

Image Interpolation by Pixel Level Data-Dependent Triangulation

Dan Su, Philip Willis

Department of Computer Science, University of Bath, Bath, BA2 7AY, U.K.
mapds, P.J.Willis@bath.ac.uk

Abstract

We present a novel image interpolation algorithm. The algorithm can be used in arbitrary resolution enhancement, arbitrary rotation and other applications of still images in continuous space. High resolution images are interpolated from the pixel level data-dependent triangulation of lower resolution images. Our results show that the new algorithm can provide better results than most of the existing interpolating algorithms. Its computing efficiency is only slightly decreased compared to bilinear interpolation. It is simpler than other methods and is adaptable to a variety of image manipulations. Our method provides a good balance between visual quality and computational complexity. We keep our method as simple and efficient as bilinear interpolation while producing results as good as other existing methods.

1. Introduction

Digital image interpolation refers to the recovery of a continuous intensity surface from discrete image data samples. It is a link between the discrete world and the continuous one. In general, almost every geometric transformation requires interpolation to be performed on an image, e.g. translating, rotating, scaling, warping or other applications. Such operations are basic to any commercial digital image processing software. Obviously, the quality of the interpolator determines the quality of the desired image.

There are several issues which affect the perceived quality of the interpolated images: sharpness of edges, freedom from artifacts, reconstruction of high frequency details. We also seek computational efficiency, both in time and in memory requirements. Classical linear interpolation techniques, such as pixel replication, bilinear or bicubic interpolation have the problems of blurring edges or artifacts around edges. Although these methods preserve the low frequency content of the sample image, they are not able to recover the high frequencies which provide a picture with visual sharpness.

Standard interpolation methods are often based on attempts to generate continuous data from a set of discrete data samples through an interpolation function. These methods attempt to improve the ultimate appearance of re-sampled images and minimize the visual defects arising from the

inevitable resampling error. Many good interpolation techniques are known^{1, 2, 3, 4, 5}.

It has been recognized that taking edge information into account will improve the interpolated image's quality^{1, 2, 3, 4}. Instead of approaching interpolation as simply fitting the interpolation function, these methods approach it in addition with information of the geometry. Li¹ asserts that the quality of an interpolated image mainly depends on *the sharpness across the edge and the smoothness along the edge*.

With these considerations in mind, we develop a new edge-directed method for image interpolation which will interpolate along the edge orientation but not across it. Our new method is based on a simple data-dependent triangulation⁶ at pixel level. Each four pixel square is divided into two triangles by the diagonal: the diagonal either goes to the NE-SW or NW-SE direction. The direction of the diagonal is chosen to correspond to the edge in the image.

Our method is thus to fit the finest triangular mesh to the source pixels. This mesh is completely regular except that the diagonals are selected to run in the same general directions as any visible edge. To generate a new image, possibly at higher resolution, the target pixels are located on the source mesh. We then evaluate each target pixel from the triangle within which it sits. Any pixel which falls in the triangle will be interpolated using only the information from

the three triangle vertices. So in a smooth region, which is actually within the triangle, bilinear interpolation keeps it smooth and saves computation. In edge areas, the interpolator won't interpolate any two pixels that fall in different triangles, which means sharpness between triangles (edges) is retained. In other words, the new high-resolution image keeps the edge sharp and the smooth area smooth.

There are several edge directed interpolation schemes^{1,2,3,4} and a data-dependent triangulation approach⁵. Li et al.¹ attempted to estimate local covariance characteristics at low resolution and used them to direct interpolation at high resolution (NEDI - New Edge Directed Interpolation) while Allebach et al.² generated a high resolution edge map and used it to direct high-resolution interpolation (EDI - Edge Directed Interpolation). Battiato et al.³ proposed a method by taking into account information about discontinuities or sharp luminance variations while doing the interpolation. B.S.Morse et al.⁴ represented a scheme that using existing interpolation techniques as an initial approximation and then iteratively reconstructs the isophotes using constrained smoothing. Yu et al.⁵ adapted the traditional data-dependent triangulation (DDT) with their new cost functions and optimization.

The NEDI¹ and the Battiato's Locally Adaptive method³ can only do direct magnification by a factor of a power of two (they require iteration for a factor more than two). EDI², DDT² and Isophote⁵ can do arbitrary interpolation but they all need some iterative progress. Additionally, four methods^{1,2,3,4} (DDT is an exception) are only designed for super resolution.

Our scheme has several advantages in contrast to these former methods. Firstly, our approach is almost as simple as bilinear interpolation while former methods are more complex. Secondly, our approach is designed for arbitrary scaling, arbitrary rotating and warping or other operations while most former methods are just defined for magnifying. Thirdly, our scheme interpolates in one go and is computationally fast while most former methods need iteration.

Our experiments show that images produced by our pixel level data-dependent triangulation have significant better visual quality than classical linear interpolation while keep as simple and efficient as bilinear interpolation. Methods which beat ours do so only marginally and are considerably slower. We demonstrate our algorithm used in colour images with arbitrary magnifying factor and arbitrary rotation.

The rest of the paper is as organized follows. In the next section we describe the principle of the method, justify it and describe the implementation. Next we present some experimental results. Finally we make some concluding remarks.

2. Pixel Level Data-Dependent Triangulation

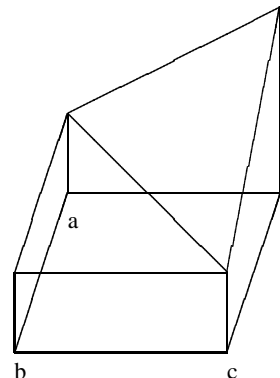


Figure 1: Triangulation in a four-pixel square

2.1. Principle of the Algorithm

Triangulation has been an active research topic during the past decade. Image reconstruction using triangles isn't widely used, probably because of its complexity. Our scheme is based on the technique of data-dependent triangulation (DDT)⁶ but we use it at pixel level. The DDT-based reconstruction algorithm attempts to minimize the visibility of the reconstruction errors thus produce pleasing reconstructions.

We wish to keep edges sharp. We first consider the case that there is an edge passing between a square of four pixels. If this edge cuts off one corner, one pixel will have a value substantially different (could be bigger or smaller) than the other three. Call this pixel the *outlier*. Imagine that we represent the brightness of the pixel as the height of a terrain. In effect, the three similar pixels define a plateau, relatively flat, while the outlier value is at the bottom of the cliff (if smaller) or the top of a peak (if higher) (Figure 1). This gives us a hint that if we want to interpolate a high resolution pixel within the relatively flat region we should not use the outlier. Classical interpolation methods like bilinear interpolation suffer from edge blurring because they use all four pixels to do interpolation.

This simple geometry motivates us to find a way to guide interpolation so that smoothness within the regions and sharpness between the flat region and cliff region can both be kept. Put another way, if the diagonal is to correspond to the edge in the image, the diagonal should be the one which does *not* connect to the outlying pixel value, the one most different to the other three. We choose a diagonal to triangulate the four pixels because of its simplicity. Sub-pixel edge triangulation could be used but we will demonstrate that triangulation by diagonal can provide excellent results.

For a grey-scale image, we represent the brightness of the pixel as height, see Figure 1. Suppose pixels *a*, *b* and *c* are the same height while *d* is higher than these three. Obviously

a , b and c define a flat region while d is the most different pixel to the other three. Thus we connect diagonal ac and get the triangles abc and adc .

In general, if b or d is the most different pixel, the edge should be ac , otherwise bd will be the edge. There are other situations if a and d are very different to b and c ; or a and b are very different to c and d . In these cases it makes little difference which diagonal is chosen.

The strength of employing triangulation is we can tune the interpolator to match an edge. In Figure 1, when interpolating the high-resolution pixel falling in triangle abc , the interpolator won't use the value of d which is very different to this plateau. For two pixels falling in different triangles, the height of the pixels will be quite different and thus the sharpness of edge is kept. It is easy to see that in very smooth regions, the interpolator is able to keep its smoothness as well, even across triangle boundaries.

2.2. Implementation and Optimization

Suppose the low-resolution image is X and the high-resolution image to be generated is Y . We first scan the sample image X to initialize a lookup table which records the edge direction of all four-pixel squares. For any super-resolution pixel we can distinguish in which triangle of X the pixel falls. The high-resolution image Y is then interpolated. For each y_{ij} we do an inverse mapping to the sample image X and decide the four pixel square. We use the lookup table to select the right triangle, then bilinear interpolate within the triangle to get y_{ij} .

We use inverse mapping because it has a number of benefits. First it can be used at arbitrary resolution. We are not constrained in any way by the resolution of the source data. Second, there is no requirement to align the target grid parallel to the source grid, so arbitrary rotation is possible at no additional cost. Third, sampling can be irregular to provide warps, although the sampling rate must not be too low because this would cause break-up.

In the initializing stage, there is an easy way to choose the direction of the edge. We simply compare the difference $|a - c|$ with $|b - d|$. We connect the pair with smaller difference as the diagonal. The proof that this is equivalent to finding the outlier pixel is in Appendix A. This method saves computing time, needing only two subtractions and a comparison.

The algorithm is very simple and easy to implement. Its complexity and its computing time are almost the same as bilinear interpolation while its result is better than bilinear and bicubic interpolation (Figure 2).

2.3. Extended Method

Close study of Figure 2 reveals a problem if we only consider the four pixel square when predicting the direction of

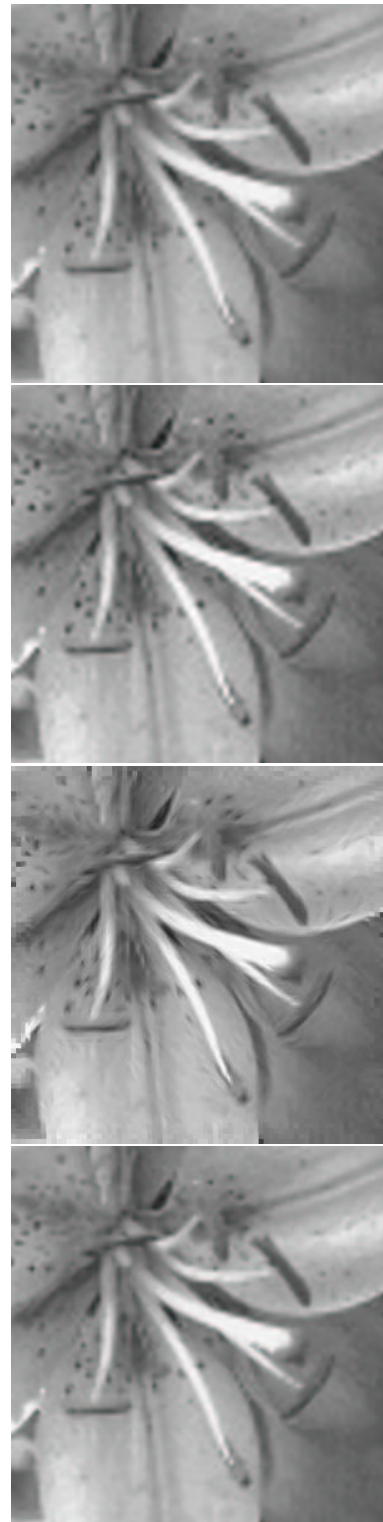


Figure 2: Detail image magnifying by 4. Top: bilinear interpolation. Second: bicubic interpolation. Third: NEDI. Bottom: our basic method

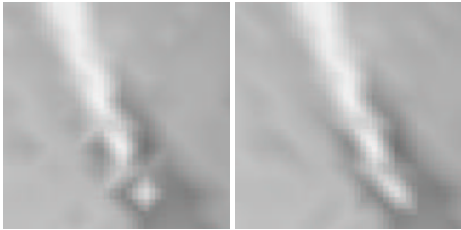


Figure 3: Magnified view of the stamen. left: selecting edge only by four-pixel squares. right: selecting edge by a 3×3 square neighbour window

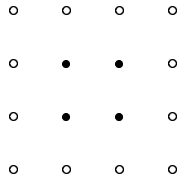


Figure 4: 3×3 square neighbour window

the edge. For example, look at the tip of lowest stamen of the flower. Figure 3 gives a magnified view of this stamen. The actual local edge goes in the NW-SE direction while some squares give the NE-SW direction causing some deterioration of edge quality. To correct this problem we need to consider information in the local area.

If we assume the image is locally stationary, we can model the image as a Markov Random Field. That is to say, the brightness of a pixel is dependent on its spatial neighborhood while independent of the rest of the image. The neighborhood of a pixel can be modeled as a window around this pixel. Instead of a normal least-square adaptive edge prediction scheme, we simply consider the neighbour window's edge direction. To predict the edge direction in a four pixel square, we look at the eight squares around the target square (Figure 4) and compute each square's edge direction. If most edges in these squares go in one direction then we set the edge in the target square to that direction. If there is no strong edge direction in the local area then we only consider the target square when choosing the edge.

3. Experimental Results

3.1. Applications

We have so far described the method for monochrome images. When selecting edge direction in colour images, we convert the RGB components of each pixel into luminance using the following formula⁵:

$$\text{Luminance} = 0.21267R + 0.71516G + 0.07217B$$

The edge direction is decided by luminance values. Interpolation is done in the R,G,B planes independently. This method generates good results because colour does not contribute as significantly as intensity to the information content of images, as Van Essen et al.⁷ say. Figure 8 shows three examples of magnification of colour images.

Due to the simplicity of our algorithm, it is easy to apply in many other image manipulations. For example, we can rotate the image by any angle (Figure 7a). We scan the target image and inverse rotate each pixel back to the sample image and interpolate the value. We can get a perspective transform of an image. On any given y scan line, we calculate the pixel at (x, y) by sampling the source image at (sx, ty) where s, t are scale factors which vary linearly with height (We are assuming the y axis is the centre of the screen). Figure 7b shows the result. We can also produce a magnifying lens effect (Figure 7c). If the lens has radius R , then its disc is filled with the image from a small disc with radius r at the same centre. For any pixel inside R , we scale down to r , evaluate the original value at r and apply it in R .

In general, these are variants on the same technique: to evaluate the target pixel p , we evaluate pixel $F(p)$ where F is a simple inverse mapping to the original image. Then we interpolate in the triangle where it falls.

3.2. Quality Assessment

From visual inspection our method produces better images than bilinear and bicubic interpolation, while the NEDI method is better still. The bicubic method introduces sharper edges but more artifacts while our method benefits from fewer artifacts, sharpness in edge areas and smoothness within smooth regions.

However, we need a more analytical assessment of the visual quality of the interpolated images. There exist some image quality measurement techniques such as degradation-based quality measures⁹ and the visible differences predictor (VDP)¹⁰. These two vision models are complicated and it is difficult to compare different methods using these vision models. We tried to find a numeric method to access the visual quality of the interpolated image. Therefore we chose the method proposed by S.Battiao⁷ et al. and T.M.Lehmann et al.⁸ to measure the picture quality. They use *cross-correlation* between the original picture and the reconstructed picture to assess the quality of reconstruction.

The cross-correlation coefficient C between two images X, Y is:

$$C = \left| \frac{(\sum_{i,j} x_i y_{ij} - IJab)}{\sqrt{((\sum_{i,j} x_{ij}^2 - IJa^2)(\sum_{i,j} y_{ij}^2 - IJb^2))}} \right|$$

where a and b denote respectively the average value of image X and Y and I and J are the image's width and height. The cross-correlation coefficient is between 0 and 1 where a higher score means better reconstruction quality.



Figure 5: Image set of three images with different edges. The angles are 30, 45 and 60 degree

We produced one sample image set of three 'edge' images with size 200×200 (Figure 5) and used twenty 768×512 real nature images as another test set. We obtained the low-resolution image of a quarter size of original image. The down-sampled images were magnified by a factor of two and compared to the original images to calculate cross-correlation.

The down-sampled images could be obtained by averaging down or sub-sampling. However, edge blurring and ringing effect are introduced by averaging down process and sub-sampling breaks down the geometry and introduce artifacts. We chose a Gaussian filter as the point-spread function with its standard deviation representing the radius of the point-spread function. Each pixel at the target image (down-sampled image) is considered as a point-spread function represented by a Gaussian distribution. It is down-sampled from some part of the source image, represented by another point-spread function. In this case the radius of the point-spread in the source image is double that of the radius in the target image. Thus, we calculate the standard deviation of the target Gaussian distribution, then double this to get that of the source image. This is then used to down-sample, by convolution. The small images in Figure 8 are produced by this method.

We used pixel replication, bilinear interpolation, bicubic interpolation, new edge-directed interpolation (NEDI)¹, our basic method and our extended method with 3×3 neighbour window to obtain the reconstructed images. The bicubic interpolation was performed by a Matlab 5 built-in function. All reconstructed images are magnified by a factor of two.

We compared the original images and the reconstructed images in the test set and calculated the averaged cross-correlation coefficients.

3.2.1. Quality of Edges

In the first test set we have generated three samples with a single edge of different angles (30, 45 and 60 degree) across the image and with black and white one each side of the edge (Figure 5). The table below shows the corresponding cross-correlation results.

The cross-correlation results report that our method gets the best score in every case. Although our triangulation gives edges of 45 degrees, it performs well on small or big angle edges. Our method produces the best result because we

don't interpolate across the edge. Bicubic and bilinear interpolation are slightly worse because they suffer from artifacts or blurring on the edge. Pixel replication doesn't catch the geometry very well and NEDI suffers from further blur because it interpolates across a bigger window.

	<i>Replication</i>	<i>Bilinear</i>	<i>Bicubic</i>
	<i>NEDI</i>	<i>Basic</i>	<i>Extended</i>
30 degree	0.995641	0.996730	0.997038
	0.992398	0.997149	0.997149
45 degree	0.996149	0.996990	0.997413
	0.994653	0.997543	0.997543
60 degree	0.995661	0.996725	0.997037
	0.995602	0.997150	0.997150

3.2.2. Quality of Real Images

We used twenty 24-bit 768×512 colour nature images as another test set. Because we use colour images, we compute the cross-correlation coefficients of the *R, G, B* planes independently and average these three. The values, averaged over the test set, are reported below.

<i>Pixel Replication</i>	<i>Bilinear</i>	<i>Bicubic</i>
0.976201	0.977699	0.978771
<i>NEDI</i>	<i>Basic Method</i>	<i>Extended Method</i>
0.972265	0.977534	0.977732

Bicubic interpolation gets the best score which means best overall reconstruction quality. Our extended method is better than using bilinear, NEDI and pixel replication. Our basic method is slightly worse than bilinear interpolation because sometimes it gives wrong edge direction. However, it generates correct triangulation in most situations and produces better visual quality than bilinear interpolation. Our extended method is slightly better than bilinear interpolation because our approach is better in edge areas and is almost the same in smooth areas. Pixel replication gets the lowest score as we expect.

For a normal image, cross-correlation only tells us how close the overall image is to the original image. We are not able to use it to predict the edge reconstruction because most of the image would be smooth. Fortunately, it is easy to observe visually that our method improves edge reconstruction in natural images compared to traditional methods. Although bicubic interpolation gets a higher cross-correlation score than our method, our extended method is visually better and much more computing efficient.

3.3. Performance Assessment

The new method's computing efficiency is only slightly decreased compared to bilinear interpolation while it is almost as simple to implement as bilinear interpolation. The following table shows the performance comparison on a Pentium II 400 machine with 256 M memory. We used the twenty 768×512 colour images test set again. We averaged every images down to 384×256 images and then magnified the

sub-sampled images by a factor of 2 and also a factor of 3.5. The results are reported below. The third column is our basic method and the fourth column is our extended method with a 3×3 square window. These three methods are implemented by C++ code. All figures are seconds.

	<i>Bilinear</i>	<i>Basic</i>	<i>Extended</i>
magnify 2	1.837	2.053	3.898
magnify 3.5	5.791	6.078	7.905

We can see from the table that the speed of our method is slightly lower than bilinear interpolation. Besides the initializing time for triangulation which is about 0.2 second for our basic method and 2 seconds for our extended method, the computing time for interpolation of our method is almost the same as bilinear and is linear with the pixels of the output image, same as bilinear interpolation.

4. Discussion

In this paper we have presented a new method of image interpolation. The new method is based on data-dependent triangulation but we use it at pixel level.

Our extended method produces visually better results than traditional interpolation schemes while it is simple and fast.

Another advantage of our method is it can be used for magnifying by an arbitrary factor in one go (Figure 8), while others former methods always need iteration which means longer the computing time.

Our method is also good for many other image manipulations such as arbitrary rotation, perspective transform, warp (Figure 7) and so on.

Bilinear and bicubic interpolation are widely used in commercial software package due to their simplicity. For example, Photoshop uses three interpolation engines: pixel replication, bilinear and bicubic interpolation. Our method produces better results with almost the same time and memory cost as bilinear interpolation.

Acknowledgments

The authors would like to thank Dr Xin Li at Sharp Labs of America at Camas, WA, USA for kindly providing his code for comparison and Dr P.W. Wong at IDzap, USA for providing the flower image shown in this paper. We would also like to thank our colleague Dr Peter Hall and Dr Man Qi for their suggestions and discussion. The work is funded under EPSRC project *Quasi-3D Compositing and Rendering* and an *Overseas Research Scholarship*.

References

1. X.Li, M.Orchard, "New Edge Directed Interpolation", *ICIP 2000*. 1, 1, 1, 2, 2, 2, 2, 5

2. J.Allebach and P.W.Wong, "Edge-directed interpolation", *ICIP' 96*, pp. 707-710 1, 1, 2, 2, 2, 2

3. S.Battiato, G.Gallo, F.Stanco "A locally-adaptive zooming algorithm for digital images", *submitted to Image vision and Computing Journal*, 2001 1, 1, 2, 2, 2, 2

4. B.S.Morse and D.Schwartzwald, "Isophote-based interpolation", *ICIP' 98* 1, 1, 2, 2, 2

5. X.Yu, B.Morse, T.W.Sederberg, "Image Reconstruction Using Data-Dependent Triangulation", *IEEE Trans. on Computer Graphics and Applications*, pp 62-68, May/June 2001 1, 2, 2, 2, 4

6. N.Dyn, D.Levin, and S.Rippa, "Data Dependent Triangulations for Piecewise Linear Interpolation", *IMAJ. Numerical Analysis*, Vol. 10, pp. 127-154, 1990. 1, 2

7. D.C.Van Essen et al, "Information Processing in the Primate Visual System: An Integrated System perspective", *Science*, Vol. 255, no. 5043, 1992, pp. 419-423 4

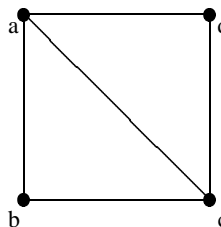
8. T.M.Lehmann, C.Gonner, K.Spitzer, "Survey: Interpolation Methods in Medical Image Processing", *IEEE Transactions on Medical Imaging*, Vol. 18, no. 11, November 1999 4

9. N.Damera-Venkata, T.D.Kite, W.S.Geisler, B.L.Evans and A.C.Bovik, "Image Quality assessment based on a degradation model." *IEEE TRansactions on Image Processing*, Vol. 9, pp. 636-650, April 2000 4

10. S.Daly, "The visible differences predictor: An algorithm for the assessment of image fidelity." *Digital Images and Human Vision* A.Watson, Ed. Cambridge, MA:MIT Press, 1993 4

Appendix A: Proof

Consider a four pixel square $abcd$. We will first prove that, if pair ac has smaller difference than bd , then b or d is the outlier pixel and we should connect ac . That is to say, if $|a - c| < |b - d|$ then b or d is either the biggest or the smallest pixel.



Suppose $|a - c| < |b - d|$, and suppose $a \geq c$, then $a - c < |b - d|$.



Figure 6: The flower image is magnified by a factor of 4. Above: bilinear interpolation. Bottom: our extended method

1. Suppose $b > d$. Then $a - c < b - d$ ($b > d, a \geq c$), hence $a - b < c - d$ ($b > d, a \geq c$).

We suppose $a > b$ and $c < d$, then $a - b > 0$ and $c - d < 0$, so we get $a - b > c - d$. However, we have the formula $a - b < c - d$ before which means our assumption that $a > b$ and $c < d$ is wrong.

Because $a > b$ and $c < d$ is wrong, either $a < b$ or $c > d$ or $a < b, c > d$ with the condition ($b > d, a \geq c$). In these cases, either b is the biggest pixel ($b > a, b > c, b > d$) or d is the smallest pixel ($d < c, d < a, d < b$).

2. Suppose $b < d$, then $a - c < d - b$ ($b < d, a \geq c$), hence $a - d < c - b$ ($b < d, a \geq c$).

We suppose $a > d$ and $c < b$. Then $a - d > 0$ and $c - b < 0$, so we get $a - d > c - b$. However, we have the formula $a - d < c - b$ before which means our assumption that $a > d$ and $c < b$ is wrong.

Because $a > d$ and $c < b$ is wrong, either $a < d$ or $c > b$ or $a < d, c > b$ with the condition ($b < d, a \geq c$). In these cases, either b is the smallest pixel ($b < c, b < a, b < d$) or d is the biggest pixel ($d > b, d > a, d > c$).

We have proved that if pair ac has the smaller difference ($|a - c| < |b - d|$), there are two situations. One is that either b is the biggest pixel or d is the smallest pixel. The second is that either b is the smallest pixel or d is the biggest pixel.



Figure 7: Above: Flower image rotated by 27 degrees. Middle: A perspective view of the flower image. Bottom: A lens effect of the flower image

In either case the outlier is either b or d and ac should be the edge. Using the same method we can prove that if pair bd has the smaller difference ($|b - d| < |a - c|$), the outlier is either a or c and bd should be the edge.

So we can conclude that drawing the edge between the least-different diagonal pair gives the same result as drawing the edge which isolates the outlier.



Figure 8: Three sample images of 192×192 are on the upper left corner. The images on corresponding positions are magnified of a factor of 2.1 by our extended method.