Mathematical Problems of Computer Science 44, 67--76, 2015.

Modification of Exemplar-Based Inpainting Algorithm for Mobile Devices With of Patch Offsets

Vahan V. Gevorgyan[1], Gevorg A. Karapetyan[2] and Hakob G. Sarukhanyan[2]

[1] Russian-Armenian University,
[2]Institute for Informatics and Automation Problems of NAS RA
e-mail: vahangev8@gmail.com, hakop@ipia.sci.am

Abstract

Inpainting (completion) of digital images is the process of filling in an unknown region with information from the known region of the image. Due to rise of mobile technologies, there is demand on usage of inpainting algorithms on mobile devices for applications such as object removal, image restoration, etc. Exemplar-based inpainting is one of the most popular and efficient inpainting algorithms, but, however, it is slow enough for mobile implementation. The intention of this paper is to develop a high performance inpainting algorithm applicable for mobile devices. We introduce a modification of the algorithm using the statistics of similar patch offsets. This approach reduces the computation time at about 10-30 times, which makes the algorithm work interactive even on mobile devices. The paper includes those experiment results and comparison of the developed algorithm with the exemplar-based inpainting algorithm and showed the advantages of our method on various images.

Keywords: Image inpainting, Image completion, Exemplar-based algorithm.

1. Introduction

Image completion is the process of recovering of missing or damaged parts of an image in such a way that the result will be visually plausible for an observer (e.g., see Fig. 1). Although the problem of image completion states very simple, the task of actually trying to solve it is far from being an easy thing. This problem is also referred as image inpainting. Origin of the term inpainting comes from art restoration, and it is also called retouching there. Starting from the Renaissance, medieval pictures were restored to bring them “up to date" by filling in any gaps.

During the development of digital images there also arose many situations in which an image should be inpainted. For example, recovering old photographs, removing unnecessary objects from an image or removing text, etc.

Fig. 1. Example of image inpainting.

Ideally, any algorithm that is solving the image completion problem should meet the following requirements [1]: (1) it should be able to complete complex natural images successfully; (2) it should be able to handle images with possibly large missing parts; (3) it should work in a automatic manner, i.e., without intervention from the user.

All these requirements above make the image completion, in general, a very challenging problem. The inpainting algorithms can be generally categorized into two main groups: diffusion-based and exemplar-based.

Diffusion-based approach fills holes in images by propagating linear structures (which are called isophotes in the inpainting literature) from the known region into the missing region via the process of diffusion. This approach was first introduced by Bertalmio *et. al* in [2]. Partial Differential Equation (PDE) usually models that problem, so sometimes it is also called a PDE-based approach. These methods are local in the sense that they use only neighboring pixels during the iteration. Drawback of that technique is that the diffusion process introduces some blur and it becomes noticeable when filling regions are large. When the region to be inpainted is small, this approach provides good results, but it is not convenient to use it when the missing region is large.

In 2003 Criminisi *et. al* introduced an exemplar-based algorithm [3]. Since then the exemplar-based approaches begin to develop and all state of the art algorithms are mainly exemplar-based. These techniques unlike the diffusion-based ones are non-local. It means that to determine the value of a current pixel the whole image might be scanned. Exemplar-based approaches fill in the missing region patch by patch. Where for target patch (patch from missing region) algorithm searches for a source patch (a patch from known region that is most similar to the target patch).

Matching patches between two images (regions from image) is also known as nearest-neighbor field problem. Computation of nearest neighbor filed is used in various computer vision problems such as image completion, retargeting and reshuffling. Suppose we have two images (regions) A and B, we should find for every patch in A a similar patch in B (in our case A is the source region and B is the target region). As computing the exact nearest-neighbor field is very expensive, algorithms calculate the approximate nearest neighbor. There are two approaches for

solving that problem tree-based (such as kd-trees and their modifications) and PatchMatch [4] (which is used in Photoshop). Tree-based methods use candidates (patches from B) distribution characteristics and according to this organize the searching space. PatchMatch does not use the candidate distributions and it uses the characteristic of queries (patches from A) that they are dependent. PatchMatch algorithm outperforms traditional tree-based algorithms. In [5] authors present propagation-assisted kd-trees, which uses both the distribution of the candidates and the dependency of the queries in their algorithm. This algorithm outperforms PatchMatch about 10-20 times.

In this paper we use the statistics of similar patches [6] (which we will describe in Section 2), to improve the original exemplar-based inpainting algorithm. Due to this modification, we improve the efficiency of the algorithm at about 10-30 times. What about the quality of a result? In some cases our modification gives better results because we don't consider unnecessary patches but in some cases our approach fails because the number of patches it considers is limited. We will show the comparison of algorithms in Section 5.

2. Exemplar-Based Algorithm

For an input image I, a user selects a target region Ω, which should be removed and inpainted. The contour of the target region is denoted by $\delta\Omega$. This contour is also called as "fill front" and it evolves inward the missing region during the algorithm. The supporting region from where the algorithm will take information, which is called a source region and is denoted by Φ, should be specified. By default it is defined as a whole image minus the target region ($\Phi = I - \Omega$). This algorithm is fragment-based, which means that it fills the missing area by patches not by pixels. The default size of a patch that authors suggest is 9×9 pixels, but this parameter may be changed or specified by the user, depending on the image size and color features. After all input parameters are set the remaining part of the algorithm is automatic and consists of the following steps:

1. Detection of boundary points, which constitute fill front and calculation of priority function for that points;
2. Selecting a patch for a point with the highest priority value (target patch);
3. Searching for the most similar patch to that patch (source patch);
4. For unknown pixels from the target patch copy color values of corresponding pixels from the source patch;
5. Update confidence term (which participate in priority function) values for filled pixels.

The algorithm iterates that steps until there is no unfilled pixels left. For every boundary point, (point between the target and the source regions) priority function is computed and it has the following representation:

$$P(p) = C(p)D(p), \forall p \in \delta\Omega,$$

where $C(p)$ stands for confidence term, D(p) – data term and they have the following representation:

$$C(p) = \frac{\sum_{q \in \Psi p \cap (I-\Omega)} C(q)}{|\Psi p|}, \qquad D(p) = \frac{\left|\nabla I_P^{\perp} \cdot n_p\right|}{\alpha},$$

where $|\Psi p|$ is the area of a patch Ψp centered at p, α is a normalization factor (its value is 255 for a typical grayscale image), n_p is a unit vector orthogonal to the front $\delta\Omega$ in the point p and $\nabla I_P^{\perp}$- the isophote (a line of equal intensity value) at p.

The function $C(p)$ initially is set to the following values: $C(p) = 0, \forall p \in \Omega$ and $C(p) = 1$, $\forall p \in I - \Omega$. This term shows how much the information surrounding the pixel p is reliable . The idea is to give preference to those patches, which have more of their pixels already filled and the earlier pixel filled (or it was never part of the target region) the more reliable it is.

The data term $D(p)$ shows the strength of an isophote hitting the front $\delta\Omega$ at point p. This term gives preference to those patches where an isophote "flows" into. Due to it, linear structures are encouraged to be inpainted first and therefore propagate in a plausible way into the target region.

After selecting the target patch, the algorithm starts to search for the most similar patch in the supporting region (source region). Patches similarity is calculated by the following formula:

$$\Psi_q = \arg\min_{\Psi_l \in \Phi} d(\Psi_l, \Psi_p),$$

where Ψ_q is the source patch, Ψ_p – the target patch, d is a function of distance (similarity) between two patches. Authors used the sum of squared difference (SSD) of known pixel color values from the target patch, with their corresponding pixels from the source patch, as the distance function.

Then for each unfilled pixel in Ψ_p, the algorithm copies the pixel value from its corresponding position inside Ψ_q. Fig. 2 illustrates the inpainting process described above.

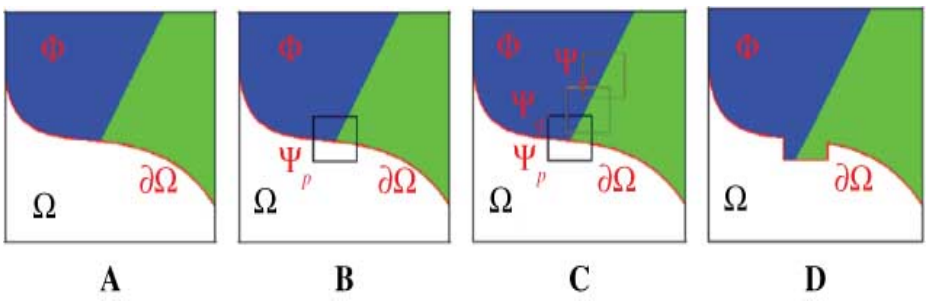


Fig. 2. Inpainting process.

Finally, the confidence values for the already filled pixels should be updated. The confidence term $C(p)$ is updated as follows: $C(t) = C(p) \ \forall t \in \Psi p \cap \Omega$.

3. Statistics Calculation

Before starting inpainting, we calculate the statistics of patch offsets. We scan the whole image and for every patch centered at point p_1, we search for the other patch from the image with center p_2, which is most similar to it. Patch similarity is computed by the sum of squared differences (SSD).

Then we compute the offset of that patch (the coordinate difference). Suppose $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. Patch offset r is: $r(p_1) = (x_1 - x_2, y_1 - y_2)$. Solving this problem with brute force for image with size s will take $O(s^s)$ time, no need to mention how slow it is. In this paper for finding the most similar patch, we use the propagation-assisted kd-trees [5], which is state of the art algorithm for finding the approximate nearest neighbor. The approximation of that algorithm does not influence the results much, because we will calculate the statistics.

Then we calculate the following statistics: for each offset, we compute the quantity of how many times it occurs. We select K most frequent offsets and fix them (O denote that set). In the future we will use this set in our inpainting algorithm. In our experiments, we use the value of K from the range 300-450.

4. Exemplar-Based Algorithm for Mobile Devices

We apply the statistics of patch offsets to improve the efficiency of the exemplar-based algorithm. We consider that the source region is the whole image (tough we can apply our approach in other cases also). While searching for the most similar patch, instead of scanning the whole source region, we run only through the K most frequent offsets and consider only their corresponding patches.

$$\Psi_q = \arg \min_{l \in ASR(p)} d(\Psi_l, \Psi_p),$$

Where $ASR(p)$ is the allowable source region for point p and is the following set of points:

$$ASR(p) = \{q = p + r \mid r \in O \text{ and } q \in \phi\}$$

Due to this optimization at every searching iteration, we scan not greater than only K patches, which is much less than scanning the whole source region. This approach improved the speed by 10-30 times on the average. In the next section, we will show our experimental results.

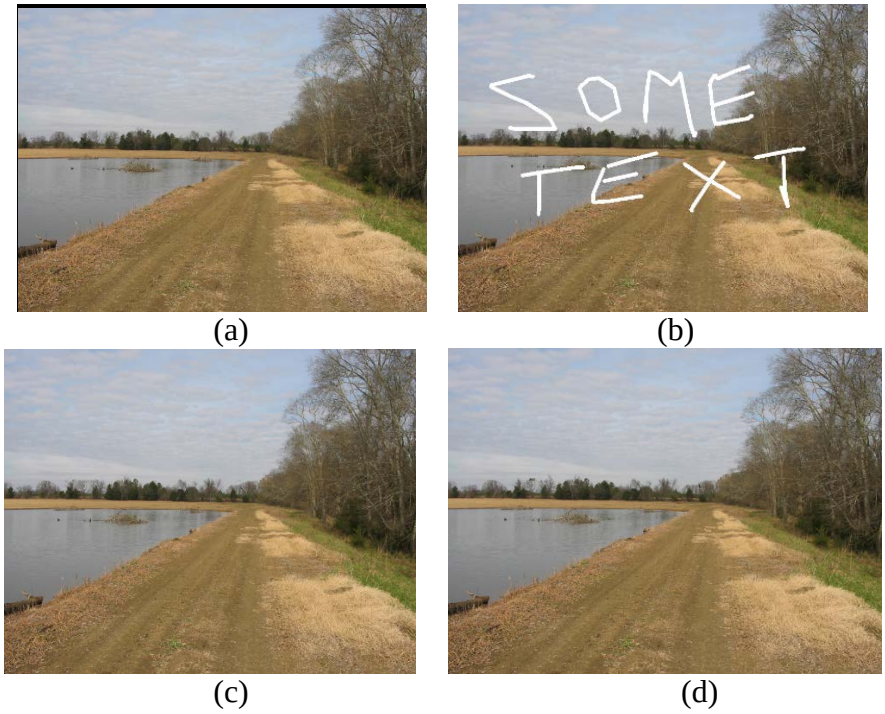
5. Implementation and Experimental Results

We implement Criminisi's exemplar-based algorithm [3] and our modification on C++ using OpenCV library [13]. We also implement propagation-assisted kd-trees for computing patch offsets statistics. Then we transform our code on Android devices using Android NDK [14]. The execution time of our approach on mobile devices for not very large sized images is competitive. For large images in mobile implementation, we scale them to a smaller size and pass this smaller image to completion algorithm. After completion, we scale the image back to the original size. In mobile implementation, we use the value of K from range 200-300 and the execution time is about 0.1 minute. Here are some examples of our experiments on PC, as well as the comparison of algorithms.

(a) (b)

(c) (d)

Fig.3. Example 1. Comparison of exemplar-based inpainting and our approach. Image size is 300x400. Target region forms 23.1525 % of whole image. Patch size is 7x7 pixels. a) the original image , b) inpainted image, c) result of Criminisi's algorithm, d) our result



(a) (b)

(c) (d)

Fig. 4. Example 2. Comparison of exemplar-based inpainting and our approach. Image size is 750x563. Target region forms 3.48988 % of whole image. Patch size is 7x7 pixels. a) the original image , b) inpainted image, c) result of Criminisi's algorithm, d) our result

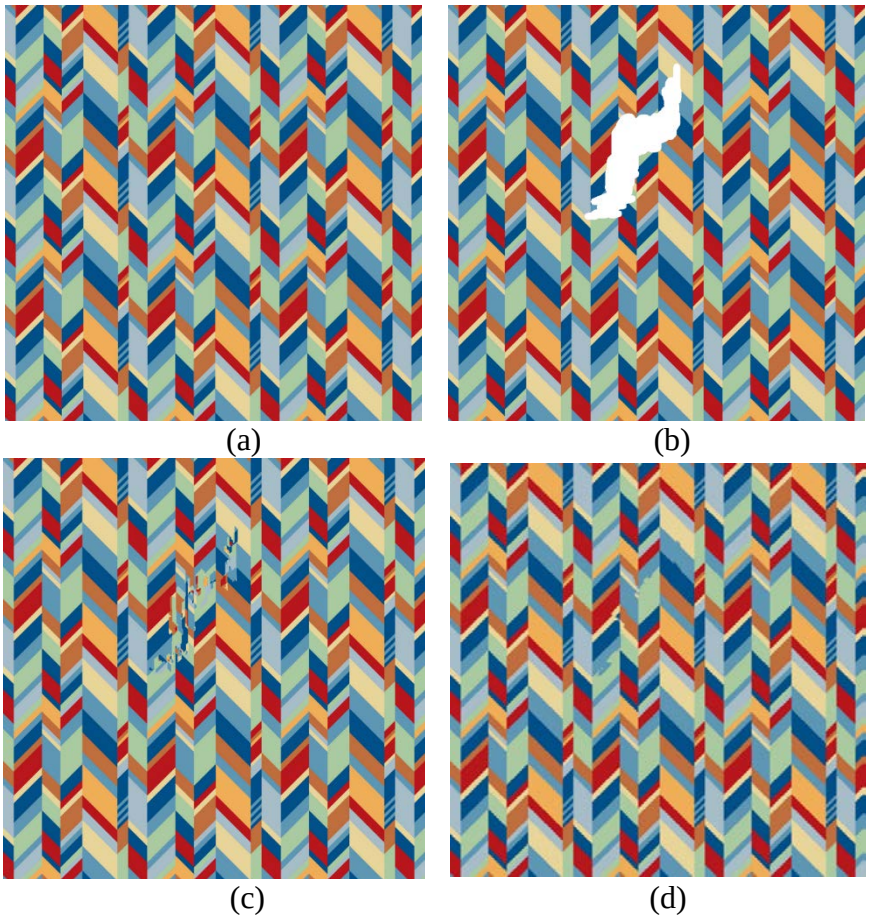


Fig. 5. Example 2. Comparison of exemplar-based inpainting and our approach. Image size is 1000x1000. Target region forms 2.9283 % of whole image. Patch size is 7x7 pixels. a) the original image , b) inpainted image, c) result of Criminisi's algorithm, d) our result.

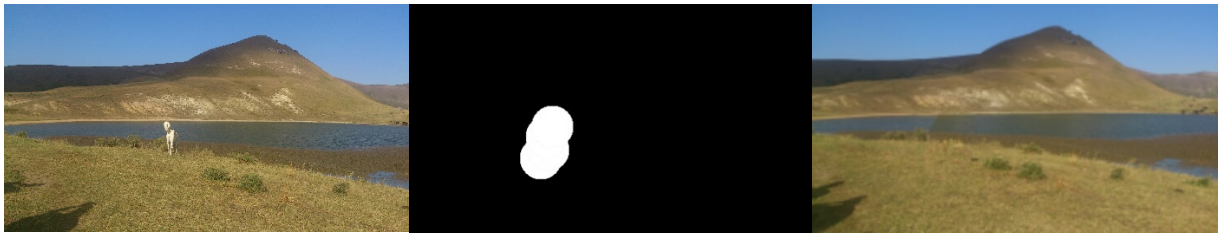
As examples above show, our algorithm gives better results when the background of missing region is uniform and when the image consists of some pattern (see Fig. 3 and Fig. 5, respectively). Both algorithms manage well with the case of removing some text from the image (see Fig. 4). The table below shows the execution time of algorithms and similarity measure of inpainted results on the mentioned examples. We calculate PSNR and SSIM similarity measures for inpainted images, where **peak signal-to-noise ratio (PSNR)** is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation [15]. The Structural Similarity (SSIM) index is a method for measuring the similarity between two images. Detailed description is given in [16]. We calculate SSIM for each RGB channel.

Table 1. Comparison of exemplar-based inpainting and our approach.

Results	Time(mins)	PSNR	SSIM(RED)	SSIM(GREEN)	SSIM(BLUE)
Image1(Criminisi)	2.53333	9.85056	79.2155%	79.2566%	79.4306%
Image1(Our)	0.416667	9.76338	79.4731%	79.5799%	79.8034%
Image2(Criminisi)	6.08333	41.5729	98.7647%	98.8749%	98.8098%
Image2(Our)	0.266667	40.2365	98.5848%	98.7253%	98.6343%
Image3(Criminisi)	25.3	28.3332	98.3342%	98.4563%	98.6232%
Image3(Our)	0.65	36.9047	99.5476%	99.5839%	99.6551%

6. Conclusion

In this paper, we introduce patch offsets statistics application to the exemplar-based inpainting algorithm. We implement this approach and the original exemplar inpainting algorithms and compare them. As our experiments shown, our approach is at average 10-30 times faster and even more when image sizes become larger. Also as examples from the previous section show, in some cases our algorithm gives better results. We implement our algorithm in android application for mobile devices and due to our speedup a user can interactively inpaint his images. Fig. 6 shows the inpainted result from our mobile application.



(a) **(b)** **(c)**

Fig. 6. Example of mobile inpainting. Image size is 3920x2206. Execution time is 0.1 minute. Patch size is 7x7 pixels. a) the original image , b) inpainted mask, c) result.

References

[1] N. Komodakis and G. Tziritas, “Image completion using efficient belief propagation via priority scheduling and dynamic pruning,” *IEEE Trans. Image Process.* , vol. 16, no. 11, pp. 2649–2661, 2007.

[2] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester. “Image inpainting”, *Proceedings SIGGRAPH 2000, Computer Graphics Proceedings*, pp. 417—424, 2000.

[3] A. Criminisi, P. Perez and K. Toyama, “Region filling and object removal by exemplar-based image inpainting,” *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[4] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, “PatchMatch: A randomized correspondence algorithm for structural image editing,” *in Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, pp. 1–8, 2009

[5] K. He and J. Sun, “Computing nearest-neighbor fields via propagation-assisted KD-trees,” *in Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 111–118, 2012

[6] K. He and J. Sun, “Image completion approaches using the statistics of similar patches”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2423-2435, 2014

[7] G.Karapetyan, H.Sarukhanyan and S. Agaian “Robust digital image inpainting algorithm in the wireless environment”, *SPIE Proceedings*, vol. 9120, 8 pages, 2014 doi:10.1117/12.2049942

[8] G.Karapetyan and H.Sarukhanyan. “Automatic detection and concealment of specular reflections for endoscopic images” *Ninth International Conference Computer Science*

and Information Technologies, Revised Selected Papers, IEEE Explore, 8 pages, 2013, 10.1109/CSITechnol.2013.6710353

[9] G.Karapetyan and H.Sarukhanyan. "Automatic detection and concealment of specular reflections for endoscopic images", *Proceedings of International Conference Computer Science and Information Technologies*, pp.197-200, 2013.

[10] G.Karapetyan and H.Sarukhanyan. "Concealment of targeted regions in digital images on mobile devices", *Transactions of IIAP NAS RA, Mathematical Problems of Computer Science*, vol. 40, pp. 68--75, 2013.

[11] G.Karapetyan, "Modification of FSE method based on coefficients of homogeneity", *Transactions of IIAP NAS RA,Mathematical Problems of Computer Science* , vol. 35, pp. 109-115, 2011.

[12] G.Karapetyan and H.Sarukhanyan, "On a modification of the frequency selective extrapolation method", *Information Models and Analyses*, vol.2, pp.139-145, 2012.

[13] [Online]. Available: http://opencv.org/

[14] [Online]. Available: http://developer.android.com/intl/ru/tools/sdk/ndk/index.html

[15] [Online]. Available:http://www.ni.com/white-paper/13306/en/

[16] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

Submitted 04.09.2015, accepted 26.11.2015

Շարժական սարքերի համար նմուշների հիման վրա պատկերների inpainting ալգորիթմի կատարելագործումը նման բլոկների տեղաշարժմամբ

Վ. Գևորգյան, Գ. Կարապետյան և Հ. Սարուխանյան

Ամփոփում

Թվային պատկերների ներկումը (ավարտումը) անհայտ տիրույթի լցման պրոցես է ինֆորմացիայով հայտնի տիրույթից: Շարժական սարքերի զարգացման շնորհիվ կա ներկման ալգորիթմների պահանջարկ շարժական սարքերի վրա այնպիսի հավելվածներում, ինչպիսիք են օբյեկտների հեռացումը, պատկերների վերականգնումը և այլն: Նմուշների հիման վրա ներկման ալգորիթմն ամենահայտնի և արդյունավետ ներկման ալգորիթմներից է, սակայն այն բավականին դանդաղ է շարժական սարքերի համար: Այս հոդվածի նպատակն է մշակել բարձր կատարողականությամբ ներկման ալգորիթմ, որը կիրառելի կլինի նաև շարժական սարքերի համար: Մենք ներկայացնում ենք ալգորիթմի ձևափոխությունը՝

օգտագործելով նման բլոկների շեղումների վիճակագրությունը: Այս մոտեցումը նվազեցնում է հաշվարկների ժամանակը մոտավորապես 10-30 անգամ, ինչը դարձնում է ալգորիթմի աշխատանքը ինտերակտիվ նույնիսկ շարժական սարքերի վրա: Հոդվածը ներառում է փորձնական արդյունքները և ալգորիթմների համեմատությունը, ինչպես նաև ցույց է տրված մեր մշակած մեթոդի առավելությունը տարբեր պատկերների վրա:

Модификация inpainting алгоритма для мобильных устройств с использованием сдвигов похожих блоков

В. Геворкян, Г. Карапетян и А. Саруханян

Аннотация

Закрашивание (завершение) цифровых изображений это процесс заполнения неизвестной области информацией с известной области изображения. Благодаря развитию мобильных технологий есть потребность в использовании алгоритмов закрашивания на мобильных устройствах для таких применений как удаление объектов, реставрация изображений и т.д. Алгоритм закрашивания основанный на экземплярах один из наиболее известных и эффективных алгоритмов закрашивания, но тем не менее он достаточно медленный для реализации на мобильных устройствах. Цель данной статьи разработать высоко эффективный алгоритм закрашивания применимый для мобильных устройств. Мы представляем модификацию алгоритма, используя статистику сдвигов похожих блоков. Данный подход сокращает вычислительное время около 10-30 раз, что делает работу алгоритма интерактивной даже на мобильных устройствах. В статье включены эти экспериментальные результаты, а так же сравнение разработанного алгоритма с алгоритмом закрашивания основанного на экземплярах и показано преимущество нашего метода на различных изображениях.