Image Interpolation using Mathematical Morphology

Alessandro Ledda[1], Hiêp Q. Luong[1], Wilfried Philips[1], Valérie De Witte[2] and Etienne E. Kerre[2]
Ghent University, Gent, Belgium
(1) Telin Department, (2) Department of Applied Mathematics & Computer Science
ledda@telin.UGent.be

Abstract

We present a new method for interpolating binary images that outperforms existing techniques. Bitmapped images have a specific horizontal and vertical resolution. When we magnify such an image, we want the resolution to be increased, allowing more details in the image. However, these extra details are not present in the original image. A blowup of the image using simple interpolation will introduce jagged edges, also called "jaggies".

We present a new interpolation technique "mmINT", which avoids these errors. It is based on mathematical morphology, a theoretical framework to alter an image while preserving the image objects' geometry. The algorithm detects jaggies in the blown up image and removes them, making the edges smoother. This is done by replacing specific black pixels with white pixels, and vice versa.

The results show that mmINT is a superior technique for the interpolation of binary images, like logos, diagrams, cartoons and maps.

1. Introduction

A digital bitmapped image consists of picture elements, *pixels*, aligned on a grid. The image resolution is the number of pixels per row and per column. When the image is magnified M times, the number of pixels is also increased (M^2 times). But we only know the pixel values of the original pixels. The values of the new pixels must be guessed using some intelligent calculation.

The easiest way is simply to copy the existing pixel values to the new neighbouring pixels (figure 1). This is called *pixel replication* or *nearest neighbour interpolation*. Every old pixel looks like a blown up pixel, introducing unwanted jagged edges, called *jaggies*.

Other techniques are for example the *bilinear* and *bicubic interpolation* [5]. Here, the (weighted) mean of respectively 4 and 16 closest neighbours is calculated for the new pixel value. Other linear (or non-adaptive) methods use higher order (piecewise) polynomials, B-splines, truncated or windowed sinc functions, etc. Most of them create a greyscale image with extra artefacts, like blurring and/or ringing. These extra artefacts do not occur if the interpolation result is a binary image.

(a) Original image (b) 4× magnified

Figure 1. Pixel replication creates "jaggies".

Adaptive or non-linear interpolation methods incorporate a priori knowledge about images. The edge-based techniques follow a philosophy that no interpolation across the edges in the image is allowed or interpolation has to be performed along the edges. This rule is employed for instance in EDI [1], NEDI [6] and AQua [10]. The restoration methods tackle unwanted interpolation artefacts. Some restoration methods are PDE-based regularization [15], isophote smoothing [9] and level curve mapping [7]. Some other adaptive methods exploit the self-similarity property of an image, e.g. iterated function systems [4]. Another class of adaptive interpolation methods is the example-based approaches, which map blocks of the low-resolution image into pre-defined interpolated patches [14, 2]. Adaptive methods still suffer from artefacts: their results often look segmented, yield important visual degradation in fine textured areas or random pixels are created in smooth areas.

In this paper we present a novel non-linear interpolation technique for black-and-white images that performs very well. In short, we remove the jaggies from a pixel replicated image, by swapping the values of the corner pixels of

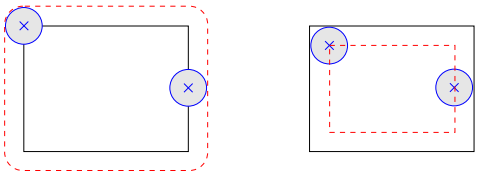


Figure 2. Schematic example of the basic morphological operators. Solid line: original object; Dashed line: result object; Disc: structuring element (cross: origin); Left: dilation; Right: erosion.

(a) Strel *B* (foreground) (b) Strel *C* (background)

Figure 3. Upper-left corner detection with the hit-miss transform. Specific structuring elements are used. The black squares are pixels of the strel; the cross is the origin of the strel.

those jagged edges. This technique is based on mathematical morphology. Therefore we call our new method *mmINT* (*Mathematical Morphological INTerpolation*).

2. Theoretical background

2.1. Morphological operators

Mathematical morphology [12, 13, 3], originally a theory for binary images, is a framework for image processing based on set theory. An object can be represented as a set, i.e. the set of the coordinates of all the pixels of the object. Objects are connected areas of pixels with value 1, whereas the background pixels have value 0. Morphological image processing can simplify image data, preserving the objects' essential shape characteristics, and can eliminate irrelevant objects.

Binary mathematical morphology is based on two basic operators: *dilation* and *erosion*. They are defined in terms of a *structuring element* (short: *strel*); this is a small window with an origin that scans the image and alters the pixels based on the window's content. A *dilation* of an image A with a structuring element B ($A \oplus B$) enlarges the objects (it increases the number of 1-pixels in the image). An *erosion* ($A \ominus B$) shrinks objects (the number of 1-pixels in the image decreases) (see figure 2).

The basic morphological operators on set A with strel B are defined as:

$$\begin{array}{rl} dilation: & A \oplus B = \bigcup_{\vec{b} \in B} T_{\vec{b}}(A) \\ erosion: & A \ominus B = \bigcap_{\vec{b} \in B} T_{-\vec{b}}(A) \end{array} \qquad (1)$$

with $T_{\vec{b}}(A)$ the translation of image A over the vector $\vec{b}$. In other words, the structuring element is positioned over every object pixel (see figure 2). With a dilation, every pixel that is part of the strel, positioned at an object pixel, will be part of the dilated object. With an erosion, only the pixels where the whole strel is part of the object will be part of the eroded object.

The structuring element B can be of any size or shape (square, cross, disc, line, ...) and is chosen depending on the image and the application. The origin of this strel is also important, as it states how the strel is positioned relative to the examined pixel.

2.2. The hit-miss transform

The *hit-miss transform* "$\otimes$" is a morphological method used in our algorithm. Two structuring elements are needed: one for the erosion of object pixels (B), and one for the erosion of background pixels (C). With this transform it is possible to detect specific shapes in the image. Its formula is:

$$A \otimes (B, C) = (A \ominus B) \cap (A^c \ominus C) \qquad (2)$$

with A^c the complement set of A (1 becomes 0 and vice versa).

For example, if we wish to detect the upper-left corner of an object, then we erode the image with a structuring element B like the one in figure 3(a), resulting in $A \ominus B$ (see figure 4). If an object pixel has a right and lower object neighbour, then this pixel is kept by the erosion. It is irrelevant if other neighbours are object pixels. We also erode the complement of the image with the structuring element C, shown in figure 3(b), resulting in $A^c \ominus C$. Notice that the origin of the structuring elements plays an important role: it sets the position of the possible corner pixel. Here, if the left and upper neighbour of the pixel are background (in the original image, foreground in A^c), then this pixel is kept by the erosion. Whether the pixel itself or its other neighbours are background pixels is not important.

The intersection of both erosions shows where upper-left corners are in the image. The result of the hit-miss transform is a set of the foreground pixels with at their right and below them foreground pixels and at their left and above them background pixels. We will further refer to this set as a *corner map*.

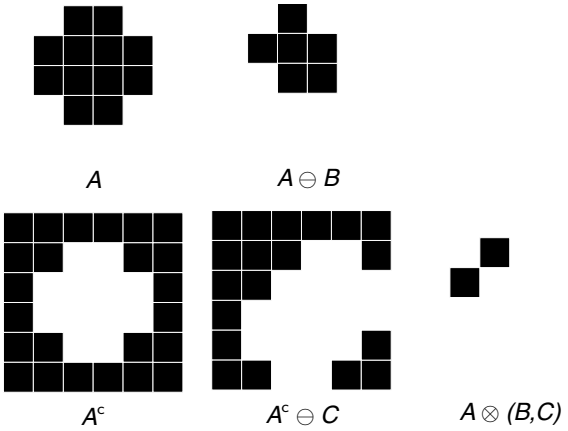


Figure 4. An example of the hit-miss transform, detecting upper-left corners (using the strels from figure 3).

3. Methodology

The purpose of *mmINT* is to remove the jagged edges from a pixel replicated image, by changing specific object pixels to background and vice versa. The pixel replication of figure 5(a) is jagged (see section 3.1). The dotted lines show the initial orientation. The ideal solution would be to fill the area between the dotted lines with object pixels and to have background pixels outside this area. This filling is done in different iteration steps.

Figure 5(c) shows the first iteration of our method. The hit-miss transform is used to detect corners in the magnified image that are possibly jaggies (see section 3.2).

Not all detected corners need to be changed: some are *real corners* of image objects, not part of a jagged edge; other introduce unwanted artefacts. We will discuss the detection and exclusion of these corners in sections 3.3 and 3.4.

We change the value of the pixels (and surrounding neighbours) that are detected as corners of a jagged edge using a morphological dilation (section 3.5), with the structuring element depending on the used magnification, the current iteration step and of course the orientation of the corner. When changing the values, black becomes white, and white becomes black.

At this point in the algorithm, the line is not yet completely smooth. The small arrows in the figure point to the next set of pixels that need to change value. The orientations of the arrows also show how the pixel swapping evolves: we start from a corner, but every iteration step the corner moves away and its shape changes. Therefore we use different structuring elements for every iteration step (section 3.6).

This procedure is repeated until all appropriate changes have been made. Lines or edges with angles near 0° and 90° will need more iteration before all jaggies are gone. We will now discuss the method in more detail.

3.1. Interpolation by pixel replication

First, we magnify the image by an integer factor using *pixel replication* (also known as *nearest neighbour interpolation*). Pixel replication is a simple interpolation technique: it merely copies every pixel to its nearest neighbours. These neighbouring pixels were not available in the original low resolution image. As a result, a blocked pattern (jagged edges or *jaggies*) is visible (see the filled object in figure 1 for an example). Next, we will remove those jaggies and let the lines appear less jaggy, like in figure 5.

3.2. Corner detection

If we examine figure 5(b), we notice that the jaggies are at the locations of object and background corners. We need the positions of these corners in order to remove the jaggies.

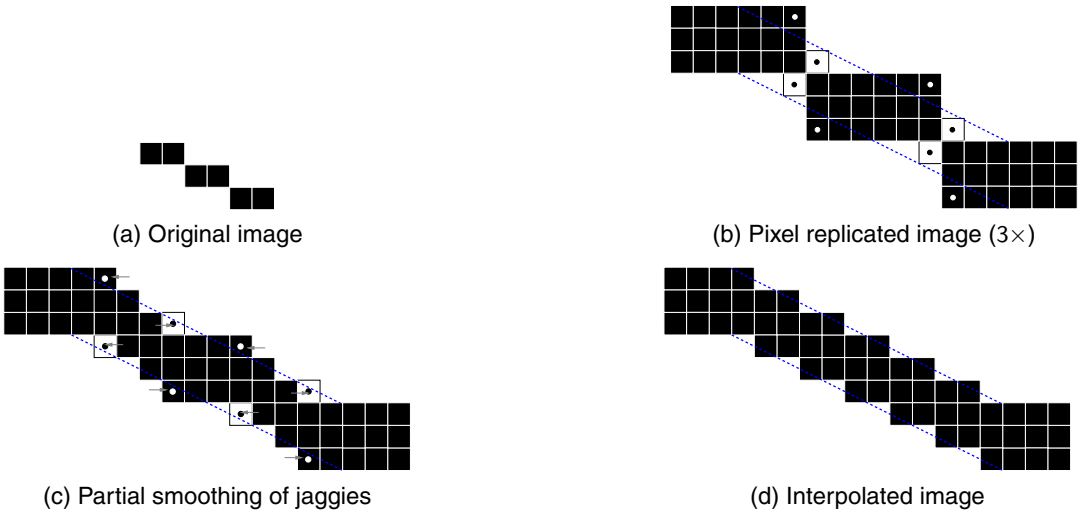
The morphological hit-miss transform is a very useful tool for corner detection. We choose a couple of structuring elements that represent an object corner. Figure 3 shows the structuring elements used for the detection of an upper-left corner. The other 3 corners (upper-right, lower-left and lower-right) are detected using rotated versions of these elements.

We not only look for corners of the objects, but also for corners of the background. This way, we will have 8 corner maps (4 *object* corner maps and 4 *background* corner maps).

3.3. Corner validation

Not all the corners found with the method described in section 3.2 will need to be changed. Some corners are *real corners*, which have to be retained in the interpolated image. For example, the corners of the door and walls in figure 6 are real corners. The corners detected at the roof are *jagged corners*, because it is a diagonal line.

We have to determine which corner pixels should be affected by the interpolation part (section 3.5) (the jagged corners) and which not (the real corners). To distinguish between both, we search for every detected corner pixel one or more *complementary corner pixels*. A complementary corner is a corner of the opposite colour that lies either in the direct neighbourhood of the corner pixel or at specific relative coordinates along the direction of the line or edge. The existence of a complementary corner suggests a jagged edge. These two types of complementary corners imply two validation rules to exclude the real corners. We need to



(a) Original image

(b) Pixel replicated image (3×)

(c) Partial smoothing of jaggies

(d) Interpolated image

Figure 5. The jagged edges have to be removed, by replacing the values of specific pixels. The dotted lines show the orientation of the original line (a). The dots show the pixels that will change value after interpolation.

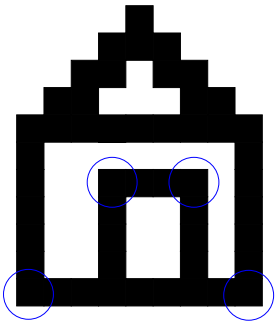


Figure 6. The difference between "jagged corners" and "real corners" (encircled).

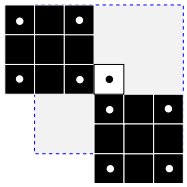


Figure 7. Complementary corners in a window (size 2*M* – 1). Black dot: background corner; White dots: object corners.

check if the corner pixels satisfy at least one of these two rules. We will now discuss both validation rules in more detail.

3.3.1. Validation rule 1: search for complements in a window. If a thin line with a thickness of 1 pixel in the original image shows jaggies in the pixel replicated image, then we can take advantage of corner pixels at the other side of the line to determine whether or not the jagged edge has to be interpolated. While the complementary pixels in section 3.3.2 move away with each iteration, here the complementary pixels move along.

Figure 7 shows a background corner pixel and the union of all the object corner maps. We take the intersection of this map union and the dilation of this background corner with a square structuring element (the window in the figure). If this intersection is not empty, then the corner is a jagged corner. The same approach is taken for the object corners.

The size of the structuring element is $2M + 1$, with M the used magnification. To classify corners like the upper-left and lower-right pixel in figure 7 as real corners, size $2M - 1$ must be used in the first iteration step. Otherwise, these pixels lie inside the window around the background corner pixel and thus will be classified as jagged corners.

The advantage of this method is that jaggies for a thin line are removed. But it only works for thin lines, since the algorithm looks at complementary corners in the direct neighbourhood of the pixel, even at a higher iteration step. In the case of filled objects (no thin lines), with each iteration step, the possible complementary corners move away from the examined corner pixel. This corner pixel will be classified as a real corner and will not be interpolated.

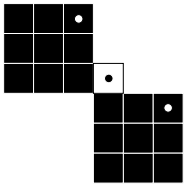


Figure 8. Complementary corners: the background corner (black dot) has 2 complementary corners (white dots) at specific relative coordinates.

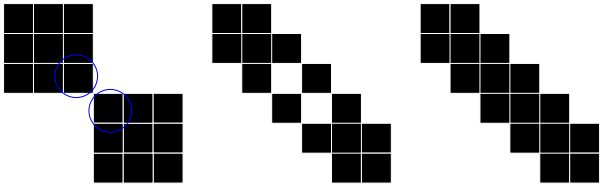


Figure 9. Barely touching pixels could give artefacts after interpolation. Hole filling (H-F) prevents this.

3.3.2. Validation rule 2: search for specific complements. A complementary corner pixel is located at specific coordinates relative to the examined corner pixel. These coordinates can be calculated by taking the magnification, the iteration step and the orientation of the corner into account. The complementary corner is located $M-1$ pixels in one direction and $1+(\theta-1)(M-1)$ (with θ the iteration step) pixels in the other direction, with the directions depending on the corner orientation.

If we have an upper-right corner pixel, then we have to look for a lower-left corner pixel of the background. Figure 8 shows a part of a diagonal line, $3\times$ magnified. If the background corner pixel changes to object, then there must also change one (or in the first iteration step possibly two, as in figure 8) object corner pixel to background, in order to keep the total number of foreground pixels, the global image intensity, (quasi) constant. This complementary pixel lies in the direction of the line or edge, as stated before.

The advantage of this method is that only one complementary pixel is needed to determine if the corner pixel is a real corner or a jagged corner, which means this part of the algorithm only takes a small amount of calculation time. The disadvantage is that in the result, some corners that are part of a jagged edge are not removed because the validation rule is too strict. Indeed, while removing jaggies along an edge, the complementary pixels move away from each other. It is possible that one of these pixels reaches the end of the edge before the other one does. If at the next iteration step the latter corner is detected, no complementary pixel will be found anymore and it will be considered a real corner, although jaggies might still be present.

3.3.3. Combination of the validation rules. By allowing corners validated by either method 3.3.1 or method 3.3.2, we can obtain smooth lines and at the same time interpolate more solid objects. This combination shows better results. The only disadvantage is an increase in calculation time.

3.4. Hole filling

After the previous algorithm steps, some of the corners detected as jagged corners are not wanted, because they cause artefacts. We will explain this type of artefacts and we will remove them. This procedure is performed once, during the first iteration step.

When we have a line with barely touching pixels, like in figure 9(a), then some detected corners should not be there. The encircled pixels in the image are detected as object corners, but also their neighbours are detected as background corners. All these corners will change values later on (section 3.5), and thus holes will be introduced (see figure 9(b)).

This hole filling part removes these corners from the corner maps obtained in section 3.3. We perform a hit-miss transform on the union of the pixel replicated image with all corner maps, both from objects and background. Within this union, we look for cross-like elements with a hole in the middle; in other words, pixels, whether or not value 1 or 0, that have four nearest neighbours with value 1. These pixels are not allowed to change during the interpolation sequence.

This part of the algorithm introduces an asymmetry between the black and white pixels: the object and background pixels are not treated in the same way. At this moment it is important to know which pixels represent the background and which ones the foreground, because figure 9(c) would look quite different if the black pixels are indicated as background. Therefore we count the white and the black pixels in the image and the colour that is most present is considered background, which indeed is mostly the case. The hole filling step adds more foreground pixels than background pixels to the magnified image.

Another solution is the use of other (more strict) structuring elements for the corner detection. This can exclude the encircled object pixels in figure 9(a) as corners during the corner detection step itself, but it will also exclude their neighbouring background pixels as corners. Therefore this solution is less desirable.

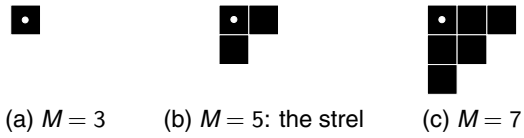


Figure 10. Not only the corner pixel value (white dot) will change its value in the interpolation part. Depending on the magnification, also neighbouring pixels change.

3.5. Interpolation

We now replace background pixels with foreground pixels, and vice versa. Not only the detected corner pixels are affected, but also some of their neighbouring pixels, depending on the used magnification.

The principle of the interpolation part is to perform a morphological dilation with a specific structuring element on the detected corner pixels, treating each corner map separately. The result is used as a mask. The pixels in the mask will be swapped from black to white, or from white to black.

All interpolation structuring elements were designed to transform the jagged edges or lines into a staircase pattern with a step size of one pixel. There are differences between each iteration step and also between odd and even magnifications.

3.5.1. Odd magnification. We take a structuring element (strel) that resembles a corner (figure 10(b)). We dilate the corners n times with this strel. In the case of an odd magnification M: $n = \lfloor \frac{M}{2} \rfloor - 1$. So, at magnification 3 the created mask will be identical to the corner map. At magnification 5 also 2 neighbours (the shape of the strel) will be added to the mask. The values of the pixels in the obtained mask are swapped.

The structuring element in figure 10 is used for an upper-left corner. Rotated versions of this strel are used for the other 3 corners.

3.5.2. Even magnification. For magnification by an even factor the principle is the same, but the object corner maps are treated different from the background corner maps, and this varies in each iteration step. Remember that only magnifications by an integer factor are possible.

In the odd iterations, $n = \lfloor \frac{M}{2} \rfloor - 1$ for the background and $n = \lfloor \frac{M}{2} \rfloor - 2$ for the objects. In the even iterations, the situation is reversed: $n = \lfloor \frac{M}{2} \rfloor - 2$ for the background and $n = \lfloor \frac{M}{2} \rfloor - 1$ for the objects.

Magnification 2 is a special case, because n can have value -1. When that is the case, those corner pixels will not change.

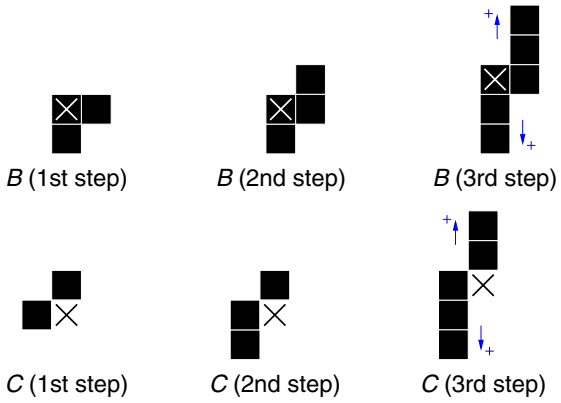


Figure 11. The hit-miss structuring elements are different for every iteration step.

3.6. Higher orders

The former steps will interpolate lines that are tilted $\pm 45°$ (and of course $0°$ and $90°$) correctly, but lines at other angles are only partly interpolated. This can be seen in figure 5(c). The jaggies are only removed in the direct neighbourhood of the original corners. Therefore, in order to obtain better results (figure 5(d)), we repeat the procedure from step 3.2 on (the corner detection). When the corner maps are all empty, then all interpolation improvements are done.

The structuring elements used in the corner detection and the interpolation sequence will be different for every iteration step. We will now look at the differences with the first iteration.

3.6.1. Corner detection. As can be seen in figure 5, the corners are not only shifted, but the shape of the corners has also changed. In the first iteration step, the shape of the corners is defined by the structuring elements B and C in figure 3. After the first iteration, corner pixels (and neighbouring pixels) are swapped, and thus new corners are created. Most of these new corners must be kept, so we cannot use the same strels again.

Therefore the structuring elements for the corner detection part (section 3.2) have to be altered (see figure 11). In the first iteration step there are 4 corner orientations, but from step 2 onwards the strels are less symmetric, which implies now 8 different corner orientations (3 rotational variants and mirrored versions). The total number of corner maps (foreground and background) hereby increases from 8 to 16.

3.6.2. Interpolation. Also for the interpolation part other structuring elements are needed. They are shown in fig-

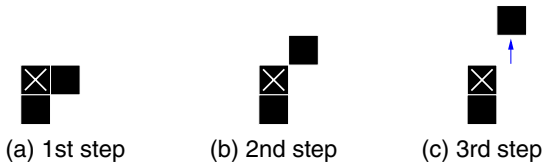


Figure 12. The interpolation structuring element is different for every iteration step.

ure 12. Also here the loss in symmetry implies 8 different orientations.

3.7. Optimizations

Several options are available to improve the speed of the algorithm. The first optimization is the localization of the possible corners in the next iteration. These locations can be calculated using the magnification and iteration step, and knowing what kind of corner (which orientation and whether or not mirrored) the pixel is. For odd magnifications the displacement is $\lfloor \frac{M}{2} \rfloor$ pixels. If the magnification is even, then the displacement is $\lfloor \frac{M}{2} \rfloor - 1$ for an odd iteration step, and $\lfloor \frac{M}{2} \rfloor$ for an even iteration. In the next iteration step, we only perform a hit-miss transform on these pixels, which reduces the calculation time. This also takes care of artefacts: only pixels that are expected to be possible corners will be investigated. Pixels that accidentally satisfy the hit-miss condition in the next cycle are now excluded.

A second optimization option is to incorporate the hole filling solution in the corner detection part, using different structuring elements for the creation of the foreground as for the background corner maps.

Another speed reduction is the omission of the validation rule in 3.3.1. But this will give a less satisfying visual result. A better option is to look at specific coordinates for a complementary pixel (as in the validation rule in 3.3.2), instead of using a morphological dilation.

In the not-optimized version of the algorithm, there are 16 different corner maps with each the size of the (magnified) image. These maps contain values 0 and 1, and only at pixels with value 1 a morphological operation is performed. If only these 1-pixels are put in lookup tables that contain their coordinates, the image doesn't need to be scanned completely, which means a further reduction of the calculation time.

Of course it is also possible to stop repeating the algorithm after a certain number of iteration steps, but this implies a loss in interpolation quality.

4. Results

We compare our technique with the method of [14] and classical linear interpolation methods [5]. These methods produce a greyscale image, but this is not always desired, as binary logos or cartoons often should remain binary. The greyscale interpolation of a binary image also often looks blurry. Our technique mmINT only produces a black-and-white result, so binarization of the output images of the other techniques is needed in order to compare. We use the well known *Otsu threshold* method [11] to define the ideal threshold for the grey values.

As can be seen in the results in figure 13 for a $3\times$ magnified binary image, our technique is visually better than other methods. The method HQ [14] also produces quite good results, but still more jaggies are visible. The contours in the figure interpolated with mmINT are smoother.

4.1. Statistical results

In order to determine the statistical significance of our premise that mmINT outperforms other techniques, we performed a traditional PSNR measurement and a small psychovisual experiment.

4.1.1. PSNR calculation. Most quality tests use a reference image as ground truth. The altered images are compared to the original image and some value is calculated, mostly the *Peak Signal-to-Noise Ratio* (PSNR). In the case of interpolation, no reference image exists: we start with a small image, but the interpolated images are magnified versions of the original. This problem can be solved by taking an image, scaling it down by subsampling it, and comparing the interpolation result of this small image with the original figure. Note that this procedure can cause problems: lines that are too thin can disappear after subsampling. Also, a line can have different thicknesses after subsampling, depending on the position of that line in the image. This affects the interpolation results. Another caveat for the use of the PSNR is the fact that the biggest PSNR value does not necessarily belong to the visually best result.

Table 1 shows the PSNR values for 4 interpolation techniques (mmINT, HQ, pixel replication and a bicubic algorithm). We took 57 different binary images, scaled down 2, 3 and/or 4 times, resulting in a set of 152 downsampled images. From this table we can conclude that the standard deviation is too high and the difference between the PSNR values is too low to draw a clear conclusion. For example, the PSNR difference between HQ and mmINT is only about 0.1%. This makes the Peak Signal-to-Noise Ratio a useless quality measure for the comparison of interpolation techniques on binary images.

(a) Original image

(b) Pixel replication

(c) mmINT

(d) Bilinear

(e) Bicubic

(f) sinc (Blackman-Harris)

(g) HQ

Figure 13. Interpolation results with $3\times$ magnification.

Table 1. PSNR calculation for the 4 interpolation techniques. The higher the PSNR value, the better. The standard deviation is indicated for each result.

Technique	Average PSNR
HQ	17.1 ± 2.8
mmINT	17.0 ± 2.8
Bicubic	16.9 ± 2.8
Pixel Replication	16.6 ± 2.6

4.1.2. Ranking experiment. We showed 8 different images, interpolated with 4 different techniques (mmINT, HQ, pixel replication and a bicubic algorithm), in random order to 35 (non-expert) persons. The test images are cartoons (both line drawings and filled drawings), text, and maps (containing both line drawings and text). The same images are used in section 4.1.3.

For this test, we asked our test public to rank the 8 sets of 4 different interpolation techniques in order of preference. We then calculated the average ranking, as can be seen in table 2.

The results from this experiment are consistent. mmINT is always ranked first, followed by HQ, except for text images. Bicubic and pixel replication follow at a rather big distance. In the case of the map images (lines and text combined), the pixel replication is preferred to the bicubic interpolation.

When we look at figure 14, we notice that the roundings of the letters "q" and "a" are not that well interpolated by mmINT. This is because different regions, that are interpolated independently, meet each other. This visually less attractive result explains the outcome of the psychovisual experiment.

4.1.3. Multidimensional scaling experiment. In [8], the *multidimensional scaling* technique (MDS) is explained. The basic assumption of MDS is that all relevant image properties (e.g. blur, noise, contrast, ...) correlate highly with geometrical properties of the stimulus positions. The stimuli are the results of the different tested interpolation techniques.

For each image, we showed the users 6 pairs of 2 different techniques and they had to give a preference score for that pair. With the MDS the stimulus was calculated in 1 dimension, representing a global image quality value for every technique. Figure 15 shows the stimulus positions (quality values) for the different techniques for all images together. The distances between the stimulus positions represent the dissimilarities between the corresponding stimuli. mmINT is clearly superior to the other methods. HQ is competitive with our interpolation technique. The bicubic and pixel replication method both give much worse results. If we look at the stimulus graphs for the images separately, then we can conclude the same as with the ranking experiment (section 4.1.2): HQ seems to work better for text; mmINT interpolates cartoons, logos and maps better than the other techniques.

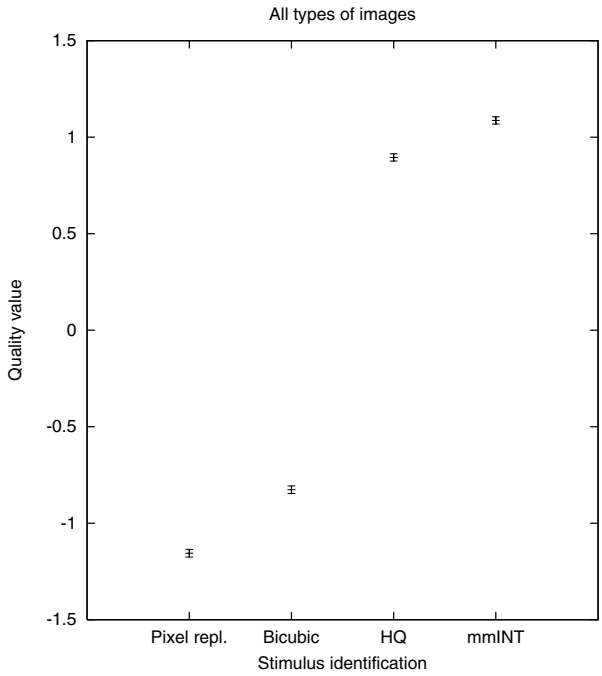


Figure 15. Stimulus position for the 4 interpolation techniques, for all 8 test images together.

5. Conclusion

Our interpolation method *mmINT* is a technique based on mathematical morphology that works very well on binary images, like logos, cartoons and maps. With a psychovisual experiment we have shown that it outperforms existing techniques. It also works quite well on text, but HQ performs better for this type of content.

A possible application is the interpolation of segmentation masks in video motion estimation applications (especially in object based coding). In the field of digital libraries, interpolation is useful when a low resolution electronic version of the document already exists. When the original paper document is not available anymore, or did not exist at all (like graphics made for a website), then interpolation is a solution to obtain a high resolution document. It could also save time to interpolate the digital documents instead of rescanning all the analog documents at a high resolution.

Table 2. Ranking of the 4 interpolation techniques. A lower number means a more preferred technique. The standard deviation is indicated for each result. Notations: BC=Bicubic; HQ=HQ; MM=mmINT; PR=Pixel replication.

	Filled 1		Filled 2		Lines 1		Lines 2	
1st	MM:	1.76 ± 0.68	MM:	1.27 ± 0.45	MM:	1.00 ± 0.00	MM:	1.24 ± 0.43
2nd	HQ:	1.84 ± 0.80	HQ:	1.78 ± 0.53	HQ:	2.00 ± 0.00	HQ:	1.84 ± 0.50
3rd	BC:	2.43 ± 0.87	BC:	2.97 ± 0.29	BC:	3.08 ± 0.28	BC:	3.11 ± 0.57
4th	PR:	3.97 ± 0.16	PR:	3.97 ± 0.16	PR:	3.92 ± 0.28	PR:	3.81 ± 0.40
	Text 1		Text 2		Map 1		Map 2	
1st	HQ:	1.19 ± 0.40	HQ:	1.41 ± 0.69	MM:	1.32 ± 0.53	MM:	1.49 ± 0.61
2nd	MM:	2.03 ± 0.44	MM:	1.78 ± 0.53	HQ:	1.76 ± 0.49	HQ:	1.59 ± 0.50
3rd	BC:	3.14 ± 0.89	BC:	3.08 ± 0.72	PR:	3.16 ± 0.60	PR:	3.41 ± 0.64
4th	PR:	3.65 ± 0.48	PR:	3.73 ± 0.45	BC:	3.76 ± 0.43	BC:	3.51 ± 0.56

(a) Pixel replication (b) mmINT (c) HQ

Figure 14. A text sample interpolated. Problems occur at roundings in letters.

In the future we plan to extend this method to greyscale and colour images. The idea is to perform the hit-miss transform on a local binarization of the intensity image. At the interpolation part, the pixels are swapped using a transfer function.

References

[1] J. Allebach and P. Wong. Edge-directed interpolation. In *Proceedings of the IEEE International Conference on Image Processing ICIP '96*, volume 3, pages 707–710, Lausanne, Switzerland, 1996.

[2] W. Freeman, T. Jones, and E. Pasztor. Example-Based Super-Resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[3] R. Haralick and L. Shapiro. *Computer and Robot Vision*, volume 1, chapter 5. Addison-Wesley, 1992.

[4] H. Honda, M. Haseyama, and H. Kitajima. Fractal Interpolation for Natural Images. In *Proceedings of the IEEE International Conference on Image Processing ICIP '99*, volume 3, pages 657–661, Kobe, Japan, 1999.

[5] T. Lehmann, C. Gönner, and K. Spitzer. Survey: Interpolations Methods In Medical Image Processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, 1999.

[6] X. Li and M. Orchard. New Edge-Directed Interpolation. *IEEE Transactions on Image Processing*, 10(10):1521–1527, 2001.

[7] H. Luong, P. De Smet, and W. Philips. Image Interpolation using Constrained Adaptive Contrast Enhancement Techniques. In *Proceedings of the IEEE International Conference on Image Processing ICIP '05*, pages 998–1001, Genova, Italy, 2005.

[8] J.-B. Martens. *Image Technology Design*. Springer, 2003.

[9] B. Morse and D. Schwartzwald. Isophote-Based Interpolation. In *Proceedings of the IEEE International Conference on Image Processing ICIP '98*, pages 227–231, Chicago, USA, 1998.

[10] D. Muresan and T. Parks. Adaptively quadratic (AQua) image interpolation. *IEEE Transactions on Image Processing*, 13(5):690–698, 2004.

[11] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[12] J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, New York, 1982.

[13] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, 2nd edition, 2003.

[14] M. Stepin. hq3x Magnification Filter, 2003. http://www.hiend3d.com/hq3x.html.

[15] D. Tschumperlé. *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Université de Nice — Sophia Antipolis, Nice, France, 2002.