On Interpolation for Triangulation-represented Digital Image

Tomáš Janák[*][†]

Department of Computer Science and Engineering
University of West Bohemia
Pilsen / Czech Republic

Abstract

Triangulation is a good tool for vector representation of raster image data. To visualize the image represented by triangulation, one needs to fit a continuous surface of colour intensity in the triangulation, i.e. to interpolate data stored in its vertices. This paper presents some interpolation methods for the purpose of use on digital images and compares them. The commonly used piecewise linear interpolation lacks means to adapt to behaviour of intensity in the surroundings of currently interpolated triangle. This leads to disturbance of continuity of the mentioned intensity surface. In this paper, two possible solutions to this problem are given. One suggests making an approximation of the surface in a surrounding area by estimation of surface normals, the other is to interpolate directly on larger formations instead of on a single triangle. Zienkiewicz's interpolation is presented as a method to use the normal information, interpolation on Bézier triangle patch and on Coons patch are presented as methods of interpolation on larger surfaces.

Keywords: Delaunay triangulation, interpolation, image representation, image reconstruction

1 Introduction

Under the term a digital image, one usually imagines a rectangular grid of pixels, an image stored as a bitmap. Though this (a matrix of pixels) is the form in which we visualize the image, scaling or other transformations of such bitmap are rather problematic as they introduce various artifacts, distortions and other unwanted changes to the resulting image.

We can avoid such problems if we convert the raster into vector representation, i.e. if we represent the image as a set of geometrical simplices. The transformation is then simplified to a change of coordinates of the points, which define individual simplices (triangles in our case). We interpolate among those points to get the remaining points, which create the image. Thus the solution leads to a representation alternative to the raster based formats (JPEG, PNG, etc.) and that is a triangulation representation.

The basic strategy is to create some triangulation from the input image and, when visualizing it, reconstruct the image with a piecewise linear interpolation of each triangle. This simple approach obviously does not generate satisfactory results. Main problems are that large, almost mono-coloured, areas are not "smooth" enough and colour edges are not "sharp" enough. By the lack of smoothness we mean discontinuity in colour intensity among individual triangles forming the area and by the lack of sharpness we mean that the colour edges, which make an individual object in the picture recognizable, are blurred. There are basically two directions in which this basic model has to be modified in order to achieve better results.

First of them is an improvement of the process of triangulation construction. The set of vertices, which forms the triangulation, resembles pixels of the original image (their x and y coordinates and colour intensity). This means that there are many different ways of how to choose which pixel should become a vertex. We also have many different types of triangulations that can be used. The construction can have various impacts on the quality of the reconstructed image, thus the choice of the type of triangulation is very important. Interpolation methods presented in this paper were tested on Delaunay triangulation, but they should be applicable to any other triangulation.

The second thing to improve is the interpolation method itself. Flaws of the commonly used piecewise linear interpolation are mainly caused by an ignorance of intensity behaviour in areas surrounding the interpolated triangle. In this paper, we discuss some options which would enable us to incorporate those areas into calculations of the resulting intesity. To achieve this, two different approaches are suggested.

First are methods, which still interpolate the triangles individually, but use vectors respective to the continuous intensity surface varying across the whole image to correlate the output. The other approach is to interpolate on more complex surface structures formed by individual

[*] tjanak@students.zcu.cz

[†] This work was supported by the Grant Agency of the Academy of Sciences of the Czech Republic (GA AV ČR), project No. KJB101470701.

triangles of the triangulation. This paper will depict some exemplary interpolation methods of those approaches, showing their advantages and disadvantages and evaluate their usability for our purposes. Methods described in this paper have already existed, but their application to the digital image is original.

After a brief review of existing work in Section 2, possibilities of interpolation on single triangle (Phong-like and Zienkiewicz's) are presented in Section 3. In Section 4 we take a closer look on interpolations on larger surfaces (Bézier triangle patch and Coons patch). Section 5 contains a comparison of results of the tried methods and their usability is then discussed in Section 6.

2 State of the art

The problem of magnification of a raster image is not new. Numerous solutions with more or less plausible results have already been developed. For example, in [9] and [11] we can read about usability of the discrete cosine transform for that purpose. In [2], assumption that luminance remains constant along the generalized path motion of a pixel in the image is used for reconstruction.

As was already told in Section 1, sharpness of colour edges is very important for a proper reconstruction. This knowledge led many research groups to putting a large emphasis on the colour edges when developing the interpolation methods. In [1], Allebach et al. suggest to create a high resolution edge map of the image and for the interpolation to use a bilinear interpolation modified to prevent interpolation across edges. In [8], this approach is enhanced by using the estimations of local covariance characteristics of the initial image to direct the interpolation of the magnified image.

Triangulations are also quite a popular tool for image reconstructions and their usability has been widely tested. In 1990, Dyn, Levin and Rippa showed in [5] that a piecewise linear interpolation on a Data-Dependent Triangulation (DDT) can lead to plausible results when used for image reconstruction. Since then, many authors enhanced their method in various ways, as can be seen, e.g., in [13], [12] or [7]. As [6] or [4] shows, also other triangulations then DDT, as the well-known Delaunay triangulation, can be used efficiently.

However, in the above stated work, the attempts to solve the problem of imperfect results are mostly narrowed to enhancement of the algorithm for triangulation construction. The usage of piecewise linear interpolation as a method to fill the pixels inside each triangle is usually mentioned as a given fact which is not a subject to discussion. But there are many interpolation methods already used in other applications (e.g. interpolations based on various splines, used successively in 3D rendering; interpolations of higher than linear degree, etc.), which could be modified and used for our purposes. This paper tries to bring some attention to this, because, as images presented in Section 5 shows, improvement of the interpolation methods themselves can lead to better results, independently of the used triangulation.

3 Interpolation on a single triangle

Following methods describe some possibilities of using normal (Phong-like) and gradient (Zienkiewicz's) vectors for better approximation of behaviour of the global colour intensity function.

3.1 Phong–like interpolation

Linear interpolation on a triangle is sometimes also referred to as Gouraud shading, a method initially developed for shading of 3D objects. Phong shading is considered as an enhancement to Gouraud shading (a brief description of both can be found in [14]) in case of 3D rendering. Since the Gouraud method is commonly used in our problem, we were curious whether we can use principles of Phong shading to achieve a similar improvement in interpolation of triangulated images. The interpolation method of Phong shading suggests to linearly interpolate not only intensities, but also normal vectors in the control vertices. Phong shading was initially meant for the Phong reflection model, which describes a 3D scene. Therefore, we also approach it as a 3D situation, with colour intensity as the third dimension.

The surface normal in each vertex is determined as an average of normals of adjacent triangles. All normal vectors are normalized before computations. This ensures that the final vector is not affected by the size of normal vectors (which are proportional to the area of their respective triangles), but only by their direction. Normal vectors in each pixel of a given triangle are then found as a linear combination of components (x and y coordinates and colour intensity) of the normal vectors in the triangle vertices. Barycentric coordinates are used as coefficients to respective normals, as shown in Eq. (1):

$$\begin{aligned} normal = (&a\cdot n_{AX} + b\cdot n_{BX} + c\cdot n_{CX}, \\ &a\cdot n_{AY} + b\cdot n_{BY} + c\cdot n_{CY},\ a\cdot n_{AI} + b\cdot n_{BI} + c\cdot n_{CI}) \end{aligned} \tag{1}$$

where a, b, c are barycentric coordinates of the triangle A,B,C, $normal_A = (n_{AX}; n_{AY}; n_{AI})$, $normal_B = (n_{BX}; n_{BY}; n_{BI})$ and $normal_C = (n_{CX}; n_{CY}; n_{CI})$ are the normal vectors in vertices A, B, C respectively, each defined by x, y and intensity components. In a similar way the basic color intensity in each pixel, as a combination of intensities of the triangle vertices, is found, see Eq. (2):

$$basic\ color = a\cdot A_I + b\cdot B_I + c\cdot C_I \tag{2}$$

A_I, B_I, C_I are intensities in the triangle vertices A, B, C and a, b, c are barycentric coordinates. This value is used as the value to be modified by the color intensity component of the normalized normal in this pixel. Note that in order to describe points inside the triangle, the barycentric coordinates a, b, c must take values from zero to one and their sum must be one.

However, our situation differs from a usual 3D case. We lack information about the light sources used by the Phong reflection model and it is highly improbable that we would be able to simulate them. Even if so, the reflection model is based on physical observations of light behaviour in 3D world, which would most likely be of no use in our coordinates (x, y and intensity). Therefore, we abandoned attempts to simulate Phong reflection model and searched for other ways to use the information presented by the normal vectors.

Let us analyze what that information actually is. Note that the surface normal in each triangle is gained as a cross product of two of its edge vectors and that the edge vector is gained as a difference between the two vertices of the edge.

Firstly, imagine a triangle inside some larger patch, which should be smooth. Then the colour intensities in its vertices are almost the same. Therefore the intensity components of the edge vectors are almost zero. Thanks to the cross product in the normal vector computation, this zero exposes in x and y coordinates of the resulting normal vector. After normalization, the intensity component will than converge to one, because the size of the normal will be almost equal to the (pre-normalization) size of its intensity component. It can be seen that in the opposite case, where there is a large difference between intensities in the individual vertices (an edge is encountered), the size of the normal vector gets much bigger than the size of its intensity component. Therefore, after normalization it becomes very small, converging to zero.

If the normals of all triangles adjacent to a vertex behave as in the first case (they lie in a smooth patch) or as in the second case (an edge is between them), than also the surface normal in the vertex behaves so. Thus the information from normal vectors is whether the interpolation should behave smoothly over edges of the triangles which surround the interpolated one or not. However, they do not give us any information about the value of colour intensity needed for the interpolation. Figure 1 illustrates this problem by visualizing colour components of normal vectors in each pixel as a grey-scale image – vectors with intensity converging to one are visualized as white, while the ones with intensity converging to zero are visualized as black. We can see that individual objects, i.e. smooth areas bordered by a colour edge, are easily recognizable. However, the colours that fill them are the same (white), even though their colours in the original image are not.

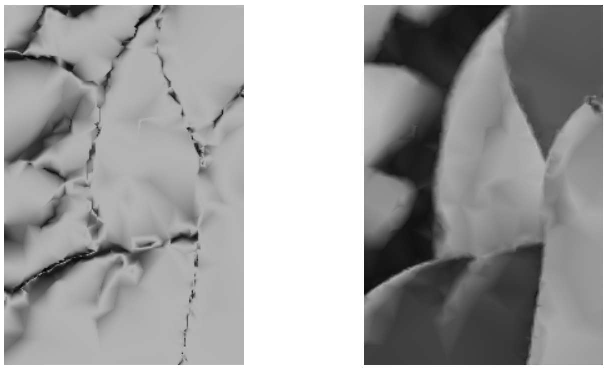
Figure 1: Part of the "Peppers" image, on the left we see intensity components of normals visualized as a grey-scale image, on the right original image for comparison.

To sum up, normal vectors can be used for quite an accurate description of global behaviour of the intensity, but do not present any direct tool for use of this knowledge. To use it, we would have to find some relation between normal vectors and intensities stored in vertices of the interpolated triangle. Section 3.2 introduces a method, which gives us such relation.

3.2 Zienkiewicz's interpolation

This method was used in [3] for interpolation of geographical data represented by triangulation. Because of satisfactory results it has in [3], it was worth trying to exploit it for our purpose.

In some way, it makes use of the knowledge presented by normals mentioned in Section 3.1 – gradient vectors are suggested for description of intensity behaviour. A simple method for a gradient estimation in a vertex is as follows: find an average of normalized surface normal vectors of each of the triangle adjacent to this vertex (as in Section 3.1), weighted by their areas, as expressed by (3):

$$normal = \sum_k \frac{A_k \cdot n_k}{A} \tag{3}$$

where n_k is the normalized surface vector of the k-th triangle, A_k is its area and A is the sum of all the areas A_k. With this estimated normal, gradient of intensity (along x and y axis) can be determined as shown in Eq. (4):

$$gradient = \left(\frac{-n_x}{n_I}, \frac{-n_y}{n_I} \right) \tag{4}$$

where n_x and n_y are normal components in the x and y coordinate and n_I is its colour component.

This estimation is then inserted into formula (5) derived for Zienkiewicz's interpolation, which cubically interpolates intensity across the triangle.

$$\begin{aligned} P(a,b,c) = {} & u_1\left[a^2 \cdot (3-2a) + k_1\right] + u_2 \cdot (a^2 \cdot b + k_2) + \\ & -u_3 \cdot (a \cdot b^2 + k_2) + \\ & +u_4\left[b^2 \cdot (3-2b) + k_1\right] + u_5 \cdot (b^2 \cdot c + k_2) + \\ & -u_6 \cdot (b \cdot c^2 + k_2) + \\ & +u_7\left[c^2 \cdot (3-2c) + k_1\right] + u_8 \cdot (c^2 \cdot a + k_2) + \\ & -u_9 \cdot (c \cdot a^2 + k_2) \end{aligned} \tag{5}$$

Result of the formula (5) is the intensity value in point (pixel) P, depending on parameters a, b, c, which are barycentric coordinates of the interpolated triangle

A,B,C. For better clarity, some terms of the interpolation formula (5) were shortened into variables *u* and *k*. Their meaning is described by Eq. (6):

$$\begin{array}{lll} u_1 = A_i & u_4 = B_i & u_7 = C_i \\ u_2 = AB \cdot g_a & u_5 = BC \cdot g_b & u_8 = \overline{CA} \cdot g_c \\ u_3 = AB \cdot g_b & u_6 = BC \cdot g_c & u_9 = \overline{CA} \cdot g_a \end{array}$$

$$k_1 = 2 \cdot a \cdot b \cdot c$$
$$k_2 = \frac{c \cdot b \cdot c}{2} \qquad (6)$$

where *AB, BC* and *CA* are the edge vectors (*B - A, C - B* and *A - C,* respectively), g_a, g_b and g_c are gradients in vertices *A, B, C*; *a, b, c* are the barycentric coordinates and A_i, B_i, C_i are colour intensities in the corresponding vertices.

4 Interpolations on patches

Following methods do not interpolate on individual triangles, but on patches created by them. This way we incorporate information held not by only three, but by more vertices, which should lead to smoother results. The first discussed method interpolates on the Bézier triangle patch, the second describes possibilities of interpolation on the Coons patch.

4.1 Interpolation on the Bézier triangle patch

Bézier triangle patch is a triangular surface, which can be made by joining three Bézier curves. Its degree is denoted by the degree of these curves, i.e. by the number of control points the curves have. For our purpose we consider patches of the second degree. Therefore, each boundary curve has three control points, which makes six control points to define the surface. A possible configuration of such a patch can be seen in Figure 2.

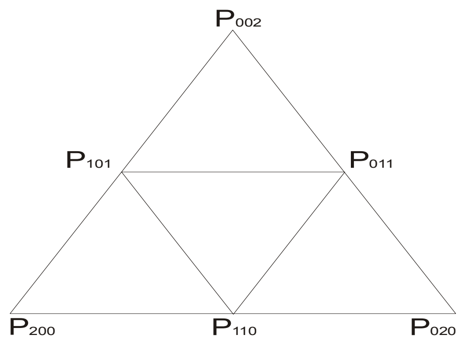

Figure 2: Bézier patch of the second degree with its control points

If we look on Figure 2, another way of interpretation of such a patch might occur. We can view it as a patch created by some triangle and its neighbours, i.e., the triangles, which share an edge with the one in center. Therefore, we can easily construct such a patch for each triangle in our triangulation. We only take the triangle and append its neighbours. Note that this approach cannot be used on triangles which are on the border of the interpolated image, because they do not have three neighbours. This singular case can be solved by using other interpolation method, e.g. bilinear, to render those triangles.

To describe points inside the patch, barycentric coordinates of vertices P_{200}, P_{020} and P_{002} can be used. We will refer to them as *a, b* and *c,* respectively. The conditions ensuring that the point *P(a, b, c)* lies inside triangle P_{200}, P_{020}, P_{002} are mentioned in Eq. (7).

$$a + b + c = 1$$
$$a \geq 0, \quad b \geq 0, \quad c \geq 0 \qquad (7)$$

Point *P* lying in the patch can than be described in the form of Bernstein polynomials with these coordinates as parameters, as Eq. (8) show:

$$P(a,b,c) = \sum_{i+j+k=n} B^n_{ijk}(a,b,c) \cdot P_{ijk}$$
$$where \quad B^n_{ijk}(a,b,c) = \frac{n!}{i! \cdot j! \cdot k!} a^i b^j c^k \qquad (8)$$

After inserting our values into Eq. (8), i.e. the degree *n = 2,* it reduces to Eq. (9):

$$P(a,b,c) = a^2 \cdot P_{200} + b^2 \cdot P_{020} + c^2 \cdot P_{002} + 2ab \cdot P_{110} + 2bc \cdot P_{011} + 2ac \cdot P_{101} \qquad (9)$$

With the resulting formula presented in Eq. (9), we are able to describe the whole surface defined by our patch. Coordinates (x, y and intensity) of each point P_{ijk} of this surface are obtained from (9) by a successive insertion of x, y and intensity coordinates of respective points P_{ijk} into the formula.

All points defined by coordinates *a, b, c,* which satisfy conditions (7), then form a surface with the following qualities. It lies in the convex hull of all control points of the patch, pass through terminal control points P_{200}, P_{020}, P_{002} and its boundaries, thanks to the quadratic interpolation function, are parabolic splines corresponding to the control points of individual boundary curves. Though these properties are welcome, they also cause problems with determining which pixels actually belong to the patch. The points of the patch do not have to project into each pixel of the triangles, which create the patch. On the contrary, some of those points can project outside of those triangles. A proposed solution to this problem is to compute intensity values for all possible triplets *a, b, c* that satisfy (7). Of course we cannot count with every number between zero and one. But since the number of pixels we want to render is finite and relatively small, we can compute a set of coordinates large enough to suffice. This can be done if we subsequently increment any two of the coordinates from zero to one by some “step”, a small (much smaller than one) real number. Note that the third coordinate is easily calculated, because we know the total sum must be one. This way the desired surface can finally be rendered.

However, the choice of this „step“ brings some difficulties. We project real values (*a, b, c*) on integer values (pixel coordinates) and therefore some pixel could be missed entirely because of rounding mistakes. Therefore, even if we calculate an approximation of number of pixels in the patch, we cannot be confident that it will suffice to compute that many intensity values. Much more values than there are pixels to render have to be computed, which result in lengthier time consumed by the computation. Though some optimization can be made (e.g. the mentioned estimation of pixel count), the quadratic time complexity of the algorithm makes it unable to compete with single triangle based methods.

On the other hand, each of the redundant value always projects onto some pixel. Thus we can get more intensity values for each pixel, which are not necessarily equal. Note that we also get those values for another reason – the patches are constructed for each triangle in the triangulation. Therefore, each triangle is involved in four patches, which means even more intensity values for pixels in the intersection of the neighbouring patches. We have to ensure that the finally displayed value is correct (that it did not end in the pixel only due to a rounding mistake). Theoretically, the most often value should be the most suitable, but experiments showed that results are almost the same as with the arithmetical average of all values projected into the individual pixels. Because computing arithmetical average has lower memory demands, we decided for this option. All results shown in Section 5 were rendered using the arithmetical average as the value to display.

4.2 Interpolation on the Coons patch

Coons patch is a surface defined by four curves as boundaries of the patch. As in the case of the Bézier triangle patch, our method suggests creating the Coons patch from quartets of triangles (one central triangle and his neighbours). The boundary curves are then defined by vertices on edges of such a configuration, i.e. P_{200}-P_{110}-P_{020}, P_{020}-P_{011}-P_{002} and P_{200}-P_{101}-P_{002} (using notation as presented in Figure 2). In order to get four curves, subdivision of the longest of those three is suggested. But before that, we have to decide what kind of curves we are actually going to lead through those vertices.

Because we have three control points per curve, it is reasonable to define the boundary curves for our patch as parabolic, i.e. defined by a quadratic polynomial. Eq. (10) shows a parametrical formula for such a curve (with the parameter *t*).

$$curve(t) = a \cdot t^2 + b \cdot t + c \qquad (10)$$

In order to set the curve uniquely, coefficients *a, b* and *c* are found according to following requirements. We want the curve to pass through the three vertices we have on each border of our triangle configuration. Simply said, we want the terminal vertices *A* and *B* (which are equal to either P_{200} and P_{020}, P_{020} and P_{002} or P_{200} and P_{002}) to be the “start” and “end” of the curve and the middle vertex *M* (P_{110}, P_{011} or P_{101}) to be the peak (or at least close to the peak) of our parabola. We get the coefficients by solving the equation system (11) for parameter *t* varying from zero to one:

$$\begin{aligned} A &= curve(0) = a \cdot 0 + b \cdot 0 + c \\ M &= curve(0.5) = a \cdot 0.25 + b \cdot 0.5 + c \\ B &= curve(1) = a \cdot 1 + b \cdot 1 + c \qquad (11) \\ &\text{--} \\ a &= 2 \cdot A + 2 \cdot B - 4 \cdot M \\ b &= 4 \cdot M - 3 \cdot A - B \\ c &= A \end{aligned}$$

Note that *A, B* and *M* represent either x, y or the intensity value of the corresponding vertex, therefore, each curve is actually defined by nine coefficients – a_x, b_x, c_x; a_y, b_y, c_y; a_I, b_I, c_I.

To select which curve to subdivide, distances from *A* to *M* and *M* to *B* are computed for all three curves and the longest curve is then subdivided into two as follows: x, y and intensity values of the longest curve for $t = 0.25$ and $t = 0.75$ are computed using formula (10) (with *a,b,c* coefficients already known). The results are used to construct vertices $T_{0.25}$ (for $t = 0.25$) and $T_{0.75}$ (for $t = 0.75$). Vertices *A*, $T_{0.75}$, *M* then define one of the new curves and *M*, $T_{0.25}$, *B* the other.

In this way we get two pairs of opposite curves. We can now use them to describe the surface delimited by them. Intensity value in each point of the surface is gained as a superposition of splines corresponding to the boundary curves in that point characterized by some parameter. The routine then looks subsequently: let a_1, a_2 and b_1, b_2 be the pairs of the opposite boundary curves.

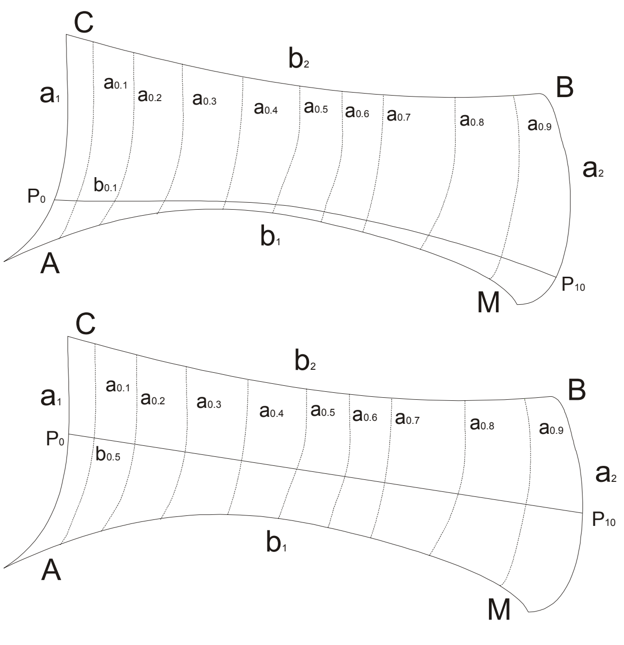

Figure 3: Two iterations of interpolation cycle on Coons patch

Values in the parameter u for the curves b_1, b_2 gives us terminal points of the spline b_u. In Figure 3, the terminal points are marked as P_0 and P_{10}. We interlace this spline with each possible spline a_v characterized by the points of pair a_1, a_2, i.e. by each parameter v from zero to one. Parameter u is then incremented (a new spline b_u is chosen) and the inner cycle repeats. Routine ends when u reaches one (every possible spline b_u has been used). In Figure 3, two iterations of this routine with step 0.1 are depicted. During each iteration, you can see the spline b_u (marked by a solid line) being interlaced with all eleven splines a_v with v varying from zero to one (marked by a dashed line). Each intersection of those splines is the interpolated point $P(u, v)$.

As in the Bézier patch interpolation, it is unknown which pixel lies on the surface. Therefore the step we add to the parameters during each iteration of the cycle should be as small as possible to ensure that all pixels are covered. Computation of a point of the surface (the superposition of splines in a certain point) can be expressed by Eq. (12).

$$\begin{bmatrix} 1-u & -1 & u \end{bmatrix} \cdot \begin{bmatrix} A & a_1(v) & M \\ b_1(u) & P(u,v) & b_2(u) \\ C & a_2(v) & B \end{bmatrix} \cdot \begin{bmatrix} 1-v \\ -1 \\ v \end{bmatrix} = 0 \tag{12}$$

As a solution of this equation, shown in Eq. (13), we get x, y or colour intensity of point P dependently on parameters u and v. A, M, B and C are x, y or intensity values in the respective vertices. $a_1(v)$, $a_2(v)$, $b_1(u)$, $b_2(u)$ are x, y or colour intensity values of points in respective boundary curves, denoted by parameters u and v.

$$\begin{aligned} P(u,v) = {} & (1-u)\cdot a_1(v) + u\cdot a_2(v) + \\ & + (1-v)\cdot b_1(u) + v\cdot b_2(u) + \\ & - (1-u)\cdot(1-v)\cdot A - u\cdot(1-v)\cdot C + \\ & - (1-u)\cdot v\cdot M - u\cdot v\cdot B \end{aligned} \tag{13}$$

Control points of the curves a_1, a_2, b_1, b_2 are written in Eq. (14). The considered longest initial curve is the curve denoted by the vertices A, M, B.

$$\begin{aligned} & a_1 = (A, M_{AC}, C),\ a_2 = (M, T_{0.25}, B), \\ & b_1 = (A, T_{0.75}, M),\ b_2 = (C, M_{BC}, B) \end{aligned} \tag{14}$$

where M_{AC} is the vertex between A and C, M_{BC} is the vertex between B and C, $T_{0.25}$ and $T_{0.75}$ are vertices created during subdivision of curve A, M, B.

The visualized value in each pixel is also gained as an arithmetic average of all values which were projected into that pixel (see Section 4.1 for details behind this decision).

Unfortunately, among other similarities, this method also shares a long computing time with the interpolation on the Bézier triangle patch. The reason is the same algorithm that is used for intensity distribution among pixels in the patch.

5 Experiments

In this Section, output images generated by methods discussed in Section 3 and 4 are presented and their qualities compared one with another and also with the most usual method, i.e. piecewise linear interpolation.

All experiments were performed on triangulations made from grey-scale images, because they better illustrate eventual artifacts. But since we can consider a colour image as an image with three different levels of grey-scale intensity, there should not be any problems to use these methods on colour images. All the methods were tested in our own implementation.

As mentioned in Section 1, the qualities we are looking for are sharpness across the colour edge and smoothness along the edge. Figures 4 and 5 focus on Lena's cheek as an example of a smooth surface (see Figure 6 for the original image). We can see that, as expected, behaviour of patch–oriented methods is better then of linear and Ziekiewicz's method. If only smoothness is considered, interpolation on Bézier triangle patch (in Figure 5, up) is clearly the favourite.

However, Zienkiewicz's interpolation still indisputably overcomes results of the common linear

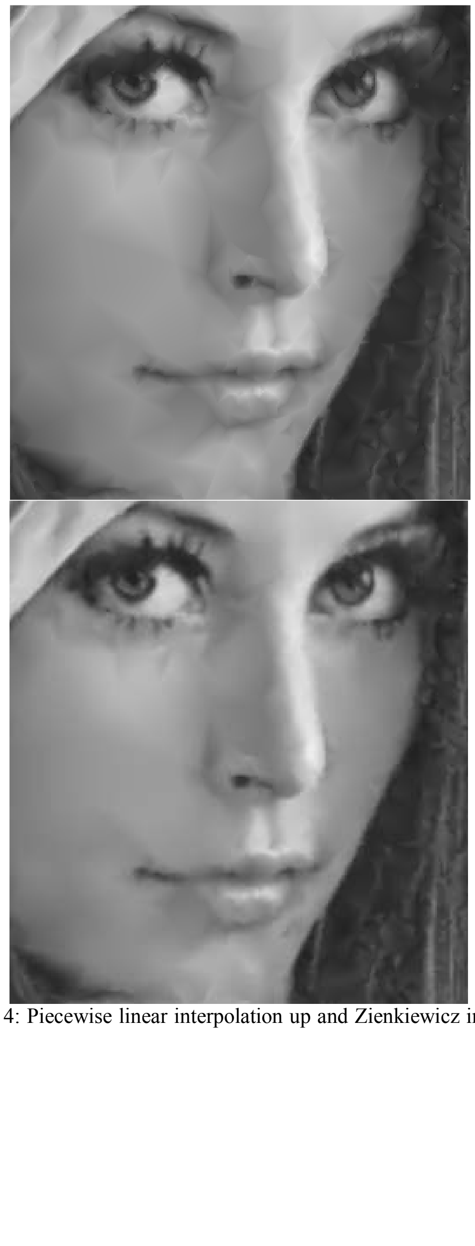
Figure 4: Piecewise linear interpolation up and Zienkiewicz interpolation down.

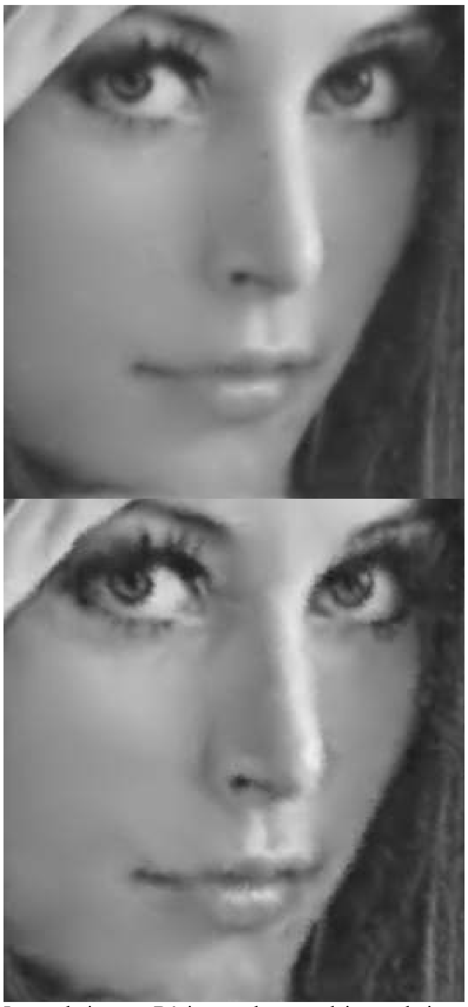

Figure 5: Interpolation on Bézier patch up and interpolation on Coons patch down.

interpolation – outlines of individual triangles, which makes the cheek looks rugged in case of linear interpolation (Figure 4, up), are flattened by Zienkiewicz's interpolation (as expected). Also, a difference in smoothness of images created by an interpolation on Coons patch and by Zienkiewicz's interpolation are rather minor (for example, area right beneath the right eye looks better in case of the Coons patch). This is quite surprising, because we presumed the patch method to get much smoother results.

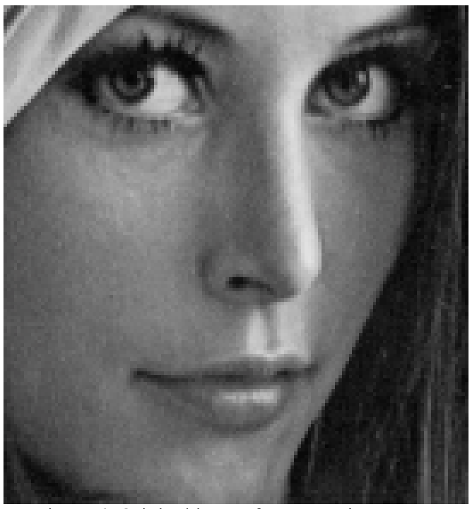

Figure 6: Original image for comparison

But the good smoothing performance of patch–oriented interpolations also has a disadvantage. Imagine some thin object described by a small count of triangles. So small, that there is almost no patch that is made entirely by triangles of this object. The object can than be almost fully erased as the colours of its surroundings blend with it. It is clear that in order to use interpolations on patches universally, we need to pay a better attention to preservation of colour edges, because these methods tend to blur them. In Figure 6 we can see a triangulation, which has colour edges expressively marked by a border of small triangles.

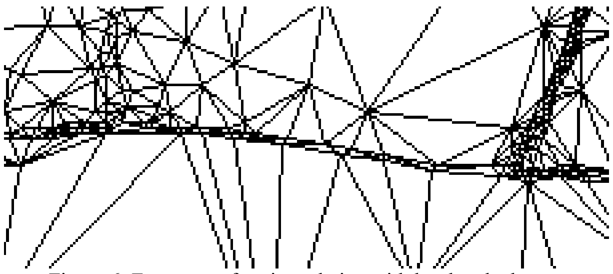

Figure 6: Fragment of a triangulation with bordered edges

Although this kind of triangulation solves the problem of blurring colour edges of the patch–oriented interpolations, Figures 7 and 8 show, that it introduces another problem instead. Patches created by triangles of markedly different sizes tend to be very deformed and far from the ideal shape. This leads to clearly visible artifacts spreading around the whole border, while Zienkiewicz's interpolation has no problem with an accurate visualization of colour edges, as can be seen in Figure 9.

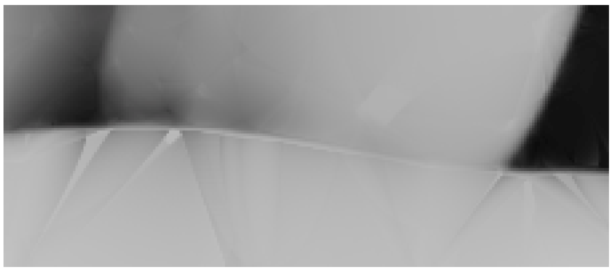

Figure 7: Interpolation on the Bézier patch of the triangulation in Figure 6

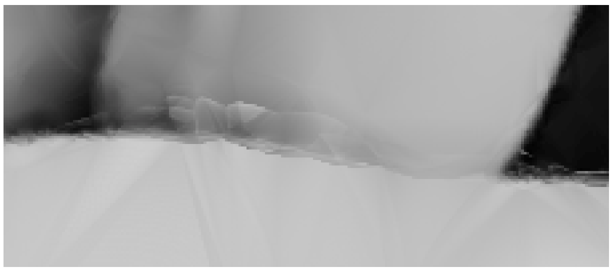

Figure 8: Interpolation on the Coons patch of the triangulation in Figure 6

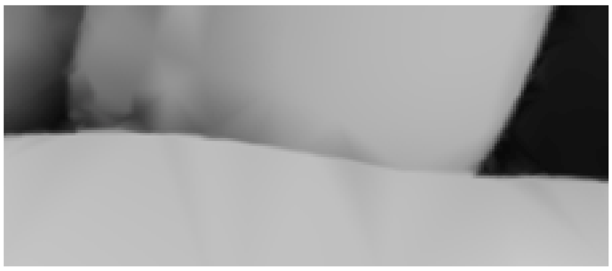
Figure 9: Zienkiewicz's interpolation of triangulation in Figure 6

6 Conclusion and future work

While not being perfectly flawless, presented Zienkiewicz's interpolation outperforms commonly used linear interpolation in smoothness of the output image and in some cases also preserves edges better. Moreover, a difference of a consumed time between both methods is insignificant. Therefore it seems like a good replacement of a linear interpolation for purposes of digital image represented by a triangulation.

Though methods interpolating on patches present better results for smooth areas, they are too dependent on triangulation construction to be used universally. However, as attempts to use Phong-like method showed, we are able to predict which areas should be smooth. Therefore an opportunity to exploit patch–oriented methods might be in a hybrid interpolation method. These would use one of such methods to render smooth patches and other, e.g. Zienkiewicz's method, to render areas, where the colour intensity is sharply varying. Unfortunately, the problem of long computation time of patch–oriented methods remains unsolved. The time can reach tenths of minutes during interpolation of high – resolution images. Therefore, those methods are not usable for the purpose of real time reconstruction (for space reasons, exact time tables have been omitted, but can be seen in [15], as well as high resolution images).

Future research possibilities include the already mentioned hybrid methods. We also plan to take a closer look on interpolations on Voronoi diagram. Voronoi diagram is a dual configuration to the Delaunay triangulation and as such can be reconstructed from the given Delaunay triangulation and used for interpolation.

Acknowledgements

I would like to thank to my supervisor Josef Kohout, Ph.D. for providing me his triangulation – processing program, triangulation samples and mainly for offering me the opportunity to work on this project. Big thanks also belong to Dr. Ivana Kolingerová for useful advices on some methods and help with the paper preparation.

References

[1] J. Allebach, P.W. Wong. *Edge-directed interpolation.* International conference on Image Processing 1996, Vol. 3, pp. 707-710.

[2] B. Ayazifar and J.S. Lim. *Pel-adaptive model-based interpolation of spatially subsampled images.* International conference on Acoustic Speech and Signal Processing proceedings, Vol. 3 (1992), pp. 181-184

[3] P. Čermák. *Výpočet vrstevnic na trojúhelníkové síti.* Diploma thesis, Faculty of Applied Sciences, University of West Bohemia in Pilsen, Czech Republic, 2002.

[4] L. Demaret, A. Iske. *Advances in Digital Image Compression by Adaptive Thinning.* Marie Curie Fellowship Association Annals, Vol. 3 (2004), pp. 105-109.

[5] N. Dyn, D. Levin, and S. Rippa. *Data Dependent Triangulations for Piecewise Linear Interpolation.* IMA Journal of Numerical Analysis, Vol. 10 (1990), pp. 137-154.

[6] J. Kohout. *On Digital Image Representation by the Delaunay Triangulation.* PSIVT 2007, December 17-19 2007, pp. 826-840

[7] B. Lehner, G. Umlauf, B. Hamann. *Image Compression Using Data-Dependent Triangulations.* ISVC 2007, November 26 – 28 2007, pp. 351 – 362

[8] X. Li, M. Orchard. *New Edge Directed Interpolation. IEEE Transactions on Image Processing*, Vol. 10, Issue 10, October 2001, pp. 1521-1527

[9] S. A. Martucci. *Image resizing in the discrete cosine transform domain.* International Conference on Image Processing proceedings Vol.2 (1995), pp. 2244

[10] T. Akenine-Möller, E. Haines. *Real-Time Rendering.* A.K. Peters Ltd., July 2002

[11] E. Shinbori, M. Takagi. *High Quality Image Magnification Applying the Gerchberg-Papoulis Iterative Algorithm with DCT.* Visual Communications and Image Processing, November 1992, pp. 311-321

[12] D. Su, P. Willis. *Image Interpolation by Pixel Level Data-Dependent Triangulation.* Computer Graphics Forum, Vol. 23 No. 2, June 2004, pp. 189 – 201

[13] X. Yu, B. Morse, T.W. Sederberg. *Image Reconstruction Using Data-Dependent Triangulation.* IEEE Computer Graphics and Applications, Vol. 21 No. 3, pp. 62-68, May/June 2001

[14] http://en.wikipedia.org/wiki/Phong_shading

[15] http://home.zcu.cz/~tjanak/CESCG