Image Interpolation by Pixel-Level Data-Dependent Triangulation on Android Platform

Jun-Yi DUAN¹,a, Cheng-Ming LIU¹,b,* and Zhi-Hui YUE¹,c

¹School of Software and Applied Technology, Zhengzhou University, Zhengzhou, China

a280467405@qq.com, bcmliu@zzu.edu.cn, c164626378@qq.com

*Corresponding author

Keywords: Image interpolation, DDT, Android.

Abstract. Smart mobile phones have become the most popular electronic equipment; many operations have been transferred from computers to mobiles. We achieved an image interpolation algorithm on android platform. This algorithm could generate high-resolution images from low-resolution images by the pixel-level data-dependent triangulation and a triangular interpolation. The algorithm we achieved on android is more lightweight and the result image is sharper.

Introduction

Image interpolation, an important component of image processing, has a wide range of application in computer vision and computer graphic. The aim of image interpolation is to Synthetic high-resolution image from low-resolution image. Accuracy of the algorithm directly affects the clarity of the image.

There are many image interpolation algorithm or super-resolution reconstruction algorithm that have been proposed in recent years[1,2]. They can roughly divided into three categories: polynomial-based interpolation, Learning-based interpolation and edge-directed interpolation[3].

Polynomial-Based Interpolation

It is widely known that Polynomial-based interpolation techniques, such as bi-cubic or bilinear interpolation are implemented by a function, which could map a relationship between known pixels in low-resolution and unknown pixels in high-resolution[4-6]. These algorithms are simple but fast, they are apply to android. However, high-resolution image interpolated by these algorithms are too smooth, and edges are blurred.

Learning-Based Interpolation

In recent years, algorithms of Learning-based have been paid more attention, these algorithms interpolate high-resolution by learning from examples. Chao Dong [7] has proposed a end-to-end deep learning algorithm, sparse representation[8] and Markov random field[9] are also included.

Edge-Directed Interpolation

Many algorithms are concentrated on edge-directed interpolation [10,11], since the edges of image are attractive to the human visual system. These algorithms make edge sharpness during the interpolation process. Cheng-ming Liu described a new image interpolation method by using scanning line algorithm which can generate C¹-curves or surfaces[3]. Lei Zhang and Xiaolin Wu[13] proposed a new edge-guided nonlinear interpolation technique through directional filtering and data fusion. Dan Su and Philip Willis [12] presented a novel image interpolation, the algorithm reconstruct high-resolution images from the pixel-level triangulation of low-resolution images.

In this paper, taking into account the universality of the smart phone and easy to operate, we achieve an image interpolation at Android by improving the algorithms of Dan Su and Philip Willis. This algorithm could roughly divide the edge by pixel-level triangulation and make high-resolution sharpness. We also perform experiments to illustrate the efficiency and accuracy of this algorithm.
Pixel-Level Date-Depend Triangulation

We first introduce the idea of Dan Su and Philip Willis. The main consideration of the idea is that there is one edge is obliquely across a square of four pixels. In this case, the edge divides the square into flat and plateau. We could use two triangulations to roughly represent the plateau and flat. In Figure 1, a, b, c and d form a square of pixels, triangulation abc form the flat, triangulation acd form the plateau. If a pixel to be interpolated falls in the flat, the interpolation only use the value of a, b and c.

![Figure 1. Triangulation in a four-pixel square.](image)

Triangular Interpolation

After dividing the known pixels, we need to interpolate inside the triangle. Commonly used triangular interpolation is Gouraud shading[14], which named after Henri Gouraud. Gourand shading could be used to compute the pixel values at any point in the triangle by the pixel values of three corners of the triangle as shown in Figure 2. In order to get the value of pixel $I_p$, values of $I_a$ and $I_b$ need to be calculated first, it can be written as[15,16]:

$$I_a = I_1 - (I_1-I_2) \frac{(Y_1-Y_p)}{(Y_1-Y_2)}$$

(1)

$$I_b = I_1 - (I_1-I_3) \frac{(Y_1-Y_p)}{(Y_1-Y_3)}$$

(2)

$$I_p = I_b - (I_b-I_a) \frac{(X_b-X_p)}{(X_b-X_a)}$$

(3)

![Figure 2. Gouraud Shading.](image)
Gouraud shading is universal, however, it is complex used in isosceles right triangle. In this paper, we use property of barycentric coordinate system to solve the problem and improve the algorithm speed.

![Barycentric coordinate system](image)

In geometry, the barycentric coordinate system is a coordinate system in which the location of a point of a simplex (a triangle, tetrahedron, etc.) is specified as the centre of mass, or barycentre, of usually unequal masses placed at its vertices[17]. In a triangle, barycentric coordinates are also known as areal coordinates or area coordinates. For a point $p$ in triangle $ABC$ (see Figure. 3), it can be determined by

$$p = (1-u-v)A + uB + vC (u \geq 0, v \geq 0, u + v \leq 1).$$

If coordinates of $A,B,C,p$ is known, values of $u$ and $v$ can be get by the following equations:

$$p.x = (1-u-v)A.x + uB.x + vC.x.$$  

$$p.y = (1-u-v)A.y + uB.y + vC.y.$$  

Then value of $p$ could be calculated by equation (4). In this paper, we use the fractional part of point to be interpolated as the parameter $u,v$ to reduce the amount of calculation and shorten the operation speed.

### Implementation

**Android NDK**

The Native Development Kit (NDK) is a set of tools that we are allowed to use C or C++ code with Android[18]. The advantages of using NDK in Android apps are as follows:

1) Better platform portability.
2) Reusability of native language libraries.
3) Improve performance in certain cases, especially computationally intensive ones like images or games.

In this paper, we build the app for NDK to make the algorithm faster.

### Design and Implement

The algorithm for developing native app for Android is as follows:

1) Loading image, getting image information and put them into an array in Java.
2) Image interpolation by pixel-level data-dependent triangulation in C++.
3) Compose image in Java.

In step 2, the detailed algorithm of Image interpolation is as follows:

1) If $|a-c| < |b-d|$, when interpolating the high-resolution pixel falling in triangle $acd$, the algorithm won’t use the value of pixel $b$, otherwise, the algorithm won’t use the value of pixel $d$.
2) If $|a-c| > |b-d|$, the high-resolution falling in triangle $abd$, the algorithm use the value of $a$, $b$ and $d$, otherwise the algorithm use the value of $b$, $c$ and $d$. 

299
3) If |a-c|=|b-d|, the algorithm is same as bilinear interpolation.

**Experiment**

We use two quality metrics, peak signal-to-noise ratio and calculating time to show the ability of algorithm we proposed. We compare the algorithm with bilinear interpolation and bicubic interpolation. In our test, we down-sample the original image by bilinear interpolation, then up-sample it to get the high-resolution by our algorithm or other method. We do experiments on an Android virtual machine with 4GB main memory created by a Lenovo computer with core i7-6700 cpu@3.40GHz and 8GB main memory. All the algorithms are implemented by NDK, and the scale factor is 2×2 or 3×3.

![Figure 4. Result of image interpolation. From left to right are: (a) Bi-cubic interpolation; (b) Bilinear interpolation; (c) Our algorithm.](image)

Figure 4 shows the results of image interpolation, compared with the bicubic interpolation and bilinear interpolation, our algorithm reduces the sawtooth phenomenon in the processing of the edge. The PSNR value and calculating time are shown in Tables 1 and 2. We can see from PSNR value, when the image is magnified twice, our algorithm is better than bilinear, close to bicubic interpolation, when the image is magnified three times, our algorithm is better than bicubic and bilinear interpolation, the greater the magnification, the better the effect of our method. In Table 2, we can find our algorithm is the fastest.

**Table 1. PSNR values.**

<table>
<thead>
<tr>
<th></th>
<th>bicubic</th>
<th>bilinear</th>
<th>our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnify 2</td>
<td>36.7112</td>
<td>35.8828</td>
<td>36.6087</td>
</tr>
<tr>
<td>Magnify 3</td>
<td>33.1495</td>
<td>32.4356</td>
<td>33.1613</td>
</tr>
</tbody>
</table>

**Table 2. Calculating time.**

<table>
<thead>
<tr>
<th></th>
<th>bicubic[ms]</th>
<th>bilinear[ms]</th>
<th>our algorithm[ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnify 2</td>
<td>95633</td>
<td>21819</td>
<td>20966</td>
</tr>
<tr>
<td>Magnify 3</td>
<td>95633</td>
<td>36865</td>
<td>31878</td>
</tr>
</tbody>
</table>

**Conclusions**

In this paper, we implement an image interpolation algorithm in Android. This algorithm can roughly divide the edge, and can keep sharpness, our algorithm is also superior to bicubic interpolation and bilinear interpolation in terms of computational speed. We do some experiments to illustrate the efficiency of the algorithm.
Acknowledgement
This research was financially supported by the National Science Foundation (No. 61402420).

References
[18] Information on https://developer.android.google.cn/ndk/guides/concepts.html