Implementing Regenerative Morphing

Jenny Lam

December 11, 2010

The goal of my project was to implement the algorithm proposed in the paper [SRAIS10] in Matlab. In this paper, we will highlight the main steps of this algorithm, point out places where ambiguity arose in the implementation, and present a few results.

1 Bidirectional Similarity

Given two source images S_1 and S_2, the regenerative morphing algorithm will generate $N-1$ intermediate images that together smoothly transition between S_1 and S_2. The hallmark of this algorithm is its lack of ghosting features, which typically occur when certain parts of an image fade away as they transition into another image.

To achieve this result, we try to minimize the bidirectional similarity between two images in several ways. The bidirectional similarity of two images S and T of possibly different sizes is defined in [SCSI08] as

$$d_{BDS}(S,T) = \overbrace{\frac{1}{N_S}\sum_{s\subset S}\min_{t\subset T} D(s,t)}^{d_{\text{Completeness}}(S,T)} + \overbrace{\frac{1}{N_T}\sum_{t\subset T}\min_{s\subset S} D(s,t)}^{d_{\text{Coherence}}(S,T)}$$

where $s \subset S$ and $t \subset T$ represent arbitrary patches in S and T respectively, and N_S and N_T represent the number of such patches. The distance $D(s,t)$ between two patches with the same dimensions $m \times n$ is the average squared difference between corresponding pixels:

$$D(s,t) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \|s(i,j) - t(i,j)\|^2$$

where $s(i,j)$ is the ij-th pixel of s represented by a vector in RGB space. Obviously, if the two images S and T have the same size, the distance is minimized when the two images coincide. However, when the two images have different sizes, the minimal distance is not achieved by a straight scaling. Instead, the completeness term requires that the second image T represent all the main features of the first image S, and the coherence term requires that T not contain anything that did not appear in S. In other words, T must represent S faithfully, without omitting essential features of S and without introducing features not already present in S. The result is that, if T is smaller than S, then T will be a summary of the features in S. And if T is larger than S, then T will synthesize parts of S in a repeated

fashion. However the features themselves will not be scaled up or down.

In the morphing algorithm, bidirectional similarity is used to synthesize a new image from an old one. Starting with a source image S and an initial guess for the target image T, we modify T to minimize d_{BDS}:

$$T_{\text{out}} = \arg\min_{T} d_{BDS}(S, T)$$

At each iteration, we compute the nearest (i.e. most similar) patch correspondence f from T to S and the nearest patch correspondence g from S to T using the approximate nearest neighbor described in the following section. Each pixel $T(i, j)$ of T is updated as follows:

1. For each patch $t \subset T$ containing pixel $T(i, j)$, find the nearest corresponding patch $f(t) \subset S$ in the source image. Record the pixel $S(i', j')$ in $f(t)$ has the same position relative to $f(t)$ as $T(i, j)$ has relative to t.
2. Conversely, for each patch $s \subset T$ whose nearest corresponding patch $g(s) \subset T$ contains $T(i, j)$, we record the pixel $S(i', j')$ in s that has the same position relative to s as $T(i, j)$ has relative to $g(s)$.
3. Pixel $T(i, j)$ is updated to be the average of all the pixels $S(i', j')$ found in parts 1 and 2. The pixels found in part 1 have weight $1/N_T$ and those found in part 2 have weight $1/N_S$.

Source Initial guess Target

In the example above, we used an image of clouds for S and an image of shells for the initial guess for T. The algorithm just described minimizes the bidirectional similarity to the third image above, where the clouds are shaped like shells. This is because all the patches used to build that image come from the source, but are chosen to be placed where they best match the patches of the initial guess. This example also illustrates the extreme sensitivity of the algorithm to initialization. For good convergence, the initial guess must be close enough to the correct output.

Source Initial guess Target (3×3) Target (7×7) Target (11×11)

In the example above, we illustrate the influence of the patch size on bidirectional similarity. We use a source image S and initial guess for the target output T which is obtained as the average of the source image and another image of flowers. We minimize bidirectional similarity on 3×3, 7×7 and 11×11 size patches. We observe that as the patch size becomes larger, the target image becomes more defined. Although the smallest patch size produces a blurrier image, it also produces one that better "remembers" the initial guess. Notice that the yellow flower in the middle of the 3×3 target image still looks split, reflecting the three yellow flowers stacked on top of each other in the initial guess. Notice that in all three output cases, the initial "ghost" flowers appearing at the top of the initial guess have become pink flowers.

The regenerative morphing algorithm will use the update rule just described to minimize d_{BDS}. It will also need to minimize the two components of d_{BDS}, coherence and completeness, separately.

To minimize $d_{\text{Completeness}}(S, T)$, we again find the nearest patch correspondence g from S to T. As in the previous update rule, each pixel $T(i, j)$ of T is updated by finding all patches of S whose corresponding patch in T contains $T(i, j)$ and find the corresponding pixels in S. We take the average of all these pixels of S equally weighted.

A variant of $d_{\text{Coherence}}$ is also needed, called the disjoint coherence term. This similarity measure works between two source images S_1 and S_2 and a target image T that must be updated. It uses a parameter $\alpha \in [0, 1]$ to adjust the similarity of T to S_1 and S_2. More specifically,

$$d_{\alpha-\text{DisjointCoherence}}(T, S_1, S_2, \alpha) = \frac{1}{N_T} \sum_{t \subset T} \min \left(\min_{s_1 \subset S_1} D(s_1, t), \min_{s_2 \subset S_2} D(s_2, t) + D_{\text{bias}}(\alpha) \right)$$

where $D_{\text{bias}}(\alpha)$ is computed as follows: compute the distances between each patch $t \in T$ and its two nearest neighbor patches $f_1(t)$ in S_1 and $f_2(t)$ in S_2. Call these distances $d_1(t)$ and $d_2(t)$. Let Diff be the distribution of values $d_1(t) - d_2(t)$ over all patches $t \subset T$. Then $D_{\text{bias}}(\alpha)$ is the α-quantile of Diff. That is, it is the value x such that

$$\Pr[\text{Diff} < x] < \alpha.$$

In other words, the term $D_{\text{bias}(\alpha)}$, when added to the second term $\min D(s_2, t)$, ensures that a percentage α of the patches of T will be matched with those in S_2 and therefore that the remaining patches of T will be matched with those of S_1. Disjoint coherence allows each patch of T to be matched with either one in S_1 or one in S_2, but not both simultaneously. This mechanism provides a way to combine two images S_1 and S_2 in a controlled fashion that furthermore preserves a crisp output image T.

2 Approximate nearest neighbor

In minimizing d_{BDS}, the most time-consuming task is to find all the nearest neighbor patch correspondences from one image S to another image T. We cannot simply scan through all patches of T to find the nearest one to a given patch $s \subset S$ because the task takes on the order of several hours.

Instead, we use the more efficient approximate algorithm described in [BSFG09]. Starting from an initial random assignment f from the set of patches of S to the set of patches of T, we repeatedly improve this assignment by alternating between a propagation step and a random search.

During the propagation step, we look at a patch s in S and its assignment $f(s) \subset T$. We also look at s's left neighbor $\tilde{s}$, and its assignment $f(\tilde{s})$. We compare the distance between s and $f(s)$, and the distance between s and the right neighbor of $f(\tilde{s})$. If the latter distance is smaller, we update $f(s)$ to be that right neighbor of $f(\tilde{s})$. We likewise look at s's top, bottom and right neighbors to improve the assignment $f(s)$.

During the random search step, we try to improve $f(s)$ by randomly looking at patches in a window around $f(s)$ and narrowing the window repeatedly until it becomes the size of a patch.

The propagation and random search steps are applied to each patch in S. The patches are scanned from the top down and left to right on odd iterations, and in the reverse order on even iterations. Also, to increase the speed of the algorithm, we only look at the left and top neighbors on odd iterations, and look at the bottom and right neighbors during even iterations.

To further improve the efficiency of this algorithm, we keep track of the current distances between every patch $s \subset S$ and its assigned patch $f(s) \subset T$. Moreover, the computation of the distance between two patches was implemented in C. If the partial computation of the distance is greater than the distance to the currently assigned patch $f(s)$, the computation halts and returns.

3 The Regenerative Morphing Algorithm

The main algorithm provided in [SRAIS10] is reproduced here.

```
Initialize: {T_n(0)}_1^{N-1}
 For each scale  s = 1 : S,
  For each iteration  i = 1 : K,
   For each frame  n = 1 : N − 1
      T^1 ← min  d_BDS(T_n(i), T_{n−1}(i))
      T^2 ← min  d_BDS(T_n(i), T_{n+1}(i))
      T^3 ← min  d_Complete(T_n(i), S_1)
      T^4 ← min  d_Complete(T_n(i), S_2)
      T^5 ← min  d_αDisjCohere(T_n(i), S_1, S_2, α)
      T_n(i+1) = (Σ_{j=1}^5 w_j T^j) / (Σ_{j=1}^5 w_j)
  Upscale  s → s + 1
Output: {T_n(i)}_1^{N-1}
```

Given two images S_1 and S_2, we produce $N-1$ transition images $T_n(i)$ $(1 \leq n \leq N-1)$ after i iterations. In the initialization step, we produce transition images $T_n(0)$ by a simple crossfading between S_1 and S_2 which we assume to be the same size. By crossfading, we mean interpolating color values at corresponding pixels. An alternative to this approach is to interpolate patch correspondence locations as suggested in [SRAIS10]. However, we were not able to implement this initialization successfully. Specifically, it was not clear how to initialize

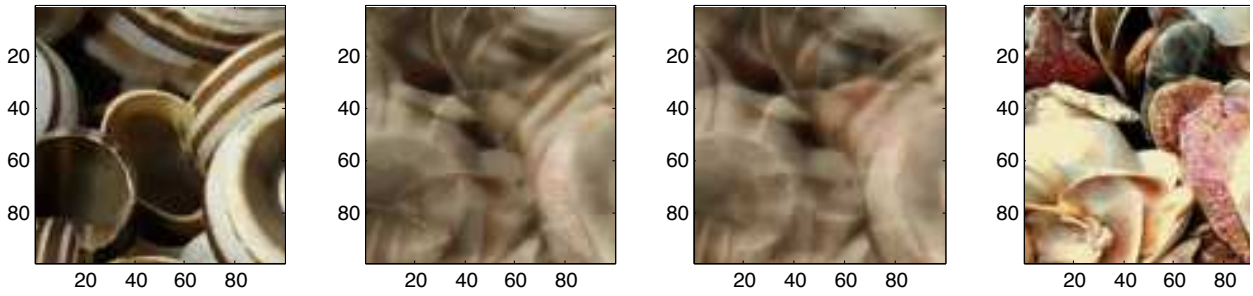
pixels that did not belong to any interpolated location between patch correspondences. Since this initialization is, according to [SRAIS10], more successful than crossfading, a possible solution is to initialize the pixels that are not defined by patch correspondence interpolation to their crossfaded values.

We did not implement the upscaling between groups of iterations through the frames because the computation were already taking about an hour. Instead we started with frames at full resolution. We performed two iterations. For each frame, we compute five images that minimize the bidirectional similarity with neighboring frames and the α-relative bidirectional similarity with the source images and take a weighted average, with weights

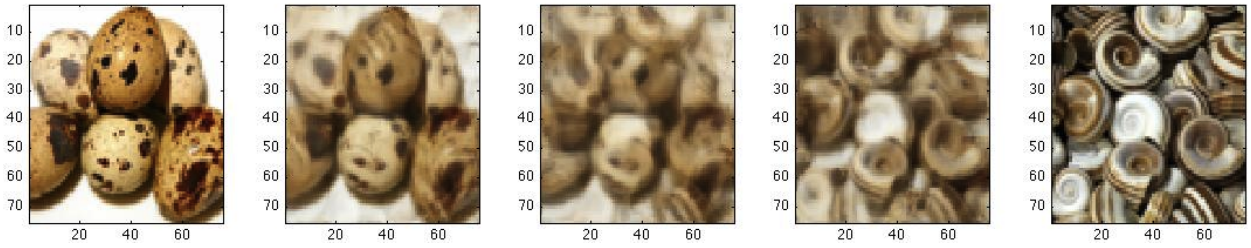
$$w_1 = w_2 = 0.5, \quad w_3 = \alpha, \quad w_4 = 1 - \alpha, \quad w_5 = 1.$$

4 Results

In the following result, we used 7×7 patches and 100×100 source images. It took 50 minutes to produce two transitional images.



In this second result, we used 7×7 patches and 75×75 source images. It took over an hour and a half to produce three transitional images.



5 Conclusion

We were able to implement the algorithm described in [SRAIS10] with partial success. While we were able to provide transitions between two images, these images were still blurry. The implementation was not efficient compared to the results achieved in [SRAIS10]. We were only able to compute transitions for small images (on the order of 100 by 100 pixels). We suspect that a faster implementation, possibly in a different language or directly on the

GPU as suggested in the paper, would achieve a better performance. We also suspect that on larger images, the blurriness will be toned down.

This project was an assignment for the course CS 216, Image Understanding, taught by Charless Fowlkes in Fall 2010 at the University of California, Irvine.

References

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[SCSI08] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *CVPR*. IEEE Computer Society, 2008.

[SRAIS10] Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steve Seitz. Regenerative morphing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San-Francisco, CA, June 2010.