Image Compression and Enlargement Algorithms

I.G.Burova, Yu.K.Dem'yanovich, A. N.Terekhov,
A.Yu.Altynova, A.D.Satanovskiy, A.A.Babushkin

St. Petersburg State University
7/9 Universitetskaya nab., St.Petersburg, 199034
Russia

Received: January 24, 2021. Revised: June 30, 2021. Accepted: August 2, 2021. Published: August 5, 2021.

Abstract—In some cases, there are problems associated with the compression and enlargement of images. The use of splines is quite effective in some cases. In this paper, a new image compression algorithm is presented. The features of increasing the size of an image when using local polynomial or non-polynomial splines are considered. The main method for enlarging an image is based on the use of splines of the second and third order of approximation. Polynomial and trigonometric splines are considered. To speed up the process of enlarging the image, we used the parallelization techniques.

Keywords—Image enlargement, Polynomial splines, Trigonometric splines.

I. Introduction

CURRENTLY, a lot of attention is paid to image processing [1]-[7]. Algorithms related to image compression and enlargement are the subject of many recently published works. This paper [1] considers several interpolation methods used for digital image enlargement. As it is written in [2] "Scaling is the major operation performed in the transformation of images. Scaling is an important operation for resizing and reshaping the images that are in digital form. Various operations can be performed with digital images out of which the shrinking and zooming are the most widely operated by any type of user in the world. The other name for shrinking is sub sampling, and the zooming operation is also called oversampling. The purpose of the zooming operation is to extend or enlarge the image in order to have a clear and efficient view. Zooming operations are mostly performed on our mobile phones for viewing images in our gallery and this operation is the most frequently performed operation by mobile phone users." In paper [3] a new approach is proposed to transform bitmaps to vector images, which is based on triangle units and consists of three steps. A new multi-scale deep learning (MDL) framework is proposed and exploited in [4] for conducting image interpolation. The interpolation method is used in [5]. Paper [6] introduces VLSI (Very Large Scale Integration) architecture of an accurate and area effectual image scalar. As stated in this paper "Image scaling is a technique to enlarge or diminish the image by the provided scale factor. Image scaling can also be discussed as image interpolation, image re-sampling, image resizing, or image zooming." As noted in paper [8] image compression is one of the most interesting fields of image processing that is used to reduce the size of an image. In paper [9] the authors suggest an additional step, in which established image-compression techniques are exploited to decrease the number of integration sub-cells. To find and apply an effective method that allows each type of pixel to be displayed in a compact form is an important problem. Two-dimensional piece-polynomial basis are used in paper [10] to determine the recovery coefficients' outcome of digital processing of radiographic images. In paper [11] an algorithm has been developed to digitally compress an image using two-dimensional Haar wavelets, reduce its size, determine the recovery coefficients, and display a higher quality image of the processed image than the original image.
Some relevant studies can be found in [12], [13], [14], [15].

The use of wavelets in image processing has the same qualities as their use in various cases of processing large amounts of numerical information. The main property of wavelets is the ability to represent the initial information flow in the form of two flows: the main one and the refinement (wavelet) one. The use of this property helps to reduce the load on computing and communication facilities, since these flows can be transmitted sequentially. The addition of the wavelet theory with the introduction of spline spaces significantly influenced the direction of further research and led to the emergence of the non-classical spline-wavelet theory (see, for example, [18]). Spline-wavelet decompositions are obtained for any pair of nested spline spaces. The enclosing spline space is represented as the sum of the enclosed spline space and its direct complement. Note that the nesting property is possessed by Haar spaces, as well as the spaces of piecewise linear functions. As a result, zero-order wavelet expansions (Haar wavelet expansions) and first-order wavelet expansions are obtained, respectively. The use of irregular grids and cellular subdivisions has led to adaptive algorithms for processing complexly structured streams of numerical information. When processing video information, the adaptability property allows us to select the main video stream by grouping close pixels of the same luminosity. Thus, the main advantages of the mentioned spline-wavelet processing are adaptability, processing speed and economical use of the video memory of the computing system. In this paper, for image processing, we use piecewise linear splines (polynomial splines of the second order of approximation), as well as trigonometric splines of the second order of approximation.

In addition, for image processing, we use polynomial and non-polynomial spline approximations of the third order of approximation.

As stated in paper [6] “Image scaling is extensively utilized in numerous image processing implementations, like digital cameras, tablets, mobile phones, and display devices.” One of the important characteristics of smartphones and TVs is the screen resolution. Screen resolution is the size of the display in pixels. It is often necessary to quickly adapt the image size to the required display size when transmitting an image. Typically, a color image uses the RGB color model. In this case, the color of each pixel is specified by three numbers. We can reduce the image by removing horizontal or vertical rows of pixels. We can enlarge the image by adding the missing color information to the rows of added pixels. Consider the features of image resizing using the spline approximation theory. In paper [16], a similar method was used using the Java programming language.

This paper considers the features of the use of other programming languages. Section 2 discusses the theoretical foundations of image compression. In Section 3, we will consider the theoretical aspects of using spline approximations (see [16]-[19]) for image processing. Section 3 will present the results of the experiments.

II. About Image Compression

First, we will propose an image compression algorithm.

Consider a rectangular screen, $\boldsymbol{M}$ with its standard rectangular pixel structure of the size $m_1 \times m_2$, where m_1, m_2 are integer numbers. Let us introduce the notation $J_m = \{0, 1, 2, \ldots, m - 1\}$. Let a set, $\boldsymbol{C} = \{ \boldsymbol{C}_i \mid i \in J_M\}$ be the original pixel subdivision of the screen $\boldsymbol{M}$; here $M = m_1 \times m_2$. Two pixels are considered adjacent if they have a common side. A connected union of any set of neighboring pixels will be called a cell. The collection of such cells will be denoted by $\mathcal{M}$. In particular, all pixels C_i are also included in this collection, since they are a special case of a cell. The pixels are called original cells. For clarity, we can assume that screen $\boldsymbol{M}$ lies in the quadrant of the coordinate plane with the integer sides of length m_1 and m_2 located on the x and y axes, when $x > 0$, $y > 0$ (it is clear that in this case one of the vertices of this rectangle is at the origin coordinates). We associate a certain natural number with each original cell (i.e., pixel). Let us accept this number as the cell brightness. Thus, a piecewise constant function with positive integer values is given on the $\mathcal{M}$. We denote it as $f(t)$, where

$$f(t) > 0 \; \forall t \in \boldsymbol{M}. \qquad (1)$$

As $f(t)$ we take a linear combination of difference ratios in mutually perpendicular directions. The coefficients of these linear combinations are inversely proportional to the side lengths of elementary rectangles (pixels).

Further we deal with the sequential enlargement of this subdivision by combining adjacent cells into one cell. The enlargement process will essentially depend on the function $f(t)$. Let $\omega \in \mathcal{M}$. Let $mes\,\omega$ be the area of set ω. Consider the set function defined by the relation

$$\varphi_f(\omega) = \max_{t \in \omega} f(t) mes\,\omega. \qquad (2)$$

Obviously, the function φ_f has the following monotonicity property:

$$\omega', \omega'' \in \mathcal{M},\ \omega' \subset \omega'' \Rightarrow \varphi_f(\omega') \leq \varphi_f(\omega''). \quad (3)$$

We will enlarge the original subdivision C in successive steps, uniting the group of cells of the mentioned subdivision so that the result is contained in the collection $\mathcal{M}$ (equivalent formulation: the set of interior points of the resulting union must be homeomorphic to the open circle). Thus, in the next step, another enlargement cell appears. The enlargement of the resulting cell can be continued by attaching the cell of the original subdivision, if the attached cell has not yet taken part in the enlargement (i.e., was not attached). At each step of the enlargement, the set of cells of the original subdivision splits into two sets.

There are two types of sets. Set $\widetilde{\mathbf{C}} = \{\mathbf{C}_i\}_{i\in\tilde{J}}$ is the set of those cells of the original subdivision that has not yet participated in the enlargement. The second set is $\widehat{\mathbf{C}} = \{\mathbf{C}_i\}_{i\in\hat{J}}$ of cells that have already participated in the enlargement. The cells of the first set are called admissible, and the cells of the second set are called excluded. It is clear that $\widetilde{\mathbf{C}}\cup\widehat{\mathbf{C}} = \mathbf{C}$, $\widetilde{\mathbf{C}} \cap\widehat{\mathbf{C}} = \emptyset$, $\tilde{J}\cup\hat{J} = J_M$, $\tilde{J}\cap\hat{J} = \emptyset$. Due to the steps above, the sets $\widehat{\mathbf{C}}$, $\hat{J}$ will expand, and the sets $\widetilde{\mathbf{C}}$, $\tilde{J}$ will shrink. The end of the enlargement process corresponds to the case when $\widetilde{\mathbf{C}} = \emptyset$, $\tilde{J} = \emptyset$, $\widehat{\mathbf{C}} = \mathbf{C}$, $\hat{J} = J_M$. To describe this process, we introduce the operation of an elementary cell enlargement.

Let the cell ω already be constructed. If for ω there exists at least one neighboring cell from $\widetilde{\mathbf{C}}$, then the set $I' = \{i'\} \subset \tilde{J}$ of indices i'satisfying the condition

$$\mathbf{C}_{i'} \approx \omega,\ \varphi_f\,(\omega \cup \mathbf{C}_{i'}) \leq \varphi_f(\omega \cup \mathbf{C}_i) \quad \forall \mathbf{C}_i \approx \omega \qquad (4)$$

is not empty. Let i_0 be the minimum index in the set I',

$$i_0 = \min_{i'\in I'}\{i'\},\quad \omega^+ = \omega \cup \mathbf{C}_{i_0} \qquad (5)$$

The operation of the elementary enlargement of the cell ω is the process of joining cells ω and $\mathbf{C}_{i_0}$; the result of this union is denoted by ω^+, $\omega^+ = \omega \cup C_{i_0}$. If cell ω has neighbors in set $\widetilde{\mathbf{C}}$, then cell ω is called extensible. If ω has no neighboring cells in set $\widetilde{\mathbf{C}}$, then this cell is called a limit cell. It follows from what has been said that the operation of elementary enlargement can be applied to a cell that has neighbors in set $\widetilde{\mathbf{C}}$.

Consider a sequence of elementary extensions, using the sign ": =" to denote the renaming of the variables in question. Let us start with cell $\mathbf{C}_0$ of the original subdivision. The corresponding algorithm is written as follows.

0. We set $\omega := \mathbf{C}_0, \widetilde{\mathbf{C}} := \mathbf{C}, \tilde{J} := J_M, \widehat{\mathbf{C}} := \emptyset, \hat{J} := \emptyset.$

1. If ω is extensible, that is, ω has at least one neighboring cell in $\widetilde{\mathbf{C}}$, then according to (3) - (4) we find i_0, ω^+. Put $\widetilde{\mathbf{C}} := \widetilde{\mathbf{C}}\backslash\mathbf{C}_{i_0}$, $\tilde{J} := \tilde{J}\backslash\{i_0\}$, $\widehat{\mathbf{C}} := \widehat{\mathbf{C}} \cup \mathbf{C}_{i_0}$, $\hat{J} := \hat{J} \cup \{i_0\}$, $\omega := \omega^+$ and go to the beginning of the cycle (that is, to step 1.).

2. Otherwise (namely, when ω is a limit cell, that is, when ω has no neighboring cells in $\widetilde{\mathbf{C}}$), the sequence of elementary extensions is complete.

Now it is clear that the operation of the described algorithm for a connected manifold ends with one cell, which turns out to be a limit cell. Thus, as the result of the operation of the algorithm, the enlarged cell subdivision D of the connected manifold $\boldsymbol{M}$ consists of one cell $D_0 = \cup_{i\in J_M}\mathbf{C}_{i'}$, and $\widetilde{\mathbf{C}} = \emptyset, \tilde{J} = \emptyset, \widehat{\mathbf{C}} = \mathbf{C}, \hat{J} = J_M$. The described algorithm is the basis for a more complex algorithm that allows some approximation properties to be taken into account. The algorithm described below will be called an approximation. By definition we put

$$\varepsilon^* = \max_{i\in I}\varphi_f\,(\mathbf{C}_i),\quad \varepsilon^{**} = \varphi_f(\boldsymbol{M})\ . (6)$$

Let the number ε be from the interval $(\varepsilon^*, \varepsilon^{**})$

$$\varepsilon \in (\varepsilon^*, \varepsilon^{**})\ . (7)$$

In contrast to the previous algorithm, where the enlargement of the cell was terminated by the exhaustion of all cells of the original subdivision in the approximation algorithm, the enlargement of cell ω can stop earlier. The cell may turn out to be expandable, but its enlargement stops if the double inequality holds

$$\varphi_f\,(\omega) \leq \varepsilon < \varphi_f\,(\omega^+). (8)$$

Inequality (8) is called the ε -criterion for stopping the enlargement. In what follows, the numbers of such cells that satisfy criterion (8) are accumulated in the set J^0, and the numbers of limit cells are accumulated in set J^1 . At first, both sets are considered empty. Let's start describing this algorithm.

ALGORITHM (A)

0. We set $\tilde{J} := J_M, j: = 0$ (thus, we put $\widetilde{\mathbf{C}} := \mathbf{C}, \widehat{\mathbf{C}} := \emptyset, \hat{J} := \emptyset, J^0 := \emptyset, J^1 := \emptyset$).

1. Define $i_0 = \min_{i\in\tilde{J}}\{i\}$.

2. Assign $\omega: = \mathbf{C}_{i_0}$ (thus, we put $\tilde{J} := \tilde{J}\backslash\{i_0\}, \hat{J} := \hat{J}\cup\{i_0\}$, $\widetilde{\mathbf{C}} := \widetilde{\mathbf{C}}\backslash\widetilde{\mathbf{C}}_{i_0}$, $\widehat{\mathbf{C}} := \widehat{\mathbf{C}} \cup \widehat{\mathbf{C}}_{i_0}$; at the beginning of the algorithm, we have $i_0 = 0, \omega: = \mathbf{C}_0$).

3. We analyze cell ω: it is expandable or a limit cell.

3.0. If ω is extensible, then we find i_0 again (see (4) - (5)) and define ω^+.

3.0.0. If (8) holds, then we put, $J^0 := J^0 \cup \{j\}$,here $\{j\}$ are the stopped expandable cells, and go to 4.
3.0.1. If (8) is not satisfied, then we put $\omega := \omega^+$, $\tilde{J} := \tilde{J} \setminus \{ i_0 \}$, $\widehat{J} := \widehat{J} \cup \{ i_0 \}$, $\widetilde{\boldsymbol{C}} := \widetilde{\boldsymbol{C}} \setminus \boldsymbol{C}_{i_0}$, $\widehat{\boldsymbol{C}} := \widehat{\boldsymbol{C}} \cup \boldsymbol{C}_{i_0}$, and go to 3.1. If ω is a limit cell, then we find $J^1 := J^1 \cup \{j\}$, (and, therefore, go to 4).

4. Make the assignments $\boldsymbol{D}_j := \omega$; $j := j + 1$; and use the (previously defined) cell type ω and (earlier founded) number i_0.
4.0. If ω is an expandable cell, then we go to 2 (we emphasize once again that we know the number i_0).
4.1. If ω is a limit cell, then we find out whether $\tilde{J}$ is empty.
4.1.0. If $\tilde{J} \neq \emptyset$ (that is, $\tilde{J}$ is not empty), then we go to 1.
4.1.1. If $\tilde{J} = \emptyset$ (i.e., $\tilde{J}$ is empty), then we go to 5.

5. END (the end of the algorithm).

The resulting enlargement is denoted *by* $\boldsymbol{D}(f, \varepsilon) = \{\boldsymbol{D}_j \mid j \in J_K\}$, where $K = |J^0| + |J^1|$, $J_K = J^0 \cup J^1$. According to the algorithm which obtained it, the following properties hold: 1) for cells $\boldsymbol{D}_j$ with indices from set J^0, the inequality

$$\varphi_f(\boldsymbol{D}_j) \leq \varepsilon < \varphi_f(\boldsymbol{D}_j^+) \; \forall j \in J^0 \qquad (9)$$

holds,
2) for cells $\boldsymbol{D}_j$ with indices from set J^1 the inequality

$$\varphi_f(\boldsymbol{D}_j) \leq \varepsilon \qquad (10)$$

holds.
Definition. A cellular subdivision $\boldsymbol{D}$ with properties (9) - (10) is called an adaptive enlargement defined by the triple $(\boldsymbol{C}, f, \varepsilon)$.
Thus, the following statement is true.

Theorem 1. Under conditions (6) - (7), the adaptive subdivision defined by the triple $(\boldsymbol{C}, f, \varepsilon)$ is realizable and uniquely determined. Moreover, the set of cells satisfying condition (9) is not empty,

$$J^0 \neq \emptyset \qquad (11)$$

Proof. The unambiguous definiteness of the adaptive subdivision follows from the unambiguousness of algorithm (A). To prove realizability, note that the algorithm starts from cell $\boldsymbol{C}_0$. In view of assumptions (6) - (7), the inequality $\varphi_f(\boldsymbol{C}_0) \leq \varepsilon < \varphi_f(\mathbf{M})$ holds. Due to the monotonicity property of the function φ_f, the sequential enlargement of this cell will certainly lead to a situation where the result ω of such an enlargement will still satisfy the condition $\varphi_f(\omega) \leq \varepsilon$, and the addition of the next cell $\boldsymbol{C}_{i_0}$ of the original subdivision $\boldsymbol{C}$ will lead to an extension $\omega^+ := \omega \cup \boldsymbol{C}_{i_0}$, for which the inequality $\varepsilon < \varphi_f(\omega^+)$ is valid. So, the realizability of the algorithm and relation (11) are proved. This completes the proof.
Remark. For $n = 1$, algorithm (A) turns into the algorithm described in [19].
Further options for compressing and enlarging images will be discussed in the following sections.

III. Spline Approximation

In the previous paper, this feature of the approximation with the splines of the third order of approximation of the rapid change function was eliminated in a more complicated way. Here we describe a simple algorithm for solving the problem of going beyond the boundaries of the allowed range of variation of numbers when using splines of the second and third orders of approximation.
The theorems about the approximation with splines of the second and third order were proved in [16]. For the convenience of the reader, we present the formulations of these theorems here.
Let a grid of nodes $\{x_j\}$ be constructed on the interval $[a, b]$.
Polynomial splines of the second order of approximation are well studied (see [16], [17]). Basic spline formulas can be given by the formulas

$$\omega_j(x) = \frac{x - x_{j+1}}{x_j - x_{j+1}}, x \in [x_j, x_{j+1}],$$
$$\omega_{j+1}(x) = \frac{x - x_j}{x_{j+1} - x_j}, x \in [x_j, x_{j+1}].$$

Here $\omega_j(x)$, $\omega_{j+1}(x)$ are the basis functions. On the interval $[x_j, x_{j+1}]$, we construct an approximation of the function $u(x)$ by the formula:

$$U(x) = u(x_j)\omega_j(x) + u(x_{j+1})\omega_{j+1}(x).$$

This type of approximation can be applied to solving the problem of image enlargement as follows. We assume that $u(x_j)$ and $u(x_{j+1})$ are the values of the intensity of the red color in pixels of the row with numbers j and $j+1$ of the original image. Our task is to enlarge the image by adding one or two pixels in a row with red intensity values. In this case, we use the values of the intensity of the red color in pixels with numbers j and $j+1$. The color intensity in the added pixels is calculated by the formula: $U(x) = u(x_j)\omega_j(x) + u(x_{j+1})\omega_{j+1}(x)$,
Let us investigate the application and the polynomial spline approximation, in addition to the non-polynomial spline approximation ([16], [17], [18]).

In this case we obtain $\omega_j(x_j+th)=\frac{sin(h-th)}{sin(h)}$, $\omega_{j+1}(x_j+th)=\frac{cos(th)}{sin(h)}$ when $x\in[x_j,x_{j+1}]$, $x=x_j+th,\ h=x_{j+1}-x_j,\ t\in[0,1]$.
We put

$$U^T(x_j+th)=u(x_j)\omega_j(x_j+th)$$
$$+u(x_{j+1})\omega_{j+1}(x_j+th),\qquad x_j+th\in[x_j,x_{j+1}].$$

In the case of the splines of the third order of approximation, we distinguish between right and left approximations. Apply basis spline approximation near the beginning of the row

$$\omega_j^R(x)=\frac{(x-x_{j+1})(x-x_{j+2})}{(x_j-x_{j+1})(x_j-x_{j+2})},x\in[x_j,x_{j+1}],$$
$$\omega_{j+1}^R(x)=\frac{(x-x_j)(x-x_{j+2})}{(x_{j+1}-x_j)(x_{j+1}-x_{j+2})},x\in[x_j,x_{j+1}],$$
$$\omega_{j+2}^R(x)=\frac{(x-x_{j+1})(x-x_j)}{(x_{j+2}-x_{j+1})(x_{j+2}-x_j)},x\in[x_j,x_{j+1}],$$

$$U^R(x)=u(x_j)\omega_j^R(x)+u(x_{j+1})\omega_{j+1}^R(x)+u(x_{j+2})\omega_{j+2}^R(x). \qquad (12)$$

We apply the basic spline approximation near the end of the line

$$\omega_j^L(x)=\frac{(x-x_{j+1})(x-x_{j-1})}{(x_j-x_{j+1})(x_j-x_{j-1})},x\in[x_j,x_{j+1}],$$
$$\omega_{j+1}^L(x)=\frac{(x-x_{j-1})(x-x_j)}{(x_{j+1}-x_{j-1})(x_{j+1}-x_j)},x\in[x_j,x_{j+1}],$$
$$\omega_{j-1}^L(x)=\frac{(x-x_{j+1})(x-x_j)}{(x_{j-1}-x_{j+1})(x_{j-1}-x_j)},x\in[x_j,x_{j+1}],$$

$$U^L(x)=u(x_j)\omega_j^L(x)+u(x_{j+1})\omega_{j+1}^L(x)+u(x_{j-1})\omega_{j-1}^L(x). \qquad (13)$$

Theorem 2 estimates the approximation error on the grid interval $[x_j,x_{j+1}]$. Denote $\parallel u \parallel_{[c,d]}=\max_{x\in[c,d]}|u(x)|$.

Theorem 2. Let function $u(x)$ be such that $u\in C^2[a,b]$. We construct the approximation U with the splines of the second order. Then for $x\in[x_j,x_{j+1}]$ we have:

$\parallel u-U\parallel_{[x_j,x_{j+1}]}\le Kh^2\parallel u''\parallel_{[x_j,x_{j+1}]}$, $K=1/8$.

Theorem 3. Let $u\in C^3[a,b]$.To approximate the function $u(x)$, $x\in[x_j,x_{j+1}]$, with spline (13), , the following inequality is valid:

$$|u(x)-U^L(x)|\le Kh^3\parallel u'''\parallel_{[x_{j-1},x_{j+1}]}.$$

To approximate the function $u(x)$, $x\in[x_j,x_{j+1}]$, with spline (12), the following inequality is valid:

$$|u(x)-U^R(x)|\le Kh^3\parallel u'''\parallel_{[x_j,x_{j+2}]}.$$

Proof. It is easy to notice that U_j^R is an interpolation polynomial of the third degree, and x_j,x_{j+1} are the interpolation nodes, $U_j^R(x_j)=u(x_j), U_j^R(x_{j+1})=u(x_{j+1}),\ U_j^R(x_{j+2})=u(x_{j+2})$. Using the remainder term we get

$u(x)-U_j^R(x)=\frac{u'''(\tau)}{3!}(x-x_j)(x-x_{j+1})(x-x_{j+2})$.

It follows that

$$\max_{x\in[x_j,x_{j+2}]}|u(x)-U_j^R(x)|\le\frac{0.385}{3!}h^3\max_{[x_j,x_{j+2}]}|u'''|.$$

Thus, $K\approx 0.064167$.

We can also use the trigonometric splines:

$$\omega_j(x)=\frac{sin(x/2-x_{j+1}/2)sin(x/2-x_{j+2}/2)}{sin(x_j/2-x_{j+1}/2)sin(x_j/2-x_{j+2}/2)},$$
$$x\in[x_j,x_{j+1}],$$
$$\omega_{j+1}(x)=\frac{sin\left(\frac{x}{2}-\frac{x_j}{2}\right)sin\left(\frac{x}{2}-\frac{x_{j+2}}{2}\right)}{sin\left(\frac{x_{j+1}}{2}-\frac{x_j}{2}\right)sin\left(\frac{x_{j+1}}{2}-\frac{x_{j+2}}{2}\right)},$$
$$x\in[x_j,x_{j+1}],$$
$$\omega_{j+2}(x)=\frac{sin\left(\frac{x}{2}-\frac{x_{j+1}}{2}\right)sin\left(\frac{x}{2}-\frac{x_j}{2}\right)}{sin\left(\frac{x_{j+2}}{2}-\frac{x_{j+1}}{2}\right)sin\left(\frac{x_{j+2}}{2}-\frac{x_j}{2}\right)},$$
$$x\in[x_j,x_{j+1}],$$
$$U^{RT}(x)=u(x_j)\omega_j(x)+u(x_{j+1})\omega_{j+1}(x)$$
$$+u(x_{j+2})\omega_{j+2}(x)$$
$$\omega_j(x)=\frac{sin(x/2-x_{j+1}/2)sin(x/2-x_{j-1}/2)}{sin(x_j/2-x_{j+1}/2)sin(x_j/2-x_{j-1}/2)},$$
$$x\in[x_j,x_{j+1}],$$
$$\omega_{j+1}(x)$$
$$=\frac{sin(x/2-x_{j-1}/2)sin(x/2-x_j/2)}{sin(x_{j+1}\,2/-x_{j-1}/2)sin(x_{j+1}/2-x_j/2)},$$
$$x\in[x_j,x_{j+1}],$$
$$\omega_{j-1}(x)$$
$$=\frac{sin(x/2-x_{j+1}/2)sin(x/2-x_j/2)}{sin(x_{j-1}/2-x_{j+1}/2)sin(x_{j-1}/2-x_j/2)},$$
$$x\in[x_j,x_{j+1}].$$
$$U^{LT}(x)=u(x_j)\omega_j(x)+u(x_{j+1})\omega_{j+1}(x)$$
$$+u(x_{j-1})\omega_{j-1}(x).$$

Let us consider the question of how to increase the speed of image processing. First, you can reduce the number of multiplicative operations. Let us go back to the formula

$$U(x)=u(x_j)\omega_j(x)+u(x_{j+1})\omega_{j+1}(x).$$

Here we have 4 multiplicative operations. We write this expression in the form:

$$U(x)=u(x_j)\frac{x-x_{j+1}}{x_j-x_{j+1}}+u(x_{j+1})\frac{x-x_j}{x_{j+1}-x_j}=$$
$$\frac{(x-x_j)}{(x_{j+1}-x_j)}\left(u(x_{j+1})-u(x_j)\right)+u(x_j).$$

There are only two multiplicative operations in this formula. Similarly, you can transform the formulas for interpolation by splines of the third order of approximation.

$$U^R(x) = u(x_j)\frac{(x-x_{j+1})(x-x_{j+2})}{(x_j-x_{j+1})(x_j-x_{j+2})} + u(x_{j+1})\frac{(x-x_j)(x-x_{j+2})}{(x_{j+1}-x_j)(x_{j+1}-x_{j+2})} + u(x_{j+2})\frac{(x-x_{j+1})(x-x_j)}{(x_{j+2}-x_{j+1})(x_{j+2}-x_j)}.$$

There are 12 multiplicative operations in this record. Denote

$$A = (x-x_{j+1})/(x_j-x_{j+2}),$$
$$B = (x-x_j)/(x_{j+1}-x_{j+2}),$$
$$C = (x-x_{j+2})/(x_j-x_{j+1}).$$

Now we have

$$U^R(x) = u(x_j)AC - u(x_{j+1})BC + u(x_{j+2})AB.$$

There are 9 multiplicative operations in this record. Now consider

$$U^L(x) = u(x_j)\frac{(x-x_{j+1})(x-x_{j-1})}{(x_j-x_{j+1})(x_j-x_{j-1})} + u(x_{j+1})\frac{(x-x_{j-1})(x-x_j)}{(x_{j+1}-x_{j-1})(x_{j+1}-x_j)} + u(x_{j-1})\frac{(x-x_{j+1})(x-x_j)}{(x_{j-1}-x_{j+1})(x_{j-1}-x_j)}.$$

Denote $D = (x-x_{j+1})/(x_j-x_{j-1})$,
$E = (x-x_j)/(x_{j+1}-x_{j-1})$,
$F = (x-x_{j-1})/(x_j-x_{j+1})$

$$U^L(x) = u(x_j)DF - u(x_{j+1})FE + u(x_{j-1})DE.$$

In the RGB model, the intensities of the red, green and blue colors range from 0 to 255. In this paper we use local spline approximations of the second or the third order of approximation when restoring a compressed image. The approximation on each grid interval uses the function values at the grid points. If the function changes rapidly, the values of the approximation of the rapid change function may go beyond the boundaries of the interval $[0, 255]$.

The algorithm for constructing approximations whose values do not go beyond the boundaries of the range is as follows. When we use the approximation with the left splines of the third order of approximation, and the values of the approximation go beyond the boundary on some grid interval, then on this interval we apply the approximation with the right splines. If this does not help, then we apply the approximation with the splines of the second order of approximation. An approximation in which the left or right splines of the third order of approximation are used on adjacent intervals, or splines of the second order of approximation are used, will be called mixed.

Consider, for example, the approximation of the function $u(x) = 75(-\sin(3.2\,x) + \cos(1.2\,x))^2$ on the interval [-1,1]. Let us construct a uniform grid $\{x_j\}$ with a step $h = 1/3$ on the interval $[-1,1]$. Let us calculate the values $u(x_j)$ at the grid points. Now, let us compare the approximations constructed using linear and quadratic splines. First, we construct an approximation with the splines of the second order of approximation. The graph of the function $u(x)$ (blue) and it's approximation (red) with the second-order splines is shown in Fig. 1, and the graph of the error of approximation when the second-order splines used is shown in Fig. 2.

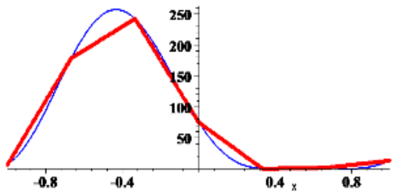

Fig.1. The graph of the function $u(x)$ (blue) and it's approximation (red) with the second-order splines

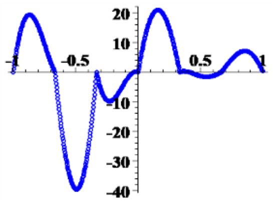

Fig.2. The graph of the error of approximation when the second-order splines are used

Now we construct an approximation with the splines of the third order of approximation. The graph of the function $u(x)$ (blue) and it's approximation (red) with the left third-order approximation splines is shown in Fig. 3, and the error of approximation with the left third-order spline approximation is shown in Fig. 4.

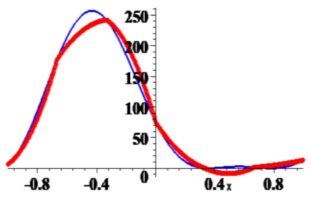

Fig.3. The graph of the function $u(x)$ (blue) and it's approximation (red) with the left third-order approximation splines

The values of the approximation with the left splines takes negative values on the interval [1/3, 2/3]. To solve this problem, we apply the approximation with the right splines, using the same values of the function on the interval [1/3, 2/3].

The graph of the function $u(x)$ (blue) and it's approximation (red) with the left and right third-order approximation splines is shown in Fig. 5. The error of the approximation using the left and right splines is shown in Fig.6.

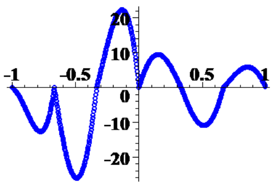

Fig.4. The graph of the error of approximation when the left third-order spline approximation is used

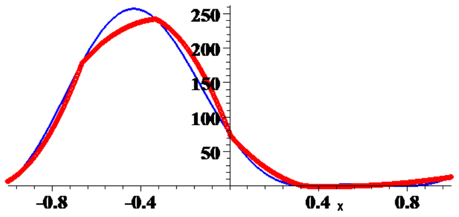

Fig.5. The graph of the function $u(x)$ (blue) and it's approximation (red) with the left and right third-order approximation splines

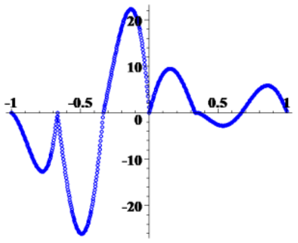

Fig.6. The graph of the error of approximation when the left third-order spline approximation is used

Table 1 shows the maxima in the absolute values of the actual approximation errors of the function $u(x)$ in the interval [-1,1] when $h = 0.01$. Table 2 shows the maxima in the absolute values of the theoretical approximation errors of the function $u(x)$ in the interval [-1,1] when $h = 0.01$. To construct the approximation, the spline approximations $U(x)$ of the second order of approximation and the left spline approximations of the third orders were used.

Table 1 The maxima in the absolute values of the actual approximation errors of the function $u(x)$ in the interval [-1,1].

$u(x)$	$U(x)$	$U^L(x)$
$75(\cos(1.2\,x) - \sin(3.2\,x))^2$	$0.39387\ 10^{-1}$	$0.10668\ 10^{-2}$
$1/(1+25x^2)$	$0.62227\ 10^{-3}$	$0.372323\ 10^{-4}$

Table 2 The maxima in the absolute values of the theoretical approximation errors of the function $u(x)$ in the interval [-1,1].

$u(x)$	$U(x)$	$U^L(x)$
$75(\cos(1.2\,x) - \sin(3.2\,x))^2$	$0.39409\ 10^{-1}$	$0.10678\ 10^{-2}$
$1/(1+25x^2)$	$0.625\ 10^{-3}$	$0.729462\ 10^{-4}$

Note that the use the trigonometric splines can give a smaller error in the approximation, compared to polynomial splines. The second column of Table 3 gives the errors of approximation with the trigonometric splines of the second order of approximation, and the third column gives the errors of approximation with the trigonometric splines of the third order of approximation.

Table 3 The maxima in the absolute values of the actual approximation errors of the function $u(x)$ in the interval [-1,1].

$u(x)$	$U^T(x)$	$U^{LT}(x)$
$75(\cos(1.2\,x) - \sin(3.2\,x))^2$	$0.3619\ 10^{-1}$	$0.1026\ 10^{-2}$
$1/(1+25x^2)$	$0.6098\ 10^{-3}$	$0.3706\ 10^{-4}$

The presented results show that by applying the spline interpolation correctly, it is possible to construct an approximation whose values do not go beyond the specified range. Tables 1, 2, 3 confirm that the obtained actual results of the approximation are consistent with the theoretical ones.

IV. Experiments

The experiment used the painting *Road with Cypress and Star* by Van Gogh [20]. The result of the magnification is shown in Figures 7, 8. Figure 7 shows the original image. Figure 8 shows a part of the enlarged image.

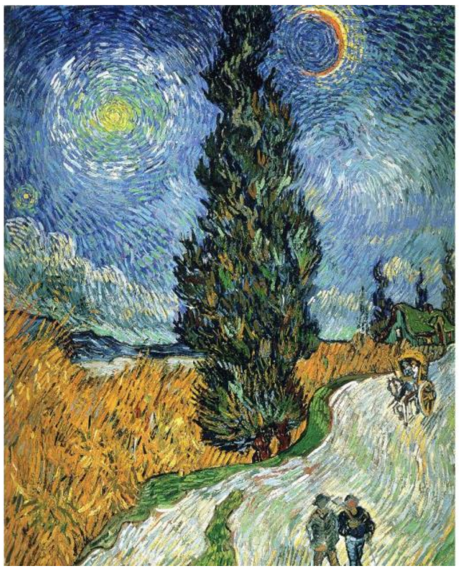

Figure 7. The original image (Van Gogh)

Let us consider how we can parallelize the process of enlarging an image. We will use the concept of geometric parallelism. In the case of splines of the second order of approximation, parallelization can be organized both along several vertical or horizontal stripes (see Figure 9), and along four squares, starting from the vertices (see Figure 10).

Figure 8. A part of the enlarged image (Van Gogh)

As we know, the range of the color channel is as follows: from 0 to 255. Note that when using splines of the third order of approximation, the value may be higher than 255 or less than 0. In this case, the developed program makes the appropriate correction.
The experiment was performed on a 1382×1808 image in the jpg format on the intel core i5 8250U processor. The average time for 5 measurements was measured directly. Image enlargement time (file reading, reduction and saving in jpg format are not taken into account), the time is indicated for optimized spline formulas with fewer multiplicative operations.
Operations in python were performed directly using the language, without using the NumPy library.
The execution time on Python 3.6 using a spline of the second order of approximation turned out to be 48.4 sec.

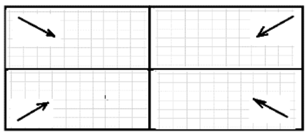
Figure 9. Parallelization can be organized along four squares, starting from the vertices

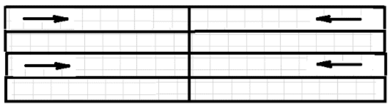
Figure 10. Parallelization can be organized along several vertical or horizontal stripes

The execution time on C++ 11 using the polynomial splines of the second order of approximation turned out to be 0.392 sec. The execution time on C++ 11 using the polynomial splines of the polynomial third order of approximation turned out to be 0.908 sec.

The experiments were performed on parallelization for 4 threads in C++11. The polynomial splines of the second order of approximation turned out to be 0.274 sec. The polynomial splines of the third order of approximation turned out to be 0.562 sec.

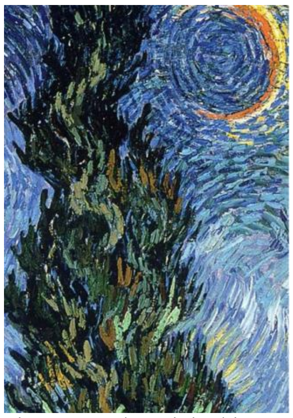
Figure 11. The original image

The results of the program are shown in Fig. 11-13. The original image is shown in Fig. 11. Columns and rows of pixels were removed through one. Figure 12 shows the result of the reconstruction using the polynomial splines of the second order of approximation. Figure 13 shows the result of reconstruction using the polynomial quadratic splines.

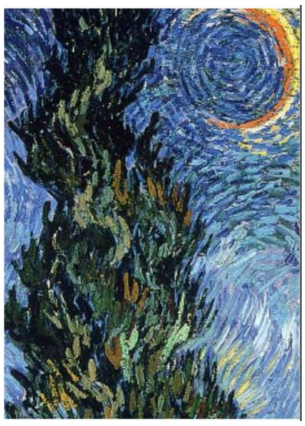
Figure 12. The reconstruction using splines of the second order of approximation

Note that the computation time according to the optimized formulas (with fewer operations) gives a reduction in the computation time even in the case of using splines of the second order of approximation.
The original image Figure 14 and the reconstruction using the polynomial splines of the third order of approximation are shown in Fig.15. The execution time on Python 3.8 using a spline of the second order of approximation turned out to be 4.31 sec. Here the modified form (with the least number of operations) of the splines were used. The reconstruction using the

polynomial splines of the second order of approximation is shown in Fig.10. The execution time on Python 3.8 using a spline of the second order of approximation turned out to be 3.46 sec. Here the modified form (with the least number of operations) of the splines were used.

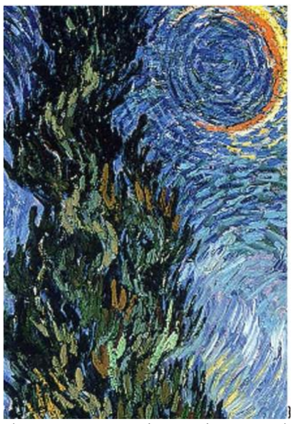

Figure 13. The reconstruction using quadratic splines

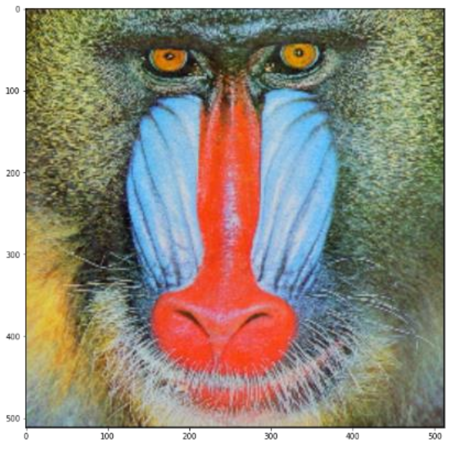

Figure 14. The original image

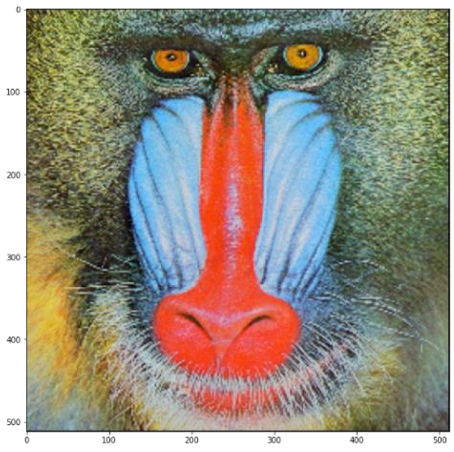

Figure 15. The reconstruction using polynomial splines of the third order of approximation

The disadvantages of using spline approximations of the third order of approximation include the fact that in some cases an excess of the permissible value is observed. In this case, we have to make an appropriate amendment.

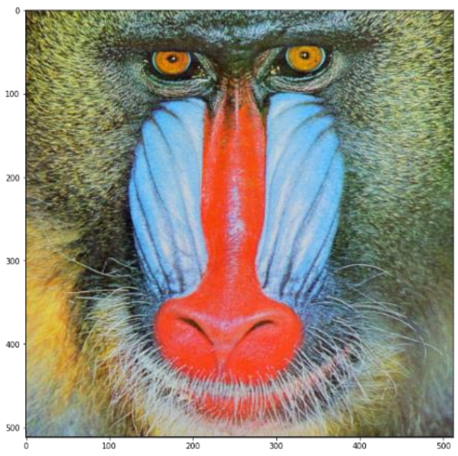

Figure 16. The reconstruction using splines of the second order of approximation

The calculations used Numpy library for Python.
The experiment was performed on a 512x512 image in jpg format on the Intel core i7 3170k processor.

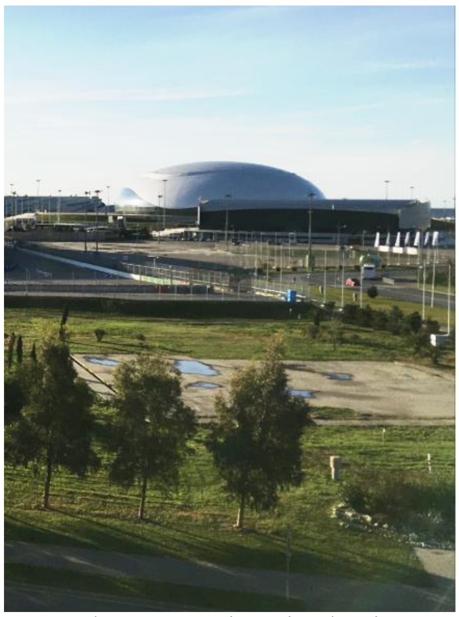

Figure 17. The reconstruction using the trigonometric splines

The execution time when Julia used the polynomial splines of the second order of approximation turned out to be 241 ms. The execution time when Julia used the polynomial splines of the third order of approximation turned out to be 345ms. The execution

time when Julia used the trigonometric splines of the third order of approximation turned out to be 568ms.
Julia 1.5.3 used the Julia Images library for the image manipulation. The tests were run on an Intel Core i5-8265U processor.
Polynomial splines show more acceptable results when applied.
An image with a size of 4032×3024 was processed. Figure 17 shows part of a double-zoomed image using the third-order trigonometric splines.
We have investigated the features of image enlargement in several programming languages. The slowest process was in python. The fastest way is to use splines of the second order of approximation. For a better image, polynomial splines can be recommended. Parallelization on several threads (for example 4) speeds up the process. Note that a multi-threaded Python application performs worse than a single-threaded one.

Acknowledgment

Acknowledgment This paper was prepared with the support by a grant from St. Petersburg State University Event 3 (Pure ID 75207094)

References

[1] A.A.Golitsyn, "Application of image interpolation algorithms in thermal surveillance devices," 2020 1st International Conference Problems of Informatics, Electronics, and Radio Engineering, PIERE 2020, paper 09314695, 2020, pp. 155-159.

[2] M.S.Devi, R.Suguna, P.K. Lbungdim, S. Kondapalli, S.K. Ambashta, D.Akhil, "Systematic image zooming and panning of graphical images using fractional replication," Journal of Computational and Theoretical Nanoscience, vol. 17 (1) , 2020, pp. 519-525.

[3] Z.Han, X.Wu, M. Chi, , J.Tang, L.Yang, "A Novel Approach to Transform Bitmap to Vector Image for Data Reliable Visualization considering the Image Triangulation and Color Selection," Security and Communication Networks, paper 8871588, 2020.

[4] J. Ji, B. Zhong, K.-K. Ma, "Image Interpolation Using Multi-Scale Attention-Aware Inception Network," IEEE Transactions on Image Processing, vol. 29, paper 9210165, 2020, pp. 9413-9428.

[5] V. Skala, M. Cervenka, "Novel RBF Approximation Method Based on Geometrical Properties for Signal Processing with a New RBF Function: Experimental Comparison," Informatics, 2019, pp.357-362.

[6] V.Ramadevi, K.Manjunatha Chari, "FPGA realization of an efficient image scalar with modified area generation technique," Multimedia Tools and Applications, vol.78 (16), 2019, pp. 23707-23732.

[7] V.Skala, S.A.A. Karim, E.A. Kadir, "Scientific Computing and Computer Graphics with GPU: Application of Projective Geometry and Principle of Duality," International Journal of Mathematics and Computer Science, vol.15, No.3, 2020, pp.769-777.

[8] W.Khalaf, D.Zaghar, N.Hashim, "Enhancement of curve-fitting image compression using hyperbolic function," *Symmetry*, vol. 11 (2), paper 291, 2019.

[9] M. Petö, F.Duvigneau, S.Eisenträger, "Enhanced numerical integration scheme based on image-compression techniques: application to fictitious domain methods," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7 (1), paper 21, 2020.

[10] H.N.Zaynidinov, I.Yusupov, J.U.Juraev, J.S. Jabbarov, "Applying two-dimensional piecewise-polynomial basis for medical image processing," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9 (4), paper 156, 2020, pp. 5259-5265.

[11] H. Zaynidinov, J. Juraev, U.Juraev, "Digital image processing with two-dimensional Haar wavelets," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9 (3), 2020, pp. 729-2734.

[12] Roumen Kountchev, Rumen Mironov, Roumiana Kountcheva, "Complexity Evaluation of Tensor Decomposition through 3D Inverse Spectrum Pyramid in Respect of Deterministic Orthogonal Transforms," WSEAS Transactions on Signal Processing, vol. 15, 2019, pp. 142-148.

[13] Roumen Kountchev, Roumiana Kountcheva, "Image Segmentation based on Adaptive Mode Quantization and 2D," Histograms Analysis WSEAS Transactions on Signal Processing, vol. 15, 2019, pp. 121-128.

[14] Luqman Hakim, Muhammad Ihsan Zul, "Implementation of Discrete Wavelet Transform on Movement Images and Recognition by Artificial Neural Network Algorithm," WSEAS Transactions on Signal Processing, vol. 15, 2019, pp. 149-154.

[15] Farzam Kharajinezhadian, Saeid Rashidi, "A Multimodal Authenticationfor Biometric Verification System using Palmprints and Fingers," WSEAS Transactions on Signal Processing, vol. 15, 2019, pp. 129-141, Art. #16.

[16] I.G. Burova, E.F.Muzafarova, I.I. Narbutovskikh, "Approximation by the third-order splines on uniform and non-uniform grids and image processing," WSEAS Transactions on Mathematics, vol.19, 2020, pp. 65-73.

[17] I.G. Burova, "On left integro-differential splines and Cauchy problem," International Journal of Mathematical Models and Methods in Applied Sciences, vol.9, 2015, pp. 683-690.

[18] Yu.K. Dem'yanovich, "Embedded Spaces of Trigonometric Splines and Their Wavelet Expansion," Math. Notes, vol. 78 (5), 2005, pp. 615-630.

[19] Yuri K. Demjanovich, "On Complexity of Adaptive Splines," International Journal of Curcuits, System and Signal Processing, vol.14, 2020, pp.607-615.

[20]https://en.wikipedia.org/wiki/Road_with_Cypress_and_Star