Image Inpainting Using Belief Propagation

Yang Yang Weixiao Fu

March 12, 2015

Abstract

In this report, we will introduce our implementation for the image inpainting algorithm based on belief propagation. The goal in our implementation is to define a cost function and try to optimize the cost function in global. To optimize the objective cost function, we use belief propagation to find one of the optimal solutions to the cost function. And then, we will also demonstrate some results using our implementation. Finally, we will discuss some further optimizations on the algorithm.

1 Introduction

Inpainting is the process of reconstructing the lost parts of images, and image inpainting also refers to the application of sophisticated algorithms to replace lost or corrupted parts of the image data (mainly small regions or to remove small defects)[1].

Suppose we have an image with missing parts called target, and remaining part called source. The goal of image completion is to fill the target area with the information from source area.

Belief propagation is a very useful iterative method to get a global optimization solution to a objective function just like gradient descent. And belief propagation is widely used in Bayesian networks and Markov random fields[2].

For our project we studied a well-established patch-based algorithm proposed by Komodakis and Tziritas[3] which solves the image completion problem. This algorithm divides the whole image into many patches, and uses patches from source area to fill target area. The structure of this algorithm can be divided into two process:

- Formulating the patch filling problem into an optimization problem with potential functions
- Solving the optimization problem using the belief propagation method as described in [4]

In the following sections, we will introduce this algorithm in details.

2 Inpainting With Belief Propagation

For a image inpainting problem, we can consider it as a machine learning problem, in which computer is trained to be able to fill the missing region(target region) using the existing region(source region) to make these two region fit best.

After doing a lot survey, we found out that there is a relative simple and intuitive model to achieve the goal in this paper[3]. And we implement a similar algorithm in this report.

2.1 MRF Model

In this report, we use the following denotation:

- I_0 - the whole original image
- S - the source region in the image
- T - the target region in the image
- L - the set including all the candidate patches

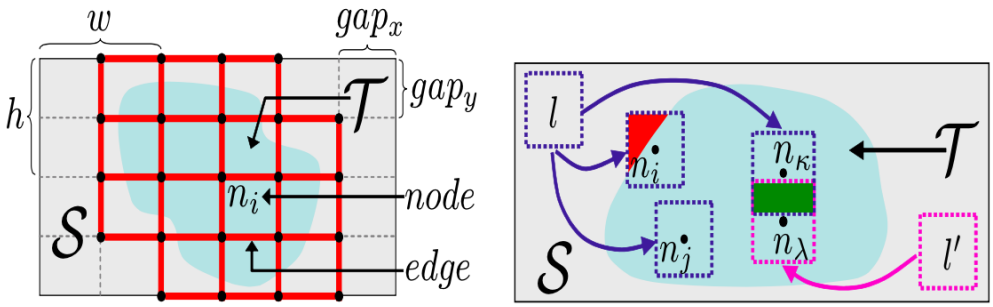
- E - the edges set of the node net(table), in this problem, the adjacent nodes have a edge between them.

Markov random field(often abbreviated as MRF) is a set of random variables having a Markov property described by an undirected graph according to the Wikipedia[5].

In the image inpainting problem, the patches set L is made of the candidate $w \times h$ patches from the source region S, and the target region could be evenly some nodes and the adjacent nodes have $gap_x = \frac{w}{2}$ and $gap_y = \frac{h}{2}$ in horizontal and vertical directions respectively. And the MRF model consists of these node $\{n_i\}_{i=1}^{N}$ where the N means the number of nodes in target region T.

MRF potentials can be considered as the cost values between two specified patches. And we use the sum of squared differences(SSD) as the MRF potentials in this project.

The following picture which illustrate the MRF model is from paper[3]:



(a) MRF nodes and edges (b) MRF potentials

Here the edge cost $V_i(l)$ needs to be defined. $V_i(l)$ is the SSD(the sum of squared differences) of the intersection region between the source region S and patch l when placing patch l over node n_i. The calculation is showed in equation (1).

$$V_i(l) = \sum_{p \in [-\frac{w}{2}, \frac{w}{2}] \times [-\frac{h}{2}, \frac{h}{2}]} M(n_i + p)(I_0(n_i + p) - I_0(l + p))^2 \tag{1}$$

In equation (1), $M(\cdot)$ is a binary mask, which is one in the target region and zero in the source region. It means that the larger intersection between the patch and the source region, the more information we can get from the source region. In our implementation, we may round up the target region a little to make the intersection region larger so that we can get more information from the neighbor source region S.

After defining the partial cost function between two specified patches, we need an MRF objective function to optimize, and that is showed in the equation (2).

$$E(\{\hat{l}_i\}) = \sum_{i=1}^{N} V_i(\hat{l}_i) + \sum_{(i,j) \in E} V_{ij}(\hat{l}_i, \hat{l}_j) \tag{2}$$

$V_{ij}(\hat{l}_i, \hat{l}_j)$ in the equation (2) is the SSD of the two patches $\hat{l}_i$ (the patch placed over n_i) and $\hat{l}_j$ (the patch placed over n_j), here another constrain is that n_i and n_j should be a pair of adjacent nodes, which can be formulated as $(i, j) \in E$.

Intuitively, for the objective function 2, the less the function result is, the better the patches sequence $\{\hat{l}_i\}$ is. That is, our goal is to minimize the objective function 2.

However, the naive method to find the global optimal solution to this objective function is brute force, which has a time complexity of $O(|L|^N)$, and $|L|$ is the number of candidate patches, which is alway larger than 300 in a 500×500 image, and N is the number of the nodes in the target region, which is about 100 if the target region is in size of 72×72(since we use 16 as our patch width and height, that is, $gap_x = 8$ and $gap_y = 8$). So the brute force method is definitely infeasible.

2.2 Belief Propagation

In the last subsection, we know that using the brute force method is infeasible, so another method should be used to minimize the objective function. According to the paper[3], since the image inpainting

problem has been modeled into a MRF problem, using belief propagation to solve it is pretty intuitive, so belief propagation is the best option.

As is described in the introduction section, belief propagation is quite a useful iterative global optimization method [2] to solve MRF(Markov random fields) problem.

Actually, belief propagation(BP) is a message passing algorithm for performing inference on graphical models[2].

BP algorithm works by propagating local messages along the nodes of an MRF[3]. Message $m_{ij}(l)$ is sent from node n_i to node n_j means indicates how likely node n_i thinks that node n_j should be assigned patch l[3], and the messages are updated as the equation (3).

$$m_{ij}(l) = \min_{l_i \in L}\{V_i(l_i) + V_{ij}(l_i, l) + \sum_{k:k\neq j,(k,i)\in E} m_{ki}(l_i)\} \tag{3}$$

The iteration should be stopped if the messages finally converge or the iteration limit is reached. In our implementation, we choose 20 as the iteration limit according to our experiments.

After the iteration phase is over, we need to calculate the belief for each patch in each node in the target region. The belief will be calculated as equation (4)

$$b_i(l) = -V_i(l) - \sum_{k:(k,i)\in E} m_{ki}(l) \tag{4}$$

Finally, we select the best patch which has the maximal belief for each node, and then we get a set of patch sequence $\{\hat{l}_i\}_{i=1}^{N}$ which should be a approximate solution.

3 Implementation

Due to the large amount of the computation is required in belief propagation algorithm, implementing the algorithm in high performance is really necessary. Since MATLAB is actually a script language and MATLAB code needs interpreter to execute, implementing this algorithm in MATLAB might be inefficient. So we decide to use C++, and for the further optimization, we also use CUDA, which will be discussed in the following sections.

3.1 SSD Table Calculation

As we all know about this algorithm, the SSD for each pair of patches such as (l_i, l_j) will be in used in the iteration message passing phase. And the calculation of SSD is really a time-consuming procedure. So pre-computing the SSD for all the pairs of patches is quite necessary.

In this phase, the SSD table calculation has a time complexity of $O(|L| \times |L|)$.

3.2 Message Passing Iteration

We use 2-dimension array(vector) to store the nodes table so that we can easily access. And in each node in the table, there will be a nested array(vector) to store the messages for each patch in four directions(up, down, left, right).

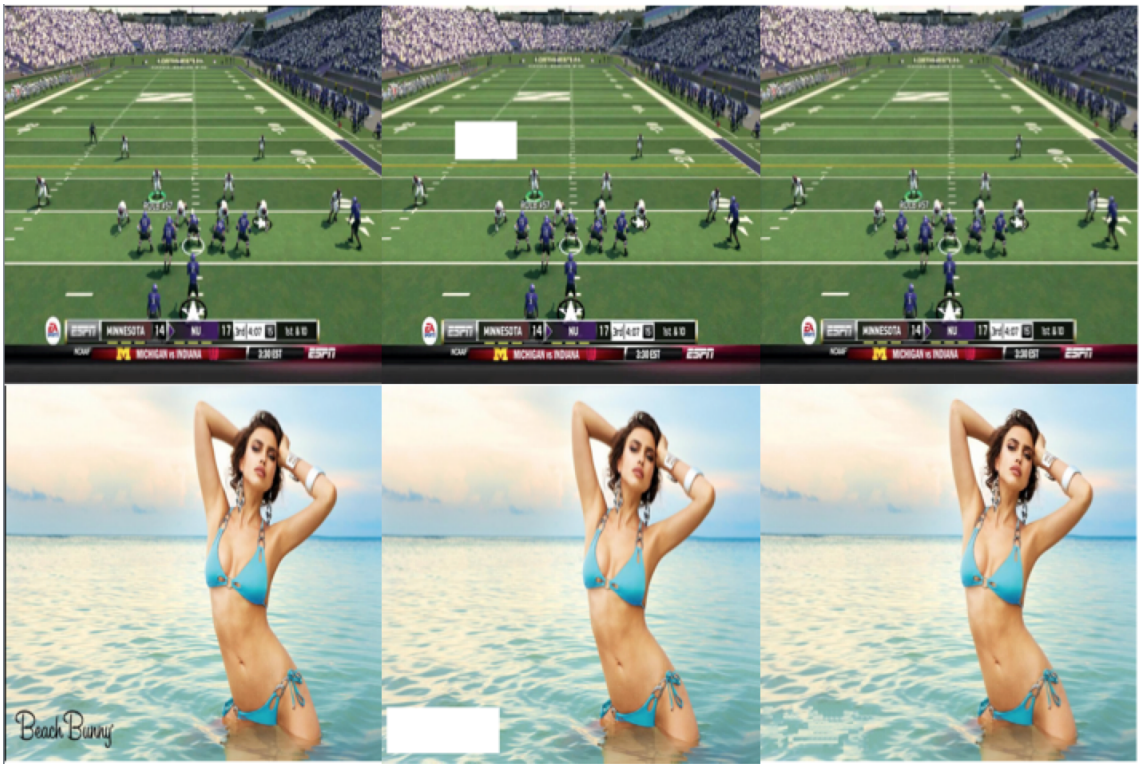
Assume that n_i is trying to pass the message to n_j and there are $|L|$ candidate patches, there will be time complexity of $O(|L| \times |L|)$ because to determine the message of one patch, we need to go through all the candidate patches to find the minimal message.

3.3 Best Patch Selection

To select the best patch for each node in the target region, we will go into each node and then calculate the belief for each patch in this node to find the best patch which has the maximal belief.

4 Experiments

For the experiments, we use several images to test how our implementation work.



In the above experiment images, the images in the left side are the original images, and the middle images are the images with the masked region as blank, and the images in the right side are the output images of our image inpainting implementation.

5 Optimizations

Since applying the original BP algorithm on this global optimization problem will consume a lot of computing time especially when the number of candidate patches is very large, we use the following methods to optimize the performance and running time:

- Adjust the weight(we can also call it ratio here) of passed message and the SSD of the adjacent patches. In [3], the passed message and the SSD of the adjacent patches are simply summed up to be the message candidates to be passed in the next iteration. After a lot experiments, we found out that the filled region didn't look really good without any adjustment to the weight, we realized that this method could be really parameters sensitive. The weight of the passed message could be considered as the learning rate, which means how much a node can learn from its adjacent node's message and the confidence of the passed message[6].

$$m_{ij}(l) = \min_{l_i \in L}\{C_1 \cdot V_i(l_i) + C_2 \cdot V_{ij}(l_i, l) + C_3 \cdot \sum_{k:k\neq j,(k,i)\in E} m_{ki}(l_i)\} \tag{5}$$

  With plenty of test, we finally get a set of parameters(weights$\{C_1, C_2, C_3\}$) in equation (5) which could make a better effect(of course, it is still not the best)

- Select less patches as candidates. Since the masked region is most likely to choose the patches from its near region and the far patches always contribute little in the message passing phase, remove these unnecessary patches is really a good way to reduce the number of the candidate patches, which might have no effect on the final result and significantly reduce the running time

- Use CUDA parallel programming to make the iterations run in parallel. CUDA parallel programming model is widely used in the modern machine learning area[7], and the BP has a really high time complexity. So using CUDA to optimize our implementation is a very natural idea. Thanks to that BP is a iterative method and only use the previous iteration result, which means that the data dependency can be easily removed and fit into the CUDA parallel model.

6 Code

We open source our code in Github, and there is a README which will tell you how to run and test our code.

References

[1] Inpainting. `http://en.wikipedia.org/wiki/Inpainting`.

[2] Belief propagation. `http://en.wikipedia.org/wiki/Belief_propagation`.

[3] Nikos Komodakis. Image completion using global optimization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 442–452. IEEE, 2006.

[4] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[5] Markov random field. `http://en.wikipedia.org/wiki/Markov_random_field`.

[6] Huang Ting, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. Image inpainting by global structure and texture propagation. In *Proceedings of the 15th international conference on Multimedia*, pages 517–520. ACM, 2007.

[7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.