

Technical Perspective

A Perfect ‘Match’

By William T. Freeman

WHAT MAKES AN array of pixel intensities look like a realistic image? How can you invent a set of plausible-looking image values in order to remove noise or to fill in missing regions of an image? These are problems the vision and image processing community has been struggling with for many years. Many different analytic approaches have been tried, but they seldom capture the richness and subtle details needed to produce realistic images.

To date, the best method to generate image data has been a surprisingly simple one: copy image values from somewhere else. This would seem to be too restrictive—how could one image patch possibly be a close enough match to another one to be useful? But it turns out that small image regions are essentially reusable parts, appearing, with small changes, many different times within an image or a set of images. If the patch is small enough, very good estimates of unknown image values can be found by extracting pixels from a patch with similar neighboring image values. The computer graphics community discovered this in the late 1990s and early 2000s, leading to an explosion of texture synthesis papers based on such sample-based methods.

This paradigm for finding good image values works not just for texture synthesis, but for many different image manipulations, too. The approach has been used for problems ranging from super-resolution to texture transfer, filling in, image editing, noise removal, and object detection. The source of image patches can be other regions of the image being processed, or other images. Sample-based image priors are ubiquitous in modern image processing and image synthesis.

Unfortunately, this powerful approach for image processing has a serious performance bottleneck: poten-

tially for each pixel to be processed, one must find the database patch that is closest to the image data surrounding that pixel. With a naive approach, and searching for matches within the same image, the search cost can be quadratic in the number of image pixels.

Fortunately, it is seldom the case that the true nearest neighbor must be found; a patch that is very similar to the target patch is often all that’s needed. This allows for existing fast, approximate nearest neighbor methods from the discrete algorithms community to be used. But while these methods help quite a bit, they often don’t help enough, and we are still left with algorithms that may be too slow for interactive applications.


Which brings us to the breakthrough contributions in the paper that follows. The authors have developed an efficient way to find approximate nearest neighbors for the case of patches within image data.

Their advance resulted from two main insights. The first is the observation, also noted by others, that the best matches from two spatially neighboring positions are usually two spatially neighboring patches from the database region. This pres-

The authors have developed an efficient way to find approximate nearest neighbors for the case of patches within image data.

ents a fast method to guess a matching patch, given the match to the spatial neighbor, but such an approach can get stuck in solutions that are only locally optimal. The authors’ fix to that comes from their second insight, delightfully counterintuitive: looking for a matching patch at *random* positions in the database region eventually finds good matches. Their “patch match” algorithm combines these approaches—deterministic update of a previous solution while allowing improvements from random guesses—to give a fast, approximate nearest neighbor algorithm for image patches that avoids getting stuck in bad solutions.

The breakthrough of the algorithm is its processing speed, which, for the first time, allows interactive use of some remarkable image editing algorithms that were previously restricted to slow, batch processing. The authors applied their algorithm to many image processing tasks, showing a broad range of applications.

The paper opens up algorithmic and theoretical questions. The scaling behavior with patch size is not known, nor is the best trade-off known for many of the choices made by the authors. But the work is having a large impact in the vision and graphics communities, both for the algorithm itself, and as an example of a class of algorithms to explore. It is unusual that commercial success follows so closely after an academic paper, but that happened in this case. The patch match algorithm is behind the release-defining “content-aware fill” feature of Adobe Photoshop CS5. 

William T. Freeman (billf@mit.edu) is a professor of computer science and Associate Department Head of the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, MA.

© 2011 ACM 0001-0782/11/11 \$10.00