Database-Backed Scene Completion

Alex Alifimoff
aja2015@cs.stanford.edu

Author's note: I liberally use the third person pronoun, "we", in this paper, as I'm used to working in group projects. Rest assured, I am the only author of this project.

Introduction

Ever have an almost-perfect photo? Maybe it's that photo of the beach that your dweeb uncle stepped in front of, or that wedding photo ruined by the donut truck driving in the background. Scene completion is the task of taking a photo and replacing a particular region of that photo with an aesthetically sensible alternative. In this work we demonstrate our implementation of a method originally implemented by Hayes & Efros which produces interesting scene completions utilizing a very large database of images.

Previous Work

There are many different approaches to the problem of scene completion. One possible approach is to use multiple images (either from multiple cameras, multiple pictures, or video) to determine exactly what kind of information was in the masked part of the image, and then adjust that information appropriately and place it back into the original image. [2, 3]

A second common approach is to utilize information from the image itself to attempt to guess what kinds of missing information should be used to fill the masked part of the image. [3] The majority of these approaches involve utilizing nearby textures and other patterns from the input image to fill the scene.

This project follows the methodology outlined by Hayes and Efros [1], who differ from previous approaches in that they try to complete the scene by finding plausible matching textures from other photographs. In particular, the implemented system searches thousands to (ideally) millions of photographs to find globally matching scenes, and then utilizes texture patterns from those scenes to fill the missing hole in the input image.

Key Improvements

The significant improvement of the Hayes & Efros system over previous image completion software is the ability to produce "novel" scene completions through the use of the large image database that is searched to find global scene matches.

Additionally, the Hayes & Efros system does not place wholly stringent restrictions on which pixels must be used from the source image and which pixels must be completed. All of the masked pixels must be replaced, but through the use of a novel application of min-cost graph cutting, the system may decide to replace more pixels in the original image if it makes for a better fit. This allows interesting completions that simply aren't possible with more stringent restrictions. We discuss this in depth in the following sections.

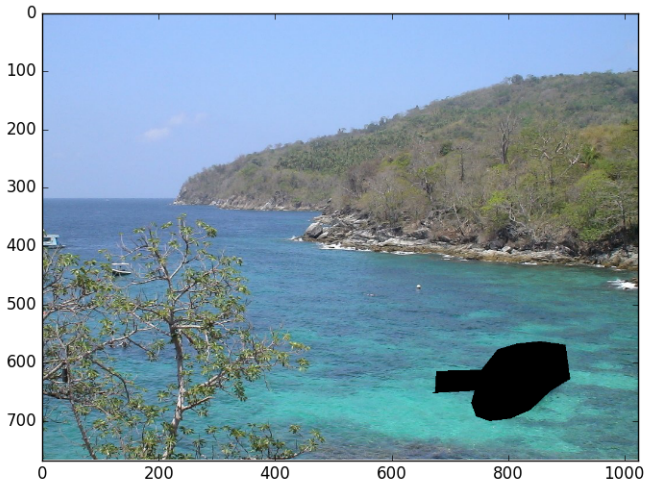

Figure 1: An example input image with corresponding mask (black region)

Technical Approach

The input to the algorithm is an image and a corresponding mask, which indicates which part of the image needs to be filled.

Generally, we then follow three steps:

1. Semantic Scene Matching. Quickly identify possible images that we could use to fill the hole in our scene.
2. Local Context Matching. Identify the local context and search all of the remaining scene matches to find the best local matches.
3. Blending. Perform graph cut and poisson blending to merge the two images.

Semantic Scene Matching

The first part of the method involves finding images which represent similar scenes to the image being filled. However, since there are potentially millions of images to search, any comparison must be done extremely quickly. To implement this part of the algorithm, we rely upon a scene descriptor called GIST. GIST descriptors build a low-dimensional representation of the scene which is designed to capture a couple of perceptual dimensions. The authors of the original GIST paper describe these dimensions as "naturalness, openness, roughness, expansion, ruggedness" [4, 5].

GIST descriptors are pre-generated for every image in the database. Once a masked image is provided, the GIST descriptor for the masked image is calculated. I use GIST descriptors with 5 oriented edge-responses at 4 scales aggregated to a 4x4 spatial resolution. These descriptors are slightly smaller than the original ones in Hayes and Efros, but the slight reduction did not impact performance while slightly improving the time it took to build the GIST descriptors for the database. We augment each GIST descriptor with a color histogram with 512 dimensions to capture color information.

The search is then performed by using the weighted combined GIST and color descriptor and an l2-distance metric. Each generated descriptor is compared to the masked input image, and the best 100 images are kept for local context matching.

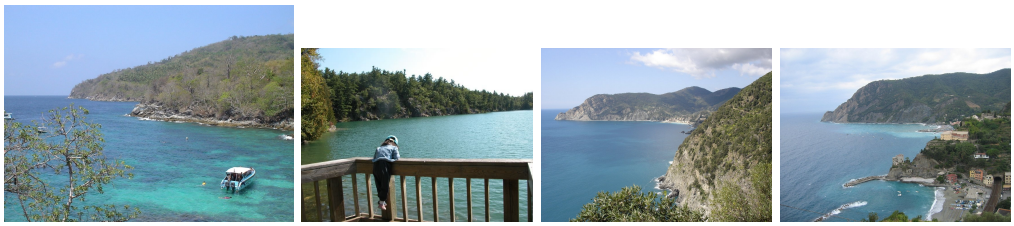

Figure 2: Best GIST matches for leftmost image

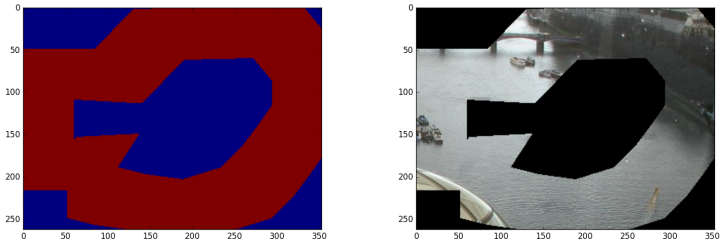

Figure 3: A local context mask and an example of a possible local context

Computational Limitations

One of the main difficulties in pursuing this project was the computational resources necessary to implement it in the same manner as the original authors. The original paper utilized a network of 15 computers to examine millions of images simultaneously. Since this computing power wasn't available to me, I downloaded a pre-filtered subset of closely matching images from Hayes and Efros' project site to augment the thousands of images I downloaded independently. This allowed me to get high quality matches to ensure the rest of the scene completion pipeline worked appropriately. The graphics in this report were generated from my own database of 200,000 images and the additional images from the original project site. My database primarily consisted of photos downloaded from the image sharing website, Flickr, that were tagged "outdoors". I restricted the category for the purpose of getting quality completions for images within the same category.

It took about 12 hours to generate all of the GIST descriptors for the small dataset with liberal use of multiprocessing. However, this computational only needs to be performed once. Performing a single nearest-neighbor search of the dataset takes approximately five minutes.

Local Context Matching

The next step is to find appropriate patches in semantically similar images to use as the scene completion content. The first step of this process is to determine exactly what the local context of a particular image is. To do this, I first dilate the mask. This effectively produces a second, slightly larger mask. Then the mask is cropped so that it is only the width and height of the dilated mask. I then subtract away the original mask and are left with an image patch which corresponds to the local context of a particular image. The remainder corresponds to the local context that we will be examining in each image. We will use this patch to find the "optimal" patch to use in our scene completion for each matching image.

We illustrate this process. On the left we show an example dilated mask, where the red corresponds to the area of local context we will be consider. Then, as we move it across the image, we consider patches like the one on the right.

We take each patch and compute a HOG descriptor and a color histogram. We utilize these as texture and color features and compare them to the local context of our source image. We use sum-of-squared distances of this feature set to select which patch to use from a particular image.

Blending

There are two main parts to the blending step. Given a mask and a dilated mask, we have to use all of the pixels from the patch image for the area of the image covered by the mask. However, for the dilated mask, we have a decision to make. One particular innovation of Hayes and Efros' method is actually choosing to remove more of the original image than the mask requires.

To determine which part of the original image to keep and which part to patch, we use a min-cost graph-cut algorithm. We assign each pixel a label, "original" or "patch". We call the set of all labels L, and we minimize:

$$C(L) = \sum_{p} C_{unary}(p, L(p)) + \sum_{p,q} C_{pair}(p, q, L(p), L(q)) \tag{1}$$

We define the unary cost functions as follows. For any pixel in the space removed by the original mask, $C_{unary}(p, original) >> 0$ (any very large number) and we set $C_{unary}(p, patch) = 0$. For any pixel that is not covered by the mask or the dilated mask, $C_{unary}(p, patch) >> 0$ and $C_{unary}(p, original) = 0$. The intuition here is that for the former category, we must choose pixels from the patch. In the latter category, we must choose pixels from the original picture. For all of the rest of the pixels, we define

$$C_{unary}(p, patch) = (k * ||f(p) - f'(mask)||)^3 \tag{2}$$

where f is a function returning the location of a pixel and f' is a function returning the nearest pixel in the original mask. Intuitively, we want to punish choosing pixels that are not in the original the further away we get from the hole. Like Hayes and Efros, we use $k = 0.002$.

The remaining part of the graph cut algorithm is determining how to define C_{pair}. In our implementation, each pixel is connected in a four-way neighbor set-up, so for pixels that are not adjacent, we have zero cost. For other pixels, we use:

$$C_{pair}(p, q, L(p), L(q)) = ||h(p_{patch}) - h(p_{original})|| + ||h(q_{patch}) - h(q_{original})|| \tag{3}$$

where h is a function returning the vectorized (RGB) representation of a pixel. The intution here is that we want to minimize the gradient of the image difference as opposed to the intensity difference along the seam.

We include a figure demonstrating the change from before the graph-cut is applied and afterwards. Generally this causes a small expansion in the size of the mask.

Finally, poisson blending is applied to the image and its patch to seamlessly blend the two images. This ensures that slight differences in color do not ruin the completion attempt. The poisson solver is allowed to run on the entire domain of the image and not just the local region it is attempting to patch.

Results

Here are a number of possible good completions from various input masks and input images. Generally, when I used the pre-seeded database of gist matches compiled from the Hayes & Efros site, I got reasonable performance. Additionally, when I used images that were "outdoorsy" (this was the image category I primarily downloaded from Flickr), I got reasonable

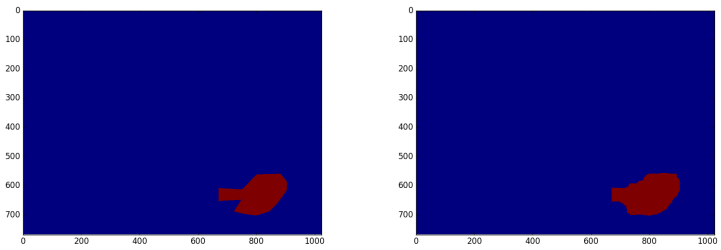

Figure 4: The patch before and after applying graph-cut

Table 1: Runtime comparison to original implementation

Phase	**Avg. Runtime**	**StdDev Runtime**	**Hayes & Efros**
Semantic Scene Matching	5.2 minutes	0.2 minutes	50 minutes [2]
Local Context Matching	22.3 minutes	1.2 minutes	20 minutes
Composition	2 minutes	0.1 minutes	4 minutes

completions. However, when trying to complete images that didn't have particularly good GIST descriptor matches in the dataset, the matches could be comically bad.

One of the main take-aways from this project is that this method highly relies upon having a large dataset available to search for completions. Hayes and Efros required a significant amount of computation power to search their dataset of 2 million images, and even they largely restricted the semantic categories in which they downloaded images. Utilizing this method as a production system for image completion would only be reasonable for companies that have significant computation power and access to many images, like a search engine provider or photo-sharing website.

Runtime is another issue of concern with this particular algorithm. As discussed previously, Hayes and Efros required fifteen CPUs to process a single image in five minutes. On a single CPU, their algorithm took 74 minutes to run. The average runtime for my implementation across a sample set of 100 photographs was as follows. For this particular experiment, we chose to use the 200 best matching scenes for local context matching. [1] My implementation was comparable to Hayes and Efros, despite being implemented in Python.

Quantitatively evaluating the performance of the algorithm in regards to how effectively it completes images is difficult, as there is no good metric for evaluating the "realness" of photographs without doing human evaluation. This evaluation is done by Hayes and Efros, but sadly I did not have the time or the access to resources to adequately conduct human trials.

Areas of Improvement

There are numerous situations in which this system fails. We generally classify these errors into a couple of different categories.

1. The first category are failures of scene matching. These are situations in which the GIST descriptor identifies scenes that just don't belong together (i.e. filling a mask in a tropical scene using an image from the snow)
2. The second category are blending issues. These errors occur typically when a sub-optimal image patch is chosen that contains superfluous artifacts, or when the graph-cut algorithm chooses to include something it should not.

[1] This was the same number as Hayes and Efros for the purposes of comparison, although for smaller databases we suspect this number should be reduced as many of the matches beyond the 20th were quite bad

Figure 5: Mask and possible completions for grassy/forest scene

Figure 6: Mask and possible completions for ocean scene

Figure 7: An example of a high-level semantic issue. Notice the partial rabbit in the grass.

Figure 8: An example of a blending issue

3. The final category of errors includes issues with high level semantics. These are situations in which partial objects are included in the patch, such a part of a rabbit filling in a grassy scene because there is otherwise a good local context match. Since the algorithm has no notion of objects, this happens quite often.

We include some illustrative examples of each error category.

Final Thoughts

In general, I was quite happy with the output of the system. There was generally at least one reasonable completion for the vast majority of images that I would input, provided I was using input images that fell into the same semantic category as input images in my database. Largely, this algorithm is effective with lots of data, but not generally effective for solving the scene completion problem on a resource-limited budget.

Acknowledgments

I would like to thank Silvio Savarese for an awesome class and to the entire teaching staff for making a really strong effort to improve the class, even throughout the quarter.

References

[1] Hayes, J. and Efros, A. Scene Completion Using Millions of Photographs. SIGGRAPH, 2007. http://graphics.cs.cmu.edu/projects/scene-completion/scene-completion.pdf

Figure 9: An example of a scene matching issue. The forest does not belong in the city!

[2] Irani, M., Anandan, P., and Hsu, S. 1995. Mosaic based representations of video sequences and their applications.

[3] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. 2004. Interactive digital photomontage. ACM Trans. Graph. 23, 3, 294–302.

[4] Oliva, A., and Torralba, A. 2006. Building the gist of a scene: The role of global image features in recognition. In Visual Perception, Progress in Brain Research, vol. 155.

[5] Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: a holistic representation of the spatial envelope. In International Journal of Computer Vision, vol 42 (3). https://people.csail.mit.edu/torralba/code/spatialenvelope/