. Su and P. Willis, "Image Interpolation by Pixel Level Data-Dependent Triangulation," *Computer Graphics Forum*, vol. 23, pp. 189-201, June 2004.

[10] K. Jensen and D. Anastassiou, "Subpixel Edge Localization and the Interpolation of Still Images," *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 285–295, 1995.

[11] S.D. Bayrakeri and R.M. Mersereau, "A New Method for directional Image Interpolation," *Proc. Int. Conf. Acoustics, Speech, Sig. Process*, vol. 4, pp. 2383–2386, 1995.

[12] S.G. Chang, Z. Cvetkovic and M. Vetterli, "Resolution Enhancement of Images Using Wavelet Transform Extrema Interpolation," *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Process*, pp. 2379–2382, May 1995.

[13] D.D. Muresan and T.W. Parks, "Adaptive, optimal-recovery image interpolation," *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Process*, vol. 3, pp. 1949–1952, May 2001.

[14] O.N. Gerek and A.E. Cetin, "Adaptive polyphase subband decomposition structures for image compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 9, pp. 1649–1660, Oct 2000.

[15] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981.