Synthesizing Structured Image Hybrids

Eric Risser[†][*] Charles Han[‡] Rozenn Dahyot[†] Eitan Grinspun[‡]

[†]Trinity College Dublin [‡]Columbia University

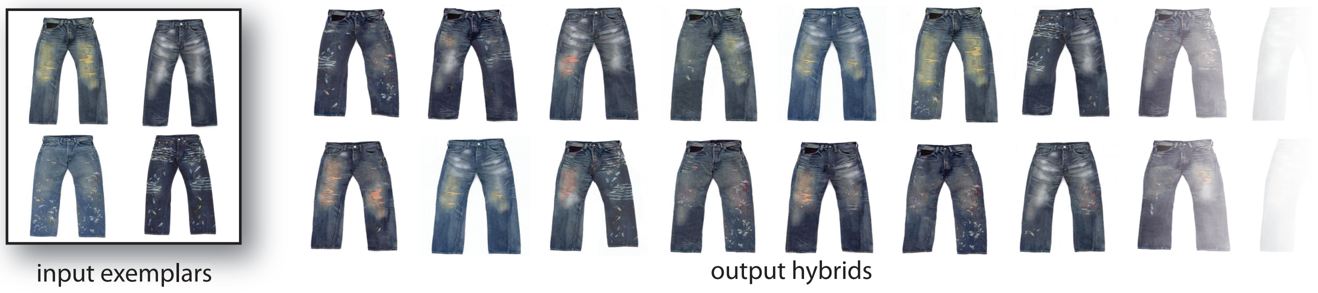


Figure 1: Image hybrids. *Given a set of input images* (left)*, our algorithm automatically produces arbitrarily many hybrid images* (right).

Abstract

Example-based texture synthesis algorithms generate novel texture images from example data. A popular hierarchical pixel-based approach uses spatial jitter to introduce diversity, at the risk of breaking coarse structure beyond repair. We propose a multiscale descriptor that enables appearance-space jitter, which retains structure. This idea enables repurposing of existing texture synthesis implementations for a qualitatively different problem statement and class of inputs: generating hybrids of structured images.

1 Introduction

We rarely see the same thing twice. No two beetles have the exact same form and markings; every human face is different; and even the most talented artist cannot exactly draw the same thing twice. Modeling the richness of the world as we see it requires incorporating diversity in generated images. We pursue an algorithm that produces arbitrarily many individual variants, or *hybrids*, given only a handful of example images.

Automated synthesis is broadly explored for the special case of *textures*, whose perceived semantics are carried by the finer scales [DeBonet 1997; Wei et al. 2009]. For data that requires "getting the structure right" at all scales, solutions are less understood. Do textures and highly structured images require *fundamentally* different synthesis algorithms? To answer this question, we attempt a *simple* extension to the parallel, controllable texture synthesis framework of Lefebvre and Hoppe [2005; 2006], extending a method designed only for textures to operate on a broader class of images whose semantic content is distributed across scales.

[*]rissere@cs.tcd.ie

Contributions We extend Lefebvre and Hoppe's work in two ways. First, we use a multiscale descriptor to measure neighborhood distance. Second, we propose a structure-preserving jitter step that operates in the *appearance* rather than *spatial* domain. The first extension helps maintain and repair synthesized image consistency; the second introduces variation *without* breaking structure. The combined extensions make for an algorithm that handles images far outside the domain of the original approach.

Related works Several earlier pixel-based approaches treated global structure and inhomogeneity, either by incorporating user input [Hertzmann et al. 2001; Ramanarayanan and Bala 2007; Wei et al. 2008], by incorporating specialized domain knowledge [Liu et al. 2004; Dischler and Zara 2006; Mohammed et al. 2009], or by automatically encouraging formation of coarse features in coarsely varying textures [Kopf et al. 2007; Han et al. 2008; Rosenberger et al. 2009]. Like the latter, we pursue an automatic approach, but we are distinct in our focus on highly structured images. Another line of works is designed to combine multiple exemplars [Wu et al. 2006; Matusik et al. 2005; Zhang et al. 2003]; this goal inspires our focus on hybridization.

Multiresolution local image descriptors are often used in vision and graphics. A number of successful early methods modeled textures using the joint coefficient distributions of various wavelet families [Heeger and Bergen 1995; DeBonet 1997; Portilla and Simoncelli 2000; Bar-Joseph et al. 2001]. Recent research in the field has been favoring operations in the pixel domain over wavelets; accordingly, there has also been a move away from the use of multiscale descriptors. While many *hierarchical* approaches operate across a range of image resolutions, these algorithms all typically use single-scale descriptors at any given stage of synthesis. In fact, the method by Wei and Levoy [2000] is the only other pixel-based synthesis method we are aware of that employs a multiresolution descriptor, albeit spanning only two resolutions.

2 Method

We build on the method of Lefebvre and Hoppe [2005], and follow the notations established therein. Following the hierarchical synthesis paradigm, the output is generated in a series of images, $S_0, S_1, \ldots, S_L$, of increasing resolution. The image pixels, $S[p]$, are not represented as RGB values but rather as coordinate pointers into the exemplar, such that the rendered color at pixel p is $E[S[p]]$. Moving from coarse to fine, three steps are executed at each pyramid level: **upsample** the previously generated level, **jit-**

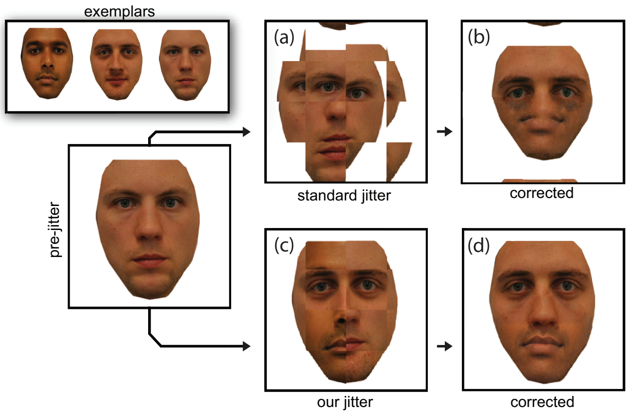


Figure 2: Jitter. *The traditional jitter method* (a) *is causes breaks in image structure which may not be repairable by correction* (b)*. Our structure-preserving jitter* (c) *adds variation while maintaining structural coherence* (d)*. Note that, although jitter occurs here at a coarse level, the prejitter and jitter steps are shown upsampled to full resolution for clarity.*

ter the coordinate image to add variation, and **correct** local appearance through a series of relaxations. The correction step requires a search for similar texture neighborhoods—an expensive high-dimensional nearest-neighbor search—which is accelerated in two ways: An offline **precomputation**, where k nearest-matching patches are selected as *candidates* for an exemplar patch, reduces the requisite number of comparisons during the online synthesis. Truncated principal component analysis (PCA) reduces the high-dimensional pixel neighborhoods to six dimensions. Following Han *et al.* [2008], we extend this framework to multiple exemplars by augmenting the coordinate pointers with exemplar indices.

Precomputation We used the PatchMatch [Barnes et al. 2009] algorithm for nearest-neighbor search during precomputation, slightly altered so that matches found in each candidate set were separated by at least 5% of the image size [Zelinka and Garland 2002]. To encourage a diversity of choices at synthesis time, we construct candidate sets with equal numbers of candidates *per exemplar* (choosing one or two candidates per exemplar usually suffices, in our experience).

Structure-preserving jitter Jitter introduces variation during synthesis. The standard, spatial approach of Lefebvre and Hoppe [2005] adds a small randomized offset to each coordinate (Fig. 2a). Jitter is applied independently to each texel, and thus typically breaks or distorts features; the correction step then attempts to repair them. However, when jitter is applied too strongly (Fig. 2b), correction fails to restore structure. By contrast, our *structure-preserving jitter* judiciously introduces variation without breaking structure (Fig. 2c,d), while serving as a drop-in replacement for spatial jitter in existing implementations.

To understand our jitter, recall that the correction step points each synthesized pixel to the closest-matching exemplar neighborhood's center. Similarly, our jitter randomly chooses *some* exemplar neighborhood from among those "sufficiently similar"—an attribute governed by the jitter parameter, J. This approach is related in spirit to earlier works in nonparametric texture synthesis which use neighborhood similarity to build candidate sets from among which random winners are chosen [Efros and Leung 1999].

At each synthesis pixel, p, we collect the 5×5 color neighborhood at the current scale level, $N_S(p)$ and project it into a truncated 6D PCA basis to obtain the reduced neighborhood $\tilde{N}_S(p)$. Mirroring the correction step, we collect precomputed candidate

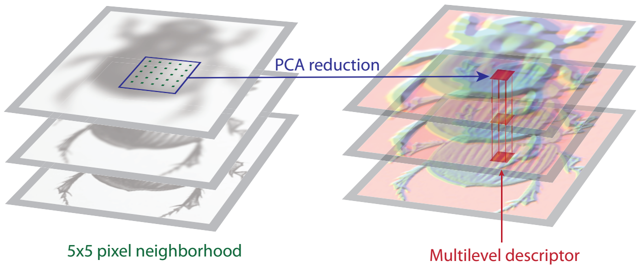


Figure 4: Schematic of descriptors. *We use PCA at each stack level to reduce the 75D ($5\times5\times3$) pixel neighborhoods to a 6D representation (shown here as RGB, right). We then concatenate PCA vectors of adjacent levels to form our multilevel descriptor. For illustration, we show here only three intermediate levels of our Gaussian stack (left).*

locations $\{c_i\}$ from each of the 3×3 neighboring texels. Employing k-coherence search [Tong et al. 2002], we evaluate the error $d_i = \|\tilde{N}_s(p) - \tilde{N}_E(c_i)\|^2$ associated with each candidate.

Letting $d_{\min}$ and $d_{\max}$ be the minimal and maximal errors over all candidates, respectively, we disqualify candidates whose normalized error $(d_i - d_{\min})/(d_{\max} - d_{\min})$ exceeds the given jitter threshold $J \in [0,1]$. We update $S[p]$ selecting uniformly at random among the qualified candidates. Intuitively, higher values of J cull fewer candidates thus encouraging variation (Fig. 3).

Observe that our jitter is heterogeneous and data-sensitive in behavior: it adds variation when safe to do so. For regions with many plausible matches, jitter selects among multiple candidates; at unique features having no close matches, jitter selects the best matching candidate—acting essentially as a correction pass.

Multiscale descriptors Our multiresolution descriptor embodies a compromise between expressiveness and simplicity. Drawing inspiration from more sophisticated models, we ultimately found that our simple PCA-based descriptor is (surprisingly) adequate for synthesis, while being simple to implement and naturally compatible with our implementation of hierarchical synthesis.

Both our new jitter step and the correction step rely on precomputed sets of best-match candidates for each pixel location in the input. The notion of "best-match" is not fixed, however, and depends on the choice of local descriptor; almost all pixel-copying texture synthesis methods use the single-scale local pixel neighborhood as the descriptor (*e.g.*, Lefebvre and Hoppe use 7×7 neighborhoods).

We extend the notion of a single-scale pixel neighborhood to a multiscale counterpart (see Fig. 4). If $\tilde{N}_{E_l}(p)$ is the neighborhood vector at pixel p in level l, our multilevel descriptor, $\hat{N}_{E_l}(p)$, is defined as the concatenation of neighborhood vectors at p from L successive finer levels (we use $L = 8$):

$$\hat{N}_{E_l}(p) = \left[\tilde{N}_{E_l}(p)\ \tilde{N}_{E_{l+1}}(p)\ \ldots\ \tilde{N}_{E_{l+L}}(p)\right].$$

Note that we use the 6D PCA-projected neighborhood vectors (hence the use of tilde notation) rather than the full 5×5 3 color 75D representations. Since precomputation already includes a PCA projection, these are available to us essentially "for free." At the finest levels where $l+L$ exceeds the Gaussian stack height, we truncate the multiscale neighborhood to include levels only up to the finest.

This multiscale descriptor improves on a straightforward starting point—the use of a larger neighborhood at a finer scale. In particular, it improves performance, since it gathers fewer pixels, and involves shorter vectors in the PCA step. As an added benefit, it avoids a potential pitfall of the large flat neighborhood approach:

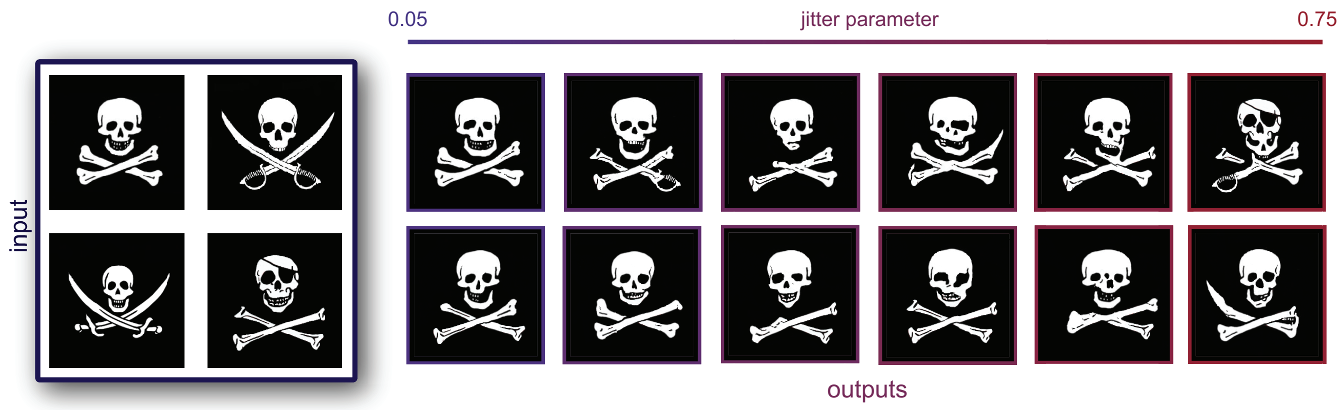


Figure 3: Variation control. *Qualitatively, higher values of the jitter parameter J (increasing towards the right) permit more variation in synthesized hybrids.*

it allows (e.g., equal) weighting of each scale, so that in particular fine features do not overpower information at coarse scales, as might occur in the most naïve flat implementation.

We provide some intuition for the role of multiscale descriptors within the overall method. *Synthesis* proceeds in a coarse-to-fine direction, at each point building on the information contained in the finest synthesized scale; thus we measure features of the synthesized image using only a single-scale descriptor. On the other hand, *analysis* has all scales available at once. Here a multiscale descriptor enables the analysis algorithm to make better choices at coarse levels, in effect anticipating finer levels. Candidate sets precomputed using the multiscale descriptor include candidates that match well both at the current level *and* at finer levels (see Fig. 6).

What is the ideal depth for a multiscale descriptor? An overly deep descriptor is costly and may introduce noise from its finest levels, while shallower descriptors retain the disadvantages of the single-scale setting. Heuristically, we have found that eight scale levels works well for all images we test, yet is efficient. It would be interesting to explore an adaptive method for choosing the descriptor depth, in particular one that focuses attention on *non-stationary features*, defined as follows. Points in an image at different scales can be classified on a gradient from completely stationary (stochastic) to completely non-stationary. When maintaining structure during synthesis it is our goal to replace highly non-stationary features with similar non-stationary features. To this end we must properly identify these similar features. However since images do not maintain the same stationary properties across scales (see Fig. 5) we are only interested in using enough scale levels to capture all non-stationary information. If we look deeper than necessary then we include stochastic information into our neighborhood which acts as noise with regards to describing structure. We have found that in practice this is not a severe problem and non-stationary features generally have more extreme differences than stationary features and thus overpower them in terms of feature distance. To illustrate the use of different scale levels we include examples of multiscale neighborhoods using one, two, four and eight scales Fig. 7.

Exemplar standardization In many applications such as object detection and recognition [Moghaddam and Pentland 1997], the scale, orientation or illumination variations across images are removed by a pre-normalization step, because they do not characterize the intrinsic properties of the object but rather the conditions of the acquisition. Similarly, we normalize our exemplars manually for scale and orientation—variations not modeled in our estimated probabilistic manifold. In addition, exemplars share a similar color palette in our experiments (e.g., faces, jeans, chess pieces, etc.): the intensity has been automatically normalized in all exemplars by computing the PCA on their (RGB) pixels and dividing by the dominant eigenvalue. More advanced automatic standardization techniques could be adopted following the object recognition literature.

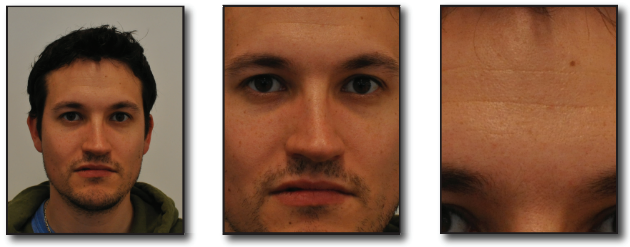


Figure 5: Stationary variation across scales. *A human face is highly non stationary at a scale which captures facial features. However, at very fine scales skin features such as pores become dominant and the image becomes highly stationary.*

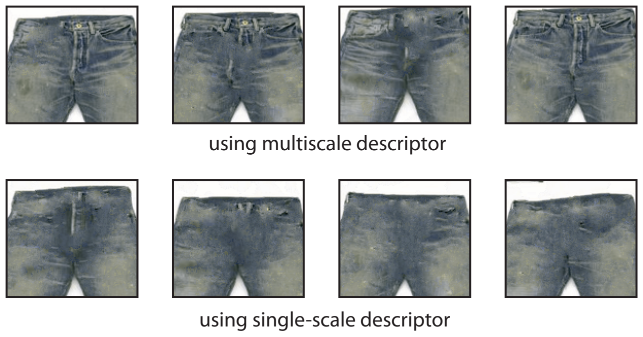


Figure 6: Multiscale vs single-scale descriptor. *Synthesis results using our multiscale descriptor* (top) *compared to the same example run with a single-scale descriptor* (bottom). *The addition of information at finer scales guides our candidate selection to more structurally meaningful location matches, better resolving sharp features such as a zipper.*

Symmetric synthesis For some examples (Figure 9, BUTTERFLIES, CHESS, FACES) we implemented an additional scheme to encourage symmetry. During the correction step, we augment the usual neighborhood distance measure with a term (shown in blue) that encourages symmetric formation of features:

$$d_i = \|\tilde{N}_s(p) - \tilde{N}_E(c_i)\|^2 + \|\tilde{N}_s(W_S - p) - \tilde{N}_E(W_E - c_i)\|^2.$$

Here, W_S and W_E are the respective widths of the synthesis and exemplar images.

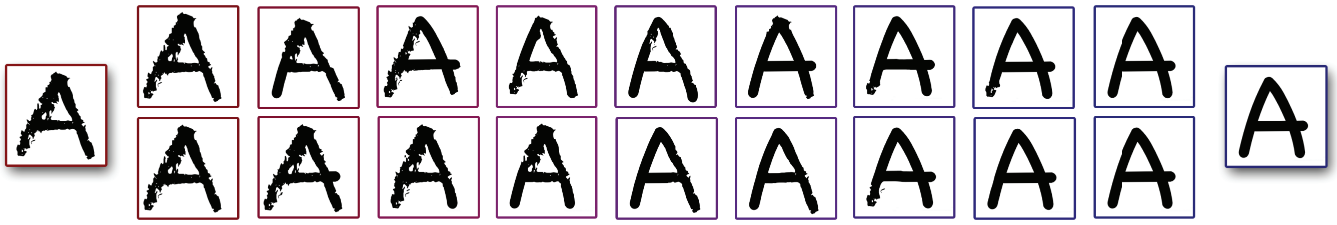


Figure 8: ***Exemplar Weighting.*** *Synthesis results from linear exemplar weight interpolation from leftmost exemplar image to rightmost.*

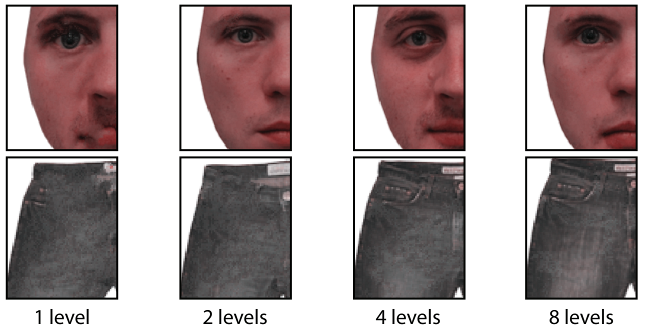


Figure 7: Progressively deeper multiscale descriptors. *Synthesis results using our multiscale descriptor with depths of one, two, four, and eight for human faces* top *and jeans* bottom*. Jean quality improves up to the eighth level, and face quality improves up to the fourth level.*

Exemplar Weighting Individual exemplar images can be given special emphasis during synthesis through the use of per-image weighting. The mechanism for this is the correction step where new link coordinates are chosen from our candidate set based on a neighborhood similarity error metric. By weighting neighborhood comparison error values based on the exemplar image the target neighborhood is taken from, we can favor or discourage a certain image's features in the final synthesized result. Exemplar weighting is a controllability mechanism which allows the user to direct the algorithm towards desired results. Exemplar weighting can exist for each image, for each image at each scale level, or per-pixel in each image using a weight map. It is also possible to randomly weight exemplars in order to promote diversity between outputs. Figure 8 illustrates the use of exemplar weights to favor specific images.

3 Discussion

Results Figure 9 depicts typical results for a variety of inputs, including drawings, photographs, objects with textured surfaces, solid shapes, both natural and anthropogenic. Our system produces an unlimited number of results, and we show representative examples from typical usage.

Inspired by *Visio-lization* [Mohammed et al. 2009], we explore the performance of our approach on images of human faces (FACES), a particularly challenging problem because of our attuned perception to facial structure. Despite that our algorithm does not "know" specifically about faces, we observe however that for highly standardized facial photographs, the resulting hybrids are surprisingly perceptually compelling. We attribute this primarily to the built-in prudence of our structure-preserving jitter. Likewise, in other organic examples (BUTTERFLIES) our system was able to recreate the natural templates defined within a given species.

We also tested our system on highly structured cases (CHESS; Fig. 8), noting that the resulting variations retain the rigid structure of the input. On slightly less structured inputs (PIRATE; Fig. 1), the system discovered useful, interesting, and even creative variations. For

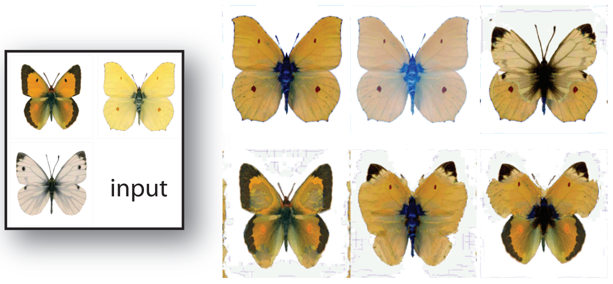


Figure 10: Color limitation. *Our rudimentary color normalization scheme can struggle when exemplars have radically different color palettes.*

even more loosely structured inputs (DOODLE, MONSTER), we were pleasantly surprised that we were able to recover not just important structural variations, but also to some extent artistic style.

Being a straightforward extension, our timings were roughly equal to those seen in our reference CPU implementation of Lefebrve and Hoppe's algorithm. Specifically we found that our method yielded a 30% computational overhead from the previous technique, which should follow for any implementation as well.

Limitations As our rudimentary color standardization scheme mainly addresses intensity normalisation between the exemplars, it can struggle when the multiple exemplars have greatly disparate color palettes (see Fig. 10). Such an input with widely varying colors carries an inherent ambiguity when defining the desired space of output colors, which will result in potentially unwanted variations. One potential solution is to increase the number of exemplars to more densely cover the desired space; however, this process can be tedious and computationally expensive. Normalization is another solution: the color variations can be reduced by transforming all exemplars so that they map to the same color palette, *e.g.* by using existing color transfer techniques [Pitié et al. 2007].

As explained in §2, the perceived randomness is controlled by a jitter parameter, J. The ideal range of values for J will vary depending on the particular input and desired randomness. Fortunately, since our structure-preserving jitter degrades gracefully into correction when randomness is inappropriate, we will still produce a usable, if not highly-varied image, for most reasonable settings.

Future work A number of methods have been developed for the direct manipulation of image structure. These allow a user to combine content from multiple source images [Agarwala et al. 2004] or to rearrange content within a given image [Cho et al. 2008; Pritch et al. 2009], possibly at interactive rates [Barnes et al. 2009]. These works are related to our method in that they seek to discover semantically plausible images, and we therefore view our algorithm as being complementary. We envision that a combination of these approaches with our structure-preserving synthesis will result in yet more powerful interactive editing tools.

Acknowledgements: We thank Carol O'Sullivan, Alex Berg, Alyosha Efros, and our anonymous reviewers for their valuable feedback on this work. This research is supported in part by

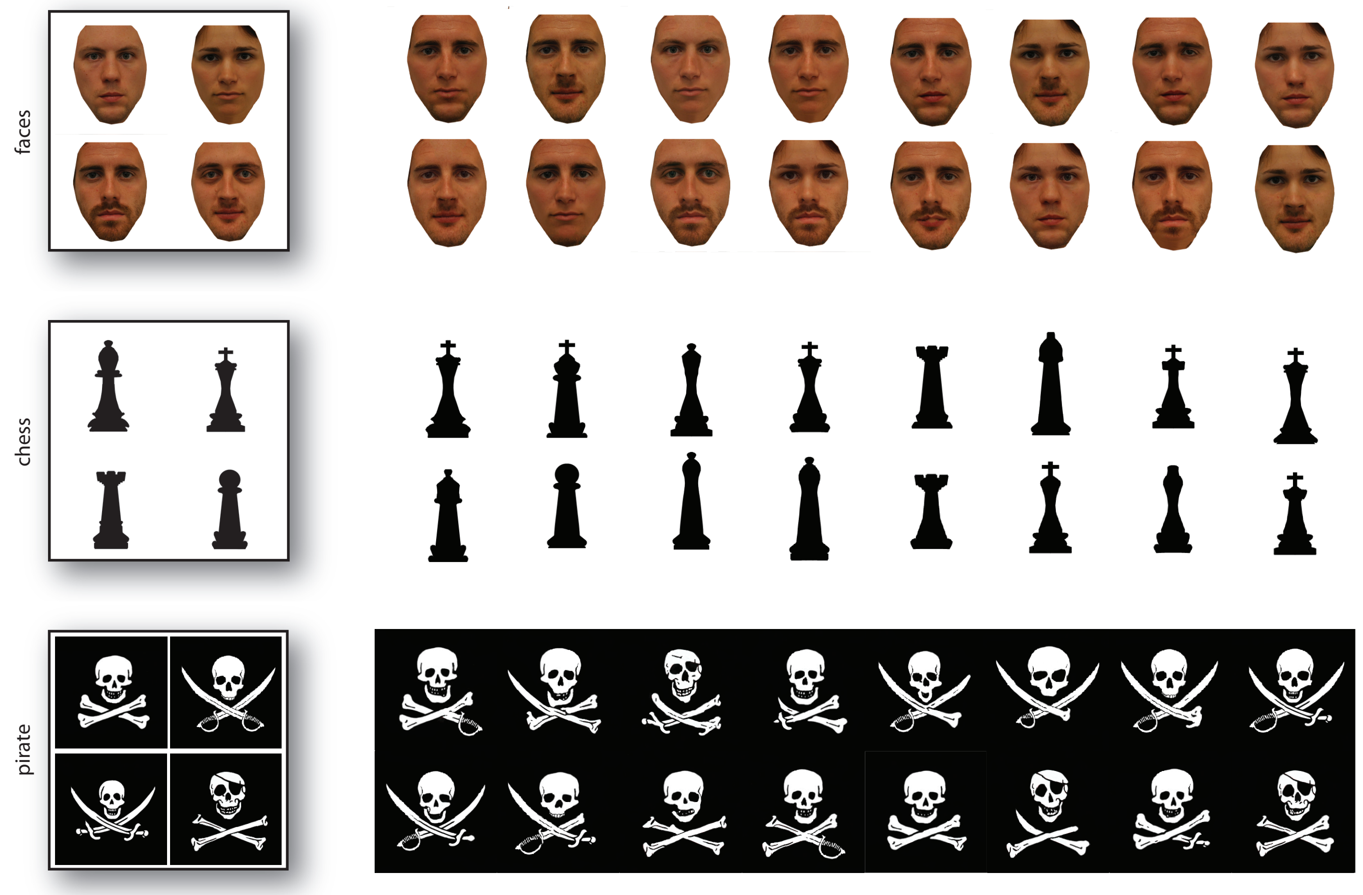


__Figure 9: Synthesis results.__ A series of example inputs and the results generated by our method.

Trinity College Dublin (Postgraduate Ussher Fellowship, Start-up Fund for New Lecturers 2009), the Sloan Foundation, the NSF (CAREER Award CCF-06-43268 and grants IIS-09-16129, IIS-05-28402, CNS-06-14770), and generous gifts from Adobe, ATI, Autodesk, mental images, NVIDIA, Side Effects Software, the Walt Disney Company, and Weta Digital.

References

Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. 2004. Interactive digital photomontage. *ACM Transactions on Graphics 23*, 3, 294–302.

Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., and Werman, M. 2001. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics 7*, 2, 120–135.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics 28*, 3, 3:1–3:11.

Cho, T. S., Butman, M., Avidan, S., and Freeman, W. T. 2008. The patch transform. In *IEEE Conference on Computer Vision and Pattern Recognition*.

DeBonet, J. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH 1997*, 361–368.

Dischler, J.-M., and Zara, F. 2006. Real-time structured texture synthesis and editing using image-mesh analogies. *The Visual Computer 22*, 9, 926–935.

Efros, A., and Leung, T. 1999. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, 1033–1038.

Han, C., Risser, E., Ramamoorthi, R., and Grinspun, E. 2008. Multiscale texture synthesis. *ACM Transactions on Graphics 27*, 3, 51:1–51:8.

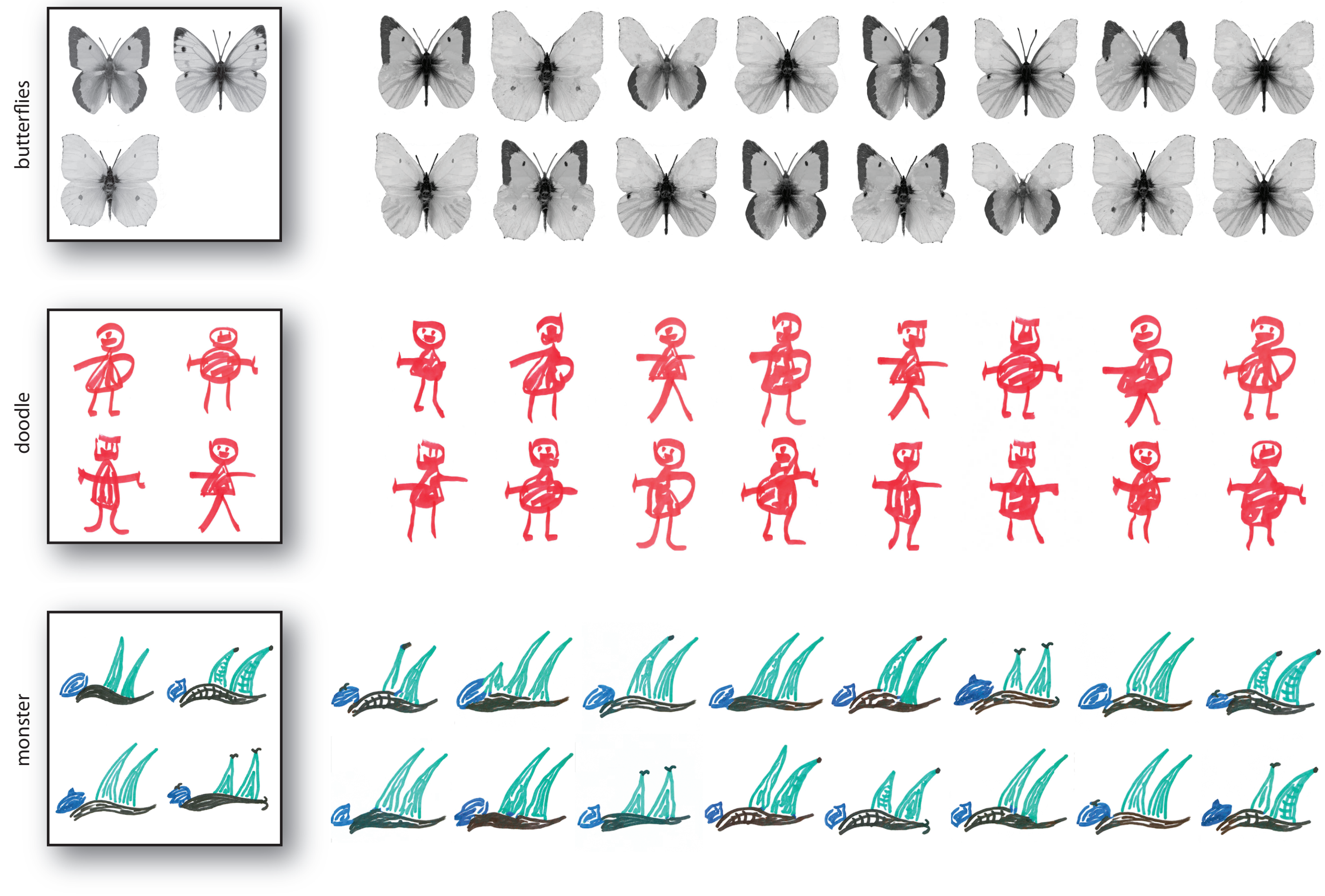
Heeger, D. J., and Bergen, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 1995*, 229–238.

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. 2001. Image analogies. In *Proceedings of SIGGRAPH 2001*, 327–340.

Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D., and Wong, T.-T. 2007. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics 26*, 3, 2:2–2:9.

Lefebvre, S., and Hoppe, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics 24*, 3, 777–786.

Lefebvre, S., and Hoppe, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3, 541–548.



Liu, Y., Lin, W.-C., and Hays, J. 2004. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics 23*, 3, 368–376.

Matusik, W., Zwicker, M., and Durand, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics 24*, 3, 787–794.

Moghaddam, B., and Pentland, A. 1997. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7.

Mohammed, U., Prince, S. J., and Kautz, J. 2009. Visio-lization: generating novel facial images. *ACM Transactions on Graphics 28*, 3, 57:1–57:8.

Pitié, F., Kokaram, A. C., and Dahyot, R. 2007. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1-2, 123–137.

Portilla, J., and Simoncelli, E. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision 40*, 1, 49–70.

Pritch, Y., Kav-Venaki, E., and Peleg, S. 2009. Shift-map image editing. In *International Conference on Computer Vision*, 151–158.

Ramanarayanan, G., and Bala, K. 2007. Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics 13*, 1, 167–178.

Rosenberger, A., Cohen-Or, D., and Lischinski, D. 2009. Layered shape synthesis: automatic generation of control maps for non-stationary textures. *ACM Transactions on Graphics 28*, 5, 107:1–107:9.

Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., and Shum, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3, 665–672.

Wei, L., and Levoy, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, 355–360.

Wei, L.-Y., Han, J., Zhou, K., Bao, H., Guo, B., and Shum, H.-Y. 2008. Inverse texture synthesis. *ACM Transactions on Graphics 27*, 3, 52:1–52:9.

Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009: State of the Art Reports*.

Wu, Q., Shi, L., Bond, S., and Yu, Y. 2006. Laplacian texture synthesis and mixing on surfaces. In *Pacific Graphics*.

Zelinka, S., and Garland, M. 2002. Towards real-time texture synthesis with the jump map. In *Eurographics Workshop on Rendering*, 99–104.

Zhang, J., Zhou, K., Velho, L., Guo, B., and Shum, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics 22*, 3, 295–302.