COMPUTER VISION & IMAGE PROCESSING [CS676A], IITK

Image Completion

with statistics of patch offsets

By Pranshu Gupta & Abhishek Jain

Instructor: Prof. Vinay P. Namboodiri

Introduction

Image Completion is the task of filling missing parts of a given image with the help of information from the known parts of the image. Our aim in this project was to implement an application that would take an image with a missing part as input and give a completed image as the result.

There are several ways to accomplish this task, they can be divided in two categories: diffusion based and exemplar based. Our method is MRF based exemplar approach.

Diffusion based approaches are good for small unknown regions whereas exemplar based approaches are good for large unknown regions.

Methodology

We observe that if we match similar patches in the image, the statistics of patch offsets are sparsely distributed and form several prominent peaks in the statistics. Such dominant offsets describe how the patterns are most possibly repeated, and thus provide clues for completing the missing region [1]. By selecting a few dominant offsets, we combine the stack of shifted images to fill in the missing region.

The process involves the following three steps:

1. Matching Similar Patches

For each patch P in the known region, we compute its offset s to its most similar patch. Here, s = (u, v) is the 2-d coordinates of the offset, x = (x, y) is the position of a patch, and P(x) is a patch centered at x. The similarity is measured by sum of squared differences between two patches. The threshold $\boldsymbol{\tau}$ is to avoid nearby patches. [1]

$$\mathbf{s}(\mathbf{x}) = \arg\min_{\mathbf{s}} \|P(\mathbf{x}+\mathbf{s}) - P(\mathbf{x})\|^2 \quad s.t. \quad |\mathbf{s}| > \tau.$$

We have implemented this part in two ways, one in which we match each pixel with every other pixel inside the constraint window, and the other in which we randomly initialize the offsets and iteratively try to converge them to an optimal value. The latter is faster but the former gives better results.

2. Computing Offsets Statistics

Given all the offsets we compute a frequency statistic in form of a 2d histogram. Then we smooth the histogram with a Gaussian filter and pick out the K highest peaks of this histogram. We have used $K = 60$ in our implementation.

3. Combining Shifted Image via Optimization

Given the K offsets, we treat them as labels and the unknown pixels as sites to be labeled. This can be formalized as a MRF energy optimization problem, where we minimize the cost of the labels assigned to the site. The cost is defined in the form of an MRF energy function.

$$E(L) = \sum_{\mathbf{x} \in \Omega} E_d(L(\mathbf{x})) + \sum_{(\mathbf{x},\mathbf{x}') \mid \mathbf{x} \in \Omega, \mathbf{x}' \in \Omega} E_s(L(\mathbf{x}), L(\mathbf{x}')).$$

Here Ω is the unknown region (with boundary conditions), and (x, x') are 4-connected neighbors. L is the labeling, $L(x) = i$, means that we copy the pixel at $x + s_i$ to the location x.

The data term E_d is 0 if the label is valid (i.e., $x + s$ is a known pixel) otherwise it is ∞. The smoothness term E_s penalizes the incoherent seams.

$$E_s(a, b) = \|I(\mathbf{x} + \mathbf{s}_a) - I(\mathbf{x} + \mathbf{s}_b)\|^2 + \|I(\mathbf{x}' + \mathbf{s}_a) - I(\mathbf{x}' + \mathbf{s}_b)\|^2.$$

Here $I(x)$ is the RGB color of x.

Minimizing the MRF cost

In order to minimize the MRF cost and find the optimal labelling for the sites, we implemented the $\alpha - \beta$ swap algorithm for multi-labeled energy minimization as described in [2]. It iteratively considers pairs of labels and applies graph cut to find out if a swap of labels between sites can minimize the cost, if it does then the labels are swapped and the iteration goes to the next label pair until the cost converges or maximum number of iterations allowed have been completed.

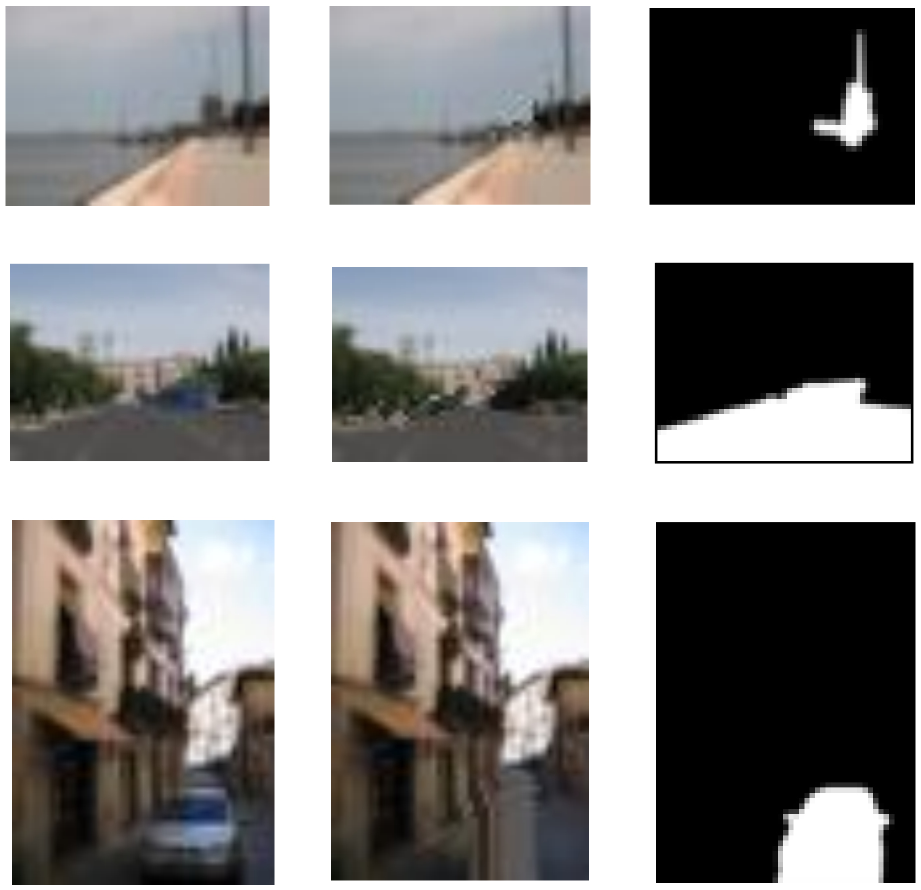
The pseudocode for the $\alpha - \beta$ swap algorithm is as follows [2]:

```
1.  Start with an arbitrary labeling $f$
2.  Set success := 0
3.  For each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$
    3.1.  Find $\hat{f} = \arg\min E(f')$ among $f'$ within one $\alpha$-$\beta$ swap of $f$
    3.2.  If $E(\hat{f}) < E(f)$, set $f$ := $\hat{f}$ and success := 1
4.  If success = 1 goto 2
5.  Return $f$
```

After the cost minimization step is done, we get labelling for each site and we construct the final image by copying the RGB values of the pixels from the known region to the sites as per their offsets.

Results

Here are some of the results we obtained from our implementation:



input output mask

Implementation

The image completion system is implemented in Python 3.4 using the OpenCV 3.1 bindings for Python. For more details please refer to the README (it has description of each function implemented).

Conclusion

We have successfully implemented an image completion system that utilizes statistics of patch offsets. Also, this method can reliably fill large unknown regions unlike the diffusion based methods.

Future Work

Our implementation is not very efficient and takes too much time in finding the patch offsets. There is a good scope of improvement in the speed. We have tried random initialization of patch offsets and then converging them but it effects the quality of result.

Most of the time is being spent in finding the offsets for each patch in the image. It can be made efficient by dimensionality reduction and by using some KD-Tree like data structure for fast lookups. Related: Computing Nearest-Neighbor Fields via Propagation-Assisted KD-Trees, Kaiming He & Jian Sun.

Source Code

The source code of the project is available on github.com at:

https://github.com/Pranshu258/image_completion

References

1. Statistics of Patch Offsets for Image Completion, *Kaiming He, Jian Sun*
2. Fast Approximate Energy Minimization via Graph Cuts, *Yuri Boykov, Olga Veksler*