Report on Seam Carving

Yifan Wu
Yuanpei College, Peking University
evanwu@pku.edu.cn

Yiping Ma
School of Electrical Engineering and Computer Science, Peking University
ma16_helen@pku.edu.cn

ABSTRACT

Seam carving is a content-aware image resizing method, which means that it takes the content of the image into account when performing resizing. It was first proposed in [1]. In this report, a python implementation of seam carving is provided, with different approaches to obtain the energy map.

1. INTRODUCTION

Given an energy map of the image, indicating the content intensity, the seam carving algorithm is to find the seam with the lowest energy. Such a seam is optimal for removal.

The performance of seam carving relies mostly on the energy map it receives. In our report, we tried out several different energy maps, including traditional RGB and Local Entropy maps, new-proposed forward energy map, and also, our new deep feature map obtained from pretrained VGG-19 model.

2. PACKAGES AND ENVIRONMENT

2.1 Libraries and frameworks

- **Numpy** v1.14.3
- **Matplotlib** v2.1.1
- **OpenCV** v3.4.1
- **Numba** v0.38.1
- **Pytorch** v0.3.1
- **Torchvision** v0.2.0

The packages can be automatically installed by running the following command:
`pip install -r requirements.txt`
or by running the script file:
`./run.sh`

If you prefer to set up the environments by yourself, you should comment the second line in file `run.sh`.

2.2 Environment

- **OS**: Linux
- **Python3.5.2**
- **GPU**: Tesla-V100
- **CPU**: Xeon(R) E5-2690 v4
- **RAM**: 512GB

3. BASICS OF SEAM CARVING

This section introduces our basic implementations of the seam carving algorithm.

3.1 Image Cutting

The optimal seam is defined:

$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

which can be found with dynamic programming:

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

This is the basis of our implementation. To cut an image of size $n \times m$ into $n' \times m$, we simply repeat the process $(n - n')$ times:

- compute the energy map for image
- find the optimal seam
- remove the seam from the image

3.2 Image Enlarging

Image enlarging is different from simply cutting. If we operate the process above with inserting a seam instead of removing, we will possibly find the same seam in each operation, and obtain a somehow weird image.

So the modified image enlarging repeats the following operations (to implement the method in paper):
Firstly make a copy C of the original image M, and a matrix P to record the coordinates of pixels in M.

- compute the energy map of C
- find the optimal seam s for removal in C.
- remove the s from C, trace its coordinates in M, add the seam into M
- modify the matrix P such that all seams right of s have there coordinates plus 1. Delete s from P.

4. TASK 1: WITH ENERGY FUNCTIONS

4.1 Energy Function RGB

The RGB energy function of pixel i is defined as following, where pixel i has 8 neighbors, each having a $val.$:

$$val_1 = (abs(R_i - R_1) + abs(G_i - G_1) + abs(B_i - B_1))/3$$
$$energy_i = val_1 + \cdots + val_8$$

For the computation of such a energy map, we use kernels and $filter2D$ in OpenCV to accelerate. With 8 kernels for $val_1, ...val_8$, we are able to add the 8 matrices (obtained by each kernel) together to get the energy map.

4.2 Energy Function Local Entropy

As the handout states, the entropy of pixel i is defined as follows:

$$H_i = -\sum_{m=i-4}^{i+4}\sum_{n=i-4}^{i+4} p_{mn} log(p_{mn})$$

where

$$p_{mn} = \frac{f(m,n)}{\sum_{k=i-4}^{i+4}\sum_{l=i-4}^{i+4} f(k,l)}$$

and $f(k,l)$ is the grayscale value of pixel (k,l).

To accelerate the computation of matrix f, we use kernel(all ones by 9×9) again, to pre-calculate $\sum_{k=i-4}^{i+4}\sum_{l=i-4}^{i+4} f(k,l)$.

5. TASK 2: WITH FORWARD ENERGY

Forward energy was first proposed in [2]. In section 5.1 of this paper, the author introduce forward energy in dynamic programming, the cost is defined as follows:

- $c_{left}(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)|$
- $c_{up}(i,j) = |I(i,j+1) - I(i,j-1)|$
- $c_{right} = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$

update rule:

$F(i,j) = energymap(i,j) + min\{F(i-1,j-1) + c_{left}(i,j), F(i-1,j) + c_{up}(i,j), F(i-1,j+1) + c_{right}(i,j)\}$. We also use kernel in the implementation, the details can be seen in code.

6. TASK 3: WITH DEEP METHOD

In this section, we implemented work of [3] to visualize VGG-19 model in Pytorch which is pretrained on ImageNet. An activation map of the whole image is obtained as an energy map for seam carving.

Figure 1: A raw energy map obtained from VGG-19.

The figure above shows an energy map for the image on the left. It is the deconvolution of output from the 7^{th} convolutional layer in the pretrained VGG-19 model. The original deconvolutional output is a 3-channel feature map, so we take an average over the channels and transform it into a greyscale map. As we can see, the energy map includes heavy noise, resulting in the instability of carving. To smoothen the energy map, we add the RGB energy with a factor of 0.5 onto the deep feature map.

7. EXPERIMENTS

7.1 Image Cutting

See fig.2 for our experiments. The deep method excels other methods in its performance, but also maintaining a comparable running speed on GPUs.

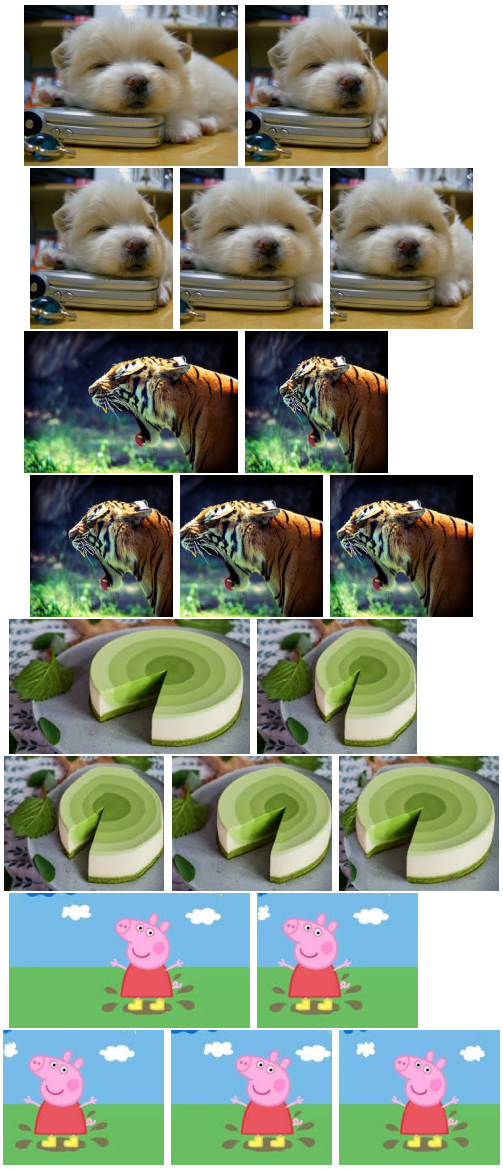

Figure 2: Cut some images. The above left is the original image. Above right, below left, below middle, below left are energy RGB, local entropy, forward and deep.

7.2 Image Enlarging

See fig.3 for our result on image enlarging operation. The deep method provides results comparable to traditional methods, on some images even better.

Figure 3: Enlarge come images. The same as cutting, with above left the original image, Above right, below left, below middle, below left respectively energy RGB, local entropy, forward and deep.

8. CONCLUSION

Comparing to traditional energy maps, deep feature map helps take the semantics of an image into account. It is a good idea to combine deep feature maps with traditional maps, balancing their performance, with removal of noise from deep energy maps, and adding semantics into traditional maps.

9. REFERENCES

[1] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007.

[2] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. *ACM transactions on graphics (TOG)*, 27(3):16, 2008.

[3] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.