Effective Gradient Domain Object Editing on Mobile Devices

Yingen Xiong*, Dingding Liu*†, Kari Pulli*
*Nokia Research Center, Palo Alto, CA, USA
Email: {yingen.xiong, dingding.liu, kari.pulli}@nokia.com
† University of Washington, Seattle, WA, USA
Email: liudd@u.washington.edu

***Abstract*— We present a gradient domain object editing approach and its implementation for mobile devices. It can be used for creating a new composite image by removing, adding, and moving objects in an image. The approach can be divided into two parts: creation and editing of a new gradient vector field, and recovery of a new composite image from the new gradient vector field. In the first part, a new gradient vector field is created from the gradients of the source image, and then updated by inserting the new object gradients, by removing object gradients and filling removed areas with the gradients of best-fit patches found in other parts of the source image, or by combining these two processes when moving objects. In the second part, a divergence vector field is computed from the gradient vector field and used for a guidance vector to construct a Poisson equation. The new composite image is recovered from the gradient vector field by solving the Poisson equation with boundary conditions.**

Our approach can merge all regions in the picture seamlessly with smooth color transition for the whole picture. It can be used for large object removal and for filling the background of the removed object. The final composite image is a globally optimal solution. The approach is implemented and runs with good performance on mobile camera phones.

I. Introduction

A. Background

As computational power and memory of mobile devices increase, and larger touch screens are introduced, editing images directly on the same device as the image was taken becomes feasible and interesting. In this paper, we are interested in editing and producing high-quality images on camera phones. Users can insert other images into their pictures and remove any part which they do not want. A simple copying and pasting produces visible artificial edges in the seams between images, due to differences in camera responses or environmental illumination. Even in one image, different parts often have such differences. Efficient tools are needed to merge all regions together and create a high-quality composite image.

We present an object editing approach which we have implemented on camera phones. It can be used for adding objects into an image seamlessly, removing objects from an image and recovering the removed areas as if the image were captured without those objects, and moving objects to new positions. The editing operations are performed in the gradient domain. Comparing with operations in the pixel color domain, the approach can merge all parts together by reducing color differences between the merged parts, making seams minimally visible. This approach can be used for producing high-quality composite images.

B. Related Work

Our approach is related to two kinds of work: image blending and image hole filling. When we add objects into the source image, we need to blend the object image and the source image together and make the seams between these images minimally visible, which is related to image blending. When we remove objects, we empty the areas of removed objects in the source image, and then we need to recover these areas with reasonable visual quality, which is related to image hole filling or inpainting.

Image blending is used for merging images together and making seams between the images minimally visible. Current image blending approaches can be divided into two main classes: transition smoothing [1] and optimal seam finding [2] algorithms. Our work is more related to the previous.

Transition smoothing algorithms reduce the color differences between images to make seams invisible. Recently, gradient domain image blending algorithms [1] have been widely used in image blending and editing. In these algorithms, source images are merged in the gradient domain, yielding a gradient vector field. A composite image can be recovered from the gradient vector field by solving a Poisson equation. The second part of our approach belongs to this category. After object editing, we need to blend all regions together by reducing the color differences and hiding the seams between the regions.

Current algorithms for image hole filling in the literature can also be categorized into two main classes: inpainting and texture synthesis algorithms.

Image inpainting algorithms [3], [4] address the hole filling problems as image restoration. Image holes are filled by propagating linear structures into the target region via diffusion. Such algorithms are good for filling small image gaps. The main disadvantage of this category is that the diffusion process introduces blur, and it does not work well for filling large regions.

Texture synthesis algorithms [5], [6] seek to replicate texture given a small source sample of pure texture to fill large image regions. In this kind of algorithms, exemplar-based approaches [7], [8] are more interesting. They generate a new texture map by sampling and copying color values from the source image.

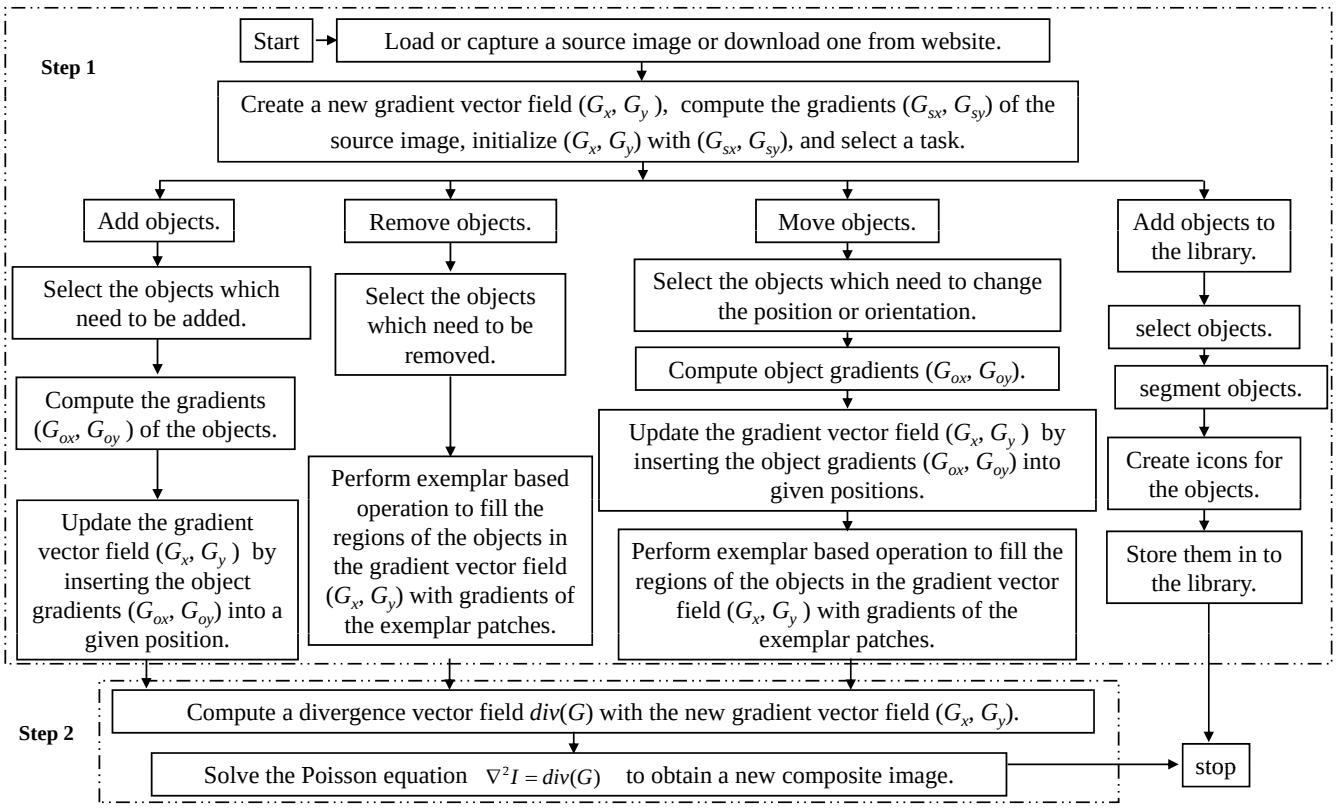

Fig. 1. Work flow of the gradient domain object editing approach.

The main advantage is that generating a new texture is quite cheap. The main problem is that the boundaries between image regions may still be visible after region filling. We address that problem by working in the gradient domain.

We combine these two kinds of hole filling algorithms in our work, so that we can use their advantages and avoid disadvantages. In order to fill large object regions, we use exemplar-based operations to find best-fit texture in the other parts of the source image. In order to solve the boundary problems, we fill the object regions not only in the pixel color domain but also in the gradient domain. Once the best-fit texture patches are found, we copy the pixel color values to fill the source image and the gradients to update the gradient vector field. A new composite image can be recovered from the gradient vector field by solving a Poisson equation, and we get a faster convergence in our iterative Poisson solver by using the filled source image as an initial estimation. We use the recovered composite image as our final result. In the following sections, we will describe the approach in detail.

II. Summary of The Approach

Figure 1 shows the work flow of the approach. It includes two steps. The first step is to create a new gradient vector field with the gradients of the source image and update it according to different editing operations. The second step is to recover a new composite image from the new gradient vector field.

The first step starts with getting a source image S which needs to be edited into memory. The image can be loaded from a disk, or captured with a camera, or downloaded from a website. We create a new gradient vector field (G_x, G_y) for a new composite image I_c. After computing the gradients of the source image (G_{sx}, G_{sy}), we initialize the new gradient vector field using the gradients of the source image. Then, we update the new gradient vector field wuth different editing operations.

For adding objects, we first select the objects to be added. The objects can be in the source image or in other images or in an object library. Then we compute the gradients of the objects (G_{ox}, G_{oy}). The new gradient vector field (G_x, G_y) can be updated by inserting the gradients of objects into given positions.

For removing objects, the regions $\Omega_0, \Omega_1, \ldots, \Omega_n$ of objects to be removed are determined by manual selection or using image segmentation algorithms. First we remove the gradients in these regions from the initial gradient vector field (G_x, G_y). Then each removed region $\Omega_i (i = 0, 1, ..., n)$ is filled with the gradients of exemplar patches found by the exemplar-based operation which is detailed in latter sections. Once all regions are filled, the update of the gradient vector field is completed.

For moving objects, we need to insert the objects into new areas and replace the original areas with best-fit texture found in the other parts of the source image. In order to do this, we combine object removal and object adding operations. Once the objects which need to be moved are selected, we specify the regions $\Omega_0, \Omega_1, \ldots, \Omega_n$ of these objects and compute their gradients. We insert them into the gradient vector field (G_x, G_y) with given positions for adding these objects into new areas. We also remove the gradients in the

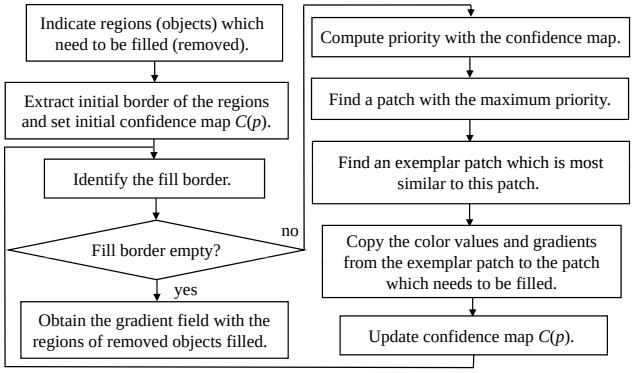

Fig. 2. Region filling operation.

specified regions. Each removed region is filled with gradients of exemplar patches found by the exemplar based operation. After all regions are filled, we obtain the updated gradient vector field (G_x, G_y).

After the new gradient vector field (G_x, G_y) is updated, the second step is to obtain the new composite image from the gradient vector field. We create a divergence field $div(G)$ from the gradient vector field and use it as guidance to construct a Poisson equation $\nabla^2 I = div(G)$. The new composite image I_c can be recovered from the gradient vector field (G_x, G_y) by solving the Poisson equation with boundary conditions.

Finally, users can segment objects from images and put them into an object library. During object editing, users can select objects from the library and add them into their pictures.

III. Object Editing in Gradient Domain

As mentioned above, the approach includes two parts, creating a new gradient vector field according to different editing operations and recovering a new composite image from the gradient vector field.

A. Create a New Gradient Vector Field

The object editing operations are performed in the gradient domain. We edit the gradients of the source image (G_{sx}, G_{sy}) with the gradients of the objects (G_{ox}, G_{oy}) to create a new gradient vector field (G_x, G_y). From the new gradient vector field, we can obtain a new composite image which is the result of object editing.

In order to do this, we first initialize the new gradient vector field with the gradients of the source image. Then we update it using different procedures according to different editing operations.

1) Object Removal: In object removal operation, the regions of objects are emptied after the objects are removed. We need to fill them to make them look "reasonable" to the human eye. The challenge is that we do not have ground-truth images for these regions. In order to solve this problem, we apply exemplar-based inpainting techniques in the pixel color and gradient domains to fill these regions.

Figure 2 shows the region filling operation. It proceeds as follows. For each pixel at the border of the region, we define a patch centered at the pixel and compute a priority value with the pixel information. A priority map can be created by the priority values of all pixels on the border. For the patch which has the maximum priority value, we search for a best-fit patch in the other parts of the image and fill all the empty pixels covered by the patch in the pixel color domain with the corresponding pixel values of the best-fit patch and in the gradient vector field with the corresponding gradients in the best-fit patch. After that, we extract a new border of the region in the updated image and repeat this process until the whole region is filled in both pixel colors and gradients. After all empty regions are filled, we can obtain a completed image and an updated gradient vector field. The details are described below.

Suppose that S_A is the whole image area, Ω is the region which needs to be filled (the empty region), Φ is other part of the image except Ω, i.e., $\Phi = S_A - \Omega$, $\delta\Omega$ is the border of region Ω. We assign a confidence value $C(p)$ to each pixel for computing filling priority. The confidence value $C(p)$ is initialize to zero if the pixel is in the empty region, otherwise, it is assigned to one.

As described above, for a pixel p on $\delta\Omega$, we define a patch Ψ_p centered at the pixel and compute its priority value $P(p)$ with

$$P(p) = R(p)F(p), \tag{1}$$

where

$R(p)$ is the confidence at pixel p;
$F(p)$ is feature information around pixel p.

The confidence $R(p)$ is related to the confidence values of all pixels in its patch Ψ_p. The more pixels in the footprint of this patch are already filled, the larger the confidence is at this pixel. $F(p)$ measures features around pixel p. The more features can be detected around pixel p, the larger is the value of $F(p)$.

We compute a priority value $P(p)$ for each pixel p on the border $\delta\Omega$ and create a priority map with the priority values of all pixels on the border. We search for a pixel p_m and its patch Ψ_{p_m} which has the maximum filling priority value $P_{max}(p_m)$. Once the pixel is found, we select its patch as the candidate for filling.

Once the pixel p_m with patch Ψ_{p_m} has been determined, we search for a patch $\Psi_q \in \Phi$ which is most similar to patch Ψ_{p_m}:

$$\Psi_q = \arg\min_{\Psi_i \in \Phi} S(\Psi_{p_m}, \Psi_i), \tag{2}$$

where $S(\Psi_{p_m}, \Psi_i)$ is a similarity measure, in our case the summed squared difference between two patches.

When all empty regions are processed, we obtain a filled image S_f and an updated gradient vector field (G_x, G_y).

2) Object Adding: For object adding, we need to update the gradient vector field (G_x, G_y) with the gradients of objects (G_{ox}, G_{oy}) which need to be added.

The objects can be selected from the source image or from other images captured with cameras, loaded from disks, or downloaded from websites. The object regions can be segmented with segmentation algorithms or by manually.

Once we obtain the object image S_o, we compute its gradients (G_{ox}, G_{oy}) and update the gradient vector field (G_x, G_y) by inserting the object gradients into a given position. As above, suppose that S_A is the whole area of the source image, Ω is the object region, and Φ is the rest of the image ($\Phi = S_A - \Omega$). We have

$$G_x = \begin{cases} G_{sx}, \forall p \in \Phi \\ G_{ox}, \forall p \in \Omega \end{cases}, G_y = \begin{cases} G_{sy}, \forall p \in \Phi \\ G_{oy}, \forall p \in \Omega \end{cases} \quad (3)$$

In the pixel color domain, we can also update the filled image S_f with the color values of the object image S_o.

$$S_f = \begin{cases} S, & \forall p \in \Phi \\ S_o, & \forall p \in \Omega \end{cases} \quad (4)$$

For each object to be added into the source image, we update the gradient vector field (G_x, G_y) and the source image S_f, until all objects are done.

3) Object moving: For object moving, the position of the object changes from one place to another in the image. In this case, we need to put the object to the new place, delete the original one, and recover the deleted region. We combine object removal and object adding operations to update the gradient vector field (G_x, G_y) and the filled image S_f

Once the object to be moved is selected, its region Ω_o can be segmented. We compute the gradients (G_{ox}, G_{oy}) of S_o and update the gradient vector field (G_x, G_y) by inserting the object gradients (G_{ox}, G_{oy}) into the new position Ω_1. The filled image is updated with equation 4.

The next step is to fill the original region Ω_o of the object. We empty the region by removing the object. After specifying the region border $\delta\Omega_o$, we compute priority $P(p)$ and patch Ψ_p for each pixel p on the border. We search in the other parts of the image for a patch Ψ_b which is the most similar to this patch. Then we update color values for the empty part of the patch in the source image and gradient values for the empty part of the patch in the gradient vector field (G_x, G_y).

For each object, we repeat this process. Finally, we obtain an updated image S_f and an updated gradient vector field (G_x, G_y).

These object editing operations can be used all together or separately for a source image. After the editing operations, we can obtain a final updated image S_f and a final updated gradient vector field (G_x, G_y). They will be used for recovering a new composite image I_c in the next step.

B. Recovering a New Composite Image

Once the new gradient vector field (G_x, G_y) is created, the next step is to recover a new composite image I_c from the gradient vector field. In order to do this, we need to create a divergence field and construct a Poisson equation using the divergence field as guidance. After solving the Poisson equation, we can recover the new composite image. We could solve the Poisson equation directly from the divergence field calculated from the gradients, but we get much faster convergence by using the color values filled from the patches and copied by pasting the insterted object as a starting point.

1) Poisson Equation Construction: In order to construct a Poisson equation, we compute divergence field $div(G)$ from the gradient vector field (G_x, G_y). Suppose $I(x, y)$ is our composite image. We use the divergence as a guidance field to construct the Poisson equation as

$$\nabla^2 I(x, y) = div(G), \quad (5)$$

where

∇^2 is the Laplacian operator,
$\nabla^2 I(x, y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}$;
div is the divergence operator;
$div(G)$ is the divergence field $\frac{\partial G_x}{\partial x} + \frac{\partial G_y}{\partial y}$.

2) Using Boundary Conditions: Equation 5 is a linear partial differential equation. In order solve this equation, we must first specify boundary conditions. We use the Dirichlet boundary condition [1].

Finally, the new composite image I_c can be recovered from the gradient vector field (G_x, G_y) by solving the Poisson equation 5 with boundary conditions. We use the result as our final composite image.

IV. Applications and Result Analysis

The gradient domain object editing approach is implemented on camera phones. It has been tested and applied to object editing for different kinds of pictures. Good performance has been obtained. In this section, we present some example applications and results which are obtained by running the approach on a Nokia N900 phone with an ARM Cortex-A8 600 MHz processor, 256 MB RAM, 768 MB virtual memory, and a 3.5 inch touch-sensitive widescreen display. It can also be run on other mobile devices. The results are satisfying.

In order to simplify object editing on mobile phones, we have designed a simple interactive user interface for object selection. The user first specifies a region that contains an object to be edited by clicking on the top-left and bottom-right corners of the bounding box for that object. Then the object will be cut out automatically and is ready for editing operations. During the cut-out process, the pixels in the bounding box are first oversegmented by the meanshift algorithm, which can better preserve the edge information than other low-level segmentation algorithms, such as watershed or superpixels. Then these small regions are clustered according to similarity measures. When the background is complex, users need to further specify the foreground by strokes. Our algorithm avoids the expensive global optimization and requires less user input. With the interface, users can select the editing objects very conveniently, especially on touch screens.

Figure 3 shows an application of the approach to a picture captured in a swimming pool scene. The top left shows the source image. In object editing, we can move the selected object from one place to another. The result is shown on the top right of Figure 3. We move the object at the bottom from right to left and recover the original area. Everything matches very well. We remove two selected objects at the bottom left and bottom right of the source image and recover the two

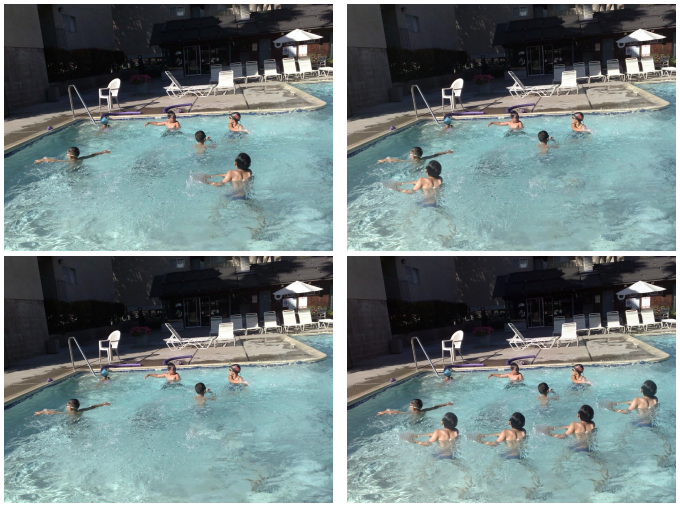

Fig. 3. Applications of object editing in a swimming pool scene.

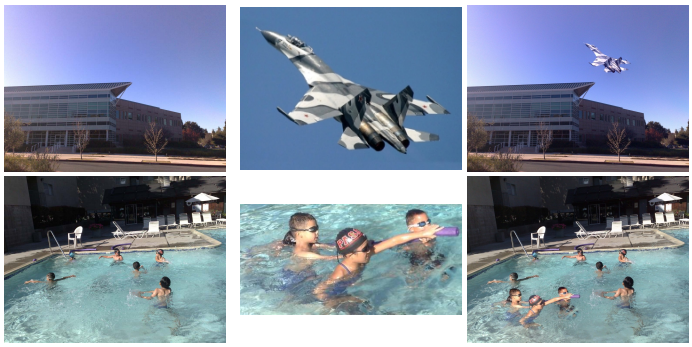

Fig. 4. Applications for adding objects from other pictures.

original areas. The result is shown on the bottom left of Figure 3. Finally, we add more objects selected in the picture and the result is shown at the bottom right of Figure 3. Again the objects match the scene very well.

Figure 4 shows applications of the approach for adding objects from other pictures. The first row of the figure shows the process of adding an airplane into a picture captured in outdoor scene. The left, middle, and right of this row shows the source image, the object image, and the result, respectively. From the result we can see that the approach can merge all regions seamlessly and the color matches perfectly. The second row shows the process of adding a large object from another picture. From left to right of this row, it shows the source image, the object image, and the result respectively. From the result we can see that the approach works well.

Figure 5 shows more applications of the approach to different scenes. The first row shows that the approach can be used for duplicating the tower and putting it to a given position. The second row shows that the approach can be used to remove the walking person in the scene and recover the removed area. The results are satisfying.

V. Conclusions and Discussion

We presented an approach for object editing in the gradient domain and its implementation on mobile devices. The method can be used for adding objects into a picture, deleting objects from a picture and recovering the deleted areas with best-fit scenes in the picture, and moving objects from one place to another inside the picture.

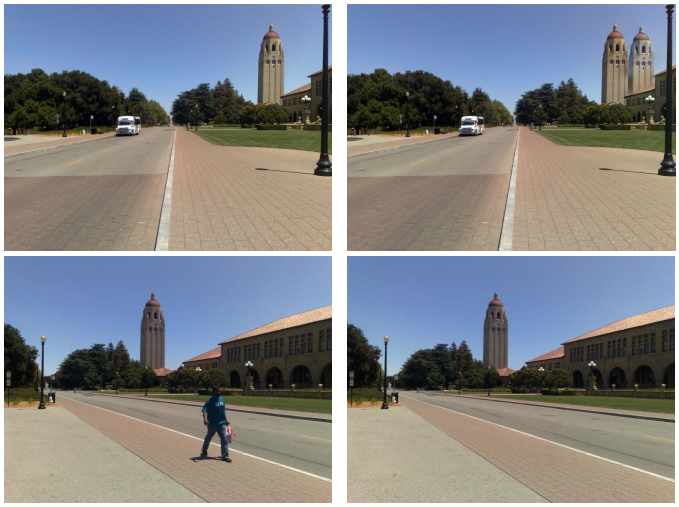

Fig. 5. Applications to other scenes for adding and removing objects.

We implemented the method on mobile devices and tested it with different images captured in different scenes. From the application of the swimming pool picture we can see that the approach can be used for adding, removing, and moving objects inside a picture and the matches between objects and the source image work very well. The removed areas can be fully recovered. From the applications of adding objects from other pictures we can see that the approach can also merge and blend the outside objects with the source image well. The result images can retain similar colors as the source images and have good color transitions. From the applications to other scenes we can see that the approach can be used for different kinds of pictures and good results can be obtained.

References

[1] Patrick Pérez, Michel Gangnet, and Andrew Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, 2003.

[2] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen, "Interactive digital photomontage," *ACM Trans. Graph*, vol. 23, no. 3, pp. 294–302, 2004.

[3] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester, "Image inpainting," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 417–424, 2000.

[4] Timothy K. Shih and Rong-Chi Chang, "Digital inpainting - survey and multilayer image inpainting algorithms," in *Proceedings of Third International Conference on Information Technology and Applications ICITA 2005*. pp. 15–24, 2005.

[5] Alexei A. Efros and William T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341-346, 2001.

[6] Michael Ashikhmin, "Synthesizing natural textures," in *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 217–226, 2001.

[7] Jason C. Hung, Chun-Hong Hwang, Yi-Chun Liao, Nick C. Tang, and Ta-Jen Chen, "Exemplar-based image inpainting base on structure construction," *JOURNAL OF SOFTWARE*, vol. 3, no. 8, pp. 57–64, 2008.

[8] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama, "Object removal by exemplar-based inpainting," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2, pp. 721–728, 2003.