J. Vis. Commun. Image R. 48 (2017) 43–53

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Non-texture image inpainting using histogram of oriented gradients ☆

Vahid K. Alilou [a], Farzin Yaghmaee [b,*]

[a] Department of Electrical and Computer Engineering, Semnan University, Semnan, Iran
[b] Faculty of Electrical and Computer Engineering, Semnan University, Semnan, Iran

ARTICLE INFO

Article history:
Received 4 September 2016
Revised 4 April 2017
Accepted 9 June 2017
Available online 11 June 2017

Keywords:
Image inpainting
Histogram of oriented gradients
Dominant orientation
Local gradients

ABSTRACT

This paper presents a novel and efficient algorithm for non-texture inpainting of images based on using the dominant orientation of local gradients. It first introduces the concept of a new matrix called *orientation matrix* and then uses it for faster and better inpainting. The process of propagating information is carried out by using a new formulation which leads to much more efficient computations than the previous methods. The gain is both in terms of computational complexity and visual quality. The promising results in contexts of text, scratch, and block loss inpainting demonstrate the effectiveness of the proposed method.

1. Introduction

Image inpainting is the problem of repairing/recovering parts of an image that have been damaged or partially occluded by undesired objects [1]. This problem has attracted the interest of the scientific community in the last decade, mainly due to the growth of important applications such as image editing [2,3], image replacement [4], noise removal [5,6], image restoration [7,8], compression [9–11], disocclusion [12], zooming and super-resolution [13], error concealment [14,15], and video inpainting [16–18].

Digital image inpainting is a technology lying at the intersection of computer graphics, image, and signal processing. The basic idea in this technology is to develop such an algorithm that imitates the way professional art repairers fix damages in paintings or sculptures.

Formally, the problem of inpainting can be defined as follows: "Given an input image $\boldsymbol{I}$ with a target region Ω try to fill-in all pixels of the target region by using the information of the known areas $\Phi = I - \Omega$" [19].

The research in this field was pioneered by Bertalmio et al. [1]. They introduced an inpainting approach based on linear partial differential equations. After that, many inpainting methods were proposed to address this problem [20–22]. The inpainting methods found in the literature can be categorized into two major categories: *diffusion-based* and *exemplar-based*.

非深度方法分类

The first category is diffusion-based. The key idea behind these methods is the smooth propagation of information from the boundary toward the interior of the missing areas along the direction of isophote lines (lines of equal gray values). It is the same way as professional painting restorators work on damaged artworks [23].

The second category is exemplar-based in which the propagation of information is carried out through a copy-and-paste texture synthesis procedure. This terminology refers to the methods that synthesize entire patches by learning from patches in the known part of the image. This idea was initially proposed by the work of Efros et al. [24]. Afterward Criminisi et al. [25] designed a revolutionary algorithm for removing large objects based on texture synthesis with giving higher priority to linear structures. After that many other exemplar-based methods have been proposed based on this work. For example, the idea of using non-local means of a set of candidate patches was proposed by Wong et al. [26]. Xu and Sun [27] proposed a method in which the priorities of patches are calculated based on the sparseness of the patch's non-zero similarities to its neighboring patches. Kumar et al. [28] added a new "edge length" term in the priority equation of [25] which leads to propagation of linear structures in a better way. In their very recent work [29], they formulated the exemplar-based image inpainting as a metric labeling problem and solved it using simulated annealing algorithm. Many other exemplar-based approaches [30–33] can be found in the literature.

The methods proposed in both categories are interesting and efficient in different situations. The patch-based texture synthesis technique of the exemplar-based methods provides significant advantages which make it possible to inpaint large holes in images.

不同的方法适用于不同的场景，力所不能及的地方在哪里呢?

☆ This paper has been recommended for acceptance by Zicheng Liu.
* Corresponding author.
E-mail address: f_yaghmaee@semnan.ac.ir (F. Yaghmaee).

http://dx.doi.org/10.1016/j.jvcir.2017.06.003
1047-3203/

Diffusion-based methods on the other hand propose different formulations which enable the algorithm to add some new information to the image rather than performing only copy-and-paste procedure.

The diffusion-based methods are well founded on the theory of partial differential equation (PDE) and variational formulation. They, in general, avoid having unconnected edges that are perceptually annoying and can achieve convincingly excellent results for filling non-textured or relatively small missing regions. However, they suffer from blur close to edges and contours. Some new works have been proposed to solve this problem. For example, Benzarti and Amiri [34] proposed a new method based on using nonlinear diffusion tensor. This technique is more effective than the use of a simple diffusion gradient due to the reliable information it can give in the presence of complex structures. Another work is proposed by Bornemann and März [35]. They propose to match the high-level quality of the methods Bertalmio et al. [1] and Tschumperl [36] to develop a very fast inpainting algorithm. Their algorithm is established based on two observations. The first is about the celebrated method of Bertalmio et al. [1] which is, neglecting some anisotropic diffusion steps that are interleaved for stabilization. The second is about discrete, linear, and single-pass inpainting method of Telea [37]. As a result, they successfully designed a fast inpainting algorithm which behaves strongly diffusive and creates peculiar transport patterns.

Thanks to advances in image processing, new methods have been proposed for automatic detection of inpainting areas. Bertalmio et al. [20] introduced a class of automated methods for digital image/video inpainting. Venkatesh et al. [38] proposed an efficient video inpainting algorithm capable of addressing inpainting under the stationary camera and moving cameras. The stationary background region is filled by a combination of adaptive background replacement and image inpainting technique. Recently, Mosleh et al. [17] presented a two-stage framework for automatic video text removal in order to detect and remove embedded video texts and fill-in their remaining regions by appropriate data.

The contribution of this paper is to develop a new practical non-texture image inpainting approach with the aim of recovering scratches, texts, or block losses from an image with a very low computational burden. This approach makes it possible to build a practical image editing software that can be used in various devices such as mobile phones and tablets. An extension of this work can also be used for the application of automatic video inpainting. The remainder of this paper is organized as follows: Section 2 describes the proposed method. In Section 3 we perform experiments and show the results and next we conclude this work in Section 4.

2. Proposed approach

The key idea behind our algorithm is to use a new matrix we call it *orientation matrix*. The overall architecture of the proposed inpainting system is shown in Fig. 1. At the first step the algorithm calculates the gradients of the input image in both x and y directions and then computes our orientation matrix using these gradients.

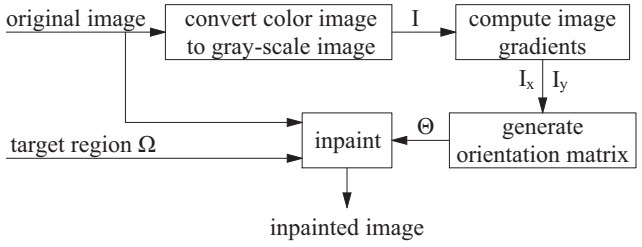


Fig. 1. The overall architecture of the proposed system.

2.1. Orientation matrix

For a given input image $\boldsymbol{I}$, we introduce the following equation for calculating our orientation matrix $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta}(x,y) = round\left(\left(\frac{\arctan\left(\frac{\boldsymbol{I}_x(x,y)}{\boldsymbol{I}_y(x,y)}\right)}{\pi} + \frac{1}{2}\right) \times n\right) \tag{1}$$

in which $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$ are gradients in x and y directions. This formula generates a matrix containing integer values ranging from 0 to n. Each value corresponds to an angle in the range of $-\frac{\pi}{2}$ to $+\frac{\pi}{2}$. In our implementations, n is considered to be 16. However, one may choose a different value for the quantization of the gradients. Fig. 2 shows these values and their corresponding angles. Since $\frac{-\pi}{2}$ and $\frac{\pi}{2}$ are at the same direction, we use:

$$\boldsymbol{\Theta}(x,y) = \begin{cases} \boldsymbol{\Theta}(x,y), & \boldsymbol{\Theta}(x,y) > 0 \\ n, & \text{otherwise} \end{cases} \tag{2}$$

This means we limit our quantization over exactly n different integer values (each corresponds to a specific orientation).

Now, we have a matrix of the same size as the original image $\boldsymbol{I}$ in which each pixel (x,y) contains an integer value in the range of 1–n. Each value represents the orientation of the gradient of the image at pixel (x,y).

To better show the characteristics of this matrix, we took the well-known *Cameraman* image and then calculated the orientation matrix for this image. Then we expanded it in n different binary images which have been shown in Fig. 3. As you can see in the images, each different value (from 1 to 16) captures the oriented gradients of the original image in the direction of its corresponding orientation.

2.2. Filling process

Here, we describe the filling process of the unknown areas using our pre-calculated orientation matrix. But, before we proceed with the description of this stage, we provide the important notations we will use in this paper (see Table 1). Fig. 4 demonstrates our notation diagram. This figure illustrates an input image $\boldsymbol{I}$ with a target region Ω. The goal is to estimate the color components of all pixels of the target region using the information exists in the source region Φ.

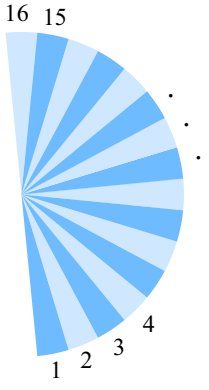


Fig. 2. Orientation angles and their corresponding values θ ($n := 16$).

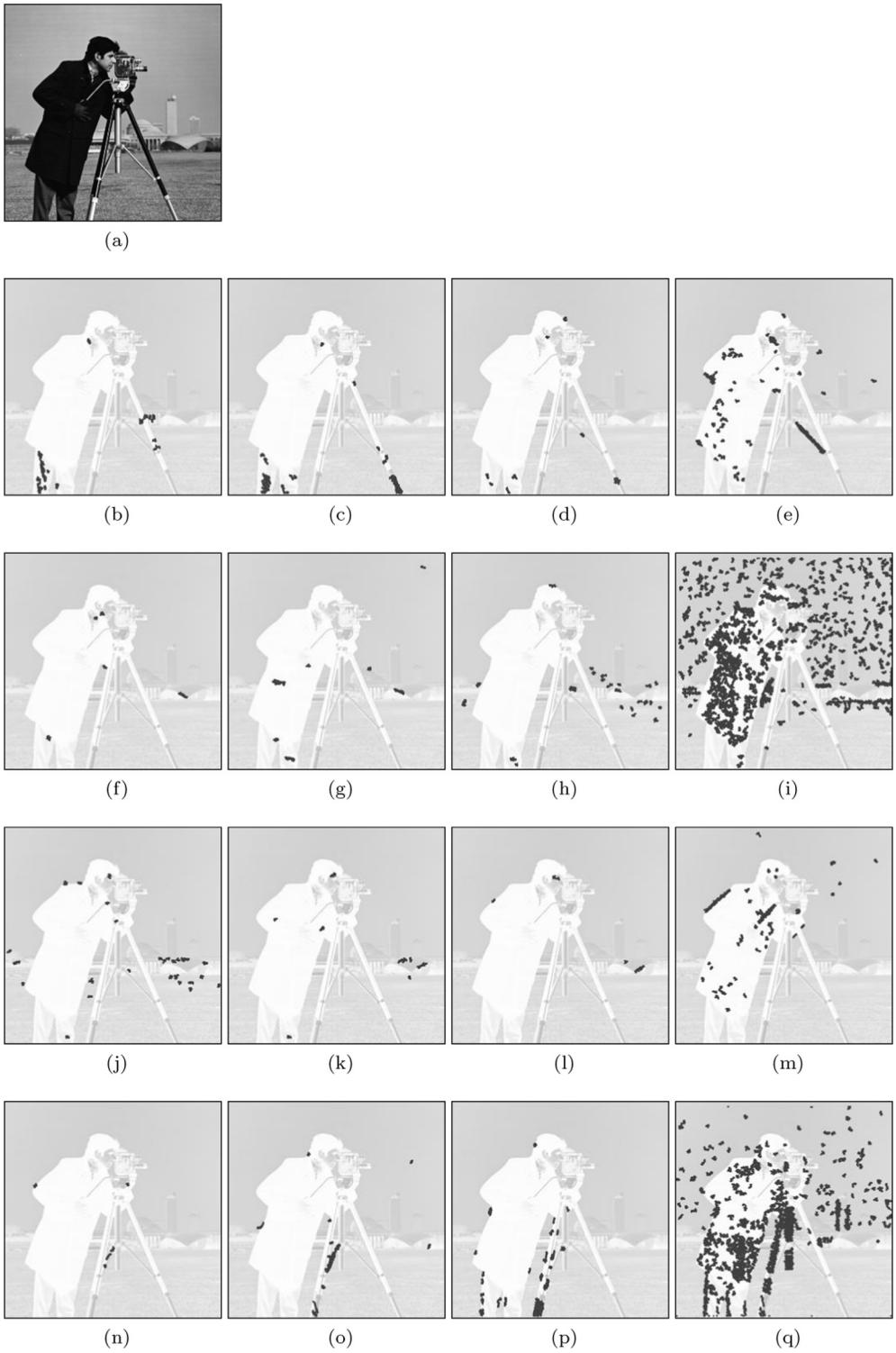


Fig. 3. Illustration of the orientation matrix for *Cameraman* (a) original image, (b-q) oriented gradients.

To do this, we present a model which gradually propagates the information from outside of the boundaries toward the interior regions. The hole diminishes pixel by pixel until no unknown pixel remains unfilled.

Here, we focus on a single iteration of the algorithm to show how our model can inpaint the unknown areas. As it is shown in Fig. 4, we first identify the one-pixel thick exterior and interior boundary regions which are denoted by $\delta\Omega^+$ and $\delta\Omega^-$ respectively. We use the following morphological operations to extract these regions:

$$\delta\Omega^+ = (\Omega \oplus B) - \Omega \tag{3}$$

$$\delta\Omega^- = \Omega - (\Omega \ominus B) \tag{4}$$

in which B is a 3×3 structural element.

Table 1
Notations and definitions.

Notation	Definition
$\boldsymbol{I}$	input image
Ω	target region
Φ	source region
$\delta\Omega$	boundary
$\delta\Omega^{-}$	1 pixel thick boundary region inside Ω
$\delta\Omega^{+}$	1 pixel thick boundary region outside Ω
Ψ_p	a patch over the boundary centered at point p
φ	source region of the selected patch
$\boldsymbol{\Theta}$	orientation matrix
θ^*	dominant orientation
$\boldsymbol{H}$	histogram of oriented gradients
B	3×3 structural element
$\boldsymbol{W}$	5×5 weight matrix
$\boldsymbol{n}_p$	8-connected neighborhood pixels of p

Then, for each pixel $p \in \delta\Omega^{+}$, the algorithm makes a patch Ψ_p centered at p and defines the following set:

$$\boldsymbol{\Theta}_p^{\varphi} = \{\boldsymbol{\theta}_i : \theta_i \in \boldsymbol{\Theta}_p \cap \varphi\} \tag{5}$$

in which

$$\boldsymbol{\Theta}_p = \boldsymbol{\Theta}(\Psi_p) \tag{6}$$

and φ is the source region of Ψ_p.

Here, we should note that, at the first iteration of the algorithm, the point p is located exactly over the boundary line of $\delta\Omega^{+}$ which is hitting the unknown area Ω and may contain an error-prone orientation. So, we nullify its orientation and let the algorithm to choice a new orientation for this point.

Now, we can make a 16-bin histogram $\boldsymbol{H}$ of oriented gradients over $\boldsymbol{\Theta}_p^{\varphi}$. The histogram of oriented gradients (HOG) can capture edges or gradient structures and thus represents one of the most important local features of an image [39–41]. The technique used in this paper is very similar to the well-known histogram of oriented gradients method of [42]. The only difference is that, instead of 9 bins of the histogram (which is defined in [42]), we create a 16-bin histogram.

Fig. 5 illustrates an example of a 7×7 patch $\boldsymbol{\Theta}_p$ of the orientation matrix centered at point p. At this moment, we need a formula that tells us what should be the orientation of point p according to the orientation values of its neighboring pixels. To find this, one may propose to make a histogram over $\boldsymbol{\Theta}_p^{\varphi}$ and choose the orientation of the highest peak. The highest peak in the histogram $\boldsymbol{H}$ corresponds to the dominant orientation of local gradients [43]. However, this is not an appropriate method for our problem because in this way all neighboring pixels would have the same impact in determining the orientation value of p. Therefore, we propose the following equation:

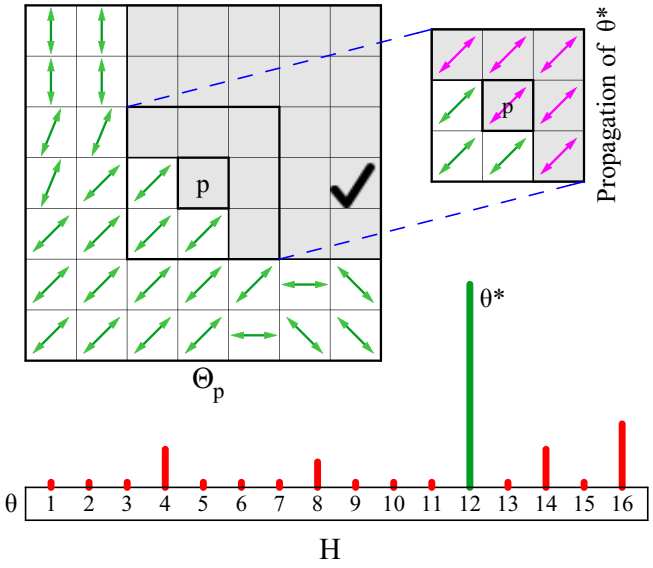


Fig. 5. Illustration of a 7×7 patch $\boldsymbol{\Theta}_p$ and its histogram. θ^* is the dominant orientation of the local gradients which is propagated into p and all its 8-connected neighboring pixels.

$$\boldsymbol{H}_k = \sum_{q \in \varphi_p^k} \exp\left(-\frac{d_q^2}{2\sigma^2}\right) \tag{7}$$

$$\varphi_p^k = \{q : q \in \Psi_p \cap \varphi, \quad \theta(q) = k\} \tag{8}$$

in which d_q is the euclidean distance between the q^{th} pixel and p. The parameter σ determines the variance of the gaussian function and can be tuned to yield better results.

Now, the highest peak of $\boldsymbol{H}$ can give us the proper dominant orientation we were looking for. However, we may not have any dominant orientation inside the patch Ψ_p, meaning that there is no isophote around that point. So, in order to determine whether there is any dominant orientation in the patch or not, we first normalize the histogram using:

$$\boldsymbol{H}_k = \frac{\boldsymbol{H}_k}{\sum_{k=1}^{n} \boldsymbol{H}_k} \tag{9}$$

The proper dominant orientation then can be found using:

$$\theta^* = \begin{cases} arg\max\{\boldsymbol{H}_k\}, & \max(\boldsymbol{H}_k) > \beta \\ \phi, & \text{otherwise} \end{cases} \tag{10}$$

the parameter β, here, is a threshold value in the range of $(0 \quad 1)$ and needs to be adjusted to give better inpainting results.

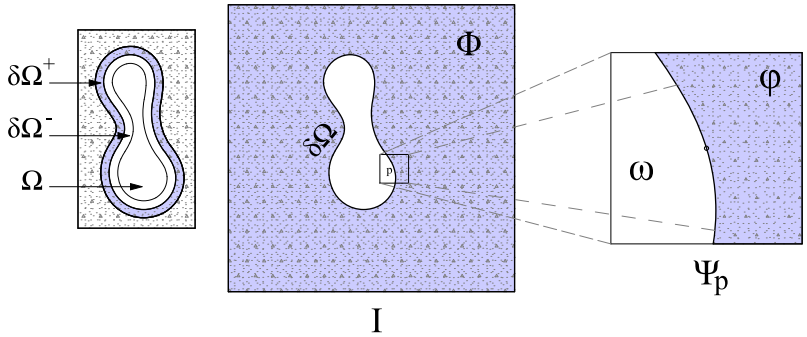


Fig. 4. Notation diagram.

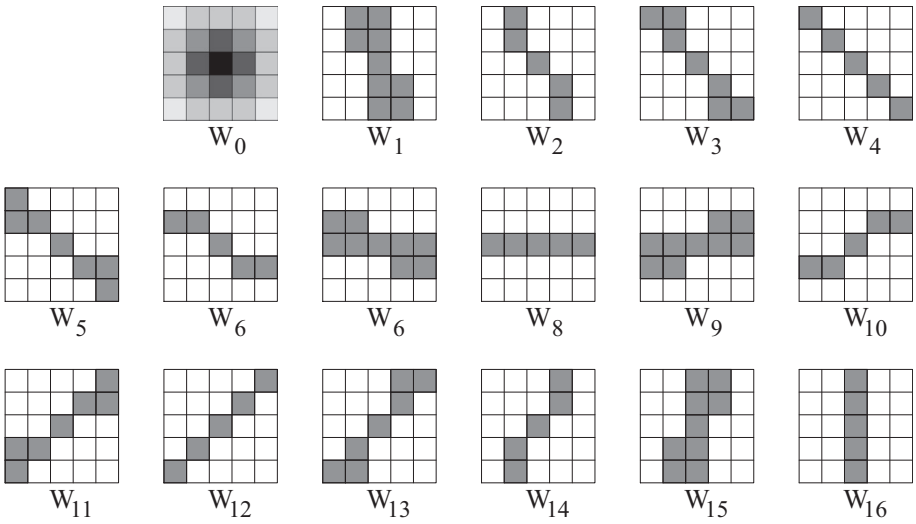


Fig. 6. Weight matrices: the white cells are 0 and the black cells are 1.

At the next step, if $\theta^* \neq \phi$, the algorithm propagates the value of θ^* into $\boldsymbol{\Theta}(p)$ and all its eight-connected neighborhood pixels $\boldsymbol{\Theta}(\boldsymbol{n}_p)$ which are not previously assigned ($\boldsymbol{n}_p = \boldsymbol{n}_{j=1,\ldots,8}(p)$). The entire process is shown in Fig. 5.

When the propagation process of the orientation values gets completed, the filling process over the interior boundary regions $\delta\Omega^-$ commences by using the following equation:

$$\boldsymbol{I}(p) = \frac{\sum_{q\in\varphi} \boldsymbol{W}_i(q) \cdot \boldsymbol{I}_p(q)}{\sum_{q\in\varphi} \boldsymbol{W}_i(q)} \tag{11}$$

in which p is a pixel on $\delta\Omega^-$, $\boldsymbol{I}_p$ is a 5×5 patch of $\boldsymbol{I}$ centered at p, $\boldsymbol{W}_i$ is a symmetric weight matrix of the same size, and φ is the source region of that patch.

The weight matrix $\boldsymbol{W}_i$, in this equation, is defined with respect to the value of θ^*. Fig. 6 demonstrates all weight matrices used in our implementations. If $\theta^* = 1$ then the algorithm uses $\boldsymbol{W}_1$. The value of θ^* determines the index of the weight matrix $\boldsymbol{W}_i$. Therefore, for each orientation θ^* we have a certain weight matrix $\boldsymbol{W}_i$. Moreover, we define $\boldsymbol{W}_0$ as below for a situation in which $\theta^* = \phi$:

$$\boldsymbol{W}_0(x,y) = \exp\left(-\frac{(x-3)^2 + (y-3)^2}{2\sigma^2}\right) \tag{12}$$

2.3. Inpainting

In the previous sub-sections, we described the basic concepts of our inpainting approach. Now, we can present our inpainting algorithm.

Algorithm 1. The proposed inpainting algorithm

Data: Input image $\boldsymbol{I}$, Target region Ω
Result: Inpainted image
Initialization: Set $\beta = 0.4$, and $\sigma = 5$
1. Computer the orientation matrix $\boldsymbol{\Theta}$ using (1) and (2)
2. While $\Omega \neq \phi$ do:
  1. Identify the interior $\delta\Omega^-$ and exterior $\delta\Omega^+$ boundary regions using (3) and (4)
  2. For each $p \in \delta\Omega^+$ do:
    (a) Make $\boldsymbol{\Theta}_p$ of (5) with regard to β
    (b) Build a normalized 16-bin histogram $\boldsymbol{H}$ of $\boldsymbol{\Theta}_p$ using (7) and (9)
    (c) Calculate θ^* using (10)
    (d) Propagate θ^* into p and all its 8-connected neighborhood pixels
  3. For each $p \in \delta\Omega^-$ do:
    (a) Specify the weight matrix $\boldsymbol{W}_i$ according to the value of $\theta^*(p)$
    (b) Fill $\boldsymbol{I}(p)$ using (11)
    (c) Update $\Omega(p)$
3. Return $\boldsymbol{I}$

At the first step, the algorithm computes the gradients of the input image in both x and y directions and makes our orientation matrix. Afterward, the inpainting process starts by identifying the interior and exterior boundary regions. Next, it performs a two-phase procedure in order to achieve the inpainting result. The first phase executes procedures associated with calculating and propagating the dominant orientations into $\delta\Omega^+$. In the second phase, the algorithm accomplishes the inpainting process by filling-in the unknown pixels of the interior boundary regions $\delta\Omega^-$. The overall pseudo-code description of the proposed algorithm is listed in Algorithm 1.

Note that our discussion is limited to non-texture inpainting. So, we only need to find the isophotes hitting the boundary regions and develop a method for propagating information in the direction of those isophotes. For a single pixel p in the boundary, the inpainting can be performed using the information of local pixels around p. If an isophote hits p, a 5×5 patch centered at p is large enough to identify it. To find the dominant orientation around that point, a patch containing one more marginal pixel can be favorable. But the larger values are not recommended because it may mislead the process of finding θ^*.

2.4. Inpainting of color images

The previous sub-section showed how to inpaint a gray-scale image. Here we extend the proposed method for inpainting color images.

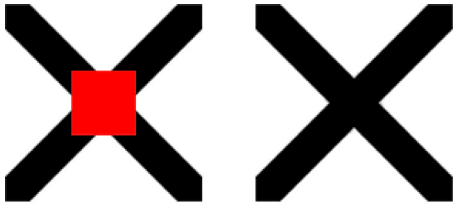
For color images, instead of having one matrix representing the image, we have three matrices. In other words, every singular pixel of a color image should be represented by a three-dimensional vector in which three dimensions correspond to the primary colors

Y

Cb

Cr

Fig. 7. A color image and its Y, Cb, and Cr components [48]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) (b)

Fig. 8. A simple synthetic image. (a) The original image (128×128 pixels) with a square mask depicted in red. (b) The result of the proposed inpainting method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

red, green, and blue (RGB). To have a better consistency with human eyes, we can separate brightness from the color information. Human eyes are more sensitive to the luminance changes than to changes in chrominance [44,45].

Beside RGB, we can find many other color spaces in the literature such as CIE Lab [46], CMC, and YCbCr. These color spaces have better consistency with the perceptual properties of human vision system. In our implementations, we used YCbCr color space which is a simple linear transform of RGB and yet imitates the properties of human vision system [47].

Fig. 7 shows a sample image with its Y, Cb, and Cr components. As you can see in the figure, the most important component that perfectly preserves edges is Y. In other words, the information in Y is enough for extracting isophotes from an image.

For color images, our algorithm first transfers the input image into the YCbCr color space. Then it computes the orientation matrix using only the Y component of that image. It also performs the same procedure of the original algorithm to find the boundary regions and determine the dominant orientations. The only difference is when the algorithm fills the pixels of the unknown region. It applies (11) to all three color components of the image and returns the result.

2.5. Time complexity

Here, we discuss the time complexity of the proposed approach. The most important parts of the proposed algorithm are listed below:

1. Computing the orientation matrix
2. Identifying the interior and exterior boundaries
3. Finding the dominant orientations
4. Filling-in the missing regions

Consider N be the number of pixels of the entire image $\boldsymbol{I}$, and M be the number of pixels of the unknown regions Ω. The question is how many arithmetic operations is required to fill-in only one pixel of the unknown areas Ω.

The answer is simple. Looking at Algorithm 1, we see that the orientation matrix is computed only one time at the beginning of the algorithm. Thus, it consumes $O(N)$. The process of identifying the interior and exterior boundary regions is also of the order $O(N)$. So, we can conclude that the most time-consuming parts deal with the processes of finding the dominant orientation and filling the missing regions. For each pixel $p \in \Omega$, the algorithm makes a 7×7 patch centered at p to find the most dominant orientation θ^* and propagate it and a 5×5 patch to calculate (11). This means that for each pixel $p \in \Omega$ we only need to apply our calculations over $49 + 25$ pixels of the image. So, the order of these parts is

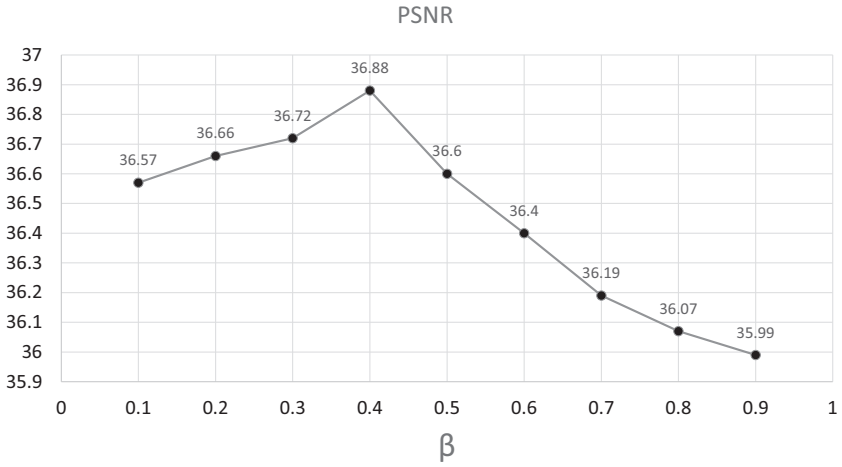


Fig. 9. The effect of choosing different values of β.

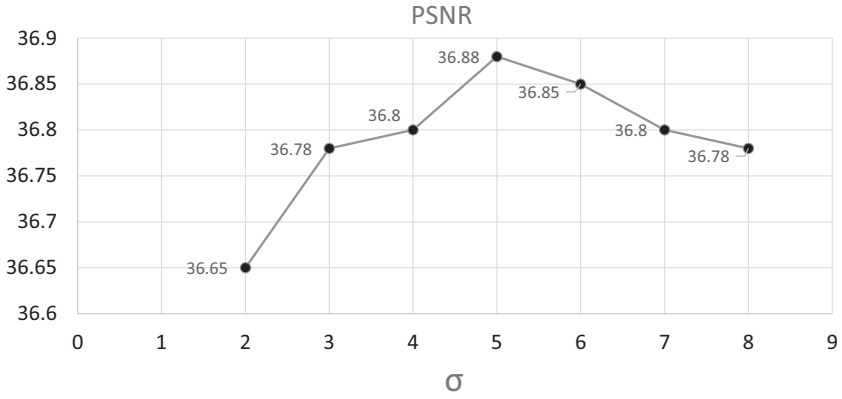


Fig. 10. The effect of choosing different values of σ.

also linear. Therefore, we can conclude that the proposed algorithm is of the order $O(N+M)$. Thus, for the proposed algorithm, we would expect to see a remarkable performance in terms of speed.

3. Experiments and comparisons

This section demonstrates the results of testing the proposed algorithm on a variety of natural images and provides comparisons with other non-texture inpainting methods.

3.1. Implementation details and parameter setup

We implemented the proposed method using MATLAB and C/C++ programming language. Beside this, we also implemented the state-of-the-art inpainting methods including GRNN-based inpainting of Alilou et al. [45], the total variation (TV) method of Dahl et al. [49], the Coherence Transport method of Bornemann et al. [35], the exemplar-based method of Criminisi et al. [25], and the sparsity method of Xu et al. [27]. All experiments were performed on a 2.4 GHz Intel system, using a dual-core CPU and 4 GByte of memory.

We applied our algorithm to the applications of text, scratch, and block loss inpainting for both gray-scale and color images. Fig. 8(a) shows a synthetic image in which the target region is shown in red. Fig. 8(b) is the output of the proposed inpainting method. This simple synthetic image shows how our algorithm inpaints without any blur effect on the image. It preserves the structures and strong edges by continuing them in the target regions.

In the next sub-sections, we provide some natural images which have taken from the state-of-the-art works and apply our method to these images. To evaluate the objective quality of the inpainted images we used the peak signal-to-noise ratio (PSNR) [50] which is defined as follows:

$$PSNR = 10\log_{10}\left(\frac{255}{RMSE(\boldsymbol{I}, \boldsymbol{I}_{ref})}\right) \tag{13}$$

where

$$RMSE = \sqrt{\frac{\sum_{x,y}(\boldsymbol{I}(x,y) - \boldsymbol{I}_{ref}(x,y))^2}{N}} \tag{14}$$

in which N denotes the number of pixels of the input image, $\boldsymbol{I}_{ref}$ indicates the original undamaged image, and $\boldsymbol{I}$ is the result of the inpainting algorithm. This metric measures the image quality degradation on a pixel by pixel basis as perceived by the human vision system [51,52].

In order to achieve the best results, we must find and fix the optimal values for the parameters β, and σ. The first parameter β which is defined in (10) is a threshold value between 0 and 1. In the proposed algorithm, this parameter controls the trade-off

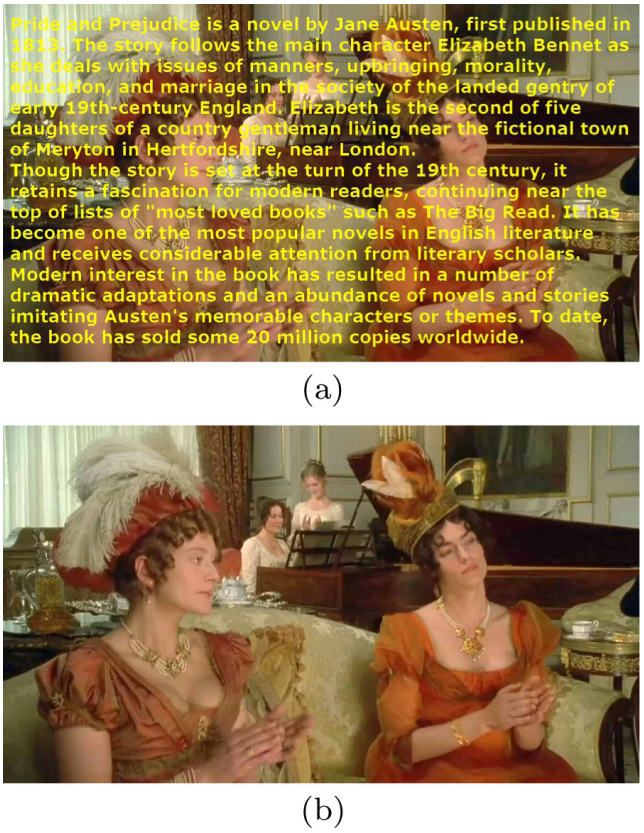


Fig. 11. Text removal. (a) The original image of size 1280×720. (b) Inpainted image.

between smooth inpainting and sharp inpainting. If we choose $\beta = 1$, then θ^* will always be ϕ. Thus, there will be no option for the algorithm to choose among different weight matrices $\boldsymbol{W}_i$ to better preserve the edges. This can result in blurring of the sharp edges. Choosing $\beta = 0$, on the other hand, enforces the algorithm to find at least one dominant orientation around the point $p \in \delta\Omega^+$. However, if there were no isophote around that point, the algorithm should inpaint it in a smooth fashion. Therefore, we need a moderate value of β to deal with this trade-off.

Fig. 9 demonstrates the effect of choosing different values for β. The PSNR achieved for each value is the average of 25 trials on the original *Lena* image with different inpainting masks. The diagram verifies that the optimal value for β is 0.4.

The next parameter is σ which is defined in (7). This parameter determines the variance of the gaussian function. Like the previous parameter, we performed a similar experiment on this parameter. The result is demonstrated in Fig. 10. As it is shown in the figure, the optimal value for σ is 5.

3.2. Text/scratch removal

Our first example is a color photograph that has been damaged by a text. Both damaged and inpainted images are shown in Fig. 11. The result looks visually pleasant which indicates that the proposed method is very successful at non-texture inpainting of images.

The next example is shown in Fig. 12. In this figure, we see the original image of *parrots* which is damaged by a text. The goal is to repair this damage by using inpainting techniques. Fig. 12(c) shows the result obtained by using Criminisi's method [25]. The result of the Sparsity method [27] is also shown in Fig. 12(d). Both of these methods are exemplar-based and use copy-and-paste texture synthesis for reconstructing the unknown areas. Instead of making

Fig. 12. (a) The original image of parrots (768×512 pixels), (b) text added image, (c) result of Criminisi's algorithm [25] (PSNR = 29.68), (d) result of Sparsity method [27] (PSNR = 29.48), (e) result of TV method [49] (PSNR = 30.52), (f) result of GRNN method [45] (PSNR = 31.51), (g) result of CT method [35] (PSNR = 31.55), (h) our result (PSNR = 32.91).

Fig. 13. Scratch/text removal. (a) Scratched/text added images of *Lena*. (b) Our results (PSNR = 40.20 db, 37.05 db, 37.50 db).

new information, they only copy some parts of the image into the unknown areas. The common problem of these methods is that they may generate undesirable artifacts which are visible in the resulting images.

The diffusion-based techniques, on the other hand, work in a different manner. Using their mathematical formulations they are capable of making new information on the unknown areas. Fig. 12(e) shows the result obtained by using the diffusion-based methods of TV [49] with the parameters $\tau = 0.85$ and $\epsilon_{rel} = 10^{-3}$. This method diffuses information from the outside of the boundary regions into interior areas. The main drawback of these methods is that they tend to introduce blur in the unknown areas [53]. For this image, the blur effect is fully visible around the eye of the left parrot.

Fig. 12(f and g) shows the results obtained by GRNN [45], and CT [35] methods, respectively. The method proposed in this paper yields a better result which is shown in Fig. 12(h). The visual comparisons reveal the fact that this new technique outperforms the previous methods in terms of quality.

To better evaluate the performance of the proposed algorithm, we took the original *Lena* image and damaged it using some text or scratch on it. The damaged images are shown in Fig. 13(a). Subtraction of these images from the original ones can give us the masks. These masks specify the regions that we want to be restored. The results of applying the proposed algorithm to these images are shown in Fig. 13(b). The achieved PSNR value for all these images are larger than 37.00 which indicates that the proposed approach is highly effective at removing/inpainting texts or scratches from an image. What is more interesting, here, is that the computational time consumed for each image is only 0.45 s. This means that the proposed approach is sufficiently fast at repairing damaged areas. Quantitative comparisons (PSNR and Time) with the state-of-the-art methods are provided in Table 2.

3.3. Recovery of consecutive block losses

When we transmit a compressed image or video over an error-prone channel, the image block loss can be a challenging problem. To resolve this issue many error concealment methods have been proposed yet [54–56]. Most of these methods use the same techniques as inpainting. Here, we perform some experiments to demonstrate the potential applicability of the proposed method for solving such problems.

Fig. 14 shows the degraded images of *Lena* in which some blocks are consecutively lost. We applied the proposed algorithm to recover the lost blocks of these images. The results are shown in Fig. 14 (c), and (d). The PSNR value achieved for each one are 36.35 and 33.95, respectively. Both qualitative and quantitative evaluations demonstrate the effectiveness of the proposed algorithm.

Table 2 compares our algorithm against the methods of TV [49], Criminisi [25], Sparsity [27], GRNN [45], and CT [35]. The comparisons include both PSNR (for objective quality) and time required for inpainting each degraded image. The first three rows evaluate the effectiveness of the proposed algorithm for the application of text/scratch removal. The next two rows evaluate its efficiency for recovering the consecutive block losses. Looking at the PSNR column of the table, we see that the proposed approach outperforms the previous methods in terms of quality. In terms of computational time, it also obtains very interesting results (see the TIME

Table 2
Comparison of PSNR and time achieved by our algorithm and the methods Criminisi [25], Sparsity [27], TV [49], GRNN [45] for the applications of text, scratch, and block loss inpainting on *Lena* image. Best result is shown in bold font.

Figure	Damage		PSNR (dB)						Time (s)					
	Type	Ratio (%)	Criminisi	Sparsity	TV	GRNN	CT	Ours	Crim	Spar	TV	GRNN	CT	Ours
Fig. 13(a)	Scratch	8	36.75	37.42	39.57	39.64	39.32	**40.20**	76	211	3.30	3.10	0.64	**0.45**
Fig. 13(c)	Text	10	32.76	34.12	36.24	36.38	36.26	**37.05**	67	186	3.60	2.30	0.84	**0.44**
Fig. 13(e)	Text	10	34.88	34.90	36.93	37.49	37.05	**37.50**	103	244	3.50	2.80	0.80	**0.45**
Fig. 14(a)	Block Loss	8	32.30	33.38	35.98	36.33	35.36	**36.35**	36	199	3.20	1.40	0.64	**0.48**
Fig. 14(b)	Block Loss	24	29.70	30.27	32.78	33.01	33.49	**33.95**	187	208	5.40	5.00	1.35	**0.61**
Fig. 12	Text	7.5	29.68	29.48	30.52	31.51	31.55	**32.91**	205	378	17.5	3.6	0.93	**0.62**

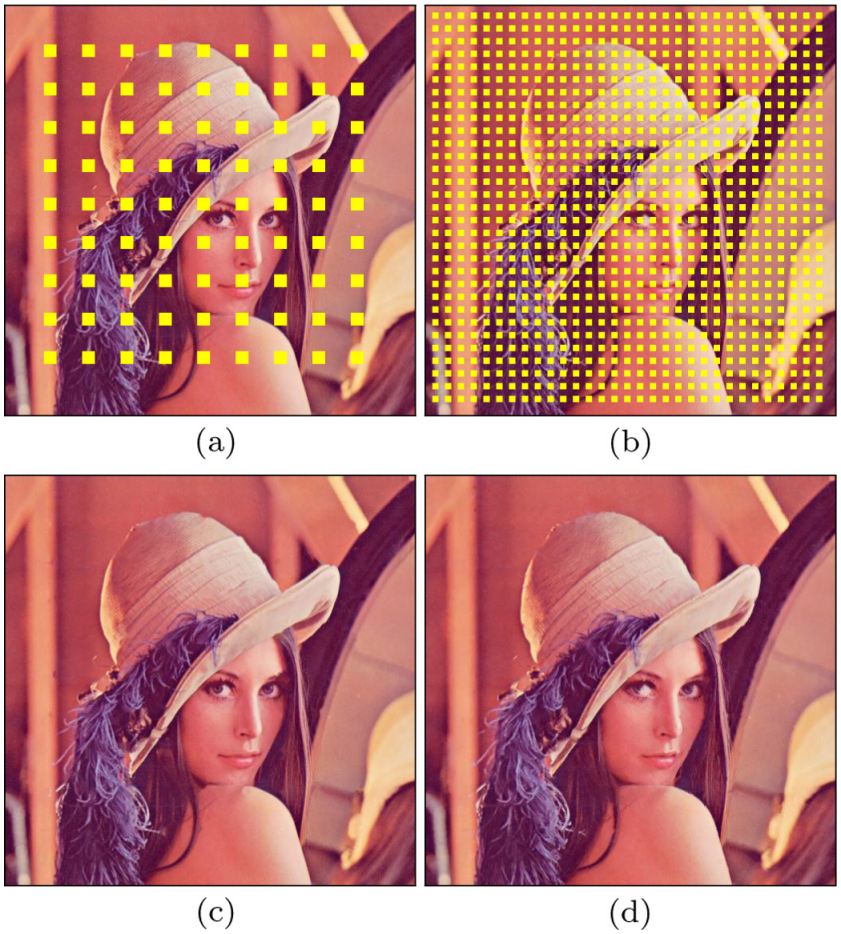


Fig. 14. Inpainting of consecutive block losses. (a, b) Degraded images of *Lena*. (c, d) Our results (PSNR = 36.35, 33.95).

column) which confirms that the proposed approach is computationally fast and noticeably time efficient.

3.4. Computational cost

Depending on the type and ratio of the damaged areas, our algorithm requires about 0.35–0.75 s to accomplish its inpainting task on a 512×512 color image.

The consumed computational times for the tested images are listed in the last column of Table 2. We can see that the average required time for these images is about 0.5 s. This means that the proposed method is capable of achieving good inpainting results (in terms of quality) with very low computational cost. To the best of our knowledge, this is one of the fastest algorithms ever invented for image inpainting.

4. Conclusion

In this paper, we described a new image inpainting approach based on using the dominant orientation of local gradients. The algorithm first computes the gradient orientation of each pixel and composes the proposed orientation matrix. Then the inpainting process commences by propagating the information from the outside of the boundary regions toward the interior areas using this matrix.

In this work, a new formulation is introduced for propagating the information based on finding the dominant orientation of the local gradients. This formulation leads to sufficiently fast and efficient computations which yield better results in terms of both quality and time. The advantages of the proposed approach are: (1) it is simple in principle; (2) it is easy to implement; (3) it is qualitatively better; (4) it is computationally fast. The results of experiments on gray-scale and color images highlight the advantages of the proposed approach.

Our future work will consist of investigating other applications of the proposed technique including inpainting based image compression, error concealment, and image restoration.

References

[1] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, 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.
[2] C. Barnes, E. Shechtman, A. Finkelstein, D. Goldman, PatchMatch: a randomized correspondence algorithm for structural image editing, ACM Trans. Graph.-TOG 28 (3) (2009) 24.
[3] Y. Pritch, E. Kav-Venaki, S. Peleg, Shift-map image editing, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 151–158.
[4] H. Igehy, L. Pereira, Image replacement through texture synthesis, Proceedings, International Conference on Image Processing, 1997, vol. 3, IEEE, 1997, pp. 186–189.
[5] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1) (1992) 259–268.
[6] X. Zhang, F. Ding, Z. Tang, C. Yu, Salt and pepper noise removal with image inpainting, AEU-Int. J. Electron. Commun. 69 (1) (2015) 307–313.
[7] M. Fornasier, Nonlinear projection recovery in digital inpainting for color image restoration, J. Math. Imag. Vis. 24 (3) (2006) 359–373.
[8] X. Huan, B. Murali, A.L. Ali, Image restoration based on the fast marching method and block based sampling, Comput. Vis. Image Underst. 114 (8) (2010) 847–856.
[9] D. Liu, X. Sun, F. Wu, Inpainting with image patches for compression, J. Vis. Commun. Image Represent. 23 (1) (2012) 100–113.

[10] Z. Xiong, X. Sun, F. Wu, Block-based image compression with parameter-assistant inpainting, IEEE Trans. Image Process. 19 (6) (2010) 1651–1657.
[11] C. Wang, X. Sun, F. Wu, H. Xiong, Image compression with structure-aware inpainting, in: Proceedings, 2006 IEEE International Symposium on Circuits and Systems, 2006, ISCAS 2006, IEEE, 2006, pp. 4–pp.
[12] S. Masnou, J.-M. Morel, Level lines based disocclusion, in: Proceedings, 1998 International Conference on Image Processing, 1998, ICIP 98, IEEE, 1998, pp. 259–263.
[13] F. Malgouyres, F. Guichard, Edge direction preserving image zooming: a mathematical and numerical analysis, SIAM J. Numer. Anal. 39 (1) (2001) 1–37.
[14] K.-H. Jung, J.-H. Chang, C. Lee, Error concealment technique using projection data for block-based image coding, in: Visual Communications and Image Processing'94, International Society for Optics and Photonics, 1994, pp. 1466–1476.
[15] Y. Wang, Q.-F. Zhu, Error control and concealment for video communication: a review, Proc. IEEE 86 (5) (1998) 974–997.
[16] K. Patwardhan, G. Sapiro, M. Bertalmío, et al., Video inpainting under constrained camera motion, IEEE Trans. Image Process. 16 (2) (2007) 545–553.
[17] A. Mosleh, N. Bouguila, A.B. Hamza, Automatic inpainting scheme for video text detection and removal, IEEE Trans. Image Process. 22 (11) (2013) 4460–4472.
[18] A. Mosleh, N. Bouguila, A. Hamza, Bandlet-based sparsity regularization in video inpainting, J. Vis. Commun. Image Represent. 25 (5) (2014) 855–863.
[19] A. Bugeau, M. Bertalmío, V. Caselles, G. Sapiro, A comprehensive framework for image inpainting, IEEE Trans. Image Process. 19 (10) (2010) 2634–2645.
[20] M. Bertalmio, A.L. Bertozzi, G. Sapiro, Navier-stokes, fluid dynamics, and image and video inpainting, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, CVPR 2001, vol. 1, IEEE, 2001, pp. I–355.
[21] T.F. Chan, J. Shen, Nontexture inpainting by curvature-driven diffusions, J. Vis. Commun. Image Represent. 12 (4) (2001) 436–449.
[22] T.F. Chan, J. Shen, H.-M. Zhou, Total variation wavelet inpainting, J. Math. Imag. Vis. 25 (1) (2006) 107–125.
[23] C. Guillemot, O. Le Meur, Image inpainting: overview and recent advances, IEEE Sign. Process. Mag. 31 (1) (2014) 127–144.
[24] A.A. Efros, T.K. Leung, Texture synthesis by non-parametric sampling, The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, IEEE, 1999, pp. 1033–1038.
[25] A. Criminisi, P. Pérez, K. Toyama, Region filling and object removal by exemplar-based image inpainting, IEEE Trans. Image Process. 13 (9) (2004) 1200–1212.
[26] A. Wong, J. Orchard, A nonlocal-means approach to exemplar-based inpainting, in: ICIP 2008, 15th IEEE International Conference on Image Processing, 2008, IEEE, 2008, pp. 2600–2603.
[27] Z. Xu, J. Sun, Image inpainting by patch propagation using patch sparsity, IEEE Trans. Image Process. 19 (5) (2010) 1153–1165.
[28] V. Kumar, J. Mukhopadhyay, S.K.D. Mandal, Modified exemplar-based image inpainting via primal-dual optimization, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2015, pp. 116–125.
[29] V. Kumar, J. Mukherjee, S.K.D. Mandal, Image inpainting through metric labeling via guided patch mixing, IEEE Trans. Image Process. 25 (11) (2016) 5212–5226.
[30] N. Komodakis, G. Tziritas, Image completion using efficient belief propagation via priority scheduling and dynamic pruning, IEEE Trans. Image Process. 16 (11) (2007) 2649–2661.
[31] K. He, J. Sun, Image completion approaches using the statistics of similar patches, IEEE Trans. Pattern Anal. Mach. Intell. 36 (12) (2014) 2423–2435.
[32] V. Kumar, J. Mukherjee, S.K.D. Mandal, Combinatorial exemplar-based image inpainting, in: International Workshop on Combinatorial Image Analysis, Springer, 2015, pp. 284–298.
[33] V.K. Alilou, F. Yaghmaee, Exemplar-based image inpainting using SVD-based approximation matrix and multi-scale analysis, Multimed. Tools Appl. (2016) 1–22, http://dx.doi.org/10.1007/s11042-016-3366-6.
[34] F. Benzarti, H. Amiri, Repairing and inpainting damaged images using diffusion tensor, CoRR abs/1305.2221. URL <http://arxiv.org/abs/1305.2221>.
[35] F. Bornemann, T. März, Fast image inpainting based on coherence transport, J. Math. Imag. Vis. 28 (3) (2007) 259–278.
[36] D. Tschumperlé, Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's, Int. J. Comput. Vis. 68 (1) (2006) 65–82.
[37] A. Telea, An image inpainting technique based on the fast marching method, J. Graph. Tools 9 (1) (2004) 23–34.
[38] M.V. Venkatesh, S.-c. S. Cheung, J. Zhao, Efficient object-based video inpainting, Pattern Recogn. Lett. 30 (2) (2009) 168–179.
[39] M. Collins, J. Zhang, P. Miller, H. Wang, Full body image feature representations for gender profiling, in: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2009, pp. 1235–1242.
[40] T. Kobayashi, A. Hidaka, T. Kurita, Selection of histograms of oriented gradients features for pedestrian detection, in: Neural Information Processing, Springer, 2008, pp. 598–607.
[41] L. Nanni, A. Lumini, Descriptors for image-based fingerprint matchers, Expert Syst. Appl. 36 (10) (2009) 12414–12422.
[42] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, CVPR 2005, vol. 1, IEEE, 2005, pp. 886–893.
[43] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[44] K. Zhang, X. Gao, D. Tao, X. Li, Single image super-resolution with non-local means and steering kernel regression, IEEE Trans. Image Process. 21 (11) (2012) 4544–4556.
[45] V.K. Alilou, F. Yaghmaee, Application of GRNN neural network in non-texture image inpainting and restoration, Pattern Recogn. Lett. 62 (2015) 24–31.
[46] M. Melgosa, Testing CIELAB-based color-difference formulas, Color Res. Appl. 25 (1) (2000) 49–55.
[47] B.-Y. Wong, K.-T. Shih, C.-K. Liang, H.H. Chen, Single image realism assessment and recoloring by color compatibility, IEEE Trans. Multimed. 14 (3) (2012) 760–769.
[48] Wikipedia, Ycbcr — Wikipedia, the Free Encyclopedia, 2017. URL <https://en.wikipedia.org/wiki/YCbCr>.
[49] J. Dahl, P.C. Hansen, S.H. Jensen, T.L. Jensen, Algorithms and software for total variation image reconstruction via first-order methods, Numer. Algorithms 53 (1) (2010) 67–92.
[50] M.P. Eckert, A.P. Bradley, Perceptual quality metrics applied to still image compression, Sign. Process. 70 (3) (1998) 177–200.
[51] A. Chalmers, A. McNamara, S. Daly, K. Myszkowski, T. Troscianko, Image quality metrics.
[52] N. Ponomarenko, F. Battisti, K. Egiazarian, J. Astola, V. Lukin, Metrics performance comparison for color image database, in: Fourth International Workshop on Video Processing and Quality Metrics for Consumer Electronics, vol. 27, 2009.
[53] P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, T. Wiegand, Depth image-based rendering with advanced texture synthesis for 3-D video, IEEE Trans. Multimed. 13 (3) (2011) 453–465.
[54] X. Li, M.T. Orchard, Novel sequential error-concealment techniques using orientation adaptive interpolation, IEEE Trans. Circ. Syst. Video Technol. 12 (10) (2002) 857–864.
[55] B. Chung, C. Yim, Hybrid error concealment method combining exemplar-based image inpainting and spatial interpolation, Sign. Process.: Image Commun. 29 (10) (2014) 1121–1137.
[56] A. Phadikar, S.P. Maity, ROI based error concealment of compressed object based image using QIM data hiding and wavelet transform, IEEE Trans. Consum. Electron. 56 (2) (2010) 971–979.