SUPER RESOLUTION IMAGE BY EDGE-CONSTRAINED CURVE FITTING IN THE THRESHOLD DECOMPOSITION DOMAIN

Tsz Chun Ho and Bing Zeng

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong. {binary,eezeng}@ust.hk

ABSTRACT

An edge-constrained curve-fitting method is proposed in this paper to produce super-resolution (SR) images from a low-resolution (LR) source image. The novelty of this method lies in the threshold decomposition applied on the source image to obtain multiple binary images and then an edge-constrained curve-fitting is applied on the resulting set of binary images. This allows us to focus on tiny objects and thin structures so as to achieve rather nice visual results even when a large zoom-in factor is used. Our results are compared with those achieved by using the bi-cubic interpolation, showing the ability of our algorithm to achieve much better visual quality in smooth areas as well as for sharp edges and small objects.

1. INTRODUCTION

Generating a super-resolution (SR) image from a low-resolution (LR) source image is a long-studied problem. Several successes have been achieved in recent years in this area via a number of approaches, such as the example-based method [1], the training-based method [2], and hyper-resolution [3]. A drawback of these methods is that they rely on some model(s) or a large database. As a result, the robustness over various kinds of images remains as a problem. Another limitation of these current techniques is that most of them can only enlarge a source image by 2 to 8 times in both dimensions, whereas problems often occur when a bigger zoom-in factor is required. This is mainly because that enlarging a source image by a big factor is usually realized via several iterations – each round of running the (same) algorithm may generate artifacts and these artifacts will propagate to all following rounds. Motivated by these existing works mentioned above, we will develop a novel technique in this paper.

1.1. A brief introduction to our algorithm

The main idea of our algorithm is to threshold-decompose a source image from its gray-scale form to a set of binary images and then an edge-constrained curve-fitting is applied on these binary images. It is known that edges in gray-scale images can take complicated structures. As a result, after converting a gray-scale source image to multiple binary images, complex edge structures are mapped to simple binary structures. In each binary image, only 1-bit of information needs to be processed for each pixel. This also allows a parallel processing since all binary images are independent to each other. Boundary of each object of relatively large size in a binary image can be extracted easily and this information will help us in rendering an SR image.

2. THE ALGORITHM

Given an LR source image \( X_{LR} \), the simplest method to produce an SR image \( X_{SR} \) is perhaps to duplicate each pixel several times (determined by the zoom-in factor). However, \( X_{SR} \) generated in this way (see Parts (b) and (e) in Fig. 1 shown above) can hardly be accepted as a good SR image, especially when a large zoom-in factor is used. Then, a bi-cubic interpolation is often used to smooth the zigzag effect that is rather annoying visually. However, blurring visualization is experienced in these interpolated results, especially along sharp edges and around small objects, see Parts (c) and (f) in Fig. 1.

We believe that the annoying zigzag effect along sharp edges and small objects can be greatly reduced if smooth curves (instead of zigzag ones) can be fitted along the boundary of all fine structures. The boundary of a large object can be determined easily, whereas problems would occur when objects are tiny, e.g., objects with the
width of one pixel only. These small objects represent the details in an image and cannot be ignored. Therefore, a pre-processing is proposed to enlarge the given source image so that the boundary of small objects can be retrieved. Nearest neighborhood (replication) method is used here because it does not change the data of the original input image. Since the smallest object is of one pixel, we propose to do the 3-times duplication in both horizontal and vertical dimensions so that each single-pixel object in the original source image now becomes $3 \times 3$ in size, which allow us to label the boundary of single pixel objects in the original image.

2.1. Threshold decomposition

Threshold decomposition [4] is a technique to decompose a gray-scale image into multiple binary images through simple thresholding operations. Suppose that a gray-scale (8-bit) image is denoted as $x(i,j)$, the set of threshold-decomposed images are obtained as

$$ y_k(i,j) = \begin{cases} 1 & \text{if } x(i,j) \geq k \\ 0 & \text{otherwise} \end{cases}, \quad \text{for } k = 1,2,\ldots,255 $$

where $y_k$ is the decomposed binary image at threshold $k$. Clearly, these binary images can be processed in parallel. Furthermore, if some binary images are completely black (zeros) or white (ones) due to the lack of dynamic range of the source image, they can be skipped from the processing.

2.2. Boundary extraction and contour following

We assume that objects in each pre-processed binary image are displayed as black (zeros), see Fig. 2 (top-left) for one example. To extract the boundary of an object, a raster scan is performed on the binary image. If the scanned pixel is zero and at least one of its 4-nearest neighbors (4NN) is one, it is defined as a boundary pixel – a value of zero is assigned to each boundary pixel. Once all boundary pixels are determined, a second raster scan is performed to track the contour of an object. Directly connecting all boundary pixels together to form a contour will yield a jagged curve, see the top-right of Fig. 2. To solve this problem, we only select significant boundary pixels (called the critical data points in the rest of the paper) in this process. In particular, for areas containing more details such as sharp corners and thin structures, more critical data points are selected in order to fit an accurate curve on them. On the other hand, less critical data points are used for smooth areas so that boundary splitting and curve fitting can be applied thereafter.

When all boundary pixels are determined, a second raster scan is performed to track the contour of an object. Directly connecting all boundary pixels together to form a contour will yield a jagged curve, see the top-right of Fig. 2. To solve this problem, we only select significant boundary pixels (called the critical data points in the rest of the paper) in this process. In particular, for areas containing more details such as sharp corners and thin structures, more critical data points are selected in order to fit an accurate curve on them. On the other hand, less critical data points are used for smooth areas so as to fit a smooth curve to represent them. How to select these critical data points is done by a pattern matching. As the original image has been enlarged by a factor of 3 in the pre-processing stage, we divide the image into three by three, non-overlapped patch. Then, each $3 \times 3$ image patch can only take one of the patterns as shown in Fig. 3, where the 1$^{\text{st}}$, 2$^{\text{nd}}$, and 4$^{\text{th}}$ patterns can be rotated by $90^\circ, 180^\circ$ and $270^\circ$, and the 3$^{\text{rd}}$ one can be rotated by $90^\circ$.

By observing the patterns shown in Fig. 3, we find that the boundary details can be directly determined by the number of boundary pixels included in each pattern. For instance, 3 boundary pixels (the left-most pattern in the 1$^{\text{st}}$ row of Fig. 3) indicate that one side is boundary; 5 boundary pixels (the second pattern in the 1$^{\text{st}}$ row) represent a corner; 6 boundary pixels (the third one in the 1$^{\text{st}}$ row) represent a part of an object of a single pixel width in the original image; 7 boundary pixels (the fourth one in the 1$^{\text{st}}$ row) represent a stick-out pixel in the original image; and 8 boundary pixels (the right-most one in the 1$^{\text{st}}$ row) represent an isolated single pixel in the original image.

From these observations, we propose that only one critical data point is selected for patterns containing 3 or 5 boundary pixels. For the patterns that contain 6 or 7 boundary pixels, 2 or 3 critical data points are selected respectively, as to represent its details more accurately. Finally, 4 critical data points are selected for patterns of eight boundary pixels. See the 2$^{\text{nd}}$ row of Fig. 3 for the exact locations in each case. Notice that these critical data points form the positions that will be fitted by curves. The bottom-left of Fig. 2 shows all critical data points for the binary image shown on the top-left part.

2.3. Enlargement and edge correction

This is the step for image enlargement. Two coordinates of each critical data point is multiplied by the desired enlargement factor so that boundary splitting and curve fitting can be applied thereafter. However, applying the same multiplication (enlargement) factor to all critical data points would yield some distortion on the structure of edges in SR images. An edge correction will be used to solve this problem, as detailed in Section 3.

2.4. Breaking components to sectors

We define the completed (closed) boundary of an object as one component. Since the curve fitted on each component is likely to be
a smooth one, it may not be very appropriate for a boundary containing some sharp corners that represent details of the original image. Therefore, some high-curvature locations should be detected as breaking points of a component so that one component can be broken into multiple sectors. At those breaking points, curves could change abruptly because curves on two sides of each breaking point are independent. A slightly modified IPAN99 [5] algorithm is used to detect these breaking points. This algorithm is simple and does not require regular spacing between data points, which is particularly important in our application.

With reference to Fig. 4, every combination of a \( k \) (\( k = 4 \) in Fig. 4) critical data points before and after \( p_k \) are used to form a triangle so as to calculate the opening angle \( \alpha \) by the cosine law. If the minimum angle of \( p_k \) is smaller than or equal to \( \alpha_{\text{min}} = 140^\circ \), \( \alpha \) is assigned to that data point. Once the calculation is finished on all critical data points, non-minima suppression is used to select the breaking points of the component. Notice that the constraints of distances \( a \) and \( b \) are not included in our application.

### 2.5. Curve fitting

Now, each component is divided into small sectors. Each sector represents a part of the smooth boundary of object. Least square fitting of cubic Bezier curve [6] is used. The maximum allowed square distance between the original and fitted data is dependent on the number of critical data points in a sector. When the sector is long, it means that the sector is smooth. Therefore, it allows fitting a loose and smooth curve on it. On the other hand, if the sector is short, it means that the boundary changes abruptly. This occurs when the boundary has more details on it. Therefore, it needs to fit a tight curve on it so as to keep the details of the boundary. For different cases, the maximum allowed square distance is defined as follow:

\[
\begin{align*}
\text{if } L \leq 4, & \quad M_x\text{AllowSqD} = 0.5 \times f \\
\text{if } 4 < L \leq 8, & \quad M_x\text{AllowSqD} = 0.75 \times f \\
\text{if } 8 < L \leq 16, & \quad M_x\text{AllowSqD} = 1 \times f \\
\text{else } & \quad M_x\text{AllowSqD} = 2 \times f
\end{align*}
\]

where \( L \) denotes the number of critical data point in the sector and \( f > 1 \) is the enlargement factor. When the limit of error bound is violated, the sector is further split into two until the error is less than or equal to the limit. Since the algorithm needs at least four data points to fit a curve, if a sector only has two data points, two extra data points will be generated by interpolation and placed in the middle positions. If a sector has three data points, the data point in the middle is duplicated to meet the minimum length requirement. Each fitted curve is represented by four control points.

### 2.6. Reconstructing SR images

This is a straightforward step. The boundary in the SR domain is drawn using the control points obtained in the curve fitting. The object is filled by the contour-based region filling [7]. When all binary images are finished, the SR gray-scale image can be obtained by adding up all binary images.

### 3. EDGE CORRECTION

As described above, the coordinates of each critical data point are multiplied to the desired range for curve fitting. However, this would distort the structure of some edges since it also enlarges the width of each edge by the same factor. For example, when the transaction of an edge from one side to another takes three pixels and the scaling factor is four, simple multiplication will generate an edge with 12 pixel width in the SR image so that the edge steepness has changed completely. Therefore, an edge correction should be done to maintain the sharpness of edges. Fig. 5 shows an example of the curve fitting with and without correction. Edges are blurred in Fig. 5(b) when direct curve fitting is applied to the critical data points. The edge image in Fig. 5(c) acts as the reference boundary of objects in the target size image. By the edge-constrained curve fitting, better edge reconstruction has been obtained in Fig. 5(d).

#### 3.1. Generating edge image

In order to identify edges of small objects and thin structure, the original gray-scale image is enlarged by three times using the bicubic interpolation. Since strong edges should be kept sharp while weak edges could be relaxed in SR images, the strength of edges is measured during the edge detection. To this end, the enlarged gray-scale image is passed to the Canny-edge detector for nine times. The threshold of detector is set at 0.1 in the first pass and increased by 0.1 in each following pass. The output of each pass is a binary image, and all nine binary images are added to produce the edge image \( e(i,j) \in [0,9] \) in which non-zero pixels are identified as edge pixels and large value represents a strong edge pixel. Finally, the edge image is normalized to \([0,1]\), i.e.

\[
e(i,j) \leftarrow \begin{cases} 0 & \text{if } e(i,j) = 0 \\ \left( e(i,j) + 1 \right) / 10 & \text{otherwise} \end{cases}
\]

Notice that the edge image obtained here is completely different from those (binary) images generated in Section 2.

For color images, the luminance is used to obtain the edge image since the Canny-edge detector is applied on gray-scale images only. Although color images are processed on RGB independently, a common edge image is used for all channels as edges must be consistent with each other.

#### 3.2. Identifying critical data points for edge correction

The edge image obtained in the pervious section is the reference boundary of objects in the target size image. Obviously, the edge pixels should only affect a part of critical data points but not all from the same object. Therefore, additional procedure is needed to filter out all unnecessary critical data points which may distort the

---

**Figure 4:** Determining if \( p_k \) is a candidate point.

**Figure 5:** From left to right: original image; output image without correction; edge image; output image with correction.
structure of objects. To this end, a window of size ±6, centered at the current critical data point, is formed to obtain a patch from the edge image generated in section 3.1. Raster scan is performed to locate the edge pixels in the patch. Same procedure is done on the critical data point image to locate it in the patch. If there are no edge pixels, the current critical data point is skipped. Otherwise, each edge pixel inside the patch has to be checked whether it is a candidate location for a coordinate correction. To do this, the Euclidean distances between all critical data points and all edge pixels are measured. The criteria for an edge pixel to be the valid candidate is that the distance from it to the current critical data point \( d_{\text{crit-edge}} \) is not bigger than the distance to any of other critical data points \( d_{\text{crit-edge}} \) plus a pre-defined overshoot \( \delta \). More specifically, the equation can be expressed as

\[
d_{\text{crit-edge}} \leq \min(d_{\text{crit-edge}}) + \delta, \quad n = 1, \ldots, N
\]

where \( N \) is the number of critical data points found in the patch. When all edge pixels are checked, only valid edge pixels are kept in the candidate list. From the list of valid candidates, the closest one to the current critical data point is selected as the location for the coordinate correction. If all the edge pixels are not valid, no correction is done for that critical data point. Notice that \( \delta \) controls the tolerant distance for validating edge pixels. The recommended value of \( \delta \) is between zero and one.

In Fig. 6(a), the current critical data point (red color) is close enough to some edge pixels when compared with other critical data points. Among all valid edge pixels (light blue), the one on the same row has the smallest distance to the red-color location. Therefore, it will be used in the coordinate correction. In Fig. 6(b), since the current critical data point has larger distance than the other critical data points, no edge pixels are valid for correction.

### 4.3. Edge-constrained curve fitting

Now, for critical data points that do not need edge correction, direct multiplication on their coordinates is done to get the desired coordinates in the target sized image. More precisely,

\[
P = \frac{p \times f}{3}
\]

where \( p, P \in \mathbb{R}^2 \) are the coordinates of one critical data point before and after the enlargement. It is divided by 3 because a 3-times up-scaling has been done during the pre-processing step. On the other hand, for a critical data point that has identified an edge pixel \( u \) for the coordinate correction, the coordinates after the enlargement is determined as

\[
U = \frac{u \times f}{3}, \quad P' = \begin{cases} \frac{U - P}{f \times e^i(i, j)} & , \text{if } f \times e^i(i, j) \geq 1 \\ \frac{P}{f \times e^i(i, j)} & , \text{otherwise} \end{cases}
\]

where \( i, j \) is the coordinate of edge pixel \( u \). \( P' \) is the final coordinates of critical data point which is fitted by curve. \( U \) is the coordinate of edge pixel in target size image. \( e^i(i, j) \) control the data point either close to the edge pixel or the coordinate of direct multiplication. When \( f \times e^i(i, j) < 1 \), \( P' \) equal to the coordinate of direct multiplication to prevent data point move away from edge pixel \( U \).

---

**Figure 6:** Examples of image patch for edge correction: red pixel – the current critical data point; green pixels – other critical data points; gray pixels – (non-critical) data points; dark blue pixels – invalid edge pixels; light blue pixels – valid edge pixels.

### 4. EXPERIMENTAL RESULTS

Numerous images are tested to evaluate the quality of results generated by our algorithm. The color images are processed on RGB domain separately. Since the source image is enlarged by a big zoom-in factor, image processed on luminance component and bicubic interpolated on chrominance components will lead to some small artifacts appeared on the resulting images.

The first testing image is a part of a manmade lantern as shown in Fig. 1. The original image is of 256 by 256 pixel. Fig. 1(b)-(d) show the 16 times up-scaled images of the square shown in Fig. 1(a). Smooth curves are fitted on the small objects in the image. The edges are much sharper than what have been obtained by using the bicubic interpolation and do not show any jagged artifacts. Fig. 1(e)-(g) display the 32 times up-scaled images of the same square drawn in Fig. 1(a). This is a very large scaling factor when comparing with other SR algorithms where 2 to 8 scaling factor is usually used.

The second and third testing images are an eye of cat and a natural image of leaf shown in Figs. 7 and 8. The original size is 64 by 64 and the image is going to be up-scaled by 8 times. These images show the different approaches of reconstructed the gradient change area and small pixel object. By pre-processing image, we could identify those small objects, separate them and fit accurate curves to represent them.

### 5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a novel method to generate SR images from an LR source image and this method is applicable to all kinds of images. By threshold-decomposing a grayscale source image into multiple binary images, edges can be represented by simple cubic Bezier curves. Each binary image is independent, which allows a massively parallel processing. The boundary of small objects and thin structures are obtained by pre-enlargement and patch matching of boundary pixels. By locating critical data points and some necessary processing on them (such as breaking into sectors, coordinate correction according to the associated edge pixels, etc.), we have been successful in fitting accurate curves on these points so as to approximate the details on each binary image as well as represent edges on large objects. Our algorithm can generate SR images by an arbitrary enlargement factor, including very large numbers, while still maintaining well the structure of edges in SR images.
One limitation of our algorithm is that the resulting images are still not very photo-realistic when the enlargement factor is big. For instance, in areas with smooth gradient change, false edges are noticeable – the reason is that the curves fitted on some binary images have crossed exactly the same location. Those concentrated boundaries lead to visible false edges with unpleasant patterns. This problem could be solved by spreading boundaries on different threshold binary image systematically. In addition, we feel that some fine details and edges are still suffering from certain false patterns that are not very photo-realistic. To solve it, we believe that it is very crucial to generate an accurate edge image which guides us to do some necessary corrections on the coordinates of critical data points in the enlarged image. If the edge image is poorly detected from the source image, the output image will also become bad. As the edge image obtained from the 3-times up-scaled image, the resolution of edge may not be good enough to cope with a large up-scaling factor. We believe that further studies on the edge correction (including detection of edge image) could improve the result.

Acknowledgement: This work was supported in part by an RGC research grant from the HKSAR government.

REFERENCES