CS276: Assignment 2

Yang Hui 2020233290
yanghui1@shanghaitech.edu.cn

ShanghaiTech University — November 7, 2020

Introduction

This project mainly contains two parts.Part1 has two tasks.The first task is to clone the area of interest from the source image onto the task image as the specified offset.The second task is to realize the Poisson Image Editing using the mix gradient field.Part2 is based on part1, using the shortest path algorithm(Dijkstra) to optimize the boundary of ROI.

1 Part 1

1.1 Poisson Image Editing Without Mix Gradient Field

Tasks to be achieved:

(a) Display the source and the target image(and a mask image if necessary).

(b) Clone the area of interest from the source image (denoted by the mask image) onto the target image at the specified offset.

(c) Display the final composited image and save the result as the specified output filename.

Implement:

(a) Get a source image,a target image and a mask image

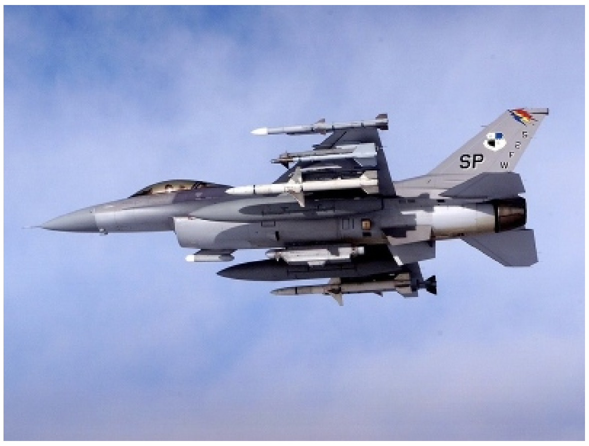
(b) Obtain the ROI in the source image according to the mask image.

(c) The ROI of the source image is derivated twice to obtain the divergence (indicated by b), and the divergence of the boundary point of the ROI is modified to the pixel value of the corresponding point in the target image.
Divergence: $\Delta . = \frac{\partial^2 .}{\partial x^2} + \frac{\partial^2 .}{\partial y^2}$
Hint:the Method to obtain boundary pixels of region of interest
For pixel f(x,y), get the number of its neighboring pixels in the region of interest (indicated by num)
$$\begin{cases} 0 < num < 4 & \text{f(x,y) is the boundary pixel of the region of interest} \\ num = 4 & \text{f(x,y) is the internal pixel of the region of interest} \\ num = 0 & \text{f(x,y) is the outer pixel of the region of interest} \end{cases}$$
The adjacent points of f(x,y): f(x-1,y),f(x+1,y),f(x,y-1),f(x,y+1)

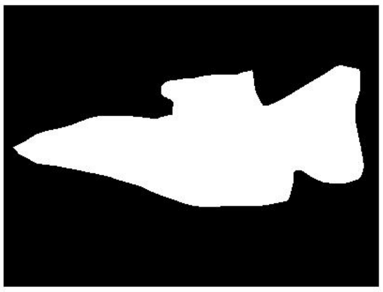
(d) Construct coefficient matrix A: the diagonal element of ROI boundary pixel in matrix A is 1, and the pixel in ROI constructs the value of A according to the Laplacian convolution kernel
Hint:the Laplacian convolution kernel used in the program
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

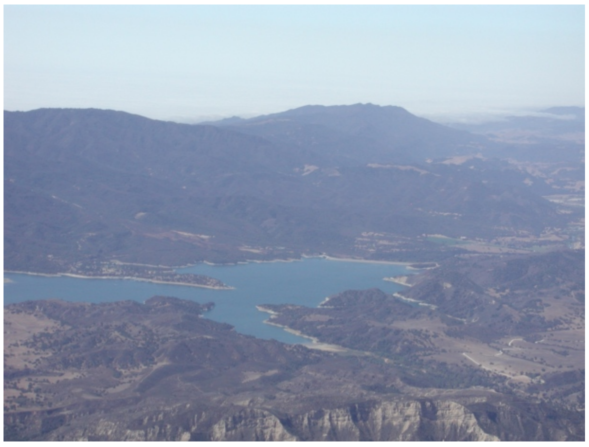
(e) Solve the equation Ax = b. In Matlab, we obtain x by $x = A\backslash b$, and x represents the RGB value of each pixel in the ROI.

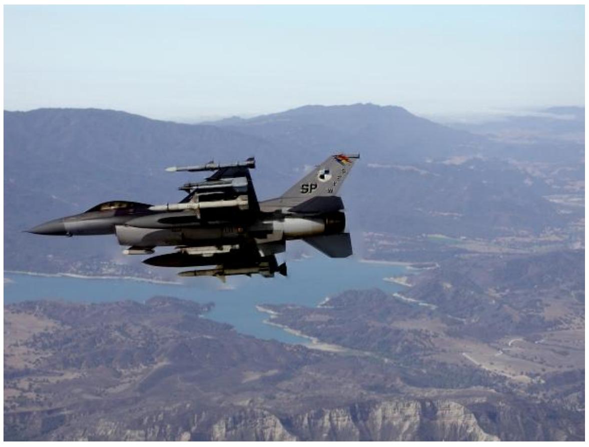
Result: The results are shown in Figure 1 and Figure 2.



(a)The source image

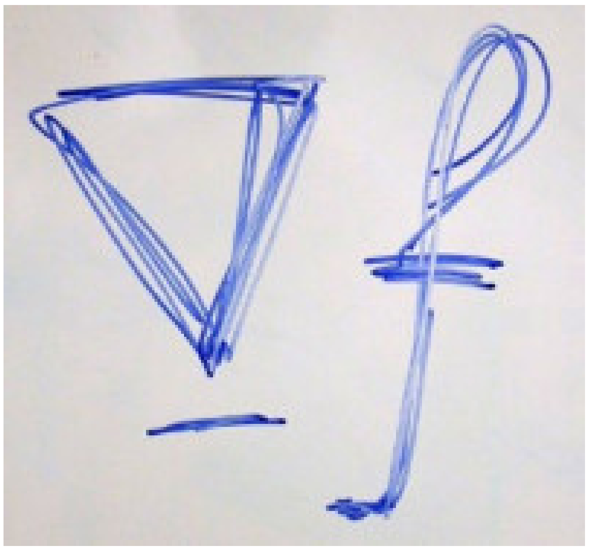

(b)Mask image generated by the program

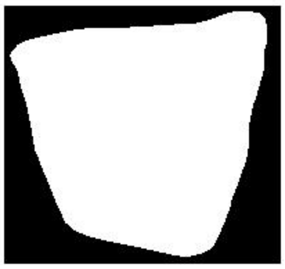

(c)The target image

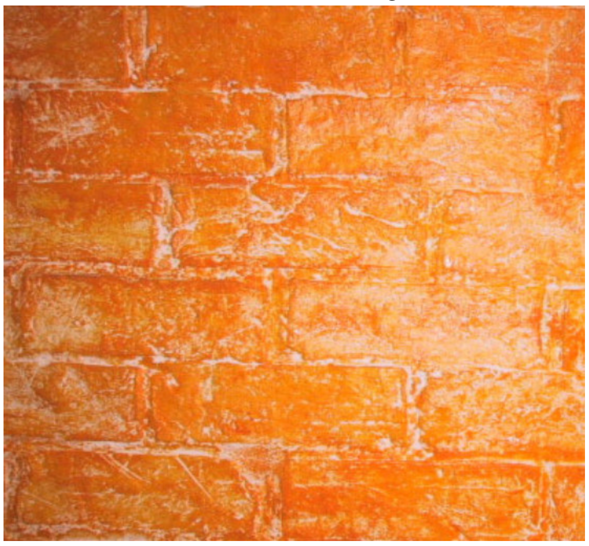


(d)The final composited image

Figure 1: the Result 1 of Poisson blending


(a)The source image


(b)Mask image generated by the program


(c)The target image

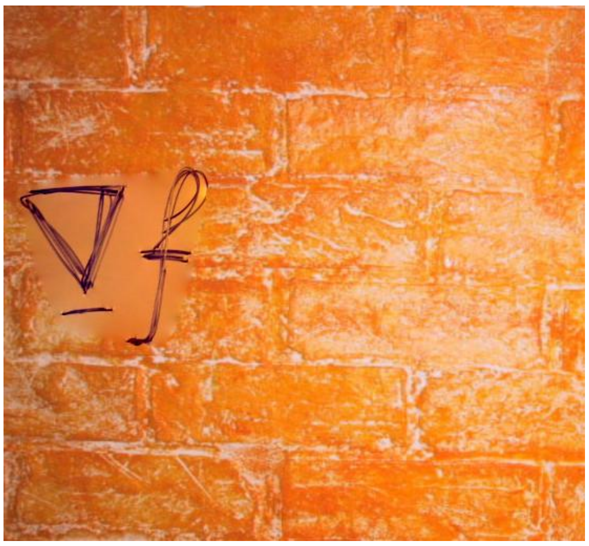

(d)The final composited image

Figure 2: the Result 2 of Poisson blending

1.2 Poisson Image Editing With Mix Gradient Field

In the previous task, the gradient of the target image at the corresponding position was not taken into account when obtaining the gradient of the ROI, as shown in the Figure2 (d), which led to the addition of holes or partially transparent objects on a textured or chaotic background would produce bad results. To avoid this situation, in this task, we will use the mix gradient field to obtain the final gradient of the ROI. Tasks to be achieved:

(a) Display the source and the target image(and a mask image if necessary).

(b) Use mixed gradient fields to achieve Poisson image editing.

(c) Display the final composited image and save the result as the specified output filename.

Implement:

(a) Get a source image,a target image and a mask image

(b) Obtain the ROI in the source image and target image respectively according to the mask image and the specified offset.

(c) Obtain the gradient of the ROI in the source image ($\nabla g(x)$) and the gradient of the ROI in the target image ($\nabla f^*(x)$) respectively. For each pixel in the ROI, compare the absolute values of the two gradients, and select the larger gradient as the gradient of the pixel. After this step, we have obtained the mixed gradient field(representation:$\mathbf{v} = (u, v)$).
for all $\mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})| \\ \nabla g(\mathbf{x}) & \text{otherwise} \end{cases}$

(d) Derivation of the mixed gradient field to obtain the divergence(indicated by b).The divergence of the boundary point of the ROI is modified to the pixel value of the corresponding point in the target image.
$\operatorname{div} \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$

(e) construct coefficient matrix A: the diagonal element of ROI boundary pixel in matrix A is 1, and thepixel in ROI constructs the value of A according to the Laplacian convolution kernel.

(f) Solve the equation Ax = b. In Matlab, we obtain x by $x = A\backslash b$, and x represents the RGB value of each pixel in the ROI.

(g) Replace the pixel value of the ROI in the target image with x.

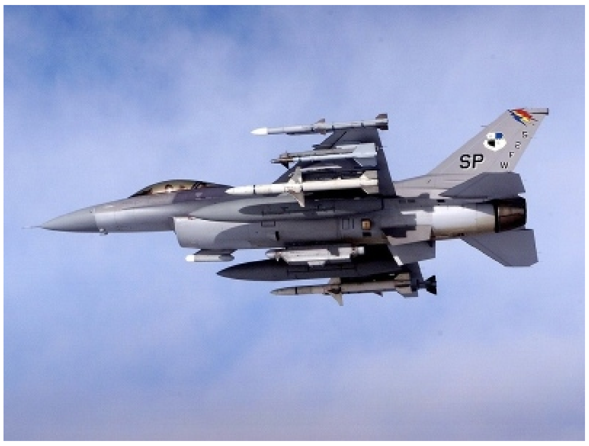
Result: The results are shown in Figure 3 and Figure 4.

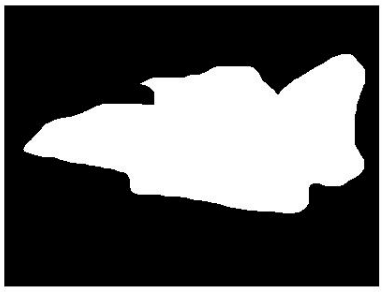
2 Part 2

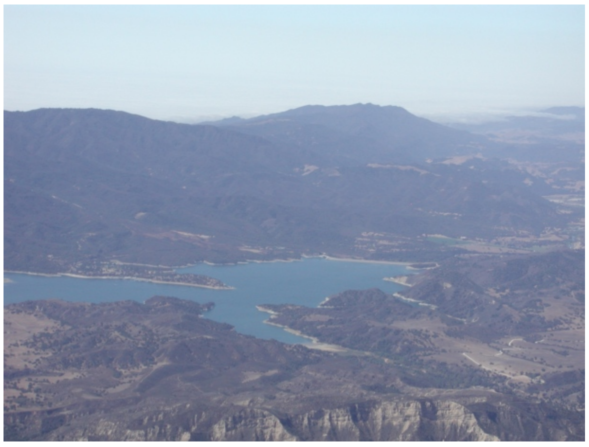
In part1, we have implemented Poisson image editing. However, in order to make Poisson image editing work well, the user must carefully draw a boundary on the source image to indicate the region of interest,such that salient structures in source and target images do not conflict with each other along the boundary. To make Poisson image editing more practical and easy to use, we need to calculate an optimized boundary before implementing Poisson Image Editing.In part2,we need to implement a shortest path algorithm(Dijkstra) for computing the optimal boundary.
The Figure 5 shows the details of boundary optimization.In figure(a),Ω_0 is the region of interest drawn by the user,which completely encloses the object of interest Ω_{obj}. The optimized boundary $\delta\Omega$ lies inside $\Omega_0\backslash\Omega_{obj}$.A cut C is used to cut the connectivity of the band in the graph G. In figure(b), the yellow and blue pixels are on different sides of C. We need to calculate the shortest path from a yellow pixel to its neighboring blue pixel.
Tasks to be achieved:

(a) Specify both the exterior boundary and the interior boundary.

(b) Compute the optimal boundary by Dijkstra algorithm.


(a)The source image


(b)Mask image generated by the program


(c)The target image

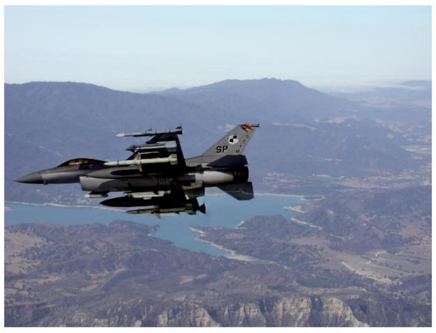
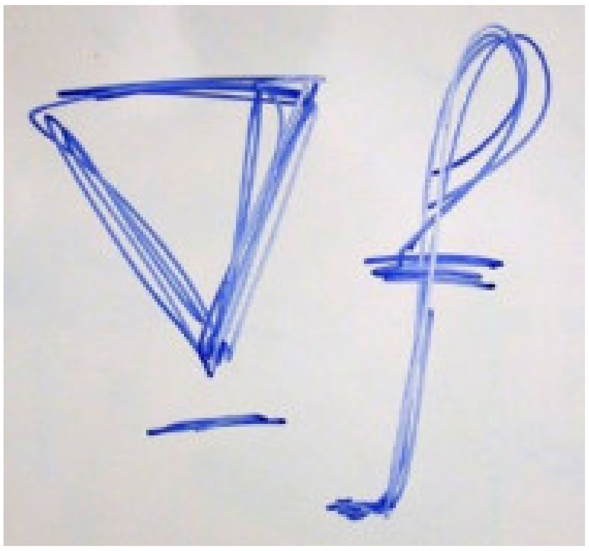
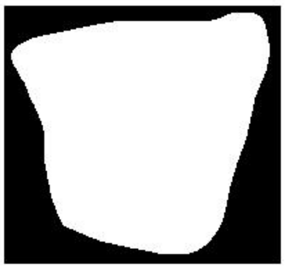
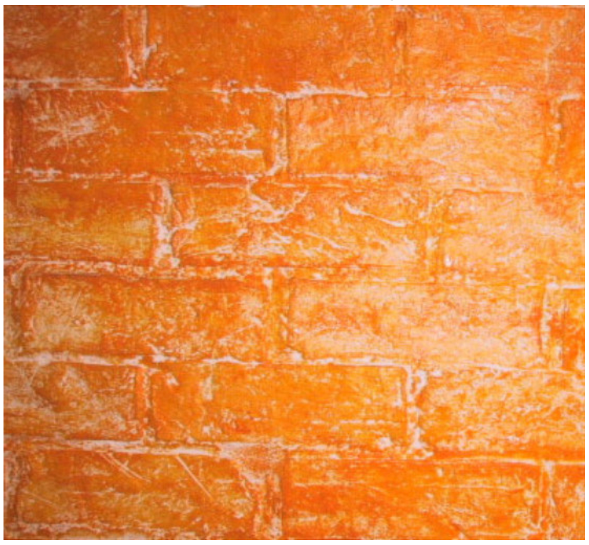

(d)The final composited image

Figure 3: the Result 1 of Poisson blending with gradient mixing


(a)The source image


(b)Mask image generated by the program


(c)The target image

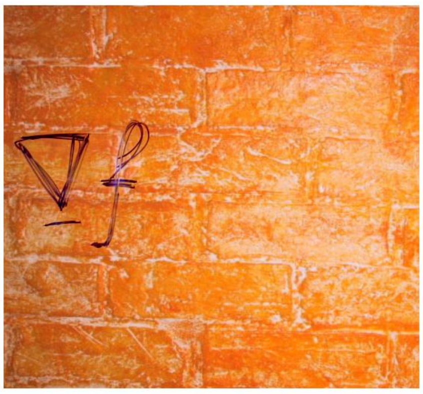

(d)The final composited image

Figure 4: the Result 2 of Poisson blending with gradient mixing

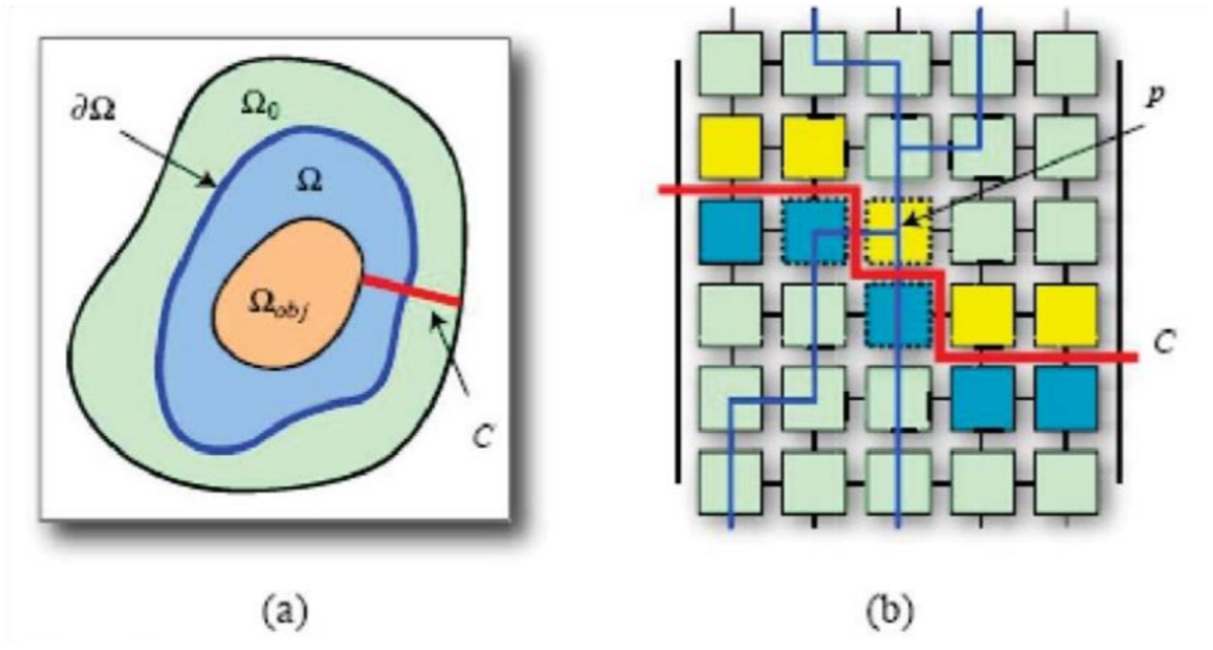


Figure 5: Boundary optimization

(c) Implement the Poisson Image Editing.

Implement:

(a) The mask image of the band area is obtained by XORing the mask image of Ω_0 with the mask image of Ω_{obj}.

(b) Build up a graph G that consists of pixels within the band.Firstly, I marked the pixels in the band, and the vertices in the graph are represented by the unique numbers of the pixels. If <s,v> represents an edge, the weight of the edge in the graph is expressed as $f_t(v) - f_s(v)$. $f_t(v) - f_s(v) = \sqrt{(R_t(v) - R_s(v))^2 + (G_t(v) - G_s(v))^2 + (B_t(v) - B_s(v))^2}$

(c) Defined C as the shortest straight line segment among all pixels pairs connecting the exterior and the interior boundary by computing the Euclidian distance.

(d) Through Bresenham's algorithm, rasterize C to obtain a list of pixels passed by C.Take the pixel list as yellow pixels, get blue pixels, and modify the weight of the edge from the yellow pixel to the blue pixel in Graph G, and set it to 0.
Hint:

(1) The method to get blue pixels
Obtain two points on the inner and outer boundaries of the line segment C, find the adjacent points of the two points respectively, and rasterize the line of the two adjacent points through the Bresenham algorithm. The points obtained by rasterization are regarded as blue points.

(2) We construct the graph through a sparse matrix, so set the weight of edges that do not exist in the graph to 0,that is, 0 is regarded as infinity.

(e) Initialize Ω as Ω_0,compute the initial value of k. $\mid \partial\Omega \mid$ is the length of the boundary $\partial\Omega$.
$k = \frac{1}{|\partial\Omega|} \sum_{p \in \partial\Omega} (f_t(p) - f_s(p))$

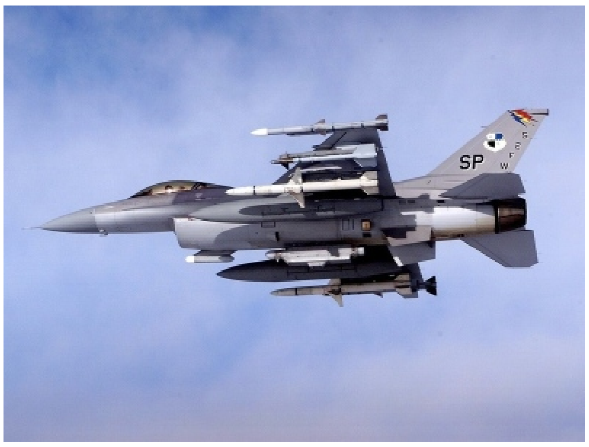
(f) Traverse the pixel list and obtain a set of shortest path from yellow pixels to blue pixels through Dijkstra.For each iteration, we need to obtain the shortest path from the yellow pixel to its neighboring blue pixel. The weight sum of each path is expressed by the following formula. After obtaining the shortest path, we then update k according to the $\delta\Omega$. $E(\partial\Omega, k) = \sum_{p \in \partial\Omega} ((f_t(p) - f_s(p)) - k)^2$, s.t. $\Omega_{obj} \subset \Omega \subset \Omega_0$
Hint:In my program, in order to avoid the squaring causing the value to be too large, I use the absolute

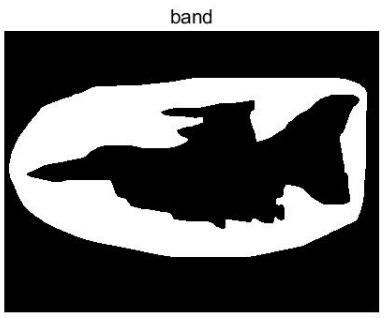
value method, that is:
$E(\partial\Omega, k) = \sum_{p \in \partial\Omega} \mid ((f_t(p) - f_s(p)) - k) \mid$, s.t. $\Omega_{obj} \subset \Omega \subset \Omega_0$

(g) The optimized boundary is assigned as one that gives the globally minimum cost.

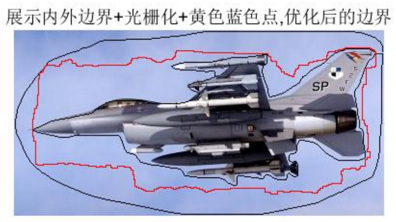
(h) Use the same code from Part1 to do Poisson Image Editing.

Result:The results are shown in Figure 6.

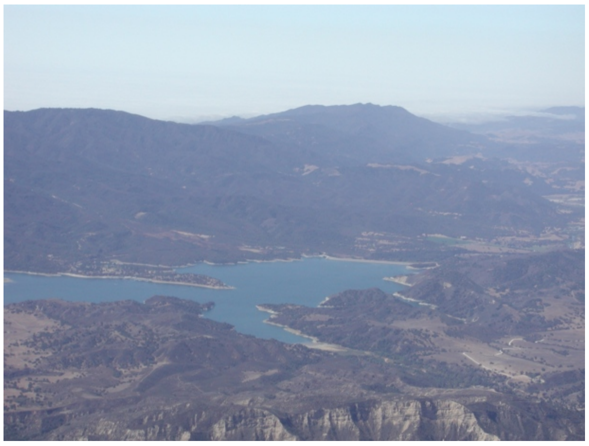

(a)The source image



(b)Ω_{obj} and Ω_0



(c)The shortest path


(d)The target image

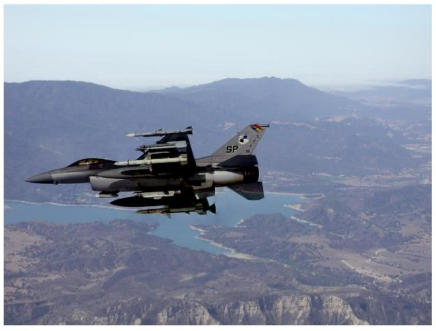

(e)The final composited image

Figure 6: the Result 1 of Possion blending with boundary optimization