Image processing algorithms construction using Local Polynomial Approximation.

Guennadi (Henry) Levkine (email: hlevkin@gmail.com)

Gibsons, BC, Canada.

Abstract. Many important algorithms in image processing are introduced empirically. For example, image filtration, edge detection and so on. Here we described new method of Local Polynomial Approximation of image brightness (LPA) for researching of existing empirical algorithms and creating new ones. We consider tasks: image noise filtration, image segmentation, gradient calculation, Laplace operator implementation, image down sampling. Local areas 5x5 are considered where empirical methods of algorithms construction do not work.

Key words: image processing, convolution kernel, noise filtration, image gradient, image resize, feature points, Laplace transform

1. Introduction.

Among different approaches to image processing some of the most popular are:
- Spectral transformation of images or their local parts-blocks (for example Fourier transform)
- Convolutions of images with square matrices of size 3x3, 5x5, 7x7. These square matrices are called convolution kernels.

The first approach was described with applications in many books (look [2], [3], [4] for example). It is based on the theory of orthogonal/unitary transformations.

The second approach was first described in [1] and after what developed in many articles (looks books [2], [3], [4] for examples). For the second approach we have to build convolution kernels of small sizes 3x3, 5x5 and so on for implementation of corresponding algorithms.

Usually it is doing empirically.

In LPA for building of such kernels we locally approximate image intensity by polynomials of two variables (rows and columns) and after that use these coefficients for kernel building.
We give here kernels for calculation of polynomial coefficients of different size and shape. We demonstrate how to use these kernels for noise filtration, gradient estimation, Laplace operator kernels, resizing and other tasks. The coefficients are calculated using the linear least squares fitting technique.
We describe how to use kernel with different sizes and polynomial powers for different pixels of image. It gives high flexibility and as a result high quality of processing.

2. Two dimensional local polynomial approximation (LPA) and pixel enumerating.

When we use LPA for every pixel in image we calculate 2D polynomial which approximate image intensity in some block around the pixel.
A common view for such a polynomial of second power is

$$A(x,y) = a + b*x + c*y + d*x*x + e*x*y + f*y*y \qquad (1)$$

For the coefficients a, b, c, d, e, f calculation we use standard Least Square Fit method for minimization of difference between our approximation A(x, y) of brightness and real brightness in image.
We do it for blocks of sizes 3x3 or 5x5, surrounding every pixel in image. Coordinates of pixels in block are (x, y).

For block 3x3 around the pixel these coordinates are

(0, 0)	(1, 0)	(2, 0)
(0, 1)	(1, 1)	(2,1)
(0, 2)	(1, 2)	(2, 2)

For block 5x5 around the pixel:

(0, 0)	(1, 0)	(2, 0)	(3,0)	(4,0)
(0, 1)	(1, 1)	(2,1)	(3,1)	(4,1)
(0,2)	(1, 2)	(2, 2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)

3. Linear local approximation in 3x3 and 5x5 blocks.

In this case we use simple linear approximation

$$I(x,y) = a + b * x + c * y \qquad \textbf{(2)}$$

For calculation of **a, b, c** coefficients 3x3 convolution kernels are (Weights are 1/9, 1/6, 1/6) :

1	**1**	**1**
1	**1**	**1**
1	**1**	**1**

-1	**0**	**1**
-1	**0**	**1**
-1	**0**	**1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

(Concept of convolution kernels is defined in [6])

If we using 5x5 blocks, corresponding convolution kernels are (with weights 1/25, 1/50, 1/50):

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2

-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

These methods are popular for noise filtration and edge detection but gives not good results for images with thin details.

4. Quadric polynomial approximation in 3x3 and 5x5 blocks

In this case we use approximation (1):

$$A(x,y) = a + b * x + c * y + d * x * x + e * x * y + f * y * y$$

Coefficients **a, b, c, d, e, f** are estimated by kernel convolution with weights 1/9, 1/6, 1/6, 1/6, 1/4, 1/6

-1	**2**	**-1**
2	**5**	**2**
-1	**2**	**-1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

-1	**-1**	**-1**
0	**0**	**0**
1	**1**	**1**

For 5x5 blocks convolution kernels are (with weights 1/175, 1/50, 1/50, 1/70, 1/50, 1 /70):

-13	2	7	2	-13
2	17	22	17	2
7	22	27	22	7
2	17	22	17	2
-13	2	7	2	-13

-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2

-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2

2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2
2	-1	-2	-1	2

+4	+2	0	-2	-4
+2	+1	0	-1	-2
0	0	0	0	0
-2	-1	0	+1	+2
-4	-2	0	+2	+4

Quadric LPA saves much more information about the image when compared with linear LPA. We see here non-standard convolution kernel for coefficient **a**. It has negative elements. Usually kernels with positive elements are used. Coefficients **b** and **c** have the same kernels as in linear LPA. Coefficients **d, f** detects vertical and horizontal ridges / valleys on image. Coefficient **e** detects saddle points.

5. LPA application to noise filtration.

Noise filtration of images using Local Polynomial Approximation (LPA) is based on assumption that the Image pixels can be presented by set of formulas:

$$\boldsymbol{A}\,(x, y) = \boldsymbol{I}\,(x, y) + \boldsymbol{n}(x, y)$$

Here $A\,(x, y)$ is an observable image, which is the sum of "ideal" image brightness $\boldsymbol{I}\,(x, y)$ without noise and brightness $n\,(x, y)$, which is the noise in pixel with coordinates (x, y) .

We assume that "ideal" image can be represented locally by set of polynomials. But it is not an analytic function: changes in one part of image are very often do not influence to other parts of the image. We also assume that the noise $n(x, y)$ is Gaussian with 0 mean value and the standard deviation $s(x, y)$ which can varies from pixel to pixel. We also have to make assumption about the power of polynomial n, which can be 1, 2, 3, 4.
If we use LPA, that means that coefficient 'a' for a selected pixel is a LSF approximation of image brightness for this pixel:

$$I(x, y) \sim a$$

It means that we can use for filtration coefficient **a** in polynomial approximation.
For example if we make assumption that in blocks of 3x3 images can be described by linear polynomials, that means we can use this kernel for noise filtration:

1	1	1
1	1	1
1	1	1

If we think that the best is a weighted linear approximation we have to use Rosenfeld kernel:

1	2	1
2	4	2
1	2	1

But if image has many thin details this algorithm destroys them. That's why Rosenfeld's filter now called "blurring filter". We can use for that case quadric LPA filter (Levkin's 3x3 filter):

-1	2	-1
2	5	2
-1	2	-1

In case 5x5 we recommend use Levkin's 5x5 filter which also helps to save details.

-13	2	7	2	-13
2	17	22	17	2
7	22	27	22	7
2	17	22	17	2
-13	2	7	2	-13

6. Image segmentation to linear and quadric parts.

If after linear LPA filtration using kernel A (W=1/6) the image does not changed, that means that image locally linear (human eye can't detect changes):

Kernel **A**

1	1	1
1	1	1
1	1	1

Image is 3x3 quadratic, if after filtration using kernel B (W=1/9) it does not change:

Kernel **B**

-1	2	-1
2	5	2
-1	2	-1

For any image it is possible to get linear part of pixels by multiple filtration of it until result will not change, Let's use standard grey Lenna image from [6] . After around 550 repetitions of filtration with the first kernel **A** result image stop changing with new repetitions. It is in Picture 1. Pixels which are not changed – linear pixels.

Now we show result of using second kernel **B**. After around 70 repetitions of filtration with the second kernel here we see that result image has stopped changing. It is shown in the Picture 2. Pixels which are not linear and in addition do not change - are quadric pixels.

The rest of the pixels are not approximated by quadric LPA filter. We call them feature points. Brightness of these pixels can be interpolated only

We developed the program which for every pixel estimates types of surrounding block. It generates the image in which every linear pixel is black, every quadric pixel is grey, and every complex approximated pixel is white. We found that complex approximated point can be treated as feature points (look at the picture 3)

This pixel division can be used for fine image processing, where we first of all define the type of pixel and after that select the proper kernel.

Similar investigations can be conducted using 5x5 LPA convolution matrices.

7. Image Gradient calculation using LPA.

Gradient of image A(x, y) is defined as a vector with coordinates

$$G_x = \frac{\partial A}{\partial x}, \qquad G_y = \frac{\partial \mathbf{A}}{\partial \mathbf{y}}$$

In polynomial approximation the components of gradient are: $G_x = b$, $G_y = c$

Using different power of the approximating polynomial and different kernel mask we can generate many different convolution kernel matrices for gradient estimation. Here we show kernels for gradient calculations for 3x3 and 5x5 image blocks .

Linear and quadric 3x3 LPA (weight $\frac{\mathbf{1}}{\mathbf{6}}$):

- 1	0	1
- 1	0	1
- 1	0	1

- 1	- 1	- 1
0	0	0
1	1	1

Rosenfeld weighted kernel (Sobel operator) (weight $\frac{\mathbf{1}}{\mathbf{8}}$):

- 1	0	1
- 2	0	2
- 1	0	1

- 1	- 2	- 1
0	0	0
1	1	1

Linear and quadric 5x5 LPA (weight $\frac{\mathbf{1}}{\mathbf{50}}$):

- 2	- 1	0	1	2
- 2	- 1	0	1	2
- 2	- 1	0	1	2
- 2	- 1	0	1	2
- 2	- 1	0	1	2

- 2	- 2	- 2	- 2	- 2
- 1	- 1	- 1	- 1	- 1
0	0	0	0	0
1	1	1	1	2
2	2	2	2	2

Cubic 5x5 LPA (weight=1/420):

31	- 5	- 17	- 5	31
-44	-62	-68	-62	-44
0	0	0	0	0
44	62	68	62	44
-31	5	17	5	-31

These gradient kernels are used in edge detection algorithms.

8. Discrete Laplace operators for 3x3 and 5x5 blocks.

Laplace Transform for continuous image A(x, y) is defined as a scalar field:

$$L(x,y) = \frac{\partial^2 A}{\partial x^2} + \frac{\partial^2 A}{\partial y^2}$$

For polynomial scalar field result of Laplace transform is equals to (2*__d__+2*__f__)

Here we consider Discrete Laplace Transform (DLT) for discrete scalar fields. In [8] (section 1.5) they demonstrate classic Laplace transform kernels

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	0
1	1	1

Our kernel (using LPA) is the sum of (**2*d+2*f**) (W equals 1/3):

2	-1	2
-1	-4	-1
2	-1	2

Here is another new DLT (W=1/3):

1	0	-1
0	0	0
-1	0	1

It also gives after convolution (2*d+2*f) for polynomial scalar fields. It is the simplest kernel 3x3 for Discrete Laplace Transform.

Which kernel from these four we have to use? Kernel action should be as much possible close to continuous Laplace transform.
For their testing we select three brightness blocks 3x3 with brightness which described by three polynomials **P(x, y) = x*x, P(x, y) = (x*x+ y*y), P(x, y) = (x*x – y*y)** correspondently:

1	0	1
1	0	1
1	0	1

1	0	1
0	0	0
1	0	1

-1	0	1
0	0	0
1	0	-1

Continuous Laplace Transform gives 0 when applied to these scalar fields.

Application of the first DLT detects to these brightness blocks gives -2, 0, and 0.
Application of the second DLT to these brightness blocks gives 6, 4, and 0.
Application of the third DLT to these brightness blocks gives 6, 8, and 0.
It means that:

- First DLT detects vertical lines in 3x3 brightness block
- Second DLT detects vertical lines and holes in 3x3 brightness block
- Third DLT (created using LPA)

Last DLT gives 0 in all three cases.

When we use first three DLT we have to keep in mind, that they are not consistent with continuous Laplace transform. But the last DLT does.

Similar situation is in the 5x5 case. We have many different DLT. The simplest one is

1	1	1	1	1
1	1	1	1	1
1	1	-24	1	1
1	1	1	1	1
1	1	1	1	1

It gives detection of many additional 5x5 brightness block use cases.

In quadric LPA convolution kernel for DLT for case 5x5 is (W=1/35):

4	1	0	1	4
1	- 2	- 3	- 2	1
0	- 3	- 4	- 3	0
1	- 2	- 3	- 2	1
4	1	0	1	4

They need special research similar which we did for 3x3 DLTs.

9. Image down-sampling based on LPA.

Here we consider image LPA method for a special case of resizing of The images: reducing image size twice for width and height. Usually it is called “down-sampling”.
Standard down-sampling algorithm is based on substitution of 2x2 blocks by one pixel. Value of a pixel is mean value of pixels in 2x2 block.
LPA based down-sampling uses 4x4 blocks for calculation of a coefficient “a”. After that internal 2x2 block in 4x4 is substituted by pixel with “a” value.
If we use cubic LPA, that convolution matrix for estimation of “a” equals (W=1/32)

- 3	2	2	- 3
2	7	7	2
2	7	7	2
- 3	2	2	- 3

Down sampling is widely used in image processing and image compression. This dawn sampling, based on LPA, essentially improve its quality.

10. Conclusions.

Local polynomial approximation (LPA) of images is a powerful method for the building of new algorithms in image processing.
Pixel classification (based on LPA) onto linear, quadric and complex pixels will be useful for image processing without image degradation and for image feature point detection.

Two new 3x3 Discrete Laplace Transforms were submitted with their description. Analyses of all four DLTs demonstrated what three of them are not consistent with continuous Laplace Transform.

11. Acknowledgements.

I gratefully acknowledge V. Levkine for the support in writing and reviewing of this article.

References:

[1] A. Rosenfeld, “Picture Processing for Computer”, Academic Press, NY, 1969
[2] W. Pratt “Digital Image Processing”, Wiley-Interscience, 2007
[3] M. Sonka, V. Hlavac, R. Boyle “Image processing, analysis and Machine Vision”,
[4] K. Castleman, Digital Image Processing, Prentice-Hall, 1979
[5] Handbook of Image and Video Processing, editor Al Bovik, Elsevier, 2005
[6] https://en.wikipedia.org/wiki/Kernel_(image_processing)
[7] www.hlevkin.com/TestImages/lenna.bmp
[8] https://en.wikipedia.org/wiki/Discrete_Laplace_operator

Picture 1. Linear 3x3 approximated Lenna.

Picture 2. Quadratic 3x3 approximated Lenna

Picture 3. Segmentation of Lenna image. (3 colors used for different segment types: black, grey and bright)