UNIVERSITY OF OSTRAVA

DOCTORAL THESIS

2014 MGR. PAVEL VLAŠÁNEK

UNIVERSITY OF OSTRAVA
FACULTY OF SCIENCE

DEPARTMENT OF INFORMATICS AND COMPUTERS

Inpainting method based on the F-transform

Doctoral thesis

Author: Mgr. Pavel Vlašánek

Supervisor: Prof. Irina Perfiljeva, CSc.

2014

OSTRAVSKÁ UNIVERZITA V OSTRAVĚ
PŘÍRODOVĚDECKÁ FAKULTA

KATEDRA INFORMATIKY A POČÍTAČŮ

Doplnění chybějících dat v obrazu využitím F-transformace

Disertační práce

Autor práce: Mgr. Pavel Vlašánek

Vedoucí práce: Prof. Irina Perfiljeva, CSc.

2014

First of all, I would like to express my huge thanks to my supervisor, Prof. Irina Perfiljeva, CSc. for her comments, suggestions, motivation and patience. It would be nearly impossible for me to write my dissertation and progress our research without this kind of valuable mentoring. Thank you. I would also like to thank my girlfriend Renata for support and tolerance over the hours, days, and years of studying, researching, writing, and programming. Last but not least I would like to thank for all opportunities for publications, travels, meeting interesting people, discussions with colleagues, etc. I think that these points are crucial for successful progressing in science research.

Já, níže podepsaný student, tímto čestně prohlašuji, že text mnou odevzdané závěrečné práce v písemné podobě i na CD nosiči je totožný s textem závěrečné práce vloženým v databázi DIPL2.

Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

V Ostravě dne 30. 5. 2014 .....

podpis

RESUME

We propose a new image inpainting technique which uses approximation properties of the fuzzy (F-)transform. The proposed technique is based on the inverse F-transform and combines it with an original image on an undamaged or missing area. We present two algorithms of the F-transform based reconstruction: one-step and multi-step. We demonstrate how these algorithms cope with various damage and compare it with interpolation and advanced inpainting techniques. We show various application in addition to inpainting, such as resampling, filtering, or denoising. Experimental results are based on testing on various sets of grayscale and color images.

Key Words: F-transform, image reconstruction, approximation, interpolation, inpainting

ANOTACE

Navrhujeme novou techniku doplnění chybějících dat v obrazu založenou na aproximaci použitím F-transformace. Technika využívá inverzní F-transformaci, jejíž výstup kombinuje s původním obrázkem. V disertační práci je naše metoda rozvedena do dvou algoritmů: jednokrokového a vícekrokového. Rekonstrukce s využitím F-transformace je porovnána s interpolací a běžně používanými metodami inpaintingu. Využití F-transformace je také demonstrováno pro další oblasti, jako je vzorkování, filtrování nebo odstranění šumu. Výsledky jsou založeny na testování množství různých barevných obrázků i obrázků v odstínech šedi.

Klíčová slova: F-transformace, rekonstrukce obrazu, aproximace, interpolace, inpainting

Contents

1 Introduction

Image inpainting is a process of filling in unknown or damaged areas. It is a technique of modifying an image and making it as close to an undamaged one as possible. Traditionally it refers to practice of professional artist. Restoration of the art is commonly a time-consuming and highly professional affair. Digital restoration tries to simulate this practice and fix digital images in a similar way. Among many fields in image reconstruction, we distinguish processes such as image inpainting, image denoising, or image resampling for example. These categories are not strictly separated.

The process of digital image inpainting and also restoration started with the selection of the damaged area in the damaged image. These areas can be very varied and thus it is necessary to select them manually. In digital image inpainting, we use an image with the same size as the damaged one called *mask*. The mask image can be considered as another layer on top of the damaged image. Black pixels in the mask are used as a marker of a damaged or unknown area in the damaged image.

The known solutions of the reconstruction problem are based on the interpolation technique [38, 21]. In many cases, interpolation function contains unknown parameters that can be found as a solution of large systems of linear equations. Therefore, the complexity of this approach is rather high. We propose to solve the problem of reconstruction with the help of an approximation technique. This means that we will be looking for an approximating image that is close to a given one and at the same time does not contain what we recognize as damage. The following are practical examples where the problem of reconstruction or inpainting is successfully applied: erasing time stamp from a photography, erasing cracks from a fresco, or erasing anything we want from a digitalized image.

We propose to reconstruct a damaged image with help of a fuzzy technique, namely the F-transform. In the last ten years, the theory of F-transforms has been intensively developed in many directions [8, 28, 25, 7, 27, 24, 33, 34]. In image processing, it has successful applications in image compression and reduction, image fusion, edge detection, noise removing, etc. [23, 8, 39, 26, 29]. F-transform can approximate the original function with an arbitrary precision [23] and thus it can be chosen as an appropriate technique.

2 Defining the issues and basic concepts

Let us explain terms used in the dissertation.

Bit Basic unit of information. A bit can have only one from two values, where the most common interpretations of these values are 0 and 1.

Byte Unit of information that consists of eight bits.

Color Derives from the spectrum of light. We will use a combination of the eight bit color channels, R as red, G as green, and B as blue. This method of color definition is called an RGB model. Every channel of the RGB model contains the amount of the specified color from the scale $\lfloor 0, 255 \rfloor$. The brightest red color is $(255, 0, 0)$, the brightest blue is $(0, 0, 255)$, or darker purple $(100, 0, 100)$.

Alpha channel Additional information for the pixel color. The alpha channel contains the level of transparency from the scale $\lfloor 0, 255 \rfloor$ where 0 stands for full transparency and 255 stands for full opacity.

Intensity Intensity $u(i, j)$ is related to a grayscale image, where $u(i, j) \in \lfloor 0, 255 \rfloor$ where 0 stands for black and 255 stands for white. Every pixel in a grayscale image has one channel with shades of grey instead of amount of color.

Pixel The smallest part of the image, one point in the raster. Every pixel has coordinates (i, j) and color/intensity.

Transformation from a color pixel to a grayscale pixel is as follows

$$u(i, j) = 0.299u_R(i, j) + 0.587u_G(i, j) + 0.114u_B(i, j),$$

where (i, j) stands for x and y coordinates of the pixel, u_R, u_G, u_B are red, green and blue channels from the input color image. In the dissertation, we will use notation $u(i, j)$ for intensity of the (i, j) pixel. The algorithm extensions for color images consist in threefold application, one per color channel.

1D/2D Abbreviations for the function of one variable (1D) $f(x)$, respectively function of two variables (2D) $f(x, y)$.

Image Two dimensional discrete function represented as a matrix. An image is composed of pixels where colors/intensities of every pixel form the matrix elements.

Binary image An image composed of pixels, where all pixels can have one out of two intensities. Most common is 0 for black and 1 for white.

Lena The image of the Lena Söderberg in Fig. 1 commonly used as etalon in computer graphic.

Figure 1: Lena Söderberg.

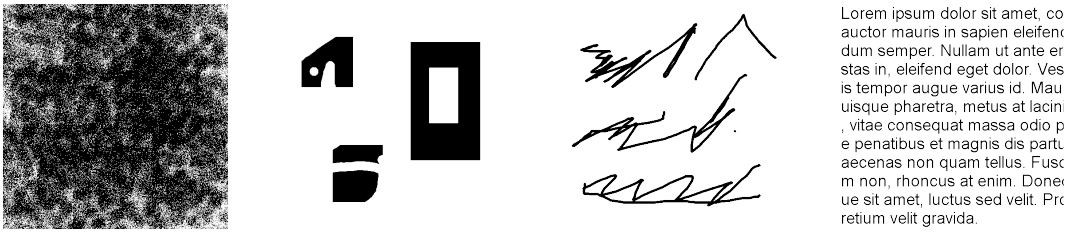
Convolution Operation producing a function from two input functions. The first function is image u and the second is kernel g. Two dimensional discrete forms are as follows

$$(u * g)(i, j) = \sum_{d=-k}^{k} \sum_{e=-k}^{k} u(i - d, j - e) \cdot g(d, e),$$

where k stands for the size of the kernel.

Fuzzy logic Generalizes common Boolean logic and appends the values between *true* and *false*. Fuzzy logic will be used as a tool to process vague object characteristic.

Damage We distinguish four types of damage: *noise*, *holes*, *scratches* and *text* in the dissertation. It can be seen in Fig. 2.



(a) Noise (b) Holes (c) Scratches (d) Text

Figure 2: Different damage types.

Noise damage in Fig. 2a corresponds to the noisy image, *holes* damage in Fig. 2b corresponds to bigger gaps which have to be filled in, *scratches* damage in Fig. 2c corresponds to cracks, folds, scribbles, and *text* damage in Fig. 2d corresponds to unwanted text in the image.

Damaged image An input image intended for inpainting or reconstruction. The damaged or incomplete image includes original information and also damaged or unknown areas.

Undamaged image The original image without damaged or unknown pixels.

Mask A binary image of the same size as the damaged image. Pixels of the first color indicate the damaged area and pixels of the second color indicate the undamaged area. A partially damaged image u is a discrete function that is defined on domain $P = \{(i, j) \mid i = 1, 2, \ldots, M; j = 1, 2, \ldots, N\}$ and is damaged on domain Ω. The characteristic function of Ω is *mask* m_Ω.

RMSD **R**oot **M**ean **S**quare **D**eviation, also RMSE[1] is a criterion used for comparison of the original and reconstructed image. It works as follows

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(u_o(i,j) - \hat{u}(i,j))^2}{M \cdot N}},$$

where $\hat{u}(i, j)$ stands for the intensity of the reconstructed pixel, $u_o(i, j)$ is the intensity of the original pixel, M is the width of the image, and N its height.

SSIM **S**tructural **SIM**ilarity. An advanced criterion taking on also natural perception features of the human eye [45].

$$SSIM(w1, w2) = \frac{(2\mu_{w1}\mu_{w2} + C_1)(2\sigma_{w1w2} + C_2)}{(\mu_{w1}^2 + \mu_{w2}^2 + C_1)(\sigma_{w1}^2 + \sigma_{w2}^2 + C_2)},$$

where $C_i = (K_i L)^2$, L is a dynamic range[2], $K_1 = 0.01$ and $K_2 = 0.03$ are constants recommended by authors and $w1, w2$ stands for various windows of the image.

Anisotropic diffusion A technique for reducing image noise without removing significant parts of the image such as edges.

2.1 Image reconstruction

Image reconstruction has many interpretations. For instance, a damaged valuable painting can be restored by a skilled professional artist. In that case, we are talking about *image restoration*. Look at Fig. 3.

Another meaning is filling in gaps called *image inpainting*. These gaps can occur due to object removing from the image, which is shown in Fig. 4. Manual inpainting commonly consists in using tools like *clone stamp* or *healing*[3].

Resampling may also be considered as an image reconstruction problem. Upsampling after resizing is in Fig. 5a, 5b and downsampling after resizing is in Fig. 5c, 5d.

In general, we have a damaged or incomplete image, see Fig. 3a. We can also say that we want to erase something from the image, see Fig. 4a. A demonstration of the inpainting and erasing an object without some artifacts is shown in Fig. 3b, 4b. In the dissertation, we propose a new technique of the image reconstruction which

[1] **R**oot **M**ean **S**quare **E**rror

[2] 255 for 8 bit channel.

[3] Available or partially available in GIMP, Photoshop, Paint.NET etc.

(a) Before restoration

(b) After restoration

Figure 3: Restoration of the painting. Figure taken from Wikipedia (`http://en.wikipedia.org/wiki/Painting_restoration`).

(a) Before inpainting

(b) After inpainting

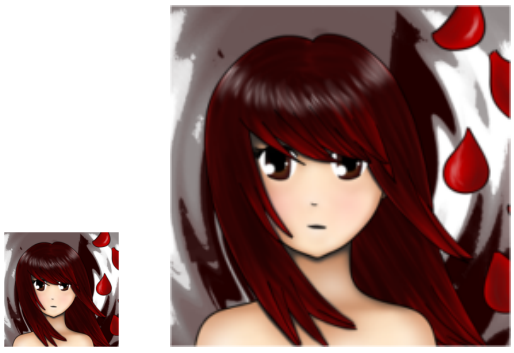
Figure 4: Inpainting of the image processed manually by clone stamp.

is able to reconstruct images, inpaint images, etc. Inputs for our algorithm are a damaged image and a mask. An example of a damaged image and a mask of the corresponding damaged area (inputs) can be seen in Fig. 6.

The reconstruction process is illustrated in Fig. 7. We see the input image, mask, and reconstructed image after application of the proposed F-transform technique. We use the term image reconstruction mainly in the image inpainting meaning in the dissertation.

2.2 Notation

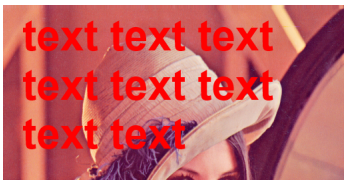
Notation in Tab. 1, 2 will be used in the dissertation. Fig. 8 shows basic terms used in image inpainting.



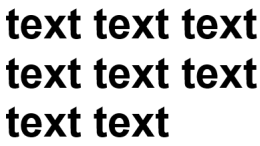
(a) (b)

(c) (d)

Figure 5: Image resampling processed hand to hand with image resizing.



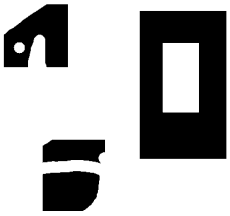
(a) Original



(b) Mask

Figure 6: Mask derived from the damage.

(a) Input damaged image



(b) Mask

(c) Reconstructed output

Figure 7: Reconstruction process.

Entity	Notation
undamaged image	u_o
input image	u
mask	m_Ω
undamaged region	Φ
damaged region	Ω
inner boundary of the damaged region	$\delta\Omega$
outer boundary of the damaged region	$\tilde{\Omega}$
interpolated/approximated image	$\hat{u}$
reconstructed/inpainted image	u^r
image in the iterative process	u^z
pixel in the $\delta\Omega$	S
pixel in the Φ	O

Table 1: Image related notation.

Entity	Notation
basic functions in x direction	A_k
basic functions in y direction	B_l
radius of the basic function	h

Table 2: Basic functions related notation.

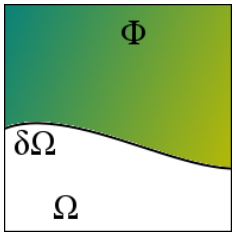


Figure 8: Basic terms.

3 Commonly used techniques

There are many methods for filling in unknown pixels or replacing damaged pixels. For instance, based on interpolation and/or inpainting. This section is divided into two subsections describing them. The first one called *Interpolation* describes algorithms based on the interpolation among undamaged (known) points. The second subsection called *Inpainting* describes advanced techniques commonly used in reconstruction of the damaged area with usage of the undamaged (known) area. In the next section, approximation methods are explained.

3.1 Interpolation

Interpolation extends the domain of a partial function and keeps the input points unchanged on the domain. Interpolation is deeply investigated and there are many available sources, for example [49, 17, 1, 36]. Computation is demonstrated on an example in Fig. 9.

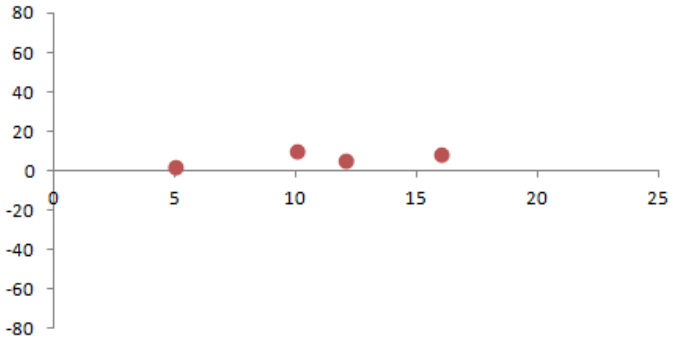


Figure 9: Input points.

The plain coordinates of the red points are as follows

$$[5, 2]; [10, 10]; [12, 5]; [16, 8]$$

In order to find an interpolation polynomial of the 3^{rd} degree, we have to solve the system of the following equations

$$\begin{aligned} y_0 &= a_0 + a_1 x_0 + a_2 x_0^2 + a_3 x_0^3, \\ y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3, \\ y_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3, \\ y_3 &= a_0 + a_1 x_3 + a_2 x_3^2 + a_3 x_3^3, \end{aligned}$$

where y_i, x_i stands for y and x coordinates of the point i. The solution leads to

$$y = -96.7792 + 33.9582x - 3.3529x^2 + 0.1025x^3.$$

Fig. 10 shows the desired interpolation by the polynomial of degree three.

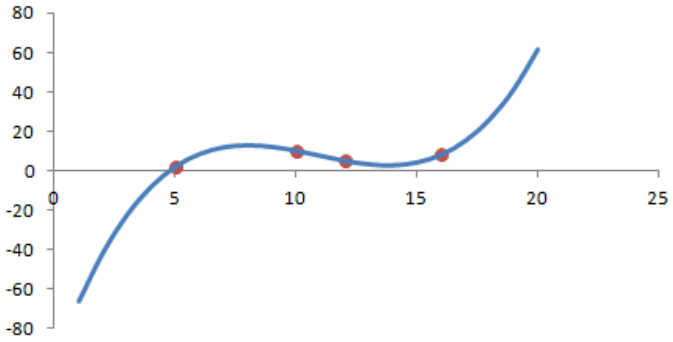
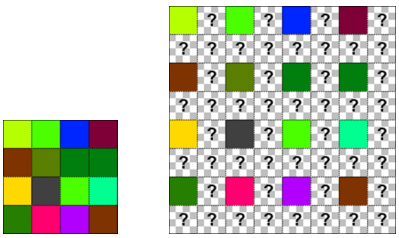


Figure 10: Interpolation by the polynomial of degree three.

Interpolation techniques are commonly used for image resampling in a regular grid. Two interpolation methods commonly used in image resampling *nearest neighbor* and *bilinear* will be extended for the simple usage in an irregular grid.

3.1.1 Regular grid

Image resampling [2] is a common task in a regular grid. In the dissertation, we are using resampling for computation of the unknown pixels of the image after resizing. Situation before upsamling is in Fig.11.



(a) Input (b) Resized

Figure 11: Situation before upsamling where "?" marks unknown pixels.

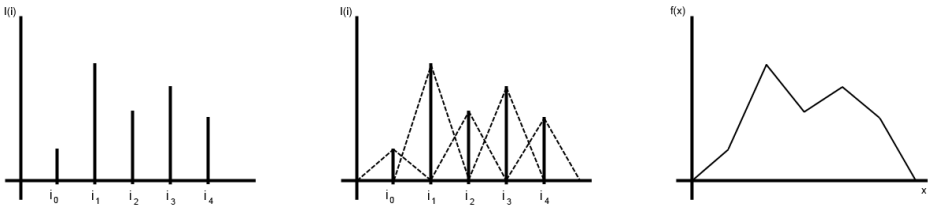
If we want to upsample an image two times, there is a given pattern of the known pixels, see Fig. 11b. Interpolation is commonly described with help of the kernel functions. We present 1D ones because of better illustration. These must be used twice for a 2D one for the x and one for y direction. A demonstration of the linear interpolation is shown in Fig. 12.

A description of the nearest neighbor, bilinear and bicubic interpolation, follows with a demonstration of the resampling from 4×4 image to 512×512 image.

Nearest neighbor The color/intensity of the unknown pixels is determined as follows

$$\hat{u}(i,j) = u(Q_{op}),$$

where Q_{op} is the nearest known pixel with respect to the pixel at position (i,j).



(a) Input function (b) Convolution kernel (c) Result

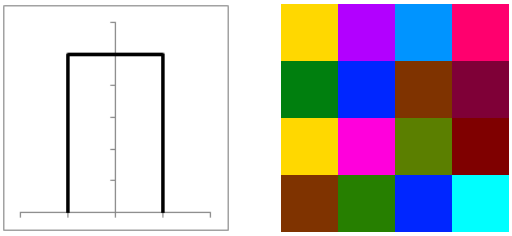
Figure 12: Convolution of the discrete function with the linear kernel.

Because we want to minimize the complexity of this technique, we will ignore diagonal directions.

The convolution kernel is as follows

$$W(x) = \begin{cases} 1 & \text{if } 0 \leq |x| < \frac{1}{2}, \\ 0 & \text{if } \frac{1}{2} \leq |x|. \end{cases}$$

A visualization is provided in Fig. 13a and an upscaled and upsampled 4×4 image is in Fig. 13b.



(a) Kernel function (b) Upsampled

Figure 13: Rectangular kernel and image upsampled by the nearest neighbor interpolation.

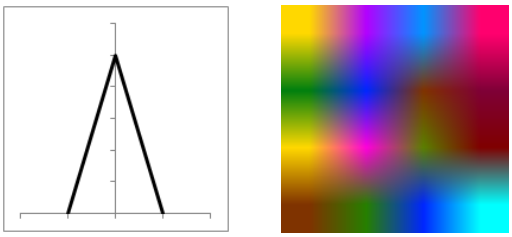
Bilinear interpolation Let us assume that we have four given pixels $Q_{00} = (i_0, j_0), Q_{01} = (i_0, j_1), Q_{10} = (i_1, j_0), Q_{11} = (i_1, j_1)$. Moreover, $i_0 \leq i \leq i_1$ and $j_0 \leq j \leq j_1$.

$$\begin{aligned} \hat{u}(i,j) = \tfrac{1}{(i_1-i_0)(j_1-j_0)}(\quad & u(Q_{00})(i_1 - i)(j_1 - j) + \\ & u(Q_{10})(i - i_0)(j_1 - j) + \\ & u(Q_{01})(i_1 - i)(j - j_0) + \\ & u(Q_{11})(i - i_0)(j - j_0)). \end{aligned}$$

The 1D convolution kernel is as follows

$$W(x) = \begin{cases} 1 - |x| & \text{if } 0 \leq |x| < 1, \\ 0 & \text{if } 1 \leq |x|. \end{cases}$$

A visualization is provided in Fig. 14a and an upsampled image is in Fig. 14b.



(a) Kernel function (b) Upsampled

Figure 14: Triangular kernel and image upsampled by the bilinear interpolation.

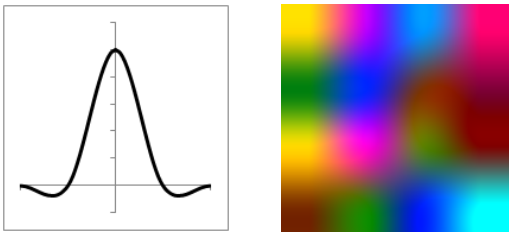
Bicubic interpolation The bicubic interpolation is as follows

$$\hat{u}(i,j) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j,$$

where 16 coefficients a_{ij} have to be determined. The 1D kernel is defined as follows

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{if } |x| \leq 1, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{if } 1 < |x| < 2, \\ 0 & \text{otherwise.} \end{cases}$$

where $a = -0.5$. The visualization is provided in Fig. 15a and an upsampled image is in Fig. 15b.



(a) Kernel function (b) Upsampled

Figure 15: Cubic kernel and image upsampled by the bicubic interpolation.

3.1.2 Irregular grid

We extend the nearest neighbor and bilinear interpolation techniques for usage in an irregular grid. The difference shown is in Fig. 11b and Fig. 16. The image of Lena with damaged areas in Fig. 17 will be used as a demonstration.

Nearest neighbor Let us assume that we want to compute the intensity of the unknown pixel marked as A in Fig. 16. First, we will find the nearest two known pixels $Q(i, z_0)$ and $Q(t_0, j)$ in the vertical and horizontal direction, where

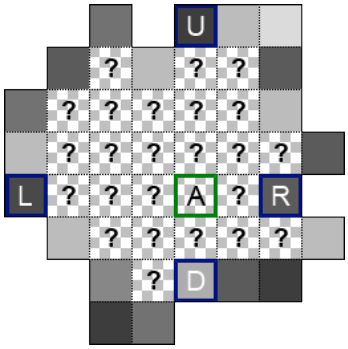


Figure 16: General irregular grid of the known and the unknown pixels.

Figure 17: The damaged image of the Lena.

$$z_0 = \arg\min_z(|A(i,j) - Q(i,z)|),$$
$$t_0 = \arg\min_t(|A(i,j) - Q(t,j)|).$$

Then, the nearest known pixel $Q(i*, j*)$ is chosen as follows

$$Q(i*, j*) = \begin{cases} Q(i, z_0) & \text{if } |j - z_0| \leq |i - t_0|, \\ Q(t_0, j) & \text{otherwise.} \end{cases}$$

The result of the Fig. 17 reconstruction is in Fig. 18. A different implementation of the nearest neighbour leads to a Voronoi diagram. Let us randomly distribute the known points (seeds) as shown in Fig. 19. The Voronoi diagram splits the area for the biggest possible cells as shown in Fig. 20. Every cell is defined by a seed specified beforehand and a region consisting of all points closer to that seed than to any other.

Bilinear interpolation In an irregular grid, we compute the linear interpolation for all unknown rows and columns. That is, for every unknown pixel, two values are available, one in each direction. The sum of these two intensities divided by 2 is used as the new intensity value of the unknown pixel. For our example in Fig. 16, we compute the linear interpolation for the row with pixel A as follows

Figure 18: Damaged Lena from Fig. 17 after nearest neighbour interpolation of the damaged parts.

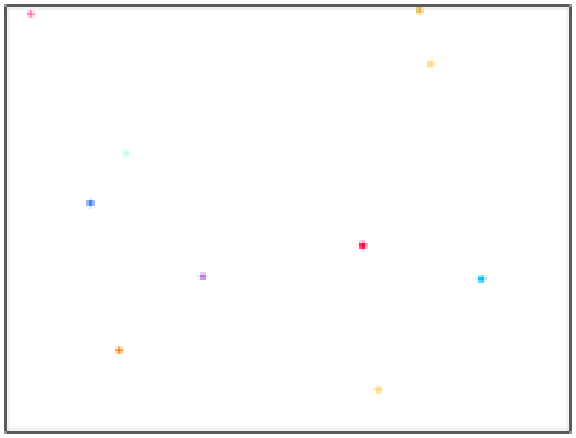
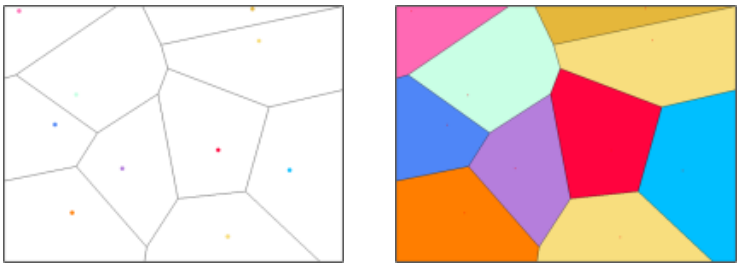


Figure 19: Randomly distributed points.



(a) Auxiliary lines (b) Voronoi diagram

Figure 20: Voronoi diagram of the randomly distributed points.

$$v_r(i,j) = v_r(i-1,j) + \frac{R_i - L_i}{u(R) - u(L)}; \quad v_r(L_i, L_j) = u(L),$$

and for the column as

$$v_c(i,j) = v_c(i,j-1) + \frac{D_j - U_j}{u(D) - u(U)}; \quad v_c(U_i, U_j) = u(U),$$

where the subscript i or j stands for the x or y coordinate of the point.

The results of the these two interpolation directions are shown in Fig. 21 for the sample image Fig. 17. The whole reconstructed image of Lena is in Fig. 22.

(a) Linear interpolation of the columns

(b) Linear interpolation of the rows

Figure 21: Two parts of the bilinear interpolation applied on Fig. 17.

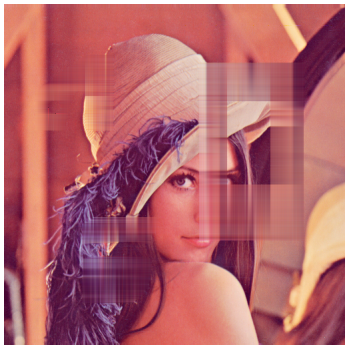


Figure 22: Damaged Lena from Fig. 17 after bilinear interpolation of the damaged parts.

Radial basis function interpolation This technique can be used in a regular and/or in an irregular grid without changes. Usage in an irregular grid is more common, therefore the technique is described in this subsection.

The radial basis functions (RBF) are a useful tool for image reconstruction. We have to choose a circular function ϕ which is symmetric around the center (i_d, j_d). There are two different RBF methods *basic* and *extended*.

The basic RBF computation is defined as

$$u(i,j) = \sum_{d=1}^{n} \lambda_d \phi(r), \tag{1}$$

$$A\overline{\lambda} = \overline{u}, \tag{2}$$

where r is $\sqrt{(i-i_d)^2+(j-j_d)^2}$ and (2) is a linear form of (1). We consider (1) at undamaged points (i,j) where points (i_d,j_d) are taken from the undamaged area as well. We can say that from the geometrical point of view we identify centers of the RBF with an undamaged pixel in order to use all of them to compute $\overline{\lambda}$.

Thus, (1) leads to a system of linear equations (2) where $\overline{\lambda}$ is the vector of unknowns $\lambda_1,\ldots,\lambda_n$, $\overline{u}$ is the vector at left-hand sides.

The extended RBF computation is as follows

$$\left|\begin{array}{cc} A & F \\ F^T & 0 \end{array}\right| \left|\begin{array}{c} \overline{\lambda} \\ \overline{\gamma} \end{array}\right| = \left|\begin{array}{c} \overline{u} \\ \overline{0} \end{array}\right|, \tag{3}$$

where $A,\overline{\lambda},\overline{u}$ are taken from (2), $\overline{\gamma}$ is the vector of unknown coefficients of a polynomial function [48], e.g., $\gamma_0+\gamma_1 x+\gamma_2 y$ and F is the matrix of the x,y coefficients from that polynomial function. The solution of (3) contains the solution $\overline{\lambda}$ of (2). We extend the applicability of equation (1) to the damaged area and consider a new function

$$\hat{u}(i,j)=\sum_{d=1}^{n}\lambda_d\phi(r),$$

where the point (i,j) belongs to the whole area (damaged and undamaged) and λ_i are the components of the solution $\overline{\lambda}$. To conclude, $\hat{u}$ is the reconstructed image with the help of the radial basis function ϕ.

An example in Fig. 17 is reconstructed in Fig. 23. The commonly used RBF are shown in Tab. 3 and illustrated in Fig. 24.

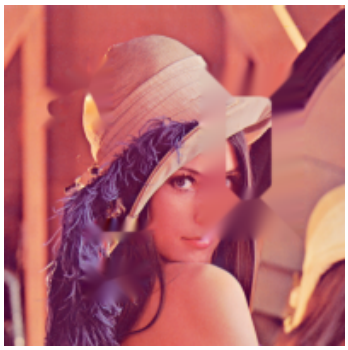


Figure 23: Image of the Lena Fig. 17 reconstructed by RBF interpolation.

We demonstrate the RBF method on a set of the input points. The plain coordinates are as follows

$$[0,5];[2,9];[4,3];[7,5];[8,1];[10,8].$$

The interpolation with usage of various RBF is shown in Fig. 25. The technique of the RBF has been investigated by the author and will be visually compared with the F-transform on the same set of the input points in section 7. More information on RBF and RBF based image reconstruction can be found in [38, 47, 3, 13]. Author's contribution is in [44].

Type	Formula
gauss	$e^{-(\epsilon r)^2}$
linear	r
quadratic	r^2
cubic	r^3
log	$r^2 \log r$
mq	$\sqrt{1+(\epsilon r)^2}$

Table 3: Commonly used RBF where usually $\epsilon = 0.25$.

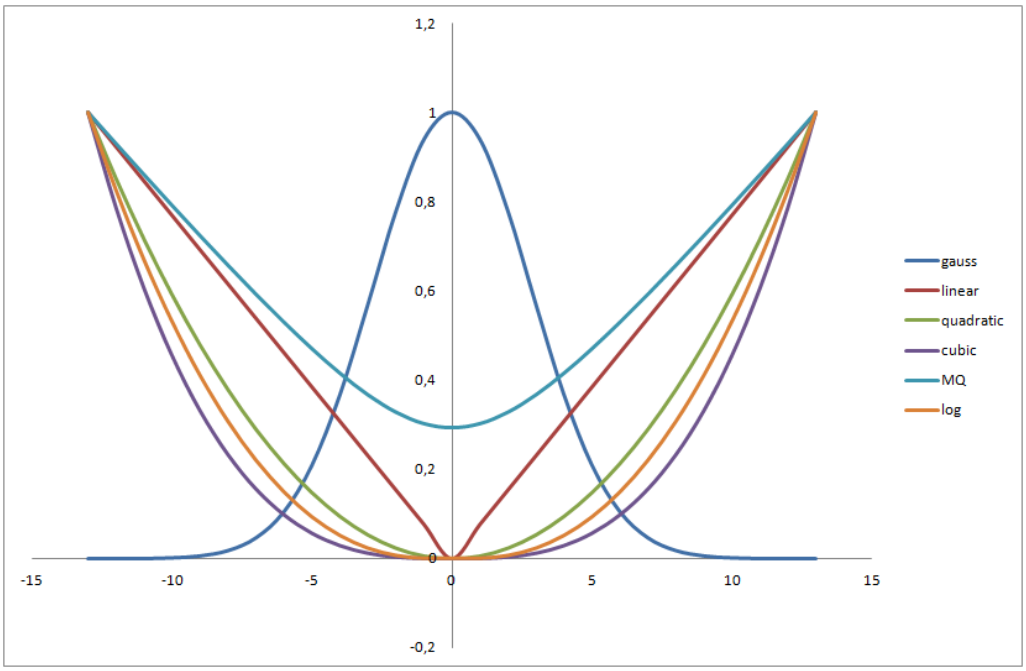


Figure 24: Demonstration of the various RBF.

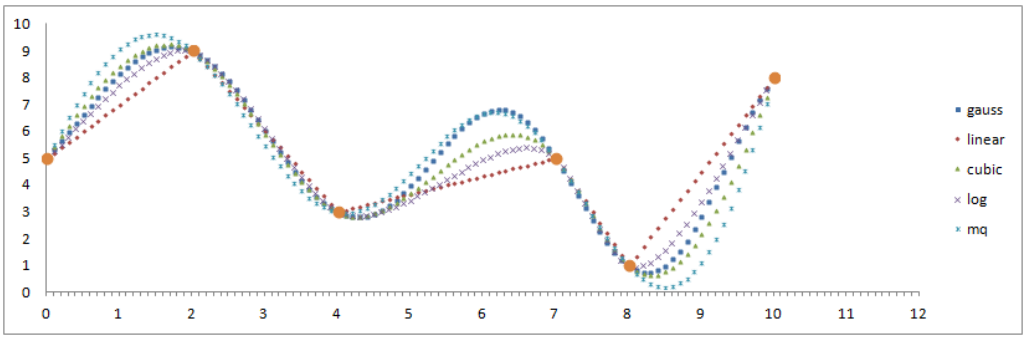


Figure 25: Demostration of 1D interpolation of various RBF.

3.2 Inpainting

Inpainting is a technique of modifying images into an undetectable form. The goals are restoration of the damaged image but also object removing from the image. The inpainting techniques are more sophisticated than the common interpolation. These

techniques attempt to replicate the techniques from the professional restorators or to use complex models. For example, models inspired in physics.

The modification of images in an undetectable form is documented in Renaissance. Bertalmio et al. [5] describes inpainting work by a professional artist as follows:

1. the global picture determines how to fill in the gap, the purpose of inpainting being to restore the unity of the work;

2. the structure of the area surrounding Ω is continued into the gap, contour lines are drawn via the prolongation of those arriving at $\tilde{\Omega}$;

3. the different regions inside Ω, as defined by the contour lines, are filled with color, matching those of $\tilde{\Omega}$;

4. the small details are painted (e.g. little white spots on an otherwise uniformly blue sky): in other words, "texture" is added.

The digital image inpainting can be described as follows. Let $u_0(i,j) : [1,M] \times [1,N] \to \mathbb{R}$ with $[1,M] \times [1,N] \subset \mathbb{N} \times \mathbb{N}$ be a discrete image. The digital inpainting process iteratively restores damaged areas by the following images $u^z(i,j) : [1,M] \times [1,N] \times \mathbb{N} \to \mathbb{R}$ where $u^0(i,j) = u_0(i,j)$ and $\lim_{z\to\infty} u^z(i,j) = u^r(i,j)$. The general algorithm can be described as follows

$$u^{z+1}(i,j) = u^z(i,j) + \Delta t u_t^z(i,j), \forall (i,j) \in \Omega,$$

where z stands for the iteration step, (i,j) are the pixel coordinates, Δt is the rate of improvement and $u_t^z(i,j)$ stands for the update of the image $u^z(i,j)$. The $u_t^z(i,j)$ step is computed by $\overline{L^z}(i,j)$, which is color/intensity propagated from the $\tilde{\Omega}$ to the $\delta\Omega$ in the direction $\overline{N^z}(i,j)$ as is shown in Fig. 26.

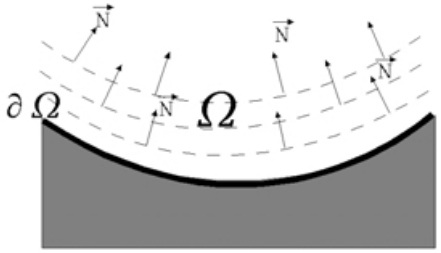


Figure 26: Filling of the Ω (figure taken from [5]).

This means that we must have

$$u_t^n(i,j) = \overline{\delta L^z}(i,j) \cdot \overline{N^z}(i,j),$$

where $\overline{\delta L^z}(i,j)$ is the measure of the change in $L^z(i,j)$. The propagation of the color/intensity must be smooth. We can use discrete Laplacian as follows

$$L^z(i,j) = \frac{\partial^2 u^z(i,j)}{\partial x^2} + \frac{\partial^2 u^z(i,j)}{\partial y^2}.$$

Then we compute the change $\overline{\delta L^z}(i,j)$ along $\overline{N}$. For that we must define $\overline{N}$ direction. One possibility is to define $\overline{N}$ as the normal vector from $\tilde{\Omega}$. This method does not consider previous isophote direction as shown in Fig. 27.

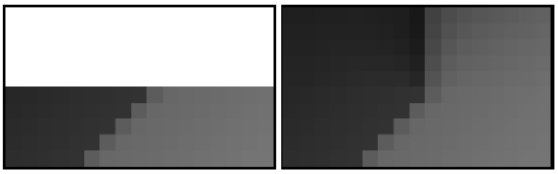


Figure 27: The normal vector application (figure taken from [5]).

It seems that a better solution is to preserve directions of the isophotes going through $\tilde{\Omega}$ do Ω. The gradient vector $\bigtriangledown u^z(i,j)$ gives the direction of the biggest spatial change. Therefore, the smallest spatial change $(\bigtriangledown u^z(i,j))^{\perp}$ is orthogonal to $\bigtriangledown u^z(i,j)$. The vector $(\bigtriangledown u^z(i,j))^{\perp}$ gives the isophote directions. We take into consideration that $\overline{N}$ varies from iteration to iteration continuously.

$$\overline{N^z}(i,j) = (\bigtriangledown u^z(i,j))^{\perp}.$$

The digital image inpainting can be roughly divided into two groups: *patch-based and sparse representation* and *partial differential equations (PDEs)/variational* methods. This division is not exclusive as some methods use principles of both of them. The following text shows some methods based on the main idea of the mentioned groups.

3.2.1 Patch-based and sparse representation methods

Ω is filled-in recursively where every pixel $S \in \delta\Omega$ is replaced by pixel $O \in \Phi$. We must define square patch $\Psi \in \Phi$. The patch $\Psi(O)$ is centered in the pixel O so that $\Psi(O)$ is the neighborhood of the O. We must choose $\Psi(S)$ which is the most similar to $\Psi(O)$ according to

$$S = \arg\min d(\Psi(S), \Psi(O)),$$

where $d(\Psi(S), \Psi(O))$ is the sum of square differences between $\Psi(S)$ and $\Psi(O)$ defined as follows

$$d(\Psi_1, \Psi_2) = \sum_{i=1}^{M}\sum_{j=1}^{N} |\Psi_1(i,j) - \Psi_2(i,j)|^2.$$

This method is used in Efros and Leung [10]. The result of their technique is shown in Fig. 28.

An improvement was given by Criminisi [6]. Firstly, the pixel-by-pixel way of processing with respect to $\delta\Omega$ was replaced by a more sophisticated version. The pixels have priority where these on the edges are processed before these on the flat regions. The second improvement is copying the whole patches instead of single pixels. A demonstration is in Fig. 29

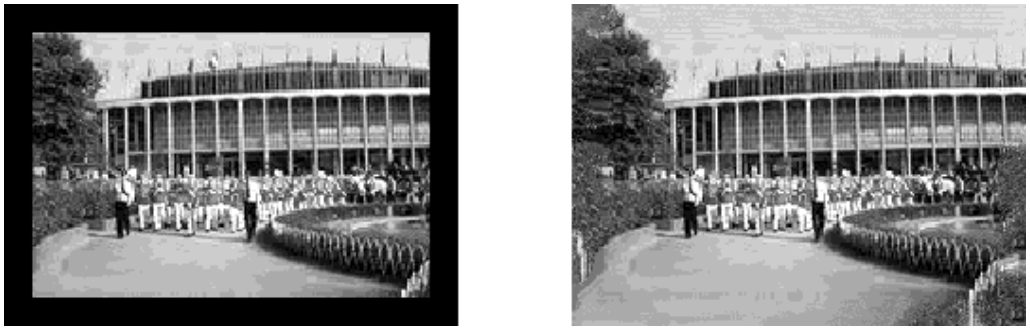


Figure 28: Algorithm applied on the real image, where the black border is replaced by the patches from the image itself (figure taken from [10]).

Figure 29: Removing large object from photography by Criminisi (figure taken from [6]).

Elad et al. [11] extends patch-based idea for the sparse coefficients for geometry and texture components of the image. Let u be the input image represented as a vector $\mathbb{R}^F; F = M \cdot N$ where M stands for the width and N stands for the height. The image is separated to two parts *geometry* and *texture* represented by matrices D_g, D_t, where inpainting is done separately in each part. The matrices have sizes $F \times k_g$ and $F \times k_t$. If $\alpha_g \in \mathbb{R}^{k_g}$ and $\alpha_t \in \mathbb{R}^{k_t}$ are the geometry and texture coefficients, then the image decomposition using the D_g and D_t dictionaries is as follows

$$u = D_g\alpha_g + D_t\alpha_t.$$

The decomposition of the image into geometry and texture representation is in Fig 30. The sparse image representation is defined as follows

$$\min_{(\alpha_g,\alpha_t):u=D_g\alpha_g+D_t\alpha_t} \|\alpha_g\|_p + \|\alpha_t\|_p,$$

where p is the coefficient of the ℓ-norm $\|\alpha\|_p = (\sum \|\alpha(q)\|^p)^{1/p}$. Elad et al. propose model

$$\min_{(\alpha_g,\alpha_t)} \|\alpha_g\|_1 + \|\alpha_t\|_1 + \lambda\|u - D_g\alpha_g - D_t\alpha_t\|_2^2 + \gamma TV(D_g\alpha_g),$$

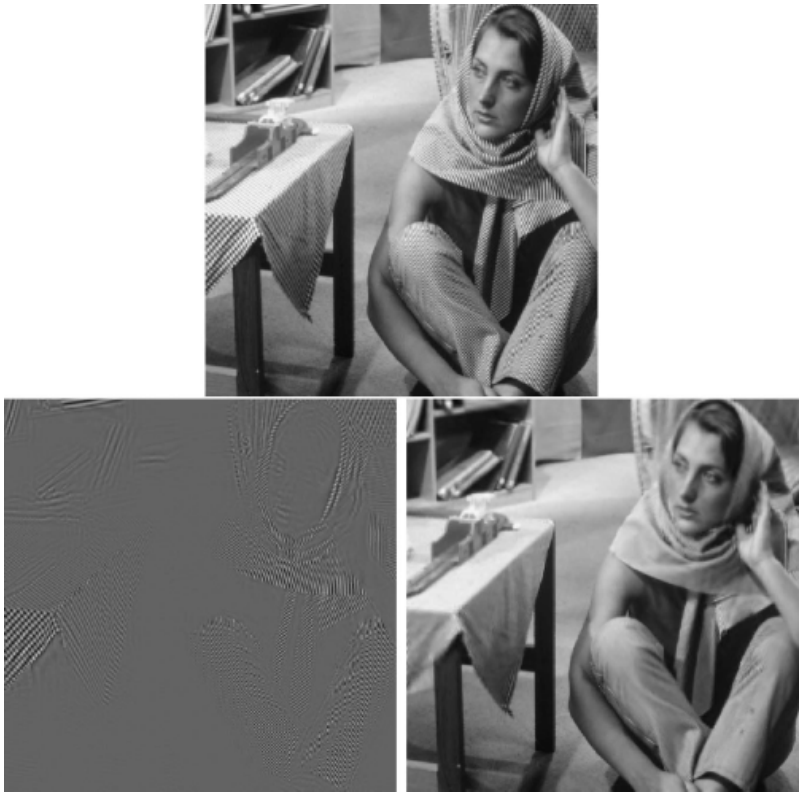


Figure 30: Separated texture (bottom left) and geometry (bottom right) from photography by Elad et al. (figure taken from [11]

where TV stands for the total variation, $p = 1$ and $\lambda, \gamma > 0$. Adaptation for image inpainting is following

$$\min_{(\alpha_g, \alpha_t)} \|\alpha_g\|_1 + \|\alpha_t\|_1 + \lambda \|C(u - D_g\alpha_g - D_t\alpha_t)\|_2^2 + \gamma TV(D_g\alpha_g),$$

where C stands for m_Ω. The undamaged pixels are marked $C = 1$ and damaged ones as $C = 0$. There are various assumptions to achieve this minimization problem for image inpainting. Details are described in [11], a brief survey follows:

- an image can be modeled as sparse combination of atom images. These images can be described by sparse composition of the two dictionaries one for the texture and second for the geometry;
- sparsity can be handled by ℓ^1.

3.2.2 PDEs and variational methods

The principles mentioned before take existing parts of the image for filling in the damaged area Ω. From a different point of view, we can determine pixels in the damaged area by computation. The new values of pixels in Ω can be determined by values of the pixels in the $\tilde{\Omega}$. Ogden et al. [20] proposed pyramid-based graphics techniques. The iterating convolution and subsampling are used for Gaussian filtering building. Successive linear interpolation, downsampling and upsamling at

different levels of the Gaussian pyramids are used for filling in the damaged area Ω. The result of this approach is in Fig. 31.

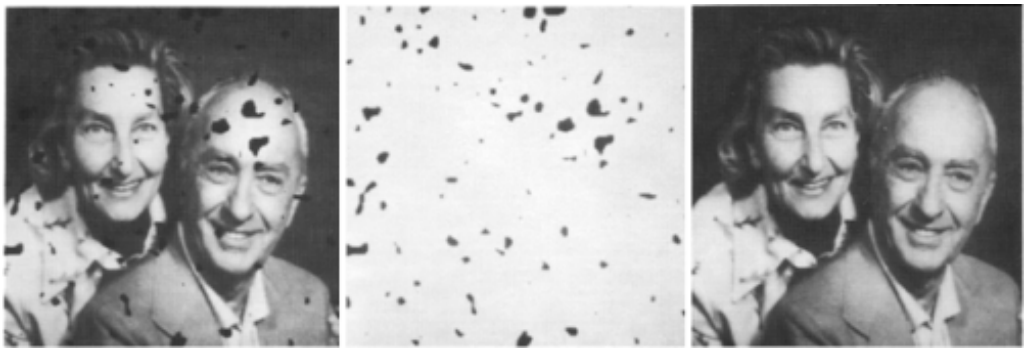


Figure 31: Inpainting by Ogden et al. (figure taken from [20]).

Another approach to inpainting was suggested by Masnou and Morel [19, 18]. It uses the ability of the human visual system to complete partially hidden edges. The ability of this restoring was studied particularly by Kanizsa [14]. He suggests that this continuation is performed between T-junctions. These T-junctions are points where edges forms the "T" as it is shown in Fig. 32.

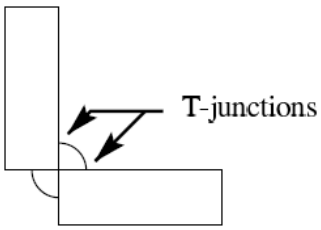


Figure 32: T-junctions (figure taken from [18]).

It seems that restored edges must be as straight and smooth as possible. Fig. 33 shows a situation where incomplete circles are restored by our perception (middle) and where rectangles are restored (right) from the same input image (left).

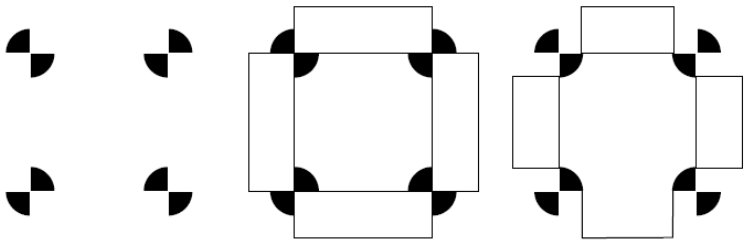


Figure 33: Difference between partially hidden edges perception (figure taken from [18]). Incomplete shapes (left) restored as circles (middle) or rectangles (right).

From the digital image processing point of view, we have to propagate isophotes from Φ to Ω. The principle is to connect two T-junctions by a curve with the minimal

length and oscillation. Approximation of the Euler's elastica is usually used for that as follows

$$\int (1 + \kappa^2) ds,$$

where s stands for the arc length and κ denotes the curvature. Unfortunately, this is not appropriate in cases where the intensity of the edges oscillates. Masnou and Morel suggest to use domain $\tilde{\Omega}$ slightly larger than Ω. The positive or negative orientation is associated for all level lines going through $\tilde{\Omega}$ based on the $\bigtriangledown u$ along the line. For all intensity levels q in $\tilde{\Omega}$, the curve is defined as

$$(L_d^q)_{d \in u(q)},$$

The solution lies in finding an optional set of curves

$$(\Gamma_l^q)_{l \in J(q)},$$

where $J(q) = u(q)/2$, connecting the T-junctions with same orientation, level, and minimizing the energy

$$E = \int_{-\infty}^{\infty} \sum l \in J(q) (\int_{\Gamma_l^q} (\alpha + \beta |\kappa|^p) ds + \varphi),$$

where α, β are positive context dependent constants, φ denotes the sum of the Γ_l^q and directions of the Γ_l^q and associated lines. Parameter p is the generalization of the κ exponent in the Euler's equation. The result of the [19] technique is in Fig. 34.

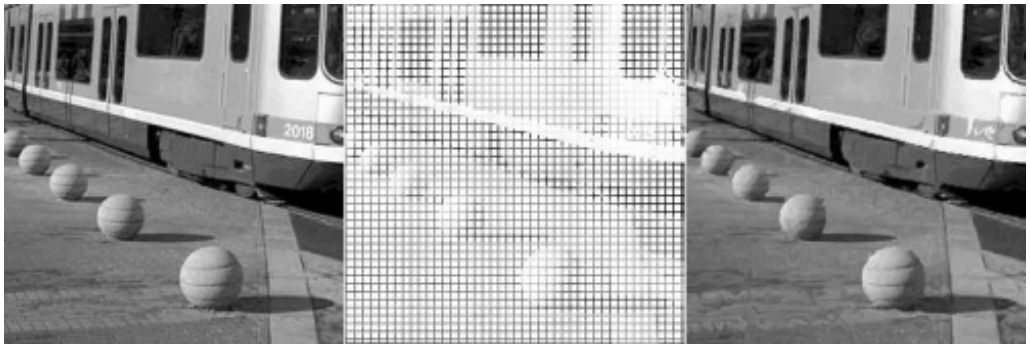


Figure 34: Masnou image inpainting of the highly damaged image (figure taken from [18]).

3.2.3 Summary

Most PDE/computational techniques are unable to restore texture properly. In Fig. 35, a demonstration is shown. On the other hand, patch-based methods are not able to restore images where good enough patches are not found. A demonstration is provided in Fig. 36. There are methods combining both of these ideas. We can mention [16, 9].

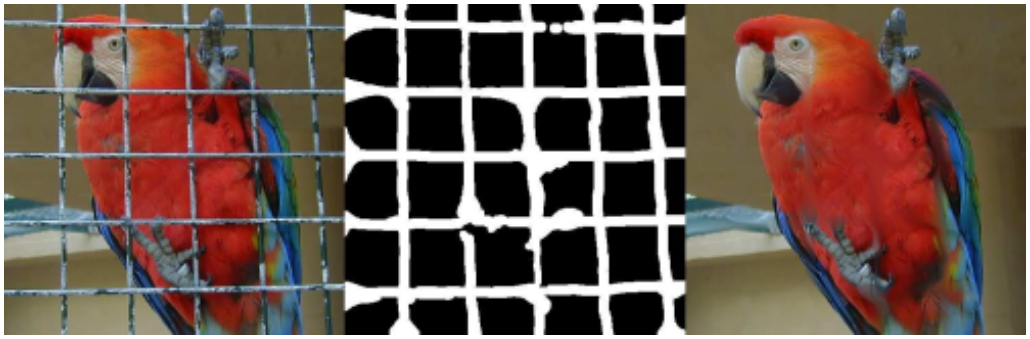


Figure 35: Image inpainting by Tschumperl used for erasing of the object from the image é (figure taken from [37]).

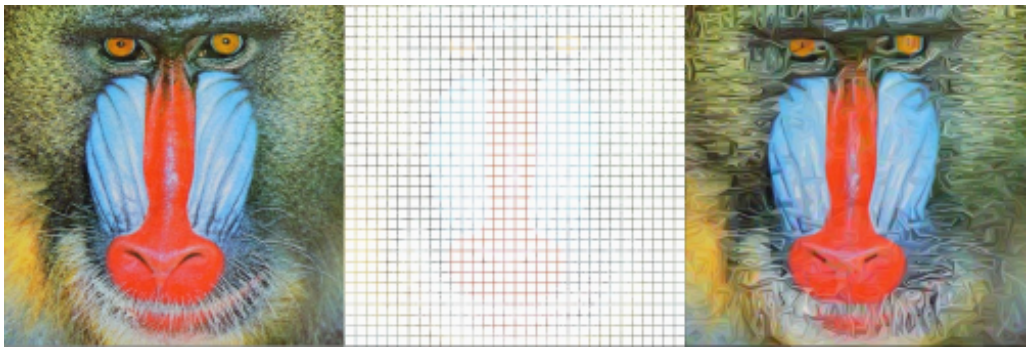


Figure 36: Masnou image inpainting in case where there are not enough known points to create proper patches. Figure taken from Survey (`http://math.univ-lyon1.fr/~masnou/fichiers/publications/survey.pdf`).

The next part of this section describes two particular methods that will be used later for comparison with the F-transform technique. The first particular inpainting technique is based on fast marching method. The second particular technique is based on Navier-Stokes equations and it simulates techniques that are used by professional restorators.

An image inpainting technique based on the fast marching method The method [35] was published in 2004. The algorithm uses small neighborhood $B_\epsilon(S)$ where ϵ stands for the radius of the neighborhood where only undamaged pixels are taken into consideration. We consider the first order approximation $u_O(S)$ as follows

$$u_O(S) = u(O) + \bigtriangledown u(O)(S - O),$$

where $\bigtriangledown u(O)$ stands for the gradient of the undamaged pixel O. Then we compute

$$u(S) = \frac{\sum_{O \in B_\epsilon(S)} w(S,O)[u(O) + \bigtriangledown u(O)(S-O)]}{\sum_{O \in B_\epsilon(S)} w(S,O)},$$

where the weighting function w is as follows

$$w(S,O) = \mathrm{dir}(S,O) \cdot \mathrm{dst}(S,O) \cdot \mathrm{lev}(S,O),$$

Where "dir" stands for the directional component, "dst" stands for the geometric distance component and "lev" stands for the level distance component. For the details see paper [35].

The pixel S that is closest to the undamaged pixels O must be filled in first. It requires a method for propagating pixels $S \in \delta\Omega$ to Ω according to their order of their distance from the initial boundary $\delta\Omega_o$. For this purpose Telea uses the fast marching method (FMM). The FFM solves Eikonal equation

$$|\bigtriangledown T| = 1 \text{ on } \Omega, \quad \text{with } T = 0 \text{ on } \delta\Omega.$$

The result of Fig. 17 inpainting is shown in Fig. 37.

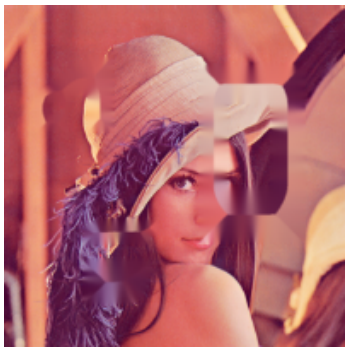


Figure 37: Image of the Lena Fig. 17 reconstructed by inpainting based on [35]

Navier-Stokes, fluid dynamics, and image and video inpainting The algorithm [4] was published in 2001 and it is a derivation of [5]. The method propagates isophotes from the Φ to $\delta\Omega$ with usage of the ideas from fluid dynamics. Image intensity is taken as a stream function for a 2D incompressible flow. The relation between image inpainting and fluid dynamics is described in Tab. 4.

Navier-Stokes	Image inpainting
stream function Ψ	image intensity u
fluid velocity $v = (\bigtriangledown\Psi)^{\perp}$	isophote direction $v = (\bigtriangledown u)^{\perp}$
vorticity $\omega = \Delta\Psi$	smoothness $\omega = \Delta u$
fluid viscosity ν	anisotropic diffusion ν

Table 4: Relation between Navier-Stokes equation and image inpainting (taken from [4]).

The authors suggest to solve vorticity transport equation for ω

$$\frac{\partial\omega}{\partial t} + v \cdot \bigtriangledown\omega = \nu \bigtriangledown \cdot(g(|\bigtriangledown\omega|) \bigtriangledown\omega),$$

where g allows anisotropic diffusion. The image u is recovered by

$$v = (\bigtriangledown u)^{\perp},$$
$$\Delta u = \omega, \quad u|_{\tilde{\Omega}} = \Phi.$$

The inpainting result of the example in Fig. 17 is shown in Fig. 38.

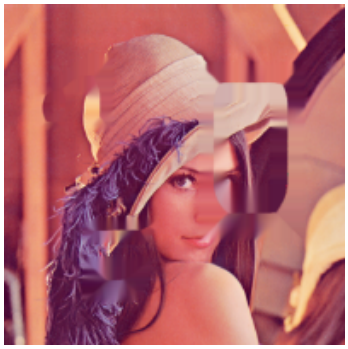


Figure 38: Fig. 17 reconstructed by inpainting based on [4]

4 Motivation and objectives of the dissertation

In this section *problem, motivation* and *objectives* of the dissertation are formulated.

4.1 State of the art

All known reconstruction techniques are based on extraction knowledge from the undamaged parts of an image with its subsequent interpolation or extrapolation to the damaged parts (with the purpose to replace them). Conventional techniques are focused on characteristics that can be extracted from an image, i.e. intensities of pixels, edges or flat areas, etc. All of these techniques use a classical (binary) representation of local areas (windows) where image characteristics are observed.

4.2 Formulation of the problem and motivation

The problem of image reconstruction consists (a) in assuming that a damaged image is given together with the information that allows to separate damaged and undamaged pixels, and (b) in replacing the former by the latter.

Let us give the technical details below. In the problem of reconstruction, it is assumed that the domain Φ of image u is a proper subset of available set of pixels P, i.e. $\Phi \subset P$, and that u is not defined (damaged) on the relative complement $\Omega = P \setminus \Phi$. The goal is to extend u to the set P, i.e. to propose a method that computes (reconstructs) values $u(i,j)$ for all $(i,j) \in \Omega$. In details, we want to obtain a new image, say $u^r : P \to \{0, 1, \ldots, 255\}$ such that $u^r|_P = u$ where $u^r|_P$ is the restriction of u^r on P.

In mathematical literature, the problem of reconstruction is known as *interpolation* or *extrapolation*, depending on whether $(i,j) \in \Omega$ is an "internal" point of P or not. In computer science literature, the problem of reconstruction is used to be solved with the help of interpolation methods. For this purpose, a class of interpolating functions is chosen a priori, e.g. bilinear, bicubic, RBF, etc. If u^r is an interpolating function, then it automatically fulfills the restriction $u^r|_P = u$.

In this contribution, we propose a new approach, that is based on utilizing approximating functions. Generally speaking, we propose to construct an extension $\hat{u} : P \to \{0, 1, \ldots, 255\}$ such that the restriction $\hat{u}|_P$ approximates u. Then the reconstructed image u^r is a combination of two functions $\hat{u}$ and u so that

$$u^r(i,j) = \begin{cases} \hat{u}(i,j) & \text{if } (i,j) \in \Omega, \\ u(i,j) & \text{otherwise.} \end{cases}$$

Our motivation can be explained simply by noticing that a class of approximating functions includes that of interpolating functions. Therefore, working with approximating functions we are less restricted and thus, have a wider choice of reconstructing functions. There is the technique of fuzzy (F)-transforms that takes local areas as areas with some additional structure. This structure is characterized by fuzzy predicates that may express any information which is relevant for a problem. In image processing, this can be, for example, a distance from a certain point, a

relationship between points, color/intensity, texture, etc. The F-transform approach creates the F-transform image of an object, to which a structure of a universe of discourse is propagated. This technique proves to be effective in the problems like image fusion, compression/reduction, edge detection etc.

In image reconstruction, we expect to utilize the following properties of the F-transform:

- a compact representation of an object and its characteristics (derivatives or partial derivatives) in a form of a sequence (matrix) of components;
- a possibility of easy processing where the F-transform representation is used instead of a (complex) object;
- an approximate reconstruction of an object from its F-transform components.

The main advantage is fast running algorithms with similar or better results than conventional ones.

4.3 Objectives

The objectives of the dissertation are as follows.

To elaborate the technique of image reconstruction on the basis of the F-transform Investigation of the F-transform technique leads to several applications in image processing. In the dissertation, we are focused on the extension of the usability to the image reconstruction field.

To develop software tools for the F-transform based reconstruction so that they are fast and easy to implement Tools must be developed for successful demonstration of the possibilities and related research. Implementation is not complicated. the algorithm is simple and easy to understand.

To analyze the influence of the F-transform parameters and to choose their optimal values for the problem of reconstruction There are many possibilities for application of the F-transform and for the various parameter settings. Optimal settings are described in the dissertation and demonstrated on examples.

To analyze a possibility of the F-transform in solving problems that are closely related to reconstruction: upsamling, denoising or filtering, inpainting Image inpainting is only one possibility for the F-transform method usage. We also investigate others, such as upsampling, denoising or filtering. The extended application is described later in the dissertation.

To compare the proposed approach with conventional ones and to analyze where it is advantageous A comparison with commonly used techniques is provided from the quality point of view. Commonly used techniques of both inpainting and interpolation technique are used for comparison based on RMSE and SSIM.

5 F-transform

Assume that we are given a partially damaged image where the damaged part is separated from the undamaged one. Our purpose is to restore this image. By this we mean that damaged pixels should be replaced by new ones, whose colors/intensities (values) are computed from the values of the undamaged pixels. To solve this problem we propose the F-transform technique [31, 41]. The technique takes the damaged image u and the mask m_Ω of the damaged part Ω as an input. The output is the reconstruction image u^r.

The F-transform is a technique putting a continuous/discrete function into a correspondence with a finite vector of its F-transform components. In image processing, where images are identified with intensity functions of two arguments, the F-transform of the latter is given by a matrix of components. We recall the definition of the F-transform [23] and give it for a function of two variables defined on a set of pixels $P = \{(i, j) \mid i = 1, 2, \ldots, M; j = 1, 2, \ldots, N\}$, respectively $P = \Omega \cup \Phi$ where Ω stands for damaged area and Φ stands for undamaged area.

We propose two algorithms: *one-step* and *multi-step* [30]. They differ in the reconstruction process where the one-step reconstructs the whole image in one iteration and the multi-step uses more iterations. The damaged area Ω is filled-in at once in the one-step algorithm. In the multi-step algorithm, the boundaries $\delta\Omega$ are computed in the first iteration. These computed pixels are partially used for another iteration. It means that area $\delta\Omega$ iteratively changes. Details are described in the following subsections.

5.1 Fuzzy partition with Ruspini condition

A *fuzzy partition with the Ruspini condition* (simply, *Ruspini partition*) was introduced in [23]. The Ruspini condition implies normality of the respective fuzzy partition, i.e. the *partition-of-unity*. It then leads to a simplified version of the inverse F-transform. In later publications [33, 24], the Ruspini condition was weakened to obtain an additional degree of freedom and a better approximation by the inverse F-transform.

Let $x_1 < \ldots < x_m$ be fixed nodes within $[a, b]$ so that $x_1 = a$, $x_m = b$ and $m \geq 2$. We say that the fuzzy sets $A_1, \ldots, A_m$, identified with their membership functions defined on $[a, b]$, establish a Ruspini partition of $[a, b]$ if they fulfill the following conditions for $k = 1, \ldots, m$:

1. $A_k : [a, b] \to [0, 1]$, $A_k(x_k) = 1$;
2. $A_k(x) = 0$ if $x \notin (x_{k-1}, x_{k+1})$, where for uniformity of notation, we set $x_0 = a$ and $x_{m+1} = b$;
3. $A_k(x)$ is continuous;
4. $A_k(x)$, for $k = 2, \ldots, m$, strictly increases on $[x_{k-1}, x_k]$ and $A_k(x)$, for $k = 1, \ldots, m-1$, strictly decreases on $[x_k, x_{k+1}]$;

5. for all $x \in [a, b]$,

$$\sum_{k=1}^{m} A_k(x) = 1. \tag{4}$$

The condition 4 is known as the Ruspini condition. The membership functions $A_1, \ldots, A_m$ are called *basic functions*. A point $x \in [a, b]$ is *covered* by basic function A_k if $A_k(x) > 0$.

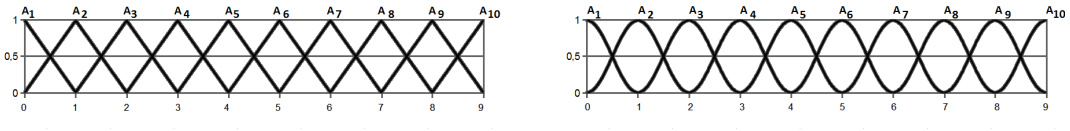
The shape of the basic functions is not predetermined and therefore it can be chosen according to additional requirements (e.g. smoothness). Let us give examples of various fuzzy partitions with the Ruspini condition. In Fig. 39, two such partitions with triangular and cosine basic functions are shown. The formulas given below represent generic fuzzy partitions with the Ruspini condition and triangular functions:

$$A_1(x) = \begin{cases} 1 - \frac{(x - x_1)}{h_1} & \text{if } x \in [x_1, x_2], \\ 0 & \text{otherwise,} \end{cases}$$

$$A_k(x) = \begin{cases} \frac{(x - x_{k-1})}{h_{k-1}} & \text{if } x \in [x_{k-1}, x_k], \\ 1 - \frac{(x - x_k)}{h_k} & \text{if } x \in [x_k, x_{k+1}], \\ 0 & \text{otherwise,} \end{cases}$$

$$A_m(x) = \begin{cases} \frac{(x - x_{m-1})}{h_{m-1}} & \text{if } x \in [x_{m-1}, x_m], \\ 0 & \text{otherwise.} \end{cases}$$

where $k = 2, \ldots m - 1$ and $h_k = x_{k+1} - x_k$.



(a) Triangular (b) Cosine

Figure 39: Two partitions following Ruspini condition.

We say that a Ruspini partition of $[a, b]$ is *h-uniform* if its nodes $x_1, \ldots, x_m$, where $m \geq 3$, are h-equidistant, i.e., $x_k = a + h(k - 1)$, for $k = 1, \ldots, m$, where $h = (b - a)/(m - 1)$, and two additional properties are met:

6. $A_k(x_k - x) = A_k(x_k + x)$, for all $x \in [0, h]$, $k = 2, \ldots, m - 1$;

7. $A_k(x) = A_{k-1}(x - h)$, for all $k = 2, \ldots, m - 1$ and $x \in [x_k, x_{k+1}]$, and $A_{k+1}(x) = A_k(x - h)$, for all $k = 2, \ldots, m - 1$ and $x \in [x_k, x_{k+1}]$.

An h-uniform fuzzy partition of $[a, b]$ can be determined by the so-called *generating function* $A_0 : [-1, 1] \to [0, 1]$, which is assumed to be *even*[4], continuous, have a

[4]The function $A_0 : [-1, 1] \to \mathbb{R}$ is even if for all $x \in [0, 1]$, $A_0(-x) = A_0(x)$.

bell shape and fulfill $A_0(0) = 1$. Basic functions A_k of an h-uniform fuzzy partition with generating function A_0 are shifted copies of A_0 in the sense that

$$A_1(x) = \begin{cases} A_0\left(\frac{x-x_1}{h}\right) & \text{if } x \in [x_1, x_2], \\ 0 & \text{otherwise,} \end{cases}$$

and for $k = 2, \ldots, m-1$,

$$A_k(x) = \begin{cases} A_0\left(\frac{x-x_k}{h}\right) & \text{if } x \in [x_{k-1}, x_{k+1}], \\ 0 & \text{otherwise,} \end{cases}$$
$$A_m(x) = \begin{cases} A_0\left(\frac{x-x_m}{h}\right) & \text{if } x \in [x_{m-1}, x_m], \\ 0 & \text{otherwise.} \end{cases}$$

As an example, we notice that the function $A_0(x) = 1 - |x|$ is a generating function for any h-uniform triangular partition. In the sequel, we will use h-uniform fuzzy partitions only and refer to h as to a *radius* of partition.

5.2 Discrete F-transform

In this subsection, we introduce the F-transform of an image u that is considered as a function $u : [1, M] \times [1, N] \to [0, 1]$ where M stands for image width and N stands for image height. It is assumed that the image is grayscale and that it is defined at points (pixels) that belong to the set P.

Let $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ be basic functions, $A_1, \ldots, A_m : [1, M] \to [0, 1]$ be fuzzy partition of $[1, M]$ and $B_1, \ldots, B_n : [1, N] \to [0, 1]$ be fuzzy partition of $[1, N]$. Assume that the set of pixels P is *sufficiently dense with respect to the chosen partitions*. This means that $(\forall k)(\exists i \in [1, M])\ A_k(i) > 0$, and $(\forall l)(\exists j \in [1, N])\ B_l(j) > 0$.

We say that the $m \times n$-matrix of real numbers $[U_{kl}]$ is called *the (discrete) F-transform* of u with respect to $\{A_1, \ldots, A_m\}$ and $\{B_1, \ldots, B_n\}$ if for all $k = 1, \ldots, m,\ l = 1, \ldots, n$

$$U_{kl} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{M} u(p_i, q_j) A_k(p_i) B_l(q_j)}{\sum_{j=1}^{N} \sum_{i=1}^{M} A_k(p_i) B_l(q_j)}. \tag{5}$$

The elements U_{kl} are called *components of the F-transform*. The *inverse F-transform* $\hat{u} : P \to [0, 1]$ of the function u with respect to $\{A_1, \ldots, A_m\}$ and $\{B_1, \ldots, B_n\}$ is defined as follows

$$\hat{u}(i, j) = \sum_{k=1}^{m} \sum_{l=1}^{n} U_{kl} A_k(i) B_l(j). \tag{6}$$

The function $\hat{u}$ approximates the original function u on the whole domain P with a given precision. Moreover, the following estimate was established in [22] for every

continuous function u on a domain P and its inverse F-transform $\hat{u}$ that is computed with respect to h-uniform fuzzy partitions $\{A_1, \ldots, A_m\}$ of $[1, M]$ and $\{B_1, \ldots, B_n\}$ of $[1, N]$:

$$\max_{t \in P} |\hat{u}(t) - u(t)| \leq C\omega(h, u), \tag{7}$$

where C is a constant, $t = (i, j)$ and $\omega(h, u)$ is the modulus of continuity of u on P.[5] Formula 7 shows that the smaller is the value of h, the better is the estimate of the difference between u and $\hat{u}$. Both these facts give a justification of the reconstruction methods described below.

1D Reconstruction The formula adjusted for 1D F-transform is as follows

$$U_k = \frac{\sum_{i=1}^{M} u(p_i) A_k(p_i)}{\sum_{i=1}^{M} A_k(p_i)}, \tag{8}$$

and for inverse F-transform as follows

$$\hat{u}(p_i) = \sum_{k=1}^{m} U_k A_k(p_i). \tag{9}$$

5.3 2D reconstruction - *one-step*

We remind that image $u : P \to [0, 255]$ is damaged (unknown) on subset $\Omega \subset P$ and undamaged (known) on the complement Φ of Ω, i.e. $\Phi = P - \Omega$. The algorithm has two input parameters: u (image) and m_Ω (mask of Ω). In the one-step algorithm, we select computation parameters at the input stage and then perform the reconstruction in one computation cycle. The following text is an informal description of an algorithm that takes u and m as inputs and computes the reconstruction u^r as a combination of the undamaged part of u with the inverse F-transform $\hat{u}$ of the partially known image u.

The main requirement of the one-step algorithm is that a uniform fuzzy partition $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ of P fulfills the following property:

P. for every damaged pixel $(i, j) \in \Omega$ there are basic functions A_k and B_l and there is pixel $(i', j') \in \Phi$ such that $A_k(i) > 0$, $A_k(i') > 0$, $B_l(j) > 0$, $B_l(j') > 0$.

This means that every damaged pixel is covered by a combination of basic functions so that this combination also covers at least one undamaged pixel. Property **P** assures that the inverse F-transform $\hat{u}$ of image u replaces the non-defined value of the intensity function u at every damaged pixel $(i, j) \in \Omega$ by the value of $\hat{u}(i, j)$ that is computed using values of u at undamaged pixels $(i', j') \in \Phi$. The (output) reconstructed image u^r is a combination of u and $\hat{u}$:

[5] Generally, $\omega(h, f) = \max_{|\delta| \leq h} \max_{x \in X} |f(x + \delta) - f(x)|$.

$$u^r(i,j) = \begin{cases} \hat{u}(i,j) & \text{if } (i,j) \in \Omega, \\ u(i,j) & \text{otherwise.} \end{cases}$$

In Fig. 40 we show the graphical illustration of the above described algorithm.

(a) The damaged image (b) The reconstructed image

Figure 40: The damage caused by *scratches* from Fig.2c or manual painting removed by one-step F-transform.

The result of the reconstruction for the example in Fig. 17 is in Fig. 41.

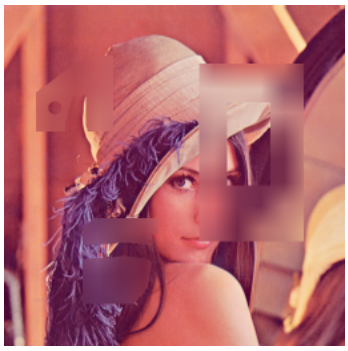


Figure 41: Lena from Fig. 17 after one-step F-transform approximation of the damaged parts.

5.4 2D reconstruction - *multi-step*

The one-step reconstruction can be successively applied to those images that are damaged on relatively small areas. In the above-given algorithm, we formalized this condition demanding that every damaged pixel should be covered by a combination of basic functions such that it covers at least one undamaged pixel (property **P**). However, it does not always happen that a fuzzy partition with this property exists. Therefore, we propose another algorithm which produces reconstruction as a result of combining an undamaged part of an original image with several inverse F-transforms, computed on a sequence of uniform fuzzy partitions with increasing radii. One step from this sequence is called iteration.

The main idea of the multi-step algorithm is as follows: in the first step we apply the F-transform with a fine partition (the smallest h) and reconstruct those damaged pixels that fulfill property **P**. Then we recompute the damaged area Ω by deleting the already reconstructed pixels and if Ω respectively m_Ω is not empty, we repeat the procedure with a bigger value of h. The multi-step reconstruction has better quality than the one-step reconstruction. This follows from the estimate (7) that shows that the smaller the value of a radius is, the closer the inverse F-transform $\hat{u}$ to u within one iteration. In Fig. 42, we show a grayscale form of image *Lena* from Fig. 1 damaged by the *text* from Fig. 2d and the multi-step reconstruction by inverse F-transforms, computed on a sequence of uniform fuzzy partitions with increasing radii $h = 2, 3, 4$ and triangular-shaped basic functions. The algorithm stops at $h = 4$, because there are no damaged pixels left.

(a) The damaged image (b) $h = 2$ (c) $h = 3$ (d) $h = 4$

Figure 42: The *text* damage from Fig.2d removed by the *multi-step* F-transform where different iterations are marked by h used in concrete one.

Reconstruction of Fig. 17 is in Fig. 43.

Figure 43: Lena from Fig. 17 after multi-step F-transform approximation of the damaged parts.

5.4.1 Error diffusion

The computed values of the pixels are used in next iterations of the multi-step algorithm for following computations. It means that *difference* (error) between original and reconstructed color/intensity value $u_o(i, j) - \hat{u}(i, j)$ is spread by an avalanche effect from the origin to the rest of the reconstructed area. This error diffusion depends on the type of damage.

Let us show an error diffusion on a 1D example shown in Fig. 44.

Figure 44: Example of 1D grayscale image.

Intensities of the used pixels are in the following vector

$$(100, 40, 30, 50, 80, 200).$$

We demonstrate the process of dithering. We have to change this vector to a binary image u_e with respect to threshold e as follows

$$u_e(i) = \begin{cases} 0 & \text{if } u_o(i) \leq e, \\ 255 & \text{if } u_o(i) > e. \end{cases}$$

The thresholded vector for $e = 49$ is as follows

$$(255, 0, 0, 255, 255, 255).$$

The outcome vector is illustrated in Fig. 45.

Figure 45: Thresholded 1D image.

The difference between the original image u_o and the thresholded image u_e is significant. We can make a comparison based on the sums of components

$$s_o = \sum_{i=1}^{m} u_o(i),$$
$$s_e = \sum_{i=1}^{m} u_e(i),$$

where s_o stands for the sum of the original image intensities and s_e stands for the sum of the thresholded image intensities. In our example

$$s_o = 500,$$
$$s_e = 1020.$$

Let us demonstrate an advanced method taking into consideration the error between the thresholded pixels $u'_e(i)$ and the original pixels $u_o(i)$. The difference for the first pixel is as follows

$$u_o(1) - u'_e(1) = -155.$$

This value is used for the second pixel as follows

$$u_o(2) = u_o(2) + (-155) = 40 - 155 = -115.$$

The intensity of $u_o(2)$ is 0 after thresholding. We also process the second pixel in the same way

$$\begin{aligned} u_o(2) - u'_e(2) &= -115 - 0 = -115, \\ u_o(3) &= u_o(3) + (-115) = 30 - 115 = -85. \end{aligned}$$

The result of this method is

$$(255, 0, 0, 0, 0, 255),$$

as can be seen in Fig. 46. The sum of the intensities is as follows

$$\begin{aligned} s_o &= 500, \\ s'_e &= 510, \end{aligned}$$

where approach of the s'_e to the s_o than the s_e to the s_o is visible.

Figure 46: Advanced method of the dithering.

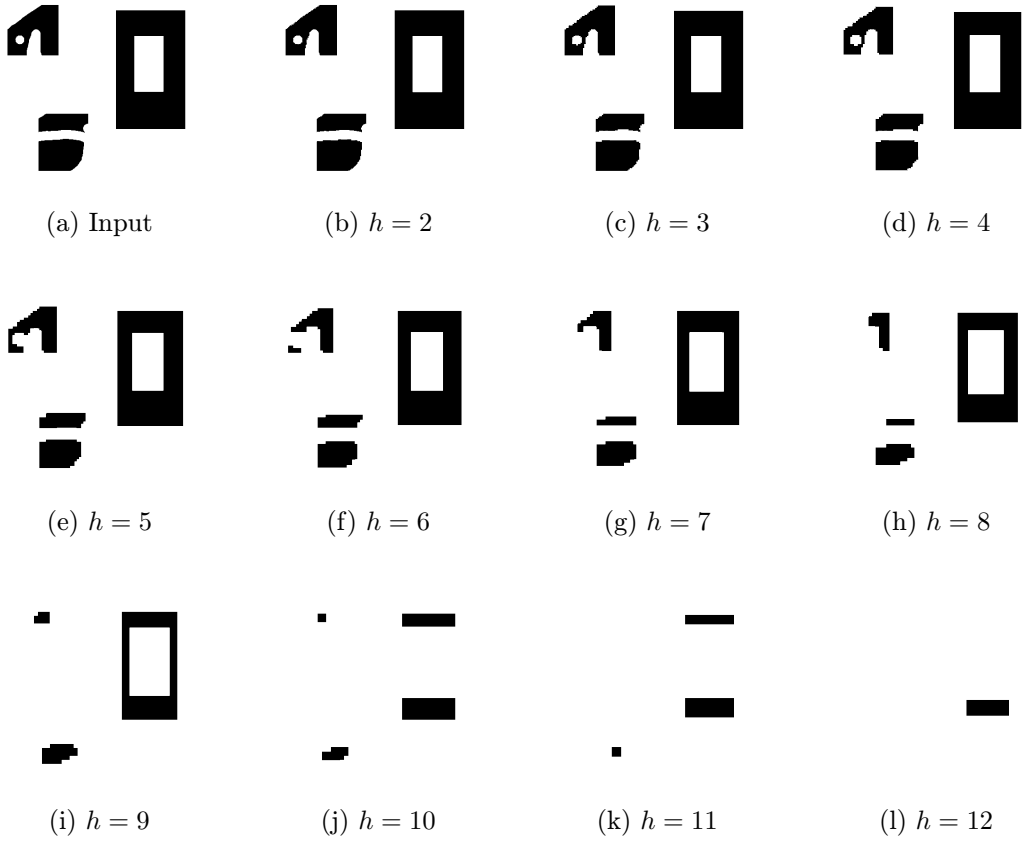
One of the methods using this error diffusion is Floyd-Steinberg [12]. A comparison is provided in Fig. 47.

(a) Thresholding (b) Floyd-Steinberg

Figure 47: The comparison between thresholding and Floyd-Steinberg processing.

Both images were dithered from 256 intensity levels to 2 intensity levels. Error diffusion, where pixel intensity computed in the one step, is taken into consideration for next iterations provides significantly better results. The error has to be distributed equally to all directions. If we distribute the error only in one direction,

from left to right for example, then we obtain the error accumulated to the last pixel, the rightest one. Error distribution in the F-transform is in Fig. 48, 49, 50, 51. Fig. 48 shows error diffusion with mask holes usage.



(a) Input (b) $h = 2$ (c) $h = 3$ (d) $h = 4$

(e) $h = 5$ (f) $h = 6$ (g) $h = 7$ (h) $h = 8$

(i) $h = 9$ (j) $h = 10$ (k) $h = 11$ (l) $h = 12$

Figure 48: Processing of the *holes* damage from Fig. 2b.

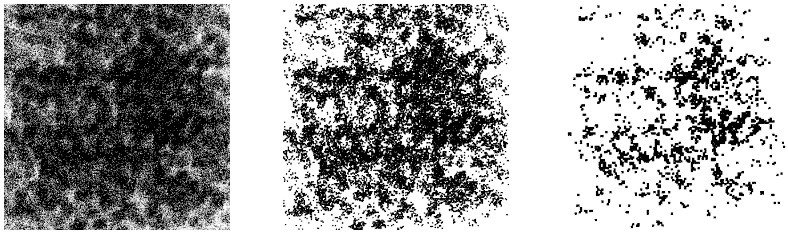
We can see that the error is not accumulated to one pixel or to one direction. This is a natural property of the F-transform.

5.5 Edge preserving

Edges of the images are not naturally preserved in the F-transform method, see Fig. 52. Edge preserving is the target of our ongoing research. The first results are achieved with support of the Sobel operator. The 5×5 kernel is used. For gradient in x direction is used kernel as follows

$$G_x = \begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 3 & 2 & 0 & -2 & -3 \\ 4 & 3 & 0 & -3 & -4 \\ 3 & 2 & 0 & -2 & -3 \\ 2 & 1 & 0 & -1 & -2 \end{bmatrix}; \quad G_y = \begin{bmatrix} -2 & -3 & -4 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 4 & 3 & 2 \end{bmatrix}.$$

As the first step, edges are detected in the input damaged image. The edge pixels surrounding the damaged area Ω are determined. The neighborhood of the

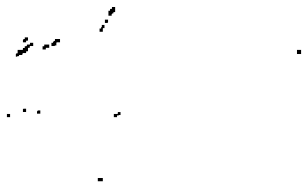


(a) Input (b) $h = 2$ (c) $h = 3$

(d) $h = 4$ (e) $h = 5$

Figure 49: Processing of the *noise* damage from Fig. 2a.

(a) Input (b) $h = 2$ (c) $h = 3$



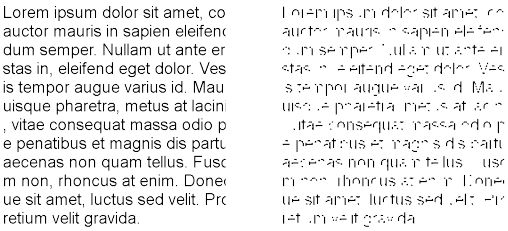
(d) $h = 4$ (e) $h = 5$

Figure 50: Processing of the *scratches* damage from Fig. 2c.

pixels is searched for another edge pixel E in the distance based on the used kernel size, see Fig. 53.

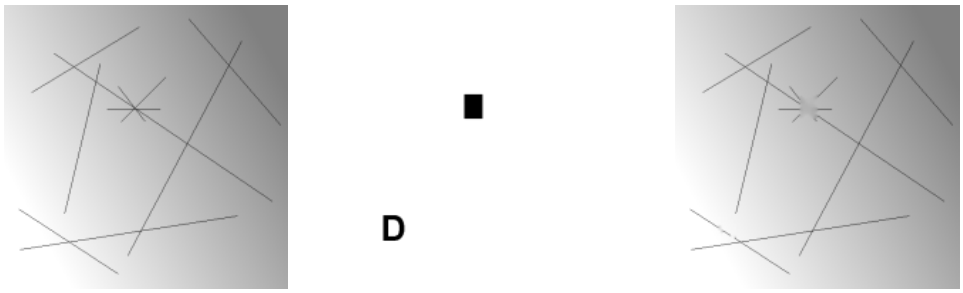
The Sobel kernel is centered to pixel E and used for the edge slope a determination. Let us demonstrate this on an example. The lowest edge from Fig. 53b is in detail in Fig. 54 with a marked area for application of the Sobel filter. The Sobel filter is centered to the purple pixel (i_p, j_p) from Fig. 53b.

Edges consist of white pixels with intensity 255 and other pixels are black with intensity 0. The kernels G_x, G_y have to be rotated by 180° to



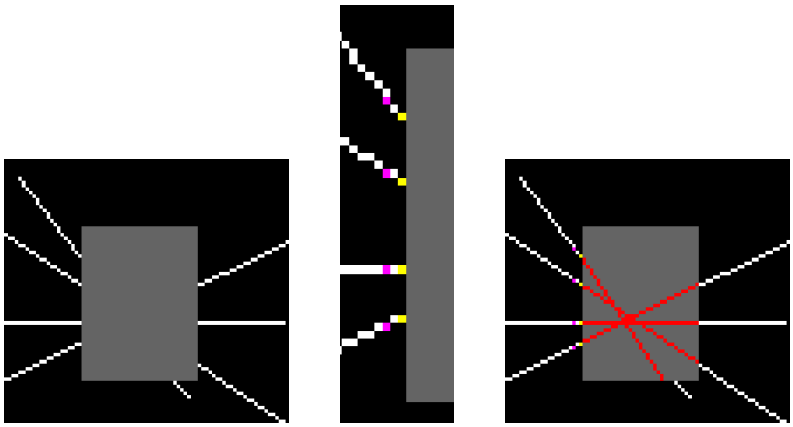
(a) Input (b) $h = 2$

Figure 51: Processing of the *text* damage from Fig. 2d.



(a) Input image (b) Mask (c) Reconstructed image

Figure 52: Demonstration of the reconstruction in the case of clearly visible edges.



(a) Missing edges (b) Marked pixels (c) Reconstructed edges

Figure 53: Demonstration of the edge preserving. Colors in the middle figure determine the following: yellow - pixel on the border of the Ω; purple - pixel in the distance based on the used kernel size.

$$G'_x = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -3 & -2 & 0 & 2 & 3 \\ -4 & -3 & 0 & 3 & 4 \\ -3 & -2 & 0 & 2 & 3 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}; \quad G'_y = \begin{bmatrix} 2 & 3 & 4 & 3 & 2 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & -3 & -2 & -1 \\ -2 & -3 & -4 & -3 & -2 \end{bmatrix}.$$

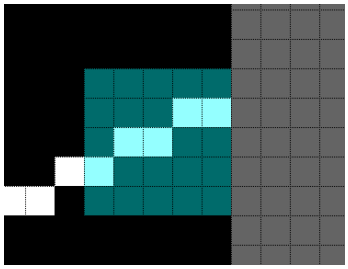


Figure 54: Detail of the processed image with applied area for Sobel mask.

It leads to following computation:

$$\begin{aligned}(u_s * G'_x)(i_p, j_p) &= (-2)\cdot 0 + (-1)\cdot 0 + 0\cdot 0 + 1\cdot 0 + 2\cdot 0 \\ &+ (-3)\cdot 0 + (-2)\cdot 0 + 0\cdot 0 + 2\cdot 255 + 3\cdot 255 \\ &+ (-4)\cdot 0 + (-3)\cdot 255 + 0\cdot 255 + 3\cdot 0 + 4\cdot 0 \\ &+ (-3)\cdot 255 + (-2)\cdot 0 + 0\cdot 0 + 2\cdot 0 + 3\cdot 0 \\ &+ (-2)\cdot 0 + (-1)\cdot 0 + 0\cdot 0 + 1\cdot 0 + 2\cdot 0 \\ &= -255,\end{aligned}$$

where $(u_s * G'_x)$ stands for the gradient for x direction, u_s stands for the image u with separated edges and G'_x stands for the rotated Sobel kernel. The rotated kernel by 90° is used for gradient determination in y direction. It leads to computation:

$$\begin{aligned}(u_s * G'_y)(i_p, j_p) &= 2\cdot 0 + 3\cdot 0 + 4\cdot 0 + 3\cdot 0 + 2\cdot 0 \\ &+ 1\cdot 0 + 2\cdot 0 + 3\cdot 0 + 2\cdot 255 + 1\cdot 255 \\ &+ 0\cdot 0 + 0\cdot 255 + 0\cdot 255 + 0\cdot 0 + 0\cdot 0 \\ &+ (-1)\cdot 255 + (-2)\cdot 0 + (-3)\cdot 0 + (-2)\cdot 0 + (-1)\cdot 0 \\ &+ (-2)\cdot 0 + (-3)\cdot 0 + (-4)\cdot 0 + (-3)\cdot 0 + (-2)\cdot 0 \\ &= 510.\end{aligned}$$

The angle between the edge and axis x is determined as follows

$$\beta = \arctan 2(-255, 510) \doteq -26.6 \text{ deg}.$$

Slope a is determined as follows

$$a = \tan(-26.6) \doteq -0.5.$$

The result of the same computation also for other detected border pixels is in Fig. 53c.

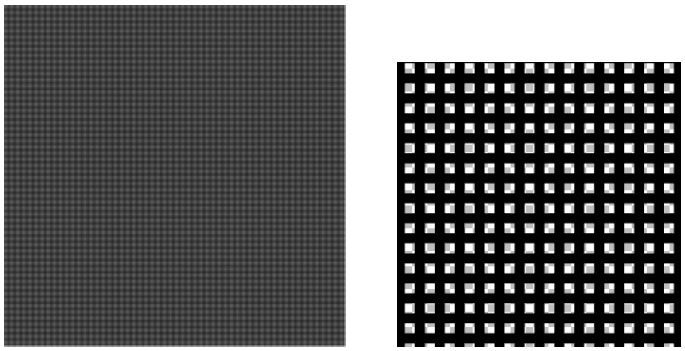
5.6 Image upsampling

The image upsamling process determines unknown pixels after size increasing of the input image as shown in Fig. 55. Techniques commonly used for that purpose are

based on interpolation, see section 3. Unknown pixels incurred as a consequence of image resampling can be considered as damage [43]. In general, we have to compute their values. The F-transform can be used for unknown pixels values determination. Unknown pixels must be marked as a damaged area Ω by mask m_Ω similar to that in Fig. 56.

(a) Original (b) Increased size

Figure 55: Size increasing of an image.



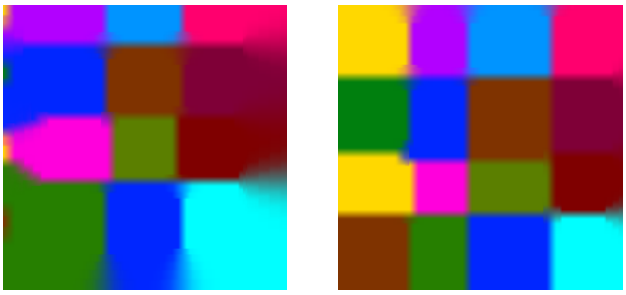
(a) Mask (b) Detail of the mask

Figure 56: Mask of the unknown pixels for image upsamling after two times resizing.

In section 3, interpolation after upscaling from 4×4 to 512×512 is shown. The same 4×4 images are also used for demonstration of the upsampling via the one-step F-transform as shown in Fig. 58. There are more possibilities to handle this task depending on the undamaged pixel positioning. Commonly used methods place known pixel to the corner of each subarea or to the centre of each subarea as shown in Fig. 57. The multi-step method is used in Fig. 57 and the one-step method in Fig. 58.

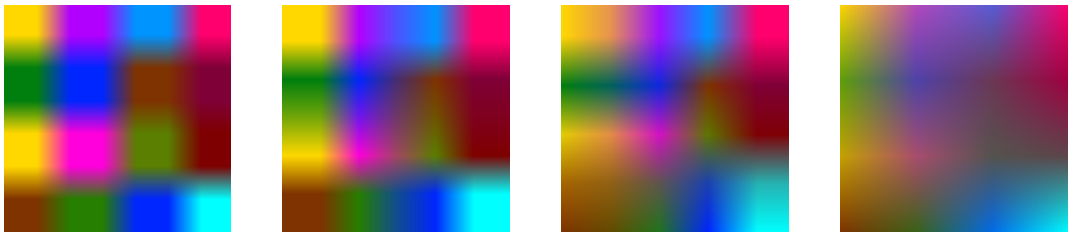
5.7 Image Filtering

The F-transform can be also used for image filtering. Photography with a semi-transparent sticker Ψ_a is shown in Fig. 59.



(a) Corner (b) Center

Figure 57: Different approaches for the known pixel preserving.



(a) $h = 74$ (b) $h = 86$ (c) $h = 110$ (d) $h = 171$

Figure 58: Image upsampling with usage of the one-step F-transform and various radii of the basic functions.

(a) Photography (b) Sticker detail

Figure 59: Photography with semi-transparent sticker.

A similar distortion can be achieved by a convolution filter. The one-step F-transform can be also used for that purpose. Fig. 60 is used as an example. There is an output of the Gaussian filtering and the one-step F-transform in Fig. 61.

The images clearly show that the F-transform achieves similar results like commonly used approaches. The processing time of this 512×512 image in our F-transform implementation and testing machine is under 80 ms. The radius of the used basic function was $h = 140$. For implementation details, see section 7.

Figure 60: Part of the photography used in Fig. 61.

(a) Gaussian filter (b) One-step F-transform

Figure 61: Comparison of the area distortion achieved by Gaussian filtering and one-step F-transform.

5.8 Noise reduction

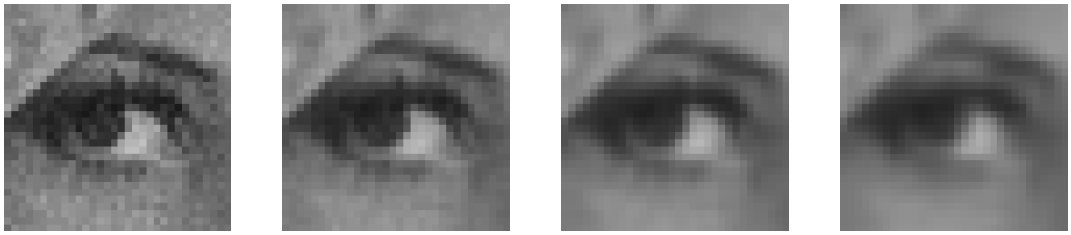
The F-transform can be also used for removing noise from the image. This is called image denoising. Noised Lena is in Fig. 62. Application of the one-step F-transform with different radii is shown in Fig. 63. More detailed images are in Fig. 64.

Figure 62: Noised Lena.

Let us remark that this solution is appropriate if noise damage is not covered by the mask. In a real situation, it is commonly very hard to mark damaged pixels

(a) $h = 2$ (b) $h = 3$ (c) $h = 4$

Figure 63: Illustration of denoising by F-transform using different basic functions radii h.



(a) Input (b) $h = 2$ (c) $h = 3$ (d) $h = 4$

Figure 64: The detail of the Lena's eye.

in that kind of distribution. In general, the one-step F-trasform can be used in the case where mask definition is a very time consuming process.

6 Optimal settings of F-transform parameters

The section describes F-transform parameters and a variety of its settings. A greatest emphasis is put on the basic functions.

6.1 Basic functions

The image reconstruction technique based on the F-transform uses clearly defined basic functions. These functions have a strong impact on the quality of reconstruction. We can use a predefined shape and radius but we can also create a new one from scratch.

6.1.1 Various types of basic functions

We use discrete basic functions A_k and B_l. A discrete basic function with respect to the Ruspini condition builds on top of the template in Fig. 65.

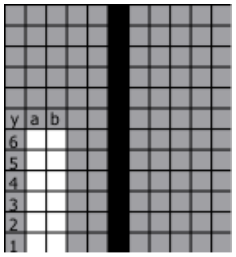
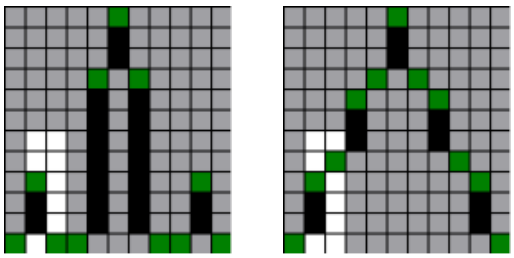


Figure 65: Template for basic function definition.

You can see white squares in Fig. 65. This squares can be marked as parts of the basic function. One mark per column. The mirrored part of the basic function is then automatically computed, see Fig. 66. We choose $y = 4$ for the column a. The value in column 10 is the same and thanks to the Ruspini condition, the values in columns 5 and 7 are easily computable as

$$f(y_5) = 1 - f(y_a),$$
$$f(y_7) = 1 - f(y_a) = f(y_5).$$



(a) a column (b) b column

Figure 66: Selection of the fourth pixel in a column and fifth pixel in b column.

Using the Ruspini condition, we can use copies of the basic function for covering the whole range. The height of the template determines values $f(y)$ on the y axis. For better usability, we choose value 0 for the first row, 1 for the last one and twice 0.5 in the middle. Two basic functions with marked values can be seen in Fig. 67.

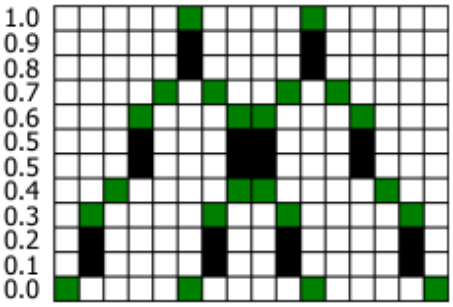


Figure 67: Basic function and shifted copy.

The part where basic functions are overlapping has its summation equal 1 in each column. For better visualization serve black vertical lines.

In experiments, we choose a generating function A_0 of a partition (with the Ruspini condition) and create a corresponding partition by shifting A_0. Because we work with uniform partitions only, the generating functions are symmetrical. Therefore, it is enough to choose their left parts. In experiments, we extend the variety of basic functions by excluding condition 4 in the definition in section 5.1. The F-transform based reconstructions with various basic functions were applied to the images of *Lena*, *cloth*, *drawing* and *nature* (see Fig. 68) which has been damaged by *text* from Fig. 2d.

We applied the multi-step reconstruction algorithm with radii from 2 to 6. The following shapes of generating functions were used: a_i, b_i, c_i, d_i (Fig. 69) and a_o, b_o, c_o, d_o (Fig. 71). As a demonstration image, *Lena* image was used. In Tab. 5, we show the values of the RMSE applied to the original (undamaged) and reconstructed images.

The reconstruction was performed by the F-transforms with basic functions generated by a_i, b_i, c_i, d_i (left nondecreasing) and a_o, b_o, c_o, d_o (oscillating). Symbols "-" in columns 5, 6 mean that all damaged pixels were reconstructed in the preceding steps. Based on Tab. 5, we can state that 4 is the maximal value of the radius of a basic function which solves the reconstruction problem for testing this kind of

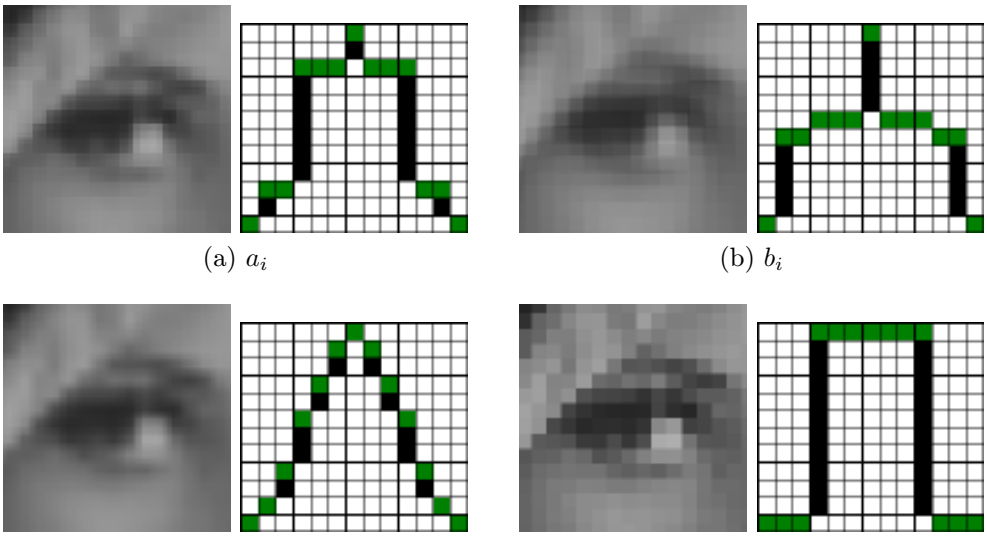
(a) Lena

(b) Cloth

(c) Drawing

(d) Nature

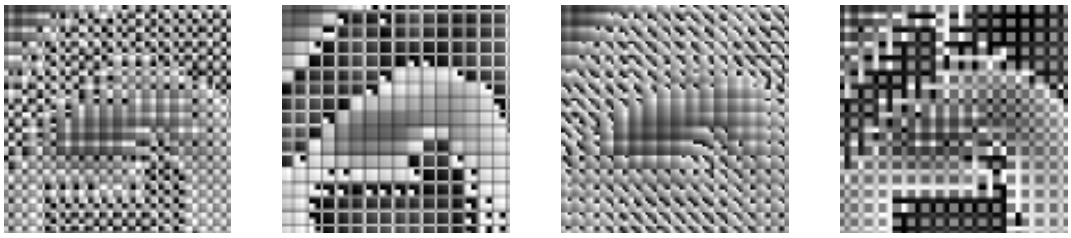
Figure 68: Images used for experiments.



(a) a_i (b) b_i

(c) c_i (d) d_i

Figure 69: Nondecreasing basic functions and the corresponding inverse F-transforms (detail of Lena's eye).



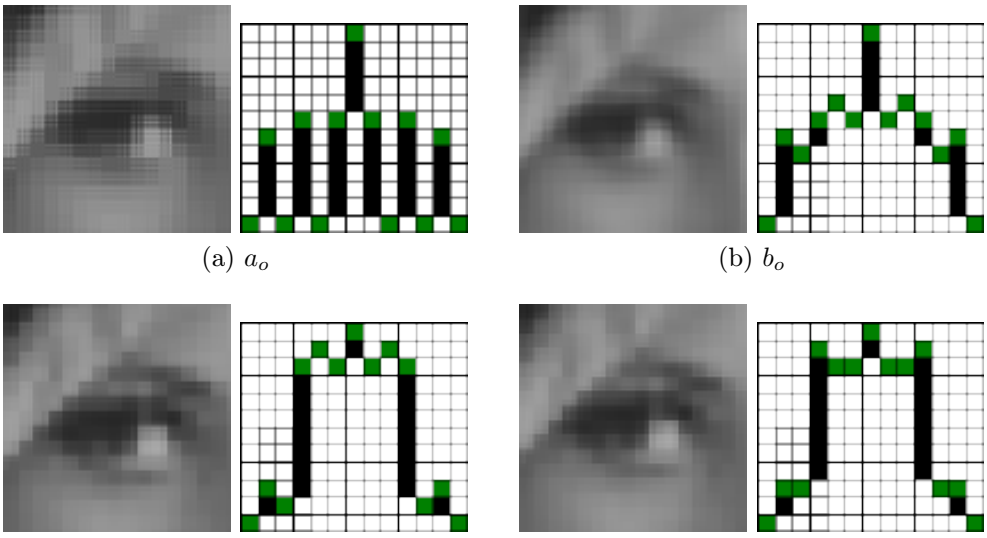
(a) a_d (b) b_d (c) c_d (d) d_d

Figure 70: Nonincreasing basic functions application (detail of Lena's eye). We used the same basic functions as in a nondecreasing example in Fig. 69 but reversed upside down.

damage. From Tab. 5, we conclude that:

- the best quality is achieved by the generated function b_o, b_i, c_i, whose shape is *triangle* or almost *triangle*;
- the worse quality is achieved by the generated function d_i;
- the quality of the reconstruction is robust with respect to small oscillations in shapes of basic functions, compare the respective RMSE values of a_i and a_o, b_i and b_o, c_i and c_o, d_i and d_o.

We analyzed the influence of various types of basic functions on image reconstruction using the F-transform. Based on the achieved results, we see that the triangular-shaped basic functions are the best ones for the reconstruction problem. The F-transform based reconstructions whose basic functions have small oscillations b_i,c_i,b_o produce visually better results than their counterparts. Related research is published in [42].



(a) a_o (b) b_o

(c) c_o (d) d_o

Figure 71: Oscillating basic functions application (detail of Lena's eye).

f/h	2	3	4	5	6	2	3	4	5	6
	Lena					Drawing				
a_i	49.98	5.59	4.75	-	-	33.91	3.97	3.90	-	-
b_i	49.96	5.50	4.64	-	-	33.90	3.92	3.84	-	-
c_i	49.97	5.51	4.65	-	-	33.90	3.83	3.76	-	-
d_i	62.99	32.71	6.66	6.66	6.66	42.47	21.98	4.82	4.82	4.82
a_o	49.96	5.93	5.10	-	-	33.90	4.43	4.37	-	-
b_o	49.96	5.47	4.60	-	-	33.90	3.90	3.82	-	-
c_o	49.98	5.64	4.80	-	-	33.91	4.04	3.97	-	-
d_o	49.98	5.59	4.75	-	-	33.91	3.97	3.90	-	-
	Cloth					Nature				
a_i	22.95	7.83	7.75	-	-	56.37	7.60	6.96	-	-
b_i	22.90	7.76	7.68	-	-	56.35	7.51	6.86	-	-
c_i	28.89	7.66	7.58	-	-	56.35	7.39	6.72	-	-
d_i	28.48	16.35	8.81	8.81	8.81	71.25	37.25	9.22	9.22	9.22
a_o	22.89	8.12	8.04	-	-	56.35	8.15	7.55	-	-
b_o	22.89	7.72	7.64	-	-	56.35	7.46	6.80	-	-
c_o	22.95	7.88	7.80	-	-	56.37	7.67	7.03	-	-
d_o	22.95	7.83	7.75	-	-	56.37	7.61	6.96	-	-

Table 5: The values of the RMSE from a comparison of the undamaged image and reconstruction of the image damaged by *text* from Fig. 2d. The values are from older currently undeveloped version of the reconstruction software.

6.1.2 Radius selection

Radius selection strongly depends on the target reconstruction. Fig. 72 shows an example on a 1D discrete function. There are rows of an image presented as a

function, where the value on axis y stands for the intensity of the pixel in the position declared by the value on axis x. A combination of the used radii is presented in Fig. 73.

6.2 Generating of suitable basic function

Shapes of the basic functions can be *nondecreasing*, *oscillating* or *nonincreasing*. Differences and influences on computation are described in [42]. We choose basic functions with radius 2 in the first step of reconstruction and radius 4 in the second step because of a fully sufficient usage on the input set of testing images. In this work, we are focused on building the shape of a basic function step by step based on the results provided by RMSE. We identified two ways of this process which will be demonstrated on a template in Fig. 65 where $max = 6$:

Column by column Find the best value for current column and continue with next column:

1. to choose radius $h = 2$;
2. to choose active column as $c = "a"$;
3. to choose current row as $y = 1$;
4. based on that values to create a basic function;
5. to use the basic function for image reconstruction;
6. to compare reconstructed image with original undamaged one by RMSE;
7. if $y = max$ then $h = 4$, $c = "b"$ and to go to step 3;
8. to change current row as $y = y + 1$;
9. to go to step 4.

Radius by radius Find the best value for current radius and continue with next radius:

1. to choose current radius as $h = 2$;
2. to choose current column as $c = "a"$;
3. to choose current row as $y = 1$;
4. based on that values to create a basic function;
5. to use the basic function for image reconstruction;
6. to compare reconstructed image with original undamaged by RMSE;
7. if $y = max$ and $h = 4$ then $c = `b'$ and to go to step 3;
8. if $y = max$ then $h = 4$ and to go to step 3;
9. to choose current row as $y = y + 1$;
10. to go to step 4.

The first way *column by column* is computed in Tab. 6. You can see that for column a there is the best RMSE value for $y = 6$. It means that we choose the sixth row for column a. For next column b the best RMSE value is in row $y = 5$ for the *Lena* and *nature* images and $y = 4$ for the *drawing* image. We choose basic function with values $y = 6$ for column a and $y = 5$ for column b. This function is in Fig. 74.

y	a	b	a	b	a	b
	Lena		Drawing		Nature	
1	62.99	5.16	46.72	4.01	71.25	7.55
2	49.99	5.04	36.63	3.89	56.39	7.39
3	49.98	4.97	36.62	3.82	56.37	7.30
4	49.97	4.93	36.62	3.79	56.36	7.27
5	49.97	4.92	36.61	3.80	56.35	7.27
6	49.96	4.95	36.61	3.84	56.35	7.32

Table 6: Values of the RMSE from a comparison of the undamaged image and reconstruction of the image damaged by *text* from Fig. 2d. *Column by column* basic function creation.

The second way of basic function creation *radius by radius* is computed in Tab. 7. The table reveals that the best value is $y = 2$ for a and $y = 5$ for column b. This basic function is shown in Fig. 75.

As a result, we can say that the step-by-step process converges to an oscilating or linear shape. For *column-by-column* way, the best solution is an oscillating basic function. Value $y = 6$ for column a and $y = 5$ for column b provides RMSE values 4.92 for *Lena*, 3.80 for *drawing* and 7.27 for *nature*. Because of very close results for *drawing* image between $b = 4$ and $b = 5$, we choose value $b = 5$ for all of them. For *radius-by-radius*, the best linear function is with $y = 2$ for column a and $y = 5$ for column b. For *Lena*, the RMSE is equal to 4.78, for *drawing* is equal to 3.61 and for *nature* is equal to 7.06. Related research is published in [40].

6.3 Usage of the one-step/multi-step F-transform method

We should distinguish between these methods based on their application. As shown in section 5, there are various applications specific for the one-step and multi-step method. We can say that the methods basically differ in the level of blurriness in the output image, which can be also handled by radius of the used basic function.

In general, the one-step F-transform can be used for the following:

- inpainting of a small area;
- image denoinsing;
- image filtering.

The multi-step F-transform is usable for the following:

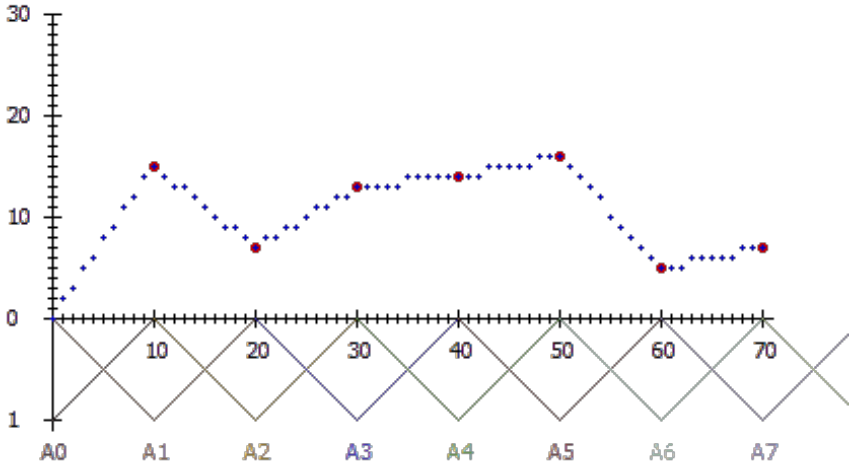
- inpainting of bigger gaps;

a / b	1	2	3	4	5	6
	Lena					
1	6.12	5.71	5.58	5.51	5.50	5.52
2	5.16	4.97	4.85	4.79	4.78	4.81
3	5.12	4.96	4.86	4.81	4.80	4.83
4	5.11	4.97	4.88	4.83	4.83	4.86
5	5.13	5.00	4.92	4.87	4.87	4.90
6	5.16	5.04	5.97	4.93	4.92	4.95
	Drawing					
1	4.93	4.32	4.21	4.16	4.15	4.19
2	3.93	3.76	3.65	3.61	3.61	3.65
3	3.91	3.76	3.67	3.63	3.64	3.68
4	3.92	3.79	3.71	3.67	3.68	3.72
5	3.96	3.83	3.76	3.73	3.73	3.77
6	4.01	3.89	3.82	3.80	3.80	3.84
	Nature					
1	8.50	7.68	7.45	7.33	7.30	7.33
2	7.63	7.33	7.16	7.07	7.06	7.10
3	7.53	7.30	7.15	7.09	7.09	7.13
4	7.50	7.30	7.18	7.13	7.14	7.19
5	7.51	7.34	7.23	7.19	7.20	7.25
6	7.55	7.39	7.30	7.27	7.27	7.32

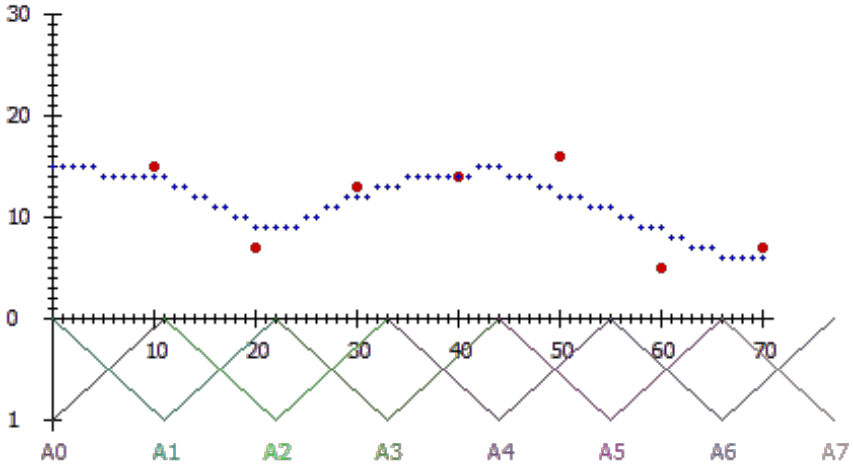
Table 7: Values of the RMSE from a comparison of the undamaged image and reconstruction of the image damaged by *text* from Fig. 2d. *Radius by radius* basic function creation.

- image upsampling;
- image filtering.

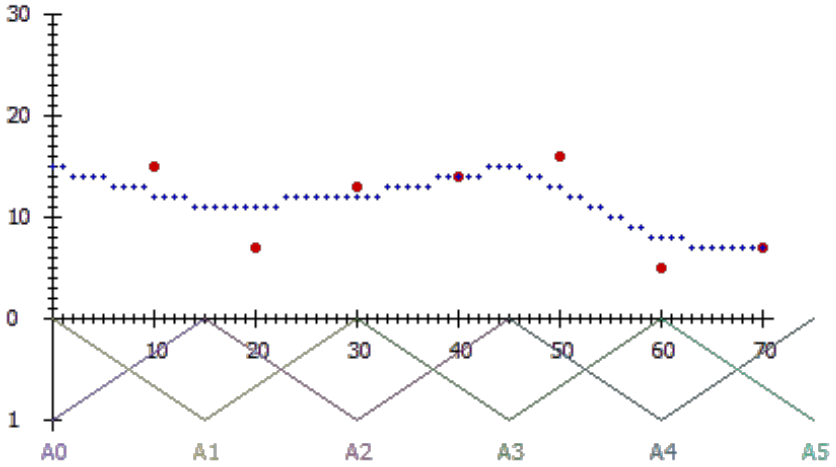
This division is not strictly separated and the distinction between one-step/multi-step depends on the desired output.



(a) $h = 10$



(b) $h = 11$



(c) $h = 15$

Figure 72: Different radii for the F-transform reconstruction applied on the same set of input points.

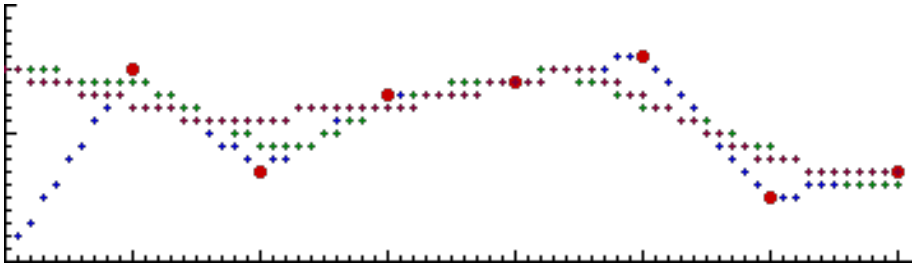
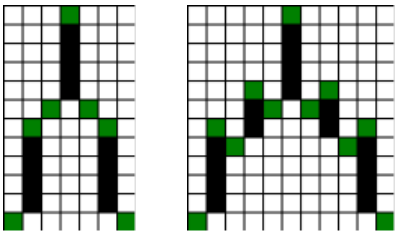
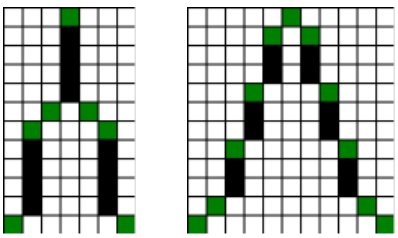


Figure 73: Combination of the various reconstructions from Fig. 72.



(a) $h = 2$ (b) $h = 4$

Figure 74: Basic function for different radii by *column by column*



(a) $h = 2$ (b) $h = 4$

Figure 75: Basic function for different radii by *radius by radius*

7 Implementation and experiments

The whole solution is developed with Qt-framework, C++, and Python. The development and testing was conducted on a computer

```
CPU: Intel(R) Core(TM)2 Duo CPU T9300 @ 2.50GHz
RAM: 2GB
OS: Windows 7 32b
```

7.1 Inpainting techniques

The inpainting techniques mentioned before *An image inpainting technique based on the fast marching method* [35] and *Navier-Stokes, fluid dynamics, and image and video inpainting* [4] were tested by implementation in the OpenCV framework. The author used Python methods

```
cv2.inpaint(img,mask,3,cv2.INPAINT_TELEA)
cv2.inpaint(img,mask,3,cv2.INPAINT_NS)
```

where `img` is the input image, `mask` is the mask, 3 stands for the inpaint radius and the last parameter determines the used algorithm.

7.2 Mask

A mask is an image with the same size as the damaged image. A mask is used to distinguish between the damaged and undamaged parts and it is therefore necessary to establish a binary distribution of the used pixel colors. We use *png* files, where the distinctive value of each pixel is the alpha channel. A transparent pixel marks the undamaged area and an opaque pixel marks the damaged area.

7.3 1D reconstruction using the F-transform

The first version of the software was aimed at the 1D reconstruction. A description of the algorithm, screenshots and example of the computation follow. A screen of the application is in Fig. 76. Fig. 77 shows the starting position. The two lines below the graph illustrate the intensity of the input (top) and reconstructed (down) pixels.

The user can define input points via *left-click*. Let us say that we want to add two points to the opposite sides of the input area. One possibility is in Fig. 78.

Coordinates are as follows

$$\begin{aligned} \text{"0"} &= [14, 6], \\ \text{"1"} &= [63, 13], \end{aligned}$$

where the first number stands for x position and y stands for the intensity value. We can use the values in the F-transform formula 5 for the involved basic functions A_1, A_2

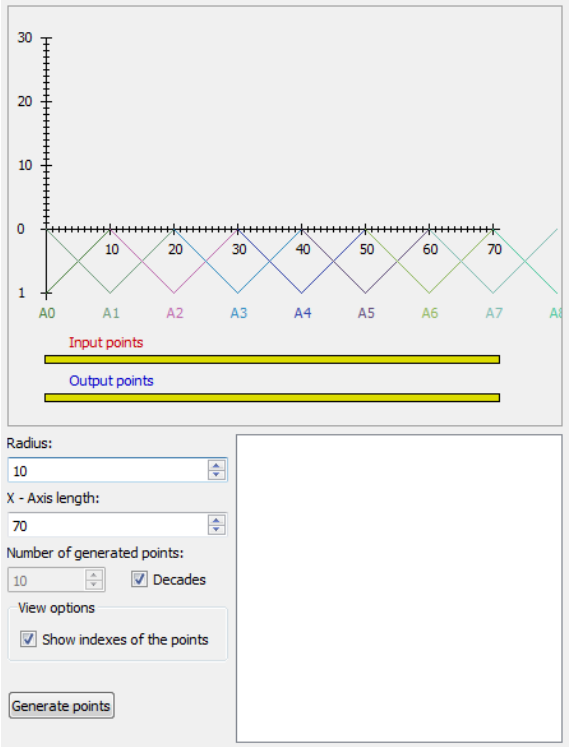


Figure 76: Default screen of the 1D reconstruction application.

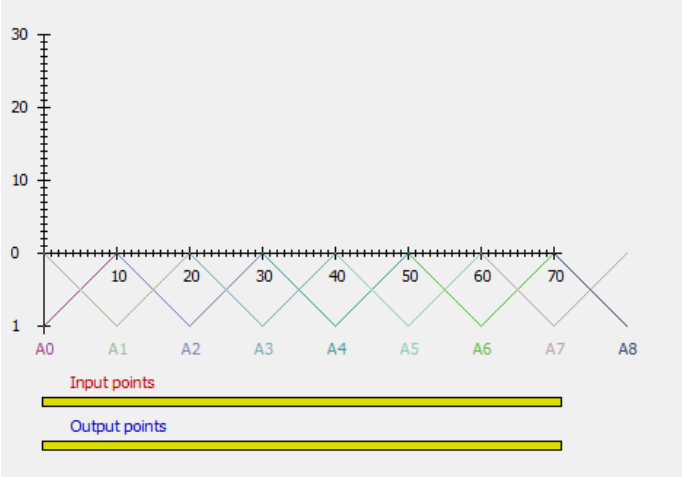


Figure 77: Default visualisation part of the application.

$$U_1 = \frac{6 \cdot 0.6}{0.6} = 6,$$
$$U_2 = \frac{6 \cdot 0.4}{0.4} = 6,$$

for the point with index 0 and basic functions A_6, A_7

$$U_6 = \frac{13 \cdot 0.7}{0.7} = 13,$$

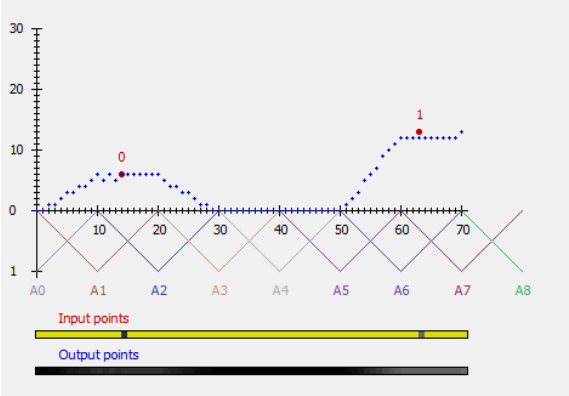


Figure 78: Function with two input points reconstructed by F-transform and illustrated as 1D image.

$$U_7 = \frac{13 \cdot 0.3}{0.3} = 13,$$

for the point with index 1. There are no other known points, well all other components are equal to zero. These results are visible in the window of the application in Fig. 79.

Point 0
[14,6]; A1 = 0.6; A2 = 0.4
Point 1
[63,13]; A6 = 0.7; A7 = 0.3

F0 = 0
F1 = 6
F2 = 6
F3 = 0
F4 = 0
F5 = 0
F6 = 13
F7 = 13
F8 = 0

Figure 79: Part of the computation for Fig. 78.

We compute an inverse F-transform formula 6 for the whole domain. Let us show a computation of the point with coordinate $x = 20$

$$\hat{u}(20) = U_2 \cdot 1 = 6.$$

Only one basic function A_2 covers the pixel with a value greater than 0. It means that only related fuzzy component U_2 is used. Next computation will be shown on pixel $x = 11$

$$\hat{u}(11) = U_1 \cdot 0.9 + U_2 \cdot 0.1 = 6.$$

Let us proceed to implementation details because if we have a close look at point $x = 11$, we can see intensity $\hat{u} = 5$ in Fig. 80.

The influence of the basic function is computed as follows

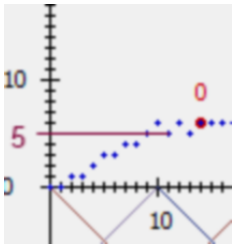


Figure 80: Detail of the approximation from Fig. 78.

```
float a = (float)1 / (float)basic_function_center;

point->bf2.weight = a * x;
point->bf1.weight = 1 - point->fw2.weight;
```

where `a` stands for the slope, `point` is the processing pixel and `bf1` respective `bf2` stands for the concrete basic function. Every pixel is covered by exactly two basic functions. Because of the Ruspini condition, we can compute only `bf2.weight`, and `bf1.weight` is taken as complement to 1.

Because of the float nature of the data, the results will be as follows

```
point->bf2.weight        0.100000001
point->bf1.weight        0.899999976
```

instead of correct ones

$$\text{bf2.weight} = 0.1,$$
$$\text{bf1.weight} = 0.9,$$

which follows to $y = 5.999...$. There are three types of float to integer conversion. Conversion *floor* rounds down, *ceil* rounds up and *round* with respect to 0.5. The default is floor, and this is why we obtained 5. We can switch from floor to round in the y computation. The result is in Fig. 81.

Computation of the fuzzy components proceeds from the basic functions point of view. There is a numerator and a denominator in U_k where the numerator is composed from basic function A_k and intensities of the related pixels. The denominator includes basic function A_k only.

The algorithm is as follows:

1. to define known input points;
2. to attach two related basic functions for every input point;
3. to compute components for all basic functions;
4. to compute inverse F-transform;

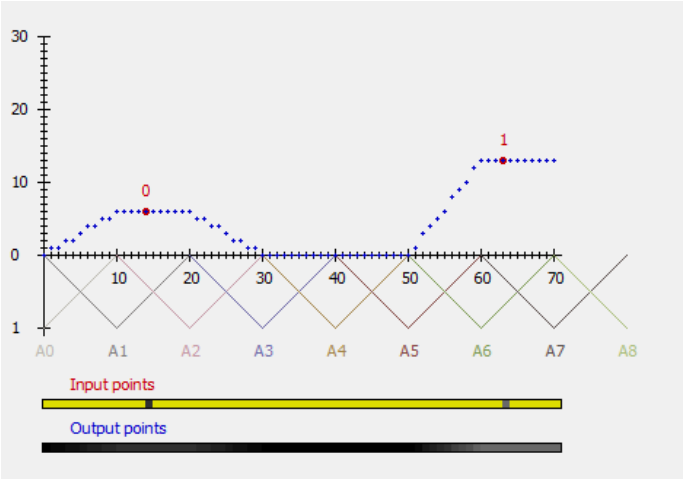
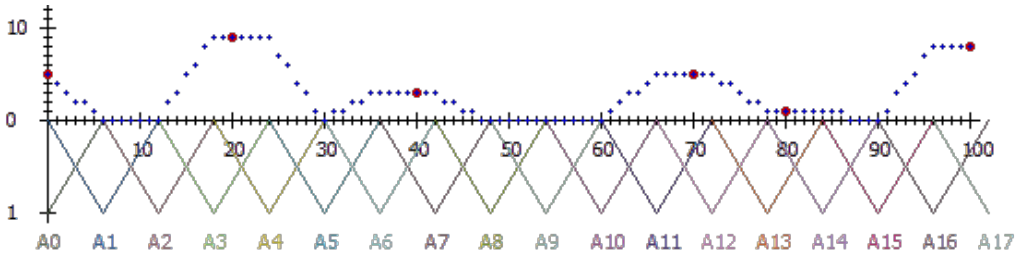
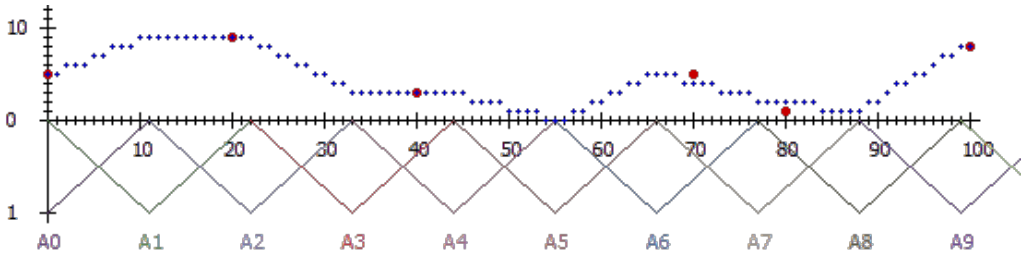


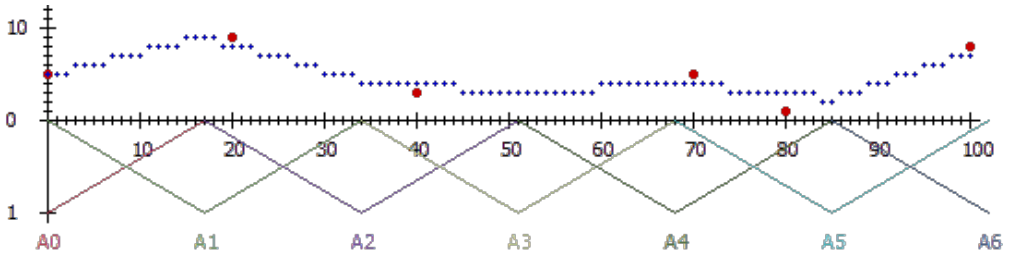
Figure 81: Approximation with *round* used for y computation.



(a) $h = 6$



(b) $h = 11$



(c) $h = 17$

Figure 82: F-transform reconstruction of the same points from Fig. 25 with different radii.

5. to produce output.

Fig. 82 shows the impact of different radii on the reconstruction.

This process is optimized in 2D usage and will be described below.

7.4 2D reconstruction using the F-transform

Let us demonstrate a 2D one-step F-transform reconstruction. A screen of the application is in Fig. 83. Lena is selected as the default input image and the user has to choose what kind of damage should be applied. After that, a radius has to be selected and the user can immediately see the output of the reconstruction.

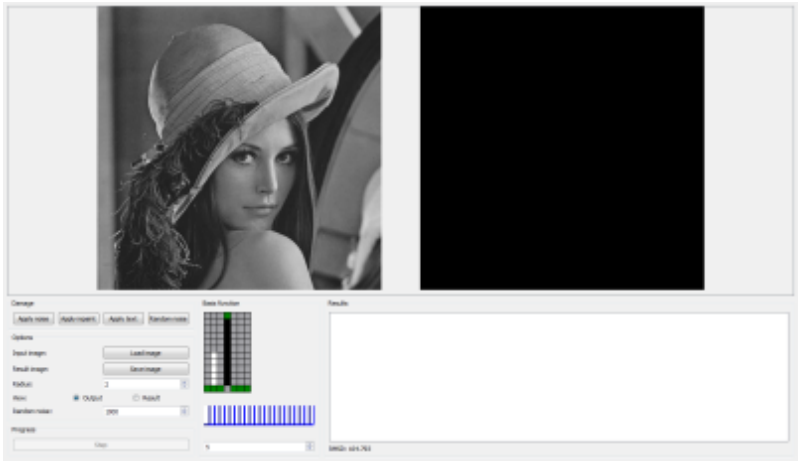


Figure 83: Default screen of the 2D reconstruction application.

In a 2D space, the image is divided by basic functions to rectangles. We use the same radius for basic functions A and B. Fig. 84 represents an example of the coverage. Let us show a concrete computation. The input is in Fig. 85.

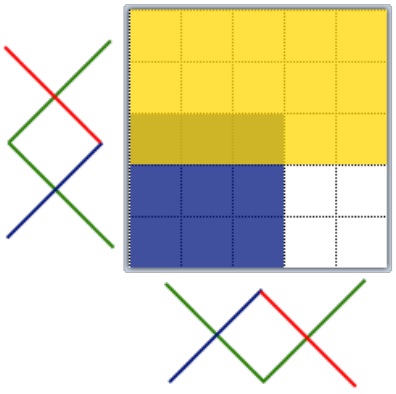
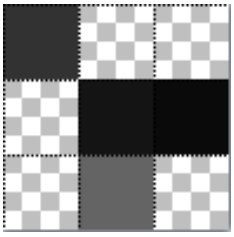


Figure 84: Blue basic function stands for A_0, B_0, green for A_1, B_1 and red for A_2, B_2.

Let us establish fuzzy partition A_0, A_1 in x direction and B_0, B_1 in y direction with a linear basic function. Every pixel is covered by 4 basic functions, 2 in each direction. It leads to:



2	50		
1		20	10
0		100	
	0	1	2

(a) Image (b) Intensities

Figure 85: Input image for 2D F-transform.

$$
\begin{aligned}
u(0,0) :& A_0(0) = 1; A_1(0) = 0 \\
& B_0(0) = 1; B_1(0) = 0 \\
u(1,0) :& A_0(1) = 0.5; A_1(1) = 0.5 \\
& B_0(0) = 1; B_1(0) = 0 \\
u(2,0) :& A_0(2) = 0; A_1(2) = 1 \\
& B_0(0) = 1; B_1(0) = 0 \\
u(0,1) :& A_0(0) = 1; A_1(0) = 0 \\
& B_0(1) = 0.5; B_1(1) = 0.5 \\
u(1,1) :& A_0(1) = 0.5; A_1(1) = 0.5 \\
& B_0(1) = 0.5; B_1(1) = 0.5 \\
u(2,1) :& A_0(2) = 0; A_1(2) = 1 \\
& B_0(1) = 0.5; B_1(1) = 0.5 \\
u(0,2) :& A_0(0) = 1; A_1(0) = 0 \\
& B_0(2) = 0; B_1(2) = 1 \\
u(1,2) :& A_0(1) = 0.5; A_1(1) = 0.5 \\
& B_0(2) = 0; B_1(2) = 1 \\
u(2,2) :& A_0(2) = 0; A_1(2) = 1 \\
& B_0(2) = 0; B_1(2) = 1
\end{aligned}
$$

Four basic functions lead to four fuzzy components. A full computation of the U_{00} will be shown.

$$U_{00} = \frac{(100 \cdot 0.5 \cdot 1) + (20 \cdot 0.5 \cdot 0.5) + (10 \cdot 0 \cdot 0.5) + (50 \cdot 1 \cdot 0)}{(0.5 \cdot 1) + (0.5 \cdot 0.5) + (0 \cdot 0.5) + (1 \cdot 0)} = \frac{55}{0.75}.$$

A computation of the other three basic functions will be shown in a shorter way.

$$
\begin{aligned}
U_{10} &= \frac{(100 \cdot 0.5 \cdot 1) + (20 \cdot 0.5 \cdot 0.5) + (10 \cdot 1 \cdot 0.5)}{0.5 + (0.5 \cdot 0.5) + 0.5} = 48, \\
U_{01} &= \frac{(20 \cdot 0.5 \cdot 0.5) + (50 \cdot 1 \cdot 1)}{(0.5 \cdot 0.5) + 1} = 44,
\end{aligned}
$$

$$U_{11} = \frac{(20 \cdot 0.5 \cdot 0.5) + (10 \cdot 1 \cdot 0.5)}{(0.5 \cdot 0.5) + (1 \cdot 0.5)} = \frac{10}{0.75}.$$

The unknown points are determined as follows

$$\begin{aligned}
\hat{u}(0,0) &= (\frac{55}{0.75} \cdot 1 \cdot 1) + (48 \cdot 0 \cdot 1) + (44 \cdot 1 \cdot 0) + (\frac{10}{0.75} \cdot 0 \cdot 0) \doteq 73, \\
\hat{u}(0,1) &= (\frac{55}{0.75} \cdot 1 \cdot 0.5) + (48 \cdot 0 \cdot 0.5) + (44 \cdot 1 \cdot 0.5) + (\frac{10}{0.75} \cdot 0 \cdot 0.5) \doteq 59, \\
\hat{u}(1,2) &= (\frac{55}{0.75} \cdot 0.5 \cdot 0) + (48 \cdot 0.5 \cdot 0) + (44 \cdot 0.5 \cdot 1) + (\frac{10}{0.75} \cdot 0.5 \cdot 1) \doteq 29, \\
\hat{u}(2,0) &= (\frac{55}{0.75} \cdot 0 \cdot 1) + (48 \cdot 0 \cdot 0) + (44 \cdot 1 \cdot 1) + (\frac{10}{0.75} \cdot 1 \cdot 0) = 44, \\
\hat{u}(2,2) &= (\frac{55}{0.75} \cdot 0 \cdot 0) + (48 \cdot 0 \cdot 1) + (44 \cdot 1 \cdot 0) + (\frac{10}{0.75} \cdot 1 \cdot 1) \doteq 13,
\end{aligned}$$

which is shown in Fig. 86.

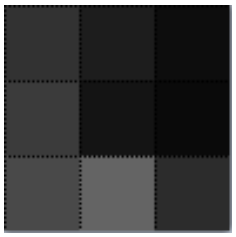


Figure 86: F-transform applied on the Fig. 85a.

We have just demonstrated how a one-step reconstruction works. Let us summarize the algorithm and show weaknesses of this approach. These weaknesses are a motivation for the multi-step approach.

7.4.1 One-step reconstruction

The algorithm is as follows:

1. to create a mask m_Ω;
2. to choose fuzzy partition of an image domain according to the size of the damaged part;
3. to compute the inverse F-transform $\hat{u}$;
4. to determine a reconstructed image u^r;
5. to produce output.

We have to determine the mask. The mask is used for a what-is-wrong definition, where the first pixel color determines the damaged area and the second determines the undamaged area. After that, the basic function shape must be selected where the default is a triangle. The radius of the function starts on $h = 2$. With these conditions, the first iteration is computed. The first iteration comes up like this:

1. to compute the inverse F-transform $\hat{u}$;

2. to check if mask covers pixels which are not processed in the preceding step.

We will repeate this process until we find a radius which completely reconstructs our image. Let us demonstrate this algorithm on a 1D example. The row of the image has some damaged (unknown) pixels as shown in Fig. 87.

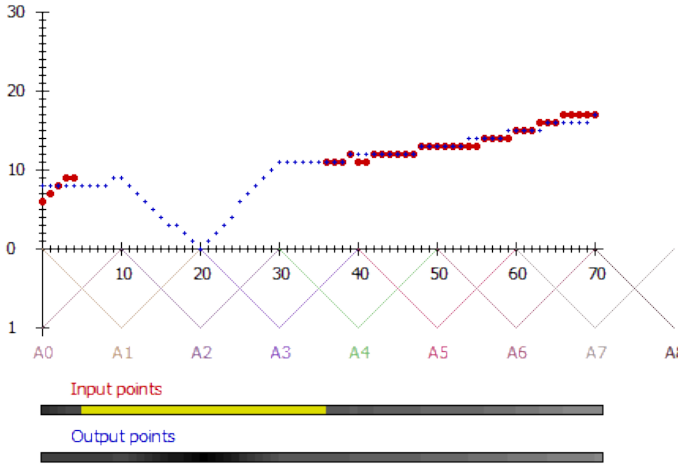


Figure 87: Row of the image with damaged (missing) part.

We can see red points illustrated in part *Input image* where the damaged ones are marked by yellow color. Blue crosses mark the output after the inverse F-transform, illustrated in part *Output image*. The scale of the y axis is transformed to pixel intensity, where 0 stands for black and 30 stands for white. We can see that the reconstruction marked by blue crosses converges to 0 in part covered by A_2. This is a problem because if we want to reconstruct a larger continuous area, the result will be converged to 0. A solution may be to choose bigger radius h and this is the idea of the one-step reconstruction. Its output will be very blurry because a bigger radius of the used basic function correlates with more intensive blurring of the reconstructed part. A better solution is to use only a part of the reconstructed pixels. We can select only these before the reconstruction starts to converge to 0, as shown in Fig. 88.

Now we must determine how to process the area covered by A_2. We can demonstrate what will happen if we also use the first pixel from the left, marked by green in Fig. 89.

We can see that the reconstruction looks more natural than before. Even better results can be achieved by the following multi-step algorithm where convergence to 0 is inhibited.

7.4.2 Multi-step reconstruction

We progress differently in that case. The algorithm is as follows:

1. to create a mask m_Ω;

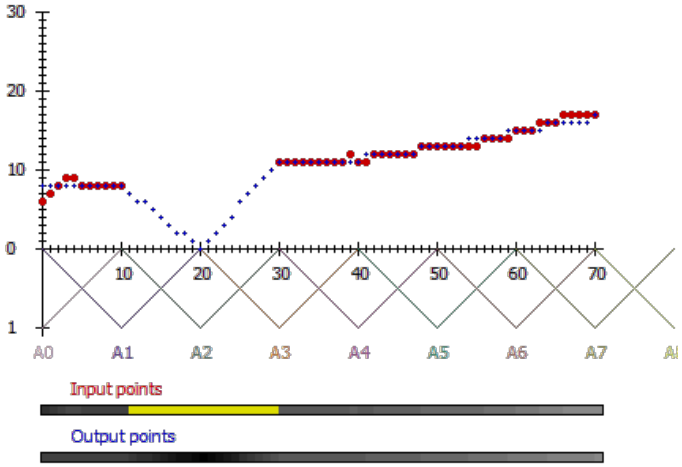


Figure 88: Row of the image with reconstructed part. You can see convergence to 0.

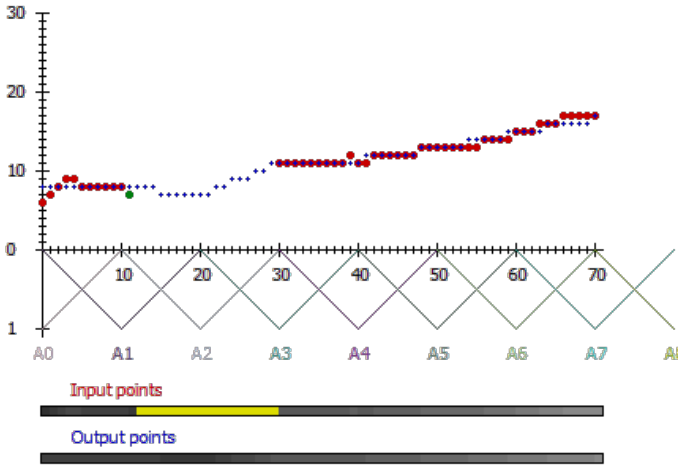


Figure 89: Row of the image with reconstructed part. One more point is used in comparison with Fig. 88.

2. to choose radius $h = 2$;
3. to establish a h-uniform fuzzy partition $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ of P;
4. to compute the inverse F-transform $\hat{u}$ of the image u;
5. to compute the combined image u^r;
6. to update the mask m_Ω by deleting pixels whose reconstructed values are strictly greater than zero;
7. if the mask is not identically equal to 0, then to update the radius $h := h + 1$ and $u = u^r$ and go to step 3. Otherwise go to step 8;
8. print output.

The main difference lies in including the previous computed pixels to the recomputing of the new ones. It means that we use various radii of the basic functions, as shown in Fig. 90, in more iterations.

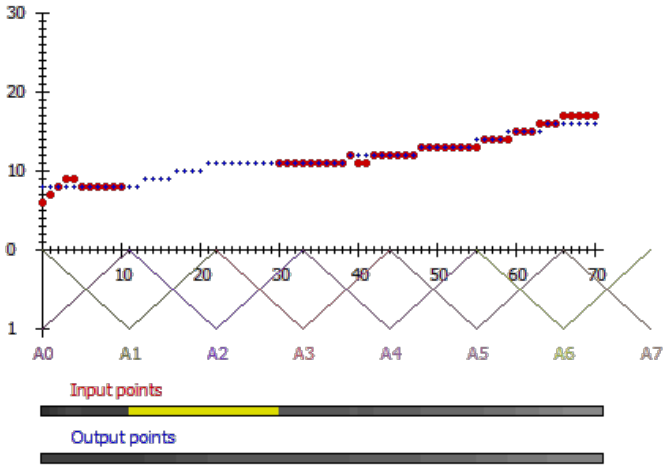


Figure 90: Row of the image with reconstructed part. Bigger radius is used in comparison with Fig. 88.

For automatic processing of many images without user interaction, a program in Python has been developed.

```
import os
import subprocess

os.chdir("./images")

for files in os.listdir("."):
    print 'processing ' + files
    subprocess.call([pCore, pInput, pMask])
```

where `pCore` is the path to the F-transform implementation, `pInput` is the path to the folder containing damaged images and `pMask` is the path for the used mask.

Let us demonstrate how a multi-step reconstruction works for 2D image in Fig. 91. Results for the various damages are in Fig. 92.

7.5 Results

We proposed to reconstruct a damaged image with the help of a fuzzy technique, namely the F-transform. Two algorithms have been proposed: one-step and multi-step. The one-step algorithm is based on the assumption that every damaged pixel is covered by a combination of basic functions so that this combination also covers at least one undamaged pixel. This assumption restricts applicability of the one-step algorithm. The multi-step algorithm has no restrictions and effectively solves

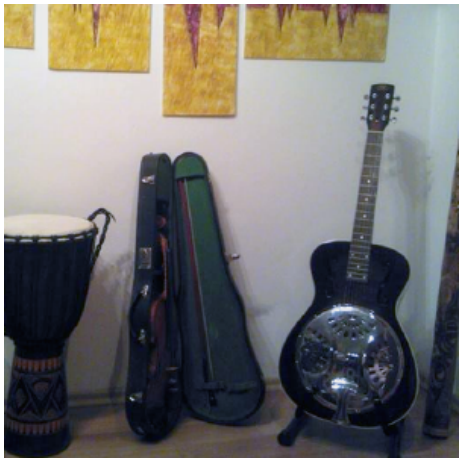


Figure 91: Demonstration image.

the problem of reconstruction. Moreover, the multi-step reconstruction has better quality than the one-step reconstruction.

We showed various experimental results on a set of 56 color images[6] and damaged by four typical types: *noise, holes, scratches, text* as is shown in Fig. 2. The reconstruction was qualified by two measures: RMSE and SSIM. The F-transform based reconstruction was compared with two traditionally applied methods of the nearest neighbor and bilinear interpolation, and with two inpainting techniques [35] and [4]. The obtained results show an absolute advantage of the F-transform over interpolation methods for all considered types of damage. The advantage over [35] and [4] techniques in the case of *noise* and *holes* damage and closeness to these methods (being slightly behind) in the case of *scratch* and *text* damage. Moreover, the comparison delineated the direction of the future research.

There are also results of the upsampling issue with attached time in Tab. 93. The testing was processed with a set of 90 grayscale images[7] with optimized algorithms for upsampling after $2\times$ resizing in the case of interpolations. The testing process consists in applying the mask from Fig. 56a and reconstructing the marked points. Processing time of the multi-step F-transform for various damage is in Tab. 94.

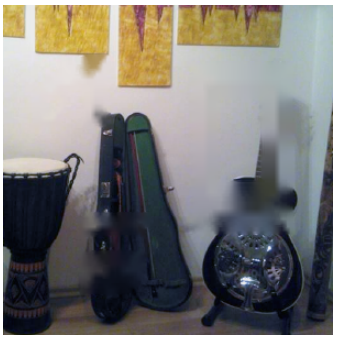
7.5.1 Images

For illustration we also append a set of demonstration images. All techniques introduced above were tested on a set of 56 color images[6] with damaged parts from Fig. 2. In Fig. 95, 96, 97, 98 there are examples of four images from the testing set damaged by masks shown in Fig. 2. There is a clear visible difference in smoothness of the reconstructed parts. The principal features are: non-connected reconstruction by the nearest neighbor interpolation, long linear stripes in the case of bilinear interpolation, and blurred connections in the case of F-transform.

From the quality of reconstruction point of view (measured by RMSE or SSIM) we can say following:

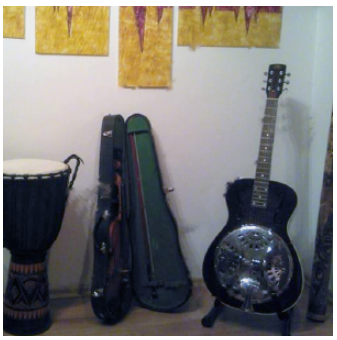
[6] `http://decsai.ugr.es/cvg/dbimagenes/c512.php`
[7] `http://decsai.ugr.es/cvg/dbimagenes/g512.php`

(a) Holes

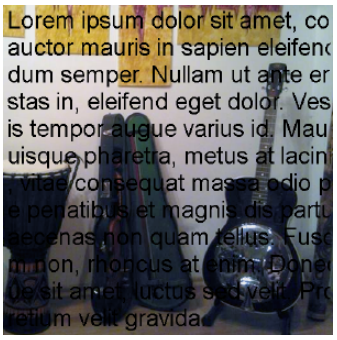


(b) Reconstruction

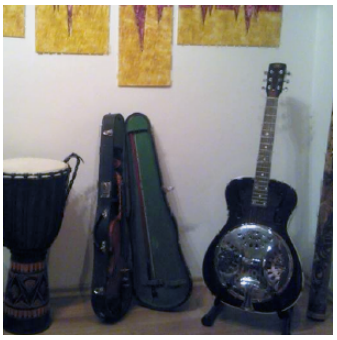
(c) Scratches



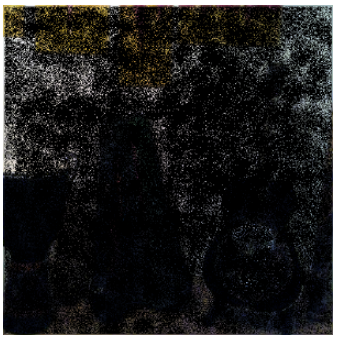
(d) Reconstruction



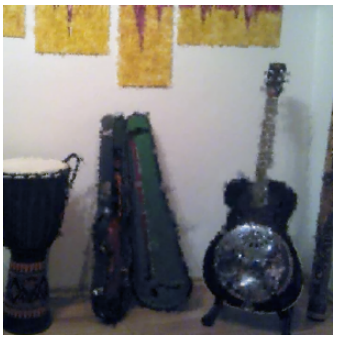
(e) Text



(f) Reconstruction



(g) Noise



(h) Reconstruction

Figure 92: Multi-step F-transform reconstruction for various damage types from Fig. 2.

	Mask: noise					
	SSIM					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	0.7071	0.7461	0.7864	0.7813	0.7555	0.7882
1st Qu.	0.8410	0.8662	0.8920	0.8928	0.8565	0.8972
Median	0.8854	0.9066	0.9334	0.9333	0.9002	0.9365
Mean	0.8796	0.8975	0.9226	0.9226	0.8921	0.9253
3rd Qu.	0.9312	0.9389	0.9618	0.9627	0.9387	0.9640
Max.	0.9763	0.9774	0.9912	0.9917	0.9837	0.9926
	RMSE					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	10.95	9.364	8.749	8.744	9.796	8.337
1st Qu.	16.30	14.429	12.092	11.958	14.726	11.738
Median	21.68	19.046	16.055	15.913	19.673	15.435
Mean	22.37	19.625	16.928	17.012	19.607	16.615
3rd Qu.	26.41	22.932	20.155	20.363	22.939	20.005
Max.	40.32	34.206	32.533	33.231	33.938	32.761

Table 8: SSIM and RMSE values of the damage in Fig. 2a

	Mask: holes					
	SSIM					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	0.8421	0.8585	0.8641	0.8570	0.8607	0.8674
1st Qu.	0.9061	0.9203	0.9290	0.9293	0.9240	0.9348
Median	0.9245	0.9382	0.9392	0.9372	0.9371	0.9451
Mean	0.9215	0.9349	0.9376	0.9372	0.9354	0.9428
3rd Qu.	0.9416	0.9481	0.9538	0.9507	0.9496	0.9573
Max.	0.9816	0.9802	0.9856	0.9848	0.9850	0.9862
	RMSE					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	8.355	6.707	7.284	7.592	7.528	7.395
1st Qu.	15.010	13.248	13.255	13.342	13.087	12.610
Median	18.051	15.580	15.689	15.689	15.376	14.800
Mean	18.695	16.604	16.517	16.568	16.482	15.704
3rd Qu.	21.432	19.413	18.786	18.851	18.744	17.997
Max.	31.906	31.429	28.727	28.370	27.957	28.671

Table 9: SSIM and RMSE values of the damage in Fig. 2b

- the F-transform technique in all the tested cases gives us great and comparable results where the largest difference is in damage in Fig. 2a;

- among interpolation methods, the bilinear interpolation shows better results than the nearest neighbor in all types of damage.

	Mask: scratches					
	SSIM					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	0.9597	0.9704	0.9739	0.9733	0.9718	0.9733
1st Qu.	0.9801	0.9859	0.9868	0.9870	0.9851	0.9864
Median	0.9882	0.9913	0.9915	0.9918	0.9887	0.9914
Mean	0.9863	0.9897	0.9906	0.9908	0.9886	0.9904
3rd Qu.	0.9927	0.9944	0.9949	0.9951	0.9939	0.9948
Max.	0.9990	0.9988	0.9994	0.9994	0.9990	0.9995
	RMSE					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	3.010	3.184	2.417	2.414	3.133	2.266
1st Qu.	5.288	4.796	4.418	4.371	5.232	4.456
Median	6.966	6.276	6.052	5.920	6.885	6.118
Mean	7.392	6.419	6.086	6.037	6.727	6.163
3rd Qu.	8.670	7.600	7.139	7.087	7.826	7.174
Max.	14.880	12.460	11.751	11.867	12.125	11.849

Table 10: SSIM and RMSE values of the damage in Fig. 2c

	Mask: text					
	SSIM					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	0.9545	0.9640	0.9689	0.9689	0.9652	0.9661
1st Qu.	0.9791	0.9822	0.9843	0.9850	0.9830	0.9837
Median	0.9874	0.9890	0.9908	0.9917	0.9899	0.9905
Mean	0.9850	0.9872	0.9888	0.9894	0.9877	0.9883
3rd Qu.	0.9933	0.9935	0.9948	0.9954	0.9945	0.9950
Max.	0.9986	0.9987	0.9992	0.9993	0.9991	0.9992
	RMSE					
Stat	Nearest	Bilinear	Telea	Navier-Stokes	FT-OS	FT-MS
Min.	3.603	3.494	2.835	2.630	2.854	2.731
1st Qu.	5.300	4.971	4.627	4.348	4.857	4.646
Median	7.096	6.794	6.062	5.808	6.453	6.117
Mean	7.656	7.097	6.566	6.357	6.872	6.682
3rd Qu.	9.228	8.508	7.956	7.675	8.241	8.224
Max.	15.736	13.701	12.771	12.883	13.315	13.329

Table 11: SSIM and RMSE values of the damage in Fig. 2d

From the processing time point of view, the F-transform is fully sufficient, where the computational time depends on the type of damage between less than a second and no more than 2.5 second. From the visual perception we can say that the F-transform provides smooth and clear output in comparison with the above-mentioned interpolations. To conclude, we say that the F-transform is the recom-

	N		
Stat	SSIM	RMSE	Time
Min.	0.5896	0.00	93
1st Qu.	0.9034	11.46	109
Median	0.9522	14.94	109
Mean	0.9255	18.55	108
3rd Qu.	0.9716	23.40	109
Max.	1.0000	81.36	140

(a)

	B		
Stat	SSIM	RMSE	Time
Min.	0.7018	3.159	109.0
1st Qu.	0.9489	7.744	109.0
Median	0.9764	10.284	125.0
Mean	0.9553	13.625	119.4
3rd Qu.	0.9868	18.066	125.0
Max.	0.9971	61.370	125.0

(b)

	FT-OS		
Stat	SSIM	RMSE	Time
Min.	0.6836	3.176	157.0
1st Qu.	0.9152	7.867	157.0
Median	0.9694	11.924	157.0
Mean	0.9393	15.488	163.1
3rd Qu.	0.9865	22.255	173.0
Max.	0.9964	62.911	173.0

(c)

	FT-MS		
Stat	SSIM	RMSE	Time
Min.	0.6897	3.176	141.0
1st Qu.	0.9462	7.823	157.0
Median	0.9751	10.535	157.0
Mean	0.9531	13.875	161.9
3rd Qu.	0.9865	18.596	172.0
Max.	0.9964	60.861	219.0

(d)

Figure 93: Processing time of various methods for image upsampling; a) stands for the nearest neighbor interpolation, b) stands for bilinear interpolation, c) stands for one-step F-transform and d) stands for multi-step F-transform.

mended solution for image reconstruction. The F-transform with linear basic functions provides smooth output and achieves high quality. Moreover, the best result in comparison with the mentioned interpolations is achieved in the case of damage *noise*. Our future research will be focused on a comparison of the F-transform with other interpolation techniques.

	Holes		
	time (ms)		
Stat	N	B	FT
Min.	249.0	124.0	2138
1st Qu.	265.0	125.0	2184
Median	266.0	140.0	2200
Mean	271.7	134.0	2210
3rd Qu.	281.0	140.2	2231
Max.	297.0	156.0	2293

(a)

	Scratches		
	time (ms)		
Stat	N	B	FT
Min.	124.0	124.0	842.0
1st Qu.	125.0	125.0	873.0
Median	140.0	125.0	874.0
Mean	133.4	130.8	879.2
3rd Qu.	140.0	140.0	889.0
Max.	156.0	156.0	920.0

(b)

	Noise		
	time (ms)		
Stat	N	B	FT
Min.	218.0	156.0	827.0
1st Qu.	234.0	171.0	842.0
Median	249.0	172.0	850.5
Mean	244.3	172.1	852.7
3rd Qu.	250.0	172.0	858.0
Max.	266.0	187.0	905.0

(c)

Figure 94: Processing time of various method for image upsampling. "N" stands for the nearest neighbor interpolation, "B" stands for bilinear interpolation, "FT" stands for multi-step F-transform.

Figure 95: Reconstruction of the image *anhinga*. The rows are from top to bottom for masks *holes*, *scratches*, *noise* from Fig. 2. The technique used for reconstruction is the same for column, where the first is the F-transform, the second is the bilinear interpolation and the third is the nearest neighbor.

Figure 96: Reconstruction of the image *athens*. The rows are from top to bottom for masks *holes*, *scratches*, *noise* from Fig. 2. The technique used for reconstruction is the same for column, where the first is the F-transform, the second is the bilinear interpolation and the third is the nearest neighbor.

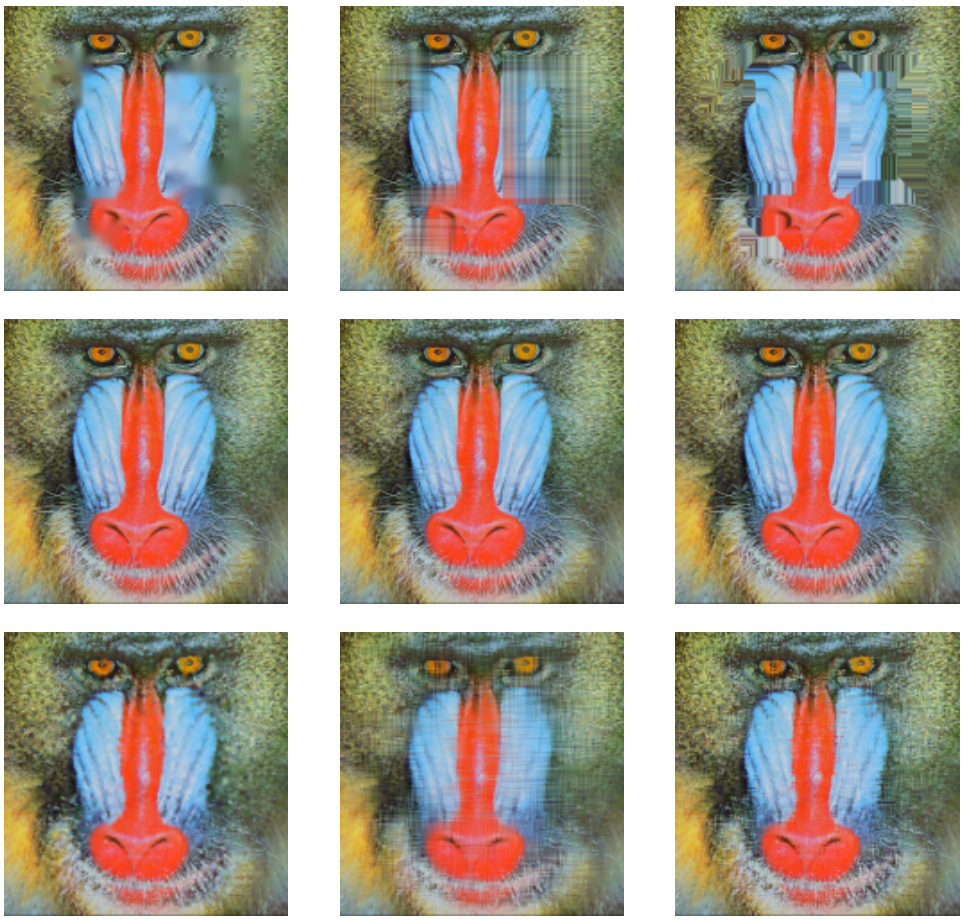


Figure 97: Reconstruction of the image *baboon*. The rows are from top to bottom for masks *holes*, *scratches*, *noise* from Fig. 2. The technique used for reconstruction is the same for column, where the first is the F-transform, the second is the bilinear interpolation and the third is the nearest neighbor.

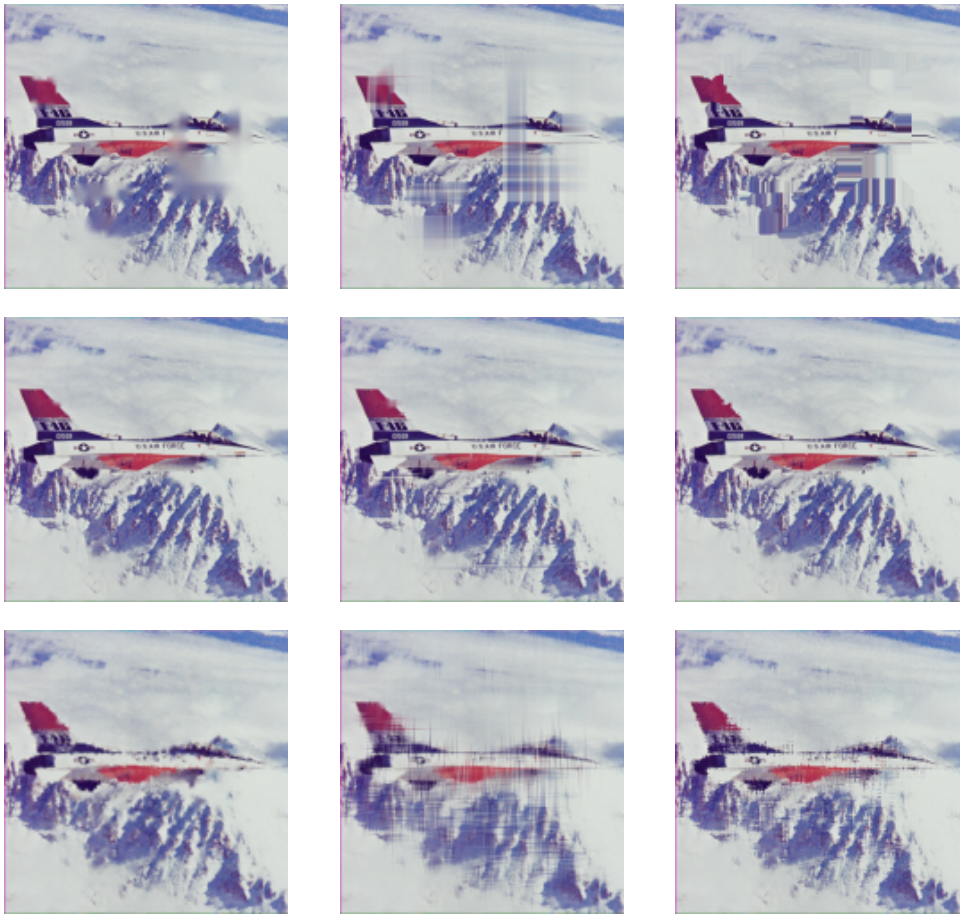


Figure 98: Reconstruction of the image *avion*. The rows are from top to bottom for masks *holes*, *scratches*, *noise* from Fig. 2. The technique used for reconstruction is the same for column, where the first is the F-transform, the second is the bilinear interpolation and the third is the nearest neighbor.

8 Conclusion

We have proposed a new technique of image reconstruction focused on image inpainting. Our technique is based on a new fuzzy technique that is named the F-transform. The undamaged neighbor of the damaged pixel is evaluated by a basic function and used for computation of fuzzy components. These components are used in the following step to determine damaged pixels. Let us recall the objectives.

To elaborate the technique of image reconstruction on the basis of the F-transform We have proven that the F-transform approximation can be used in image inpainting or image reconstruction process in general.

To develop software tools for the F-transform based reconstruction so that they are fast and easy to implement Tools have been developed for a 1D and 2D F-transform application.

To analyze the influence of the F-transform parameters and to choose their optimal values for the problem of reconstruction The influence of the various settings of the F-transform parameters has been investigated and described.

To analyze a possibility of the F-transform in solving problems that are closely related to reconstruction: upsamling, denoising or filtering, inpainting Demonstration of these problems related to image reconstruction is described and demonstrated in the dissertation. The solution is based on the F-transform.

To compare the proposed approach with conventional ones and to analyze where it is advantageous A comparison based on SSIM and RMSE is available for big sets of various color and grayscales images.

Two algorithms are described *one-step* and *multi-step*. These two algorithms have different usage based on the demanded output. In the dissertation some examples are shown. Let us recall it with a remark that all of them are applicable on the grayscale and/or color images.

One-step One-step reconstruction based on the F-transform can be used for:

- inpainting of the small area,
- image denoising,
- image filtering.

Processing in that way consists in using the wide basic function for covering all damaged area. Every basic function must cover at least one undamaged (known) pixel.

Multi-step Multi-step reconstruction based on the F-transform determines the result in more iterations. This is possible because of narrow basic functions. The damaged area is reconstructed in smaller pieces, where the reconstructed pieces from the preceding iterations are used for the following reconstruction iterations. Its possible usage is as follows:

- inpainting of the bigger gaps;
- image upsampling;
- image filtering.

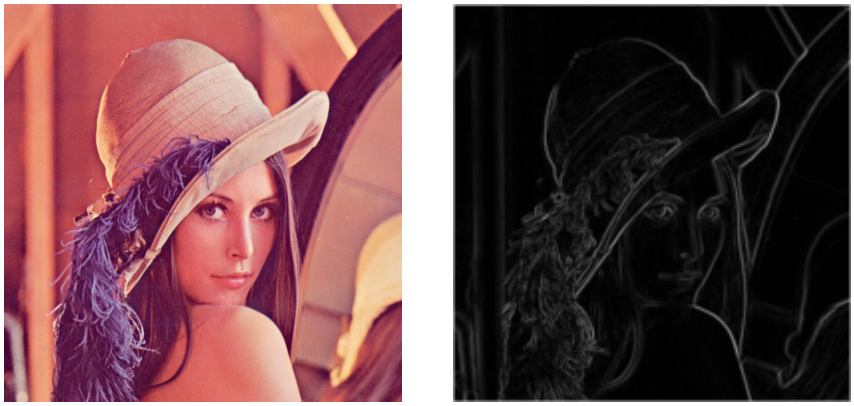
If some basic function does not cover at least one undamaged pixel, then this function is excluded from the computation. It means that damaged area covered by excluded basic functions will be reconstructed in some of the following iteration.

Implementation of the software was divided to more functional parts. The GUI and core is developed in Qt framework. In the last version, the GUI is removed because of automatization possibility. This automatization is provided by a Python script which is able to run automatic processing of many images.

The F-transform is fully sufficient to be used in the field of image inpainting. We have compared this solution with the interpolation, such as the nearest neighbor, bilinear, and inpainting techniques. There is also a possibility to use it for other image processing tasks, such as image resampling, denoising, or filtering. Our goal was to establish a powerful technique for image reconstruction, which has been satisfied.

9 Further Development

Current solutions of image reconstruction with usage of the F-transform covers only still images. An extension for video reconstruction will be one of the targets of further investigation. Examples of current solutions that are not based on F-transform are in [15, 46, 32]. From the processing point of view, where image reconstruction is a reconstruction of a 2D discrete function, video reconstruction is a reconstruction of a 3D discrete function. The third dimension is reached by using more following frames for one reconstruction iteration. Squares from 2D reconstruction are replaced by cubes or blocks in 3D. Next important improvement may be advanced edge preserving. We have tried only basic preserving based on the slope of the detected edges. For advanced purpose, F^1 transform described in [26] may be used. Determining the gradient is shown in Fig. 99.



(a) Lena (b) Gradient image

Figure 99: F^1 transform applied for gradient obtaining.

Attention will be focused on the damage characteristic research. Currently, the mask definition process may be very time consuming and not bullet proof due to human factor. It is possible that we can somehow determine which area is damaged and which is not, at least partially, automatically.

References

[1] A. Amanatiadis and I. Andreadis. "A survey on evaluation methods for image interpolation". In: *Measurement Science and Technology* 20.10 (2009), p. 104015.

[2] A. Amanatiadis and I. Andreadis. "Performance evaluation techniques for image scaling algorithms". In: *Imaging Systems and Techniques, 2008. IST 2008. IEEE International Workshop on*. IEEE. 2008, pp. 114–118.

[3] B. Baxter. "The interpolation theory of radial basis functions". In: *arXiv preprint arXiv:1006.2443* (2010).

[4] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–355.

[5] M. Bertalmio et al. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 417–424.

[6] A. Criminisi, P. Pérez, and K. Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *Image Processing, IEEE Transactions on* 13.9 (2004), pp. 1200–1212.

[7] F. Di Martino, V. Loia, and S. Sessa. "A segmentation method for images compressed by fuzzy transforms". In: *Fuzzy Sets and Systems* 161.1 (2010), pp. 56–74.

[8] F. Di Martino et al. "An image coding/decoding method based on direct and inverse fuzzy transforms". In: *International Journal of Approximate Reasoning* 48.1 (2008), pp. 110–131.

[9] I. Drori, D. Cohen-Or, and H. Yeshurun. "Fragment-based image completion". In: *ACM Transactions on Graphics (TOG)*. Vol. 22. 3. ACM. 2003, pp. 303–312.

[10] A. A. Efros and T. K. Leung. "Texture synthesis by non-parametric sampling". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 1033–1038.

[11] M. Elad et al. "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)". In: *Applied and Computational Harmonic Analysis* 19.3 (2005), pp. 340–358.

[12] R. W. Floyd and L. Steinberg. "An Adaptive Algorithm for Spatial Greyscale". In: *Proceedings of the Society for Information Display* 17.2 (1976), pp. 75–77.

[13] B. Fornberg et al. "Observations on the behavior of radial basis function approximations near boundaries". In: *Computers & Mathematics with Applications* 43.3 (2002), pp. 473–490.

[14] G. Kanizsa and G. Kanizsa. *Organization in vision: Essays on Gestalt perception*. Praeger New York, 1979.

[15] A. C. Kokaram. *Motion picture restoration: digital algorithms for artefact suppression in degraded motion picture film and video*. Springer-Verlag, 1998.

[16] N. Komodakis and G. Tziritas. "Image completion using efficient belief propagation via priority scheduling and dynamic pruning". In: *Image Processing, IEEE Transactions on* 16.11 (2007), pp. 2649–2661.

[17] T. M. Lehmann, C. Gonner, and K. Spitzer. "Survey: Interpolation methods in medical image processing". In: *Medical Imaging, IEEE Transactions on* 18.11 (1999), pp. 1049–1075.

[18] S. Masnou. "Disocclusion: a variational approach using level lines". In: *Image Processing, IEEE Transactions on* 11.2 (2002), pp. 68–76.

[19] S. Masnou and J.-M. Morel. "Level lines based disocclusion". In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. IEEE. 1998, pp. 259–263.

[20] J. M. Ogden et al. "Pyramid-based computer graphics". In: *RCA Engineer* 30.5 (1985), pp. 4–15.

[21] J. A. Parker, R. V. Kenyon, and D. Troxel. "Comparison of interpolating methods for image resampling". In: *Medical Imaging, IEEE Transactions on* 2.1 (1983), pp. 31–39.

[22] I. Perfilieva. "Fuzzy Transforms: A Challenge to Conventional Transforms". In: ed. by P. Hawkes. Vol. 147. Advances in Imaging and Electron Physics. Elsevier, 2007, pp. 137–196. DOI: http://dx.doi.org/10.1016/S1076-5670(07)47002-1. URL: http://www.sciencedirect.com/science/article/pii/S1076567007470021.

[23] I. Perfilieva. "Fuzzy transforms: Theory and applications". In: *Fuzzy sets and systems* 157.8 (2006), pp. 993–1023.

[24] I. Perfilieva and M. Danková. "Towards F-transform of a Higher Degree." In: *IFSA/EUSFLAT Conf.* Citeseer. 2009, pp. 585–588.

[25] I. Perfilieva and B. De Baets. "Fuzzy transforms of monotone functions with application to image compression". In: *Information Sciences* 180.17 (2010), pp. 3304–3315.

[26] I. Perfilieva, P. Hoďáková, and P. Hurtík. "F^1-transform edge detector inspired by canny's algorithm". In: *Advances on Computational Intelligence*. Springer, 2012, pp. 230–239.

[27] I. Perfilieva and V. Kreinovich. "Fuzzy transforms of higher order approximate derivatives: A theorem". In: *Fuzzy Sets and Systems* 180.1 (2011), pp. 55–68.

[28] I. Perfilieva, V. Novák, and A. Dvořák. "Fuzzy transform in the analysis of data". In: *International Journal of Approximate Reasoning* 48.1 (2008), pp. 36–46.

[29] I. Perfilieva and R. Valášek. "Fuzzy transforms in removing noise". In: *Computational Intelligence, Theory and Applications*. Springer, 2005, pp. 221–230.

[30] I. Perfilieva and P. Vlašánek. "Image Reconstruction by means of F-transform". In: *Knowledge-Based Systems* (2014). DOI: 10.1016/j.knosys.2014.04.007.

[31] I. Perfilieva, P. Vlašánek, and M. Wrublová. “Fuzzy transform for image reconstruction”. In: *Uncertainty Modeling in Knowledge Engineering and Decision Making*. Singapore: World Scientific, 2012.

[32] T. Shiratori et al. “Video completion by motion field transfer”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 411–418.

[33] L. Stefanini. “F-transform with parametric generalized fuzzy partitions”. In: *Fuzzy Sets and Systems* 180.1 (2011), pp. 98–120.

[34] M. Štěpnička et al. “A linguistic approach to time series modeling with the help of F-transform”. In: *Fuzzy sets and systems* 180.1 (2011), pp. 164–184.

[35] A. Telea. “An image inpainting technique based on the fast marching method”. In: *Journal of graphics tools* 9.1 (2004), pp. 23–34.

[36] P. Thévenaz, T. Blu, and M. Unser. “Interpolation revisited [medical images application]”. In: *Medical Imaging, IEEE Transactions on* 19.7 (2000), pp. 739–758.

[37] D. Tschumperlé. “Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE’s”. In: *International Journal of Computer Vision* 68.1 (2006), pp. 65–82.

[38] K. Uhlíř and V. Skala. “Radial basis function use for the restoration of damaged images”. In: *Computer Vision and Graphics*. Ed. by K. Wojciechowski et al. Vol. 32. Computational Imaging and Vision. Springer Netherlands, 2006, pp. 839–844. ISBN: 978-1-4020-4178-5. DOI: 10.1007/1-4020-4179-9_122. URL: http://dx.doi.org/10.1007/1-4020-4179-9_122.

[39] M. Vajgl, I. Perfilieva, and P. Hod’áková. “Advanced F-transform-based image fusion”. In: *Advances in Fuzzy Systems* 2012 (2012), p. 4.

[40] P. Vlašánek. “Generating Suitable Basic Functions Used in Image Reconstruction by F-Transform”. In: *Advances in Fuzzy Systems* 2013 (2013), pp. 1–6.

[41] P. Vlašánek and I. Perfilieva. “Image reconstruction with usage of the F-Transform”. In: *International Joint Conference CISIS’12-ICEUTE’12-SOCO’12 Special Sessions*. Berlin: Springer, 2013, pp. 507–514.

[42] P. Vlašánek and I. Perfilieva. “Influence of various types of basic functions on image reconstruction using F-transform”. In: *European Society for Fuzzy Logic and Technology*. Atlantis Press, 2013, pp. 497–502.

[43] P. Vlašánek and I. Perfilieva. “Interpolation techniques versus F-transform in application to image reconstruction”. In: *IEEE World Congress On Computational intelligence*. 2014.

[44] P. Vlašánek and A. Ronovský. “Using Radial-Basis Functions on Image Reconstruction”. In: *WOFEX 2011*. Ostrava: VŠB - TU Ostrava, 2011.

[45] Z. Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *Image Processing, IEEE Transactions on* 13.4 (2004), pp. 600–612.

[46] Y. Wexler, E. Shechtman, and M. Irani. “Space-time video completion”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2004, pp. I–120.

[47] J. Zapletal, P. Vaněček, and V. Skala. “RBF-based image restoration utilising auxiliary points”. In: *Proceedings of the 2009 Computer Graphics International Conference*. ACM. 2009, pp. 39–43.

[48] J. Zapletal, P. Vaněček, and V. Skala. “Influence of essential parameters on the rbf based image reconstruction”. In: *Proceedings of the 24th Spring Conference on Computer Graphics*. ACM. 2008, pp. 163–170.

[49] J. Žára et al. *Moderní počítačová grafika*. Computer press, 2005.

Author's contributions

[30] I. Perfilieva and P. Vlašánek. "Image Reconstruction by means of F-transform". In: *Knowledge-Based Systems* (2014). DOI: 10.1016/j.knosys.2014.04.007.

[31] I. Perfilieva, P. Vlašánek, and M. Wrublová. "Fuzzy transform for image reconstruction". In: *Uncertainty Modeling in Knowledge Engineering and Decision Making*. Singapore: World Scientific, 2012.

[40] P. Vlašánek. "Generating Suitable Basic Functions Used in Image Reconstruction by F-Transform". In: *Advances in Fuzzy Systems* 2013 (2013), pp. 1–6.

[41] P. Vlašánek and I. Perfilieva. "Image reconstruction with usage of the F-Transform". In: *International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions*. Berlin: Springer, 2013, pp. 507–514.

[42] P. Vlašánek and I. Perfilieva. "Influence of various types of basic functions on image reconstruction using F-transform". In: *European Society for Fuzzy Logic and Technology*. Atlantis Press, 2013, pp. 497–502.

[43] P. Vlašánek and I. Perfilieva. "Interpolation techniques versus F-transform in application to image reconstruction". In: *IEEE World Congress On Computational intelligence*. 2014.

[44] P. Vlašánek and A. Ronovský. "Using Radial-Basis Functions on Image Reconstruction". In: *WOFEX 2011*. Ostrava: VŠB - TU Ostrava, 2011.