Interactive Evolution of Camouflage

Craig Reynolds*
Sony Computer Entertainment, US R&D

Keywords
Biology, interactive hybrid systems, evolutionary computation, genetic programming, texture synthesis, vision

A version of this paper with color figures is available online at http://dx.doi.org/10.1162/artl_a_00023. Subscription required.

Abstract This article presents an abstract computation model of the evolution of camouflage in nature. The 2D model uses evolved textures for *prey*, a background texture representing the *environment*, and a visual *predator*. A human observer, acting as the predator, is shown a *cohort* of 10 evolved textures overlaid on the background texture. The observer clicks on the five most conspicuous prey to remove ("eat") them. These lower-fitness textures are removed from the population and replaced with newly bred textures. Biological morphogenesis is represented in this model by *procedural texture synthesis*. Nested expressions of generators and operators form a texture description language. Natural evolution is represented by *genetic programming* (GP), a variant of the *genetic algorithm*. GP searches the space of texture description programs for those that appear least conspicuous to the predator.

1 Introduction

That animals often resemble their environment has been observed since ancient times. This sometimes startling visual similarity highlights the adaptation of life to its environment. Since the earliest publication on evolution, camouflage has been cited as a key illustration of natural selection's effect:

> When we see leaf-eating insects green, and bark-feeders mottled-gray; alpine ptarmigan white in winter, the red-grouse the colour of heather, and the black-grouse that of peaty earth, we must believe that these tints are of service to these birds and insects in preserving them from danger.—Charles Darwin, 1859, *On the Origin of Species* [9]

Natural camouflage appears to result from coevolution between predator and prey. Many predators use vision to locate their prey, so prey have a survival advantage if they are harder to see. Predators with superior vision are better able to find prey, giving them a survival advantage. Over time this leads to well-camouflaged prey and to predators with excellent eyesight and a talent for breaking camouflage.

The hypothesis for these experiments was that selection pressure from a visual predator will gradually eliminate the most conspicuous (least well camouflaged) prey from the evolving population. Prey would then converge on more effective camouflage. The results presented here in Figures 1, 2, and 3 lend support to this idea and point the way to more powerful human-computer

* Sony Computer Entertainment, US R&D, 989 E. Hillsdale Blvd., Foster City, California, USA. E-mail: craig_reynolds@playstation.sony.com

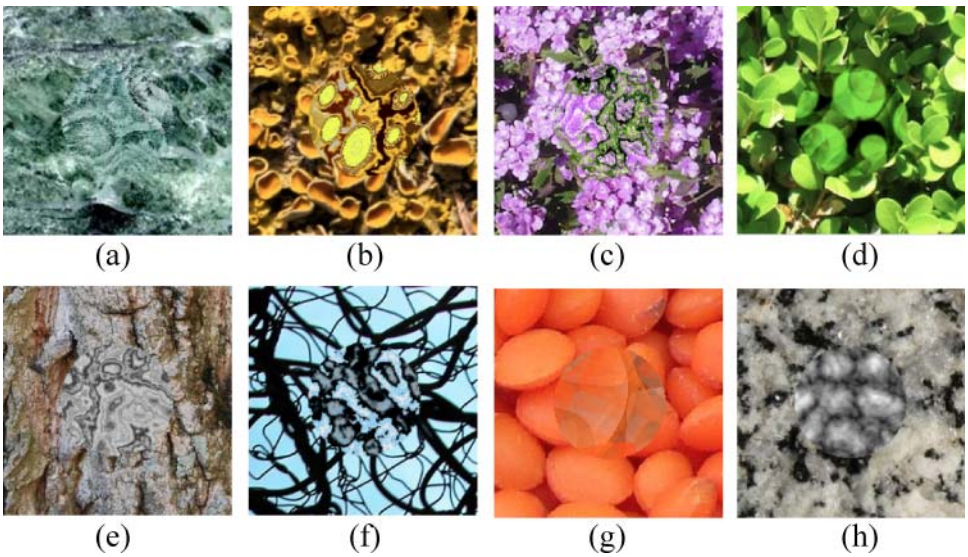


Figure 1. Camouflaged prey overlaid on the background image for which they were evolved: (a) "Serpentine," (b) "Yellow Lichen," (c) "Flowers," (d) "Green Hedge," (e) "Bark," (f) "Twisty Wires," (g) "Lentils," and (h) "Granite Yosemite." Each image shows a single circular prey (100 pixels in diameter) centered on a square neighborhood of the background image (200 × 200 pixels).

hybrid systems as well as future simulation studies of the coevolution of prey camouflage and predator vision.

As defined in [30], the term *camouflage* includes all strategies of *concealment*. To distinguish it from *hiding*, this is taken to mean reducing the chance of recognizing an animal that is otherwise in plain sight. Thayer describes [33] a bird "in plain sight but invisible." The more specific term *crypsis* refers to preventing initial detection, including the sort of *cryptic coloration* commonly implied by the term camouflage. For comparison, crypsis helps prey avoid detection, while *mimicry* protects by leading predators to misclassify prey after detection.

A common misconception about camouflage is that ideally it should match the background. This is generally untrue except for homogeneous environments like white snow or green leaves. Consider a large photograph of a natural texture such as the surface of a rock. Take a small square region of the large image and place it in a different random location on the large image. It will not be perfectly cryptic. Even small discontinuities of pattern at the edge of the cutout stimulate low-level edge detectors in the visual system, causing a strong perception of a square boundary. The goal of effective camouflage is to hide that edge.

Much recent work on camouflage (see next section) has focused on the importance of *disruptive camouflage*. While these patterns often echo colors and textures from the environment, their effectiveness comes from their ability to visually disrupt the silhouette of an animal. This can prevent a predator from recognizing that an object is an animal, or even prevent the detection of an object in the first place [26]. Surprisingly, even camouflage that does not match the colors of the background can be quite effective if it incorporates strong visual features (*false edges*) that intersect the object's real edges [29]. Most of the effective camouflage patterns evolved in these experiments appear to have disruptive qualities.

The work described here lies between computer science and evolutionary biology. This multidisciplinary middle ground is variously called *theoretical biology*, *mathematical biology*, or *artificial life*. Research in this middle area has the potential to benefit all related fields. From a computer graphics perspective, this could be seen as a special case of *goal-oriented texture synthesis* where new textures can be created from a description of desired image properties. To biologists, a computational model of camouflage evolution could allow new types of theoretical experiments to be conducted in simulation that are not subject to

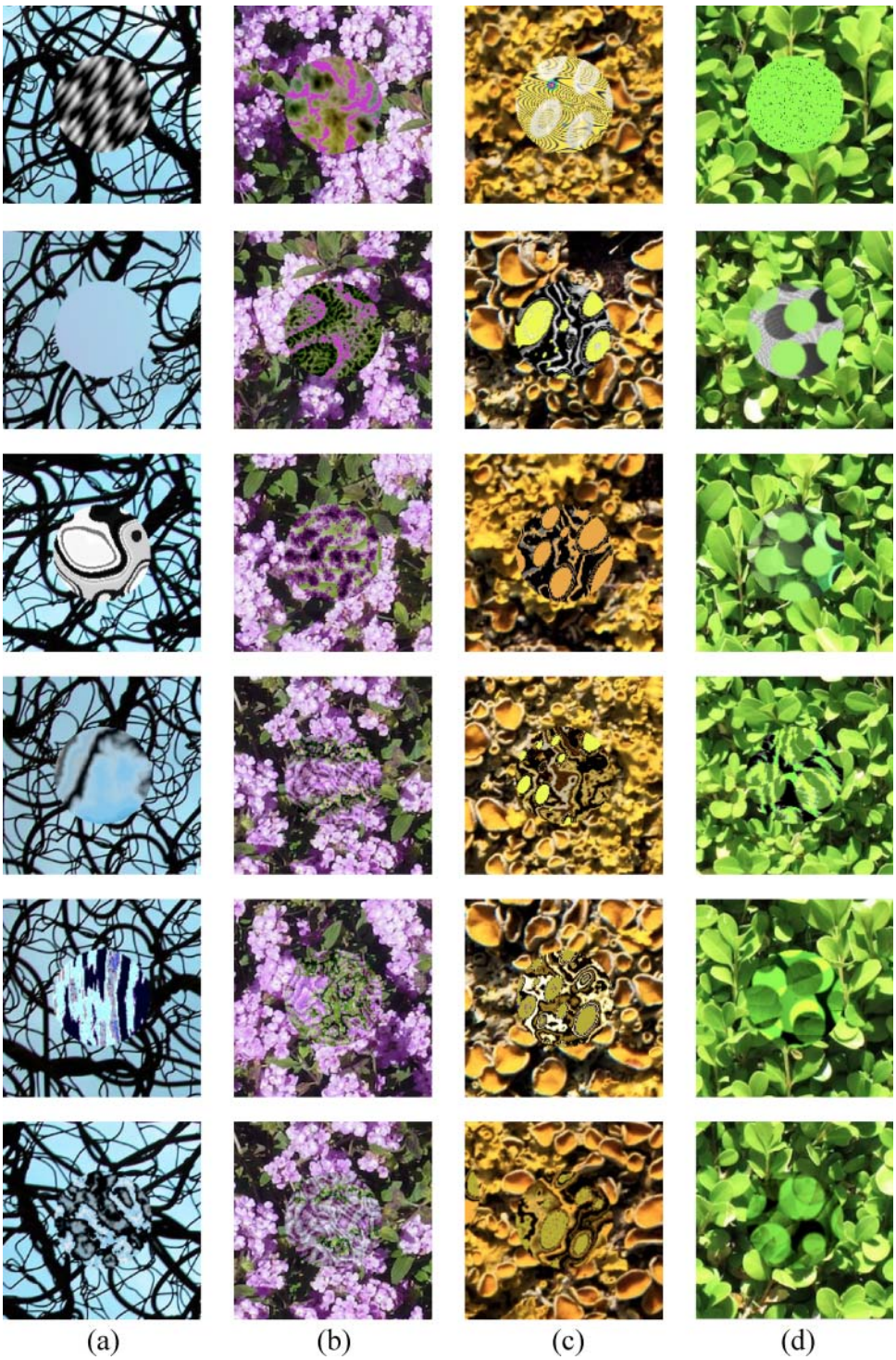


Figure 2. Time series showing progression (from top to bottom) of camouflage patterns during four runs of interactive evolution: (a) "Twisty Wires," (b) "Flowers," (c) "Yellow Lichen," and (d) "Green Hedge."

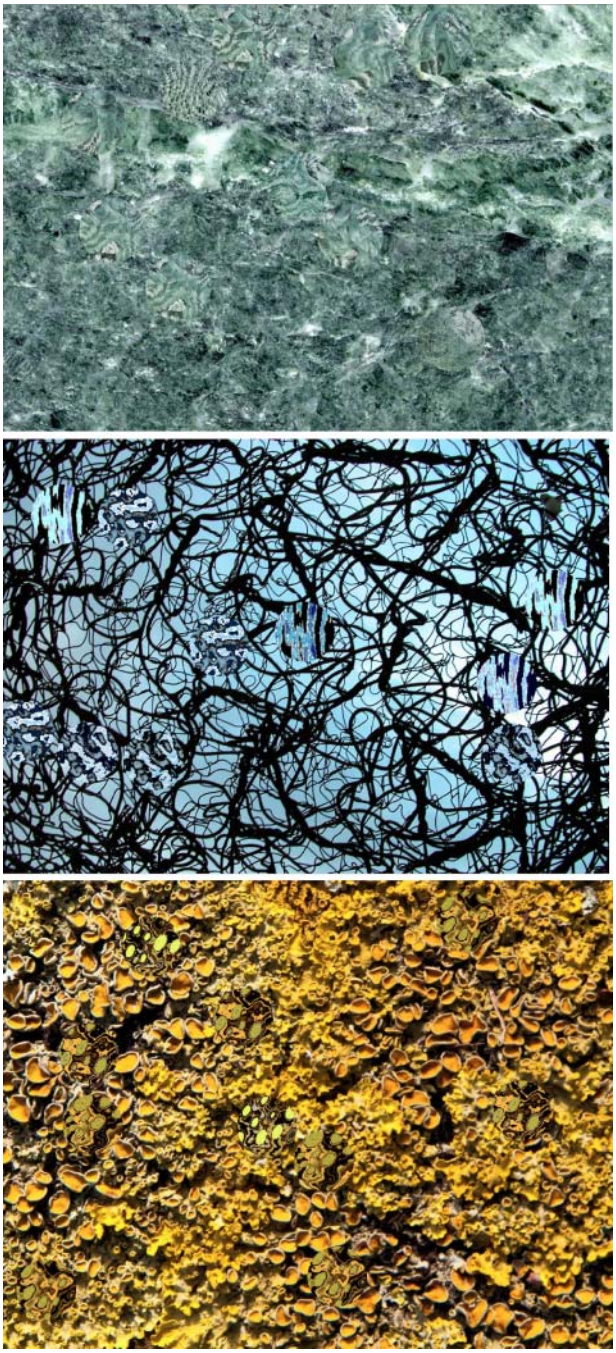


Figure 3. Screen captures showing interactive sessions with "Serpentine" (top) and "Twisty Wires" (middle), and "Lichen" (bottom) environments. In all three, we see the entire background texture, overlaid with a new cohort of 10 circular prey, each with its own evolved camouflage texture.

constraints imposed by working in the field, or with live animals, and in general such a model is not limited to examples found in Earth's biosphere.

2 Related Work

Over the last century several seminal works have surveyed the broad topic of camouflage in nature. These include Beddard's 1895 book [3], Thayer's 1909 work [33], and Cott's 1940 classic [6]. The last two continue to be widely cited today. Over the last 20–30 years there has been a significant renaissance in the understanding of camouflage. Before that, work in this area tended to be more descriptive than experimental. It is challenging to design well-controlled studies of the effectiveness of camouflage in either the field or the laboratory. Still, with careful design and patient experimentation, studies providing new insights have appeared regularly in the biological literature. For an excellent recent survey, see [30].

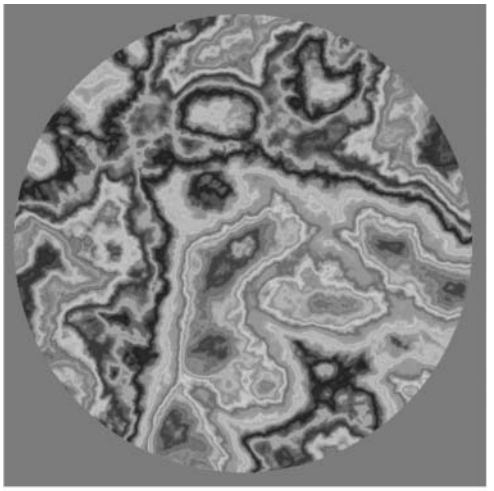
In the biologist John Endler's extensive field studies of guppies (polymorphic tropical fresh water fish) he saw wide variation in colorations from one stream to the next. Males in some streams were drab and inconspicuous, while those in other streams were elaborately colored. He came to correlate this with the presence or absence of predators. Drab color helps avoid being seen by predators, while bright colors appeal to the female guppies. Endler was able to reproduce this system in the lab [13]. Colorful guppies evolved to have cryptic coloration when kept in tanks with predators. Conversely, the drab-colored wild fish would evolve bright coloration in tanks without predators. These evolutionary changes took place over about a year, a remarkably short amount of time.

Of particular relevance to the work presented here are various experiments offering *artificial prey* to real predators. Many valuable results have been obtained with a similar experimental design involving cardboard "moths" [7] and avian insectivores: wild birds that naturally prey on moths. During the day these nocturnal moths rest on tree trunks protected by their cryptic wing coloration. Artificial moths are constructed with cardboard wings decorated using a color printer, a worm is attached to serve as an edible body, and the "moth" is attached to a tree trunk. A missing worm is taken to indicate that a wild bird detected and attacked the artificial moth. This technique has shown the key importance of disruptive coloration [26], measured the disadvantage of symmetrical camouflage [8], and shed light on several related topics.

Other experiments have used live captive birds [4] and humans [28] as predators of "virtual artificial prey" on a display screen. In both cases this predation was used to drive an evolutionary computation as in the work described here. In [21] artificial predators learn to detect artificial prey whose camouflage evolves to avoid detection. However the textures used are quite small, 4 to 8 symbolic pixels. A recent simulation-based study looked at a unique three-player camouflage game based on evolution of flower color [1].

The original idea of using an interactive task as the fitness function for an evolutionary computation goes back to the *Blind Watchmaker* software that accompanied [10]. That application displayed a grid of *biomorphs*, small tree-structured line drawings with a genetic description. The user picked a favorite, which was mutated several times to produce a new generation. Dawkins introduced the idea of intentionally evolving toward a goal, a biomorph he called the "holy grail." Karl Sims combined a similar approach with genetic programming and a rich set of image-processing functions to create an interactive system for aesthetic evolution of texture patterns [27]. In [14] and other articles, Jordan Pollack's DEMO group describe their TRON project, where game-playing agents were evolved in competition with each other and then in competition with human players over the Web. A survey of related techniques used to create game content is presented in [32]. A deep survey of the whole field of *interactive evolutionary computation* is found in [31].

This work conceptually overlaps the large body of work in *example-based texture synthesis*, also known as *texture extension*, which creates arbitrarily large texture patterns to match a small exemplar texture [36]. Using this technique to generate camouflage image puzzles is described in [5], where new textures are pieced together from samples of the background image. In contrast, the synthesis



```
Colorize (Ring (5.80532,
                Vec2 (-2.12073, 0.411024),
                Stretch (0.0449509,
                         -1.06448,
                         Vec2 (-1.37922, 0.946741),
                         Furbulence (1.21806,
                                     Vec2 (1.62529, 2.9815)))),
          Furbulence (1.21806,
                      Vec2 (-2.94693, -1.86416)))
```

Figure 4. Disruptive oak bark camouflage of Figure 1e, re-rendered at 600 × 600 resolution, with its evolved source code.

of camouflage texture described in this article does not "see" or otherwise access the input texture. Instead, the background can only be *inferred* from the indirect evidence of predation, as it presumably is in evolution of natural camouflage.

3 Texture Synthesis

In nature, patterns of surface coloration on plants and animals (see, e.g., [12]) result from complex genetic and developmental processes collectively called *morphogenesis*. In the camouflage evolution model described here, pattern formation is represented by procedural texture synthesis [11]. More specifically, this work uses *programmatic* texture synthesis. Textures are defined by nested expressions of generators and operators, forming a programming language for textures. *Generators* produce results of type Texture from simple parameter types (numbers, 2D vectors, and RGB colors). *Operators* are similar and have one or more Texture parameters. These nested expressions look like composition of functions (see Figure 4 and Figure 5),[1] although in this implementation they are actually constructors for C++ classes representing the various types of procedural textures. Once the tree of procedural texture objects is constructed, its root provides an interface for rendering pixels.

This texture synthesis library [25] brings together several preexisting techniques. Its generators include uniform colors and simple patterns like spots and color gradations. There are are several types of striped *gratings* (e.g., a sine wave grating) and an assortment of noise patterns such as *noise* and *turbulence* [24] plus several novel variations on these. The library's collection of texture operators includes simple geometrical transformations (such as scale, rotate, and translate), simple image-processing operations

1 Images in the printed edition of the journal are in black and white; full color images are available in the online version of the article and at the project's website: http://www.red3d.com/cwr/iec/.

```
Invert (SoftMatte (HueIfAny (Colorize (Twist (-1.76008, Vec2 (-2.90822, -1.26208),
Multiply (Brownian (0.880861, Vec2 (2.80615, 1.14405)), Wrap (6.21909, 5.55726,
Vec2 (1.88101, -1.10475), Add (VortexSpot (-2.95874, 4.37424, Vec2 (-2.24113,
-0.804409), Row (Vec2 (-1.20827, -0.80333), Wrapulence (5.81646, Vec2 (1.46969,
0.464754)))), Multiply (TriangleWaveGrating (15.0552, 0.251605, 4.92253), Wrap
(6.21909, 5.25948, Vec2 (-2.90822, -1.26208), Add (ColoredSpotsInCircle (146.485,
0.573184, 0.103147, Stretch (1.92016, 0.932767, Vec2 (0.994563, 1.8778), SineGrat-
ing (17.4233, 0.477075)), Translate (Vec2 (1.3634, -3.05406), Colorize (SineGrat-
ing (87.1581, 1.2438), SoftEdgedSquareWaveGrating (138.03, 0.0101831, 0.894823,
1.03307))), SliceToRadial (Vec2 (-1.20827, -0.80333), ColorNoise (1.09284, Vec2
(1.24907, -3.11514)))), Brownian (4.15562, Vec2 (-1.20827, -0.80333)))))))),
Brownian (0.880861, Vec2 (2.80615, 1.14405)))), SliceToRadial (Vec2 (-1.20827,
-0.80333), ColorNoise (1.09284, Vec2 (1.24907, -3.11514))), Colorize (Twist
(-1.90423, Vec2 (0.977825, -0.533419), Twist (-1.90423, Vec2 (0.977825,
-0.533419), RadialGrad (195.316, Vec2 (1.24907, -3.11514)))), Wrapulence (5.81646,
Vec2 (0.0918581, -0.543768))))
```

Figure 5. Camouflaged prey evolved on "Serpentine" background with its evolved source code.

(add; subtract; multiply; and adjust intensity, hue, and saturation), convolution-based operations (blur, edge detect, edge enhance), operators to produce multiple copies of a texture (row, array, ring), and a collection of image-warping operators (stretch, wrap, twist, etc.). Several operators use a 1D "slice" of a texture, such as colorizing one texture by mapping its brightness into colors along the $y = 0$ axis of another texture. Only convolution-based texture operators have fixed pixel resolution; all others use floating point coordinates. The complete texture synthesis API used in these experiments is listed in Appendix 1. Missing from the library are reaction-diffusion and other computationally expensive textures, awaiting a GPGPU implementation.

4 Evolutionary Computation

The texture synthesis library described in the previous section was designed for use with *genetic programming* (GP) [20]. Like the closely related *genetic algorithm* (GA) [17], GP is a stochastic technique for population-based (parallel) search and optimization in high-dimensional spaces. These *evolutionary computation* (EC) techniques are inspired by evolution in the natural world and share some of its attributes. While GP is used in this work as an abstract model of natural evolution, it is important to keep in mind the vast differences between the two. For example, natural evolution works with very large populations and very long time scales. Much of the engineering in evolutionary computation has to do with getting useful results without requiring billions of individuals or waiting millions of years.

A genetic programming system maintains a population of *individuals*, each of which represent a program expressed in the given grammar. In this work, each program defines a procedural texture. These programs can be thought of either as nested textual expressions of composed functions, or as a tree of functional nodes with constants as terminal leaves. The GP population is initialized to randomly generated programs. GP uses a given fitness function (objective function) to evaluate each individual. Fitness is used to select which individuals will reproduce to create new offspring programs to replace lower-fitness individuals in the population. New individuals are created by genetic operators such as *crossover* and *mutation*. GP crossover involves replacing a subnode of one program with a subnode of

another program. It is essentially random syntax-aware cut-and-paste between programs. Over time, programs containing beneficial code fragments become more numerous in the population. Crossover tweaks these programs, juxtaposing code fragments in new ways. While some changes improve fitness and some reduce fitness, the population is biased to collect the good and discard the bad.

For these experiments, genetic programming was implemented with the excellent open source library Open BEAGLE [15, 23]. This flexible framework provides support for many common types of evolutionary computation while also allowing customization of all aspects of the process. For example, Open BEAGLE supports the variation on GP used here that allows mixtures of data types known as *strongly typed genetic programming* (STGP) [22]. In addition, Open BEAGLE's structure allows changing its population replacement strategy operator and fitness evaluation operator to implement the novel *cohort fitness* used for interactive evaluation of relative camouflage effectiveness.

In these experiments GP populations consist of 100 or 120 individual texture programs. These are run, on average, for the equivalent of 100 generations using *steady state* replacement. So roughly 10,000 individuals are bred and have their fitness tested in 1000 cohorts of 10 individuals each. The population is divided into 4 or 5 *demes* (islands, isolated breeding populations, with occasional migration) of 20 or 30 individuals each. In addition to GP crossover between programs, the floating point constants in each program were subjected to incremental ("jiggle") mutation. Figure 6 shows early tests (before the interactive camouflage experiments) of evolved textures using an ad hoc fitness function. This fitness function merely measures simple image properties such as a somewhat uniform brightness histogram and some color variation. These textures were created automatically with no human in the loop; then interesting results were hand-selected for Figure 6.

5 Interactive Evaluation of Camouflage

The role of predator in these experiments is played by a human observer who visually compares the quality of evolved camouflage patterns. This happens in a simple graphical user interface. The user sees a blank window and clicks the mouse to begin a round of the camouflage game. The window is redrawn to show a background texture overlaid with a *cohort* of circular prey objects, each with its own evolved

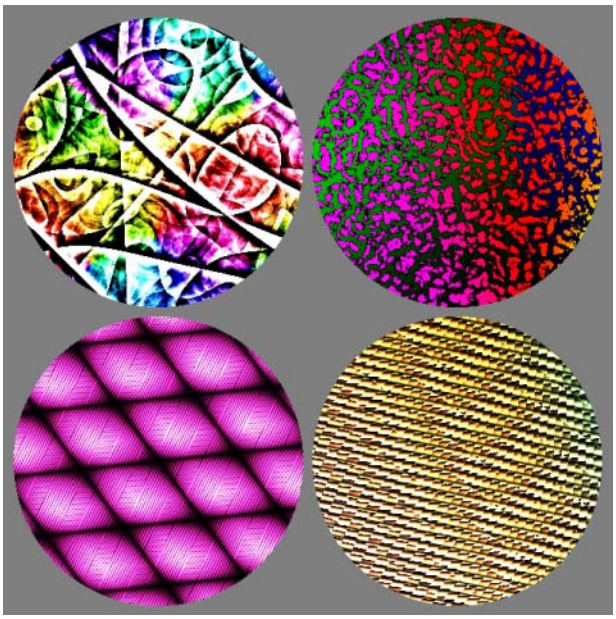


Figure 6. "Random" textures automatically evolved with genetic programming using a noninteractive ad hoc fitness function.

camouflage texture; see examples in Figure 7. In these experiments a cohort contains ten individuals. The prey are circular disks, 100 pixels in diameter. Each prey's camouflage texture is rendered at this resolution, transformed so the prey's body corresponds to a unit-diameter disk at the origin of texture space. Prey are placed on the background in random nonoverlapping positions. They were allowed to extend partially outside the window—perhaps a poor design choice, since sometimes interesting textures were only partly recorded.

The user's task is to inspect the scene, locate prey objects, and select the one that appears most conspicuous—that contrasts most strongly with the background. This selection is indicated with a mouse click on the prey object, signaling the act of abstract predation. In response the GUI records the selection, removes the selected prey from the cohort, and redisplays, erasing the prey. Now the scene consists of the background with $n - 1$ prey objects, and the user selects the next most conspicuous. This process is repeated five times, leaving five survivors from the original cohort of 10. (Cohort size and the number eaten can be varied, 10 and five, respectively, seemed to work well in these experiments.) The window returns to its blank state and awaits the next round.

In a typical GA/GP application, fitness conveys fine gradations of quality. In this model, fitness is binary: life or death. Individuals selected by the predator are removed from the population. This is similar to the *selective breeding* of [34]. Survivors, spared by the predator, retain their high fitness and pass into the next generation (called *elitism* in evolutionary computation). For each round of the camouflage game, the predator looks at a cohort of 10 textured prey. These are selected randomly from a deme's population, which is (on average) half newly bred and half survivors from earlier generations. From this cohort of 10 new and old prey, the predator eats those with the least effective camouflage in the cohort. This culls out both ineffective new prey and obsolete old prey. Figure 8 shows the "fossil record" of improvement during several runs.

The original plan was that a static image of prey over background would be presented to the user, who would then click on the prey in order of conspicuousness. However, it seemed the user might lose track of which prey had already been selected. Some sort of mark could be drawn to indicate which had been selected. But the presence of those already selected (more conspicuous) prey, if not

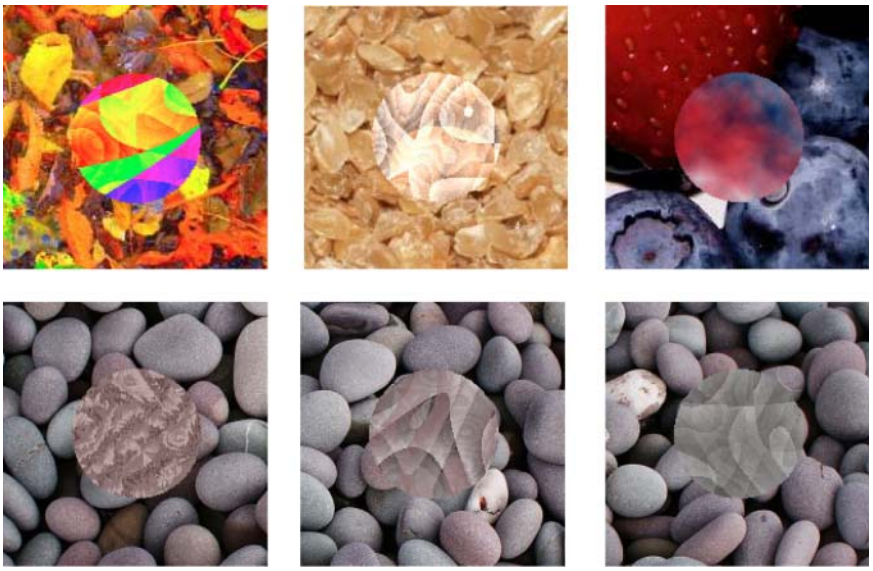


Figure 7. Early results on "Leaves" (top left) and "Cracked Wheat" (top middle) are both based on the Wrapulence texture, which provides edges at many scales and so helps create disruptive camouflage. The top right texture appears to be based on the cloudlike Brownian texture, which is not appropriate for the "Mixed Berries" background but managed to match three colors of the environment: red, white, and blue. The bottom row shows three near misses in a run on the "Pebbles" environment.

Figure 8. Two instances of random "proto-mimicry": prey that look a bit like purple blueberries, and pattern on a prey similar to the green stem on a red pepper just above it.

the marks themselves, might interfere with finding the *n*th most conspicuous prey. Erasing prey as they are selected removes this potential distraction. It gives the user a less demanding cognitive task: Scan the image and identify the most conspicuous remaining prey. This kind of *salience* [18] detection seems to happen at a low level in the vision system and requires little or no abstract reasoning.

Still, this task can be ambiguous for the human observer. Given a green background and a collection of red, purple, and checkerboard camouflage textures—as might happen in the early stages of an evolution run—it can be difficult to decide which of the conspicuous prey is the worst match to the background.

6 Results

While not all evolutionary runs found convincing results, many produced effective camouflage. In fact, some evolved camouflage was so effective that it was missed in the user's initial scan for prey. They were overlooked until a count revealed a missing prey and a second, more careful, visual search was made. It should be noted that in all experiments described here, the user (human, predator) is the author of this article. That a jaded experimenter was actually fooled by evolved camouflage is a significant success. This first happened with the "Bark" background in Figure 1e. Similarly it was often hard to pick out some of the prey in the run with the "Serpentine," "Lichen," and "Green Hedge" backgrounds (Figures 2 and 3).

In these experiments, the evolving prey population usually moved toward matching the typical color or texture of the background. Matching on multiple characteristics seems harder for evolution to achieve. Sometimes a run would find the exact color but never really get the pattern right (see Figure 7, top right), and vice versa (Figure 7, top middle). A few times both came together to produce a compelling result. Combinations of multiple colors seemed a much harder target for adaptation. This was especially true when features in the background were larger than the prey, as for example with "Mixed Berries" (Figure 8) and "Peppers" (Figure 9). The "Lentils" background (Figure 1g) seemed to be close to the upper limit on feature size. Two different runs on the "Lentils" background produced results that were only partially effective. Prey body size places an upper bound on the size of features (lower bound on the spatial frequencies) that can be matched. In the extreme—imagine a beetle on a checkerboard—an environment made up of large areas of uniform appearance allows no effective camouflage for small prey.

Evolutionary computation commonly produces a mixture of successful and unsuccessful runs. Some variability is inevitable when using a stochastic optimization technique. When too many bad runs are seen, a typical fix is to run the evolutionary computation with a larger population. For a standard EC application this is just a matter of investing more processors or time. With an interactive fitness function there is a tradeoff between bigger populations and the limits of human endurance. In these experiments, a typical run has 1000 cohorts, so it requires about 5000 mouse clicks. If the user could keep up a blistering pace of one evaluation and click per second, a run would require about 1.5 hr of mind-numbing

Figure 9. Features in the "Peppers" background were larger than the body size of prey. This prevented the evolution of effective camouflage patterns. Some of the best results, shown here, depend on accidental alignments between camouflage and background.

work. My rate is significantly slower, plus I cannot work steadily at it for more than 15–30 min at a sitting. See Section 7 about addressing this problem with distributed human computation.

These experiments are based on the hypothesis that camouflage can be evolved, given only that an observer can identify the most conspicuous prey in a group. While effective camouflage patterns have been found, this idea is not clearly proved. The methodology used here presents a risk of *experimenter bias*. The same person advances the hypothesis and serves as the subject in an experiment to test it. With knowledge of how the interactive task is mapped into fitness, it is possible to game the task, using it for aesthetic selection as in [27]. For example, the user might be reluctant to eat a prey with a particularly interesting camouflage pattern, even if it were more conspicuous than others in the cohort.

It would be inappropriate to call it an instance of mimicry, but some interesting shapes evolved in a run using the "Mixed Berries" background (see Figure 8). While the colors are wrong and the shapes and textures are off, some of the prey looked a bit like blueberries, with a frosted white surface and a suggestion of the crown (remains of the flower) at the end of a blueberry. Similarly in a "Peppers" run a prey was found that looked a lot like the top of a red bell pepper with its green stem (see Figure 8). These chance similarities do not say much about mimicry in nature, except that one can see how easily it can initially arise and then be amplified and refined by conferring even a small survival advantage.

See http://www.red3d.com/cwr/iec/ for additional results.

7 Future Work

These initial experiments were intended as the first steps in a more comprehensive study of camouflage evolution. Beyond refining this technique, two new research directions are planned.

Refinements on the current approach include improvements to the texture synthesis library and modified user interaction. Cohorts now contain a fixed number of camouflaged prey. It may be helpful to vary this number to remove a clue that well-camouflaged prey have been overlooked. Evolutionary computation seems to progress faster and more robustly when the goal is periodically changed [19; see especially Figure 5]. For camouflage evolution, this might mean periodically (say every fifth generation) cycling between several related background images, perhaps several photographs of a similar environment. It would also be possible to alternate between contrasting environments (say green leaves and brown bark) to produce hybrid camouflage for mixed environments.

The first new research direction is to use *distributed human computation* over the Internet to allow using larger genetic populations. This should provide stronger results and allow tackling more challenging kinds of background images. One approach is simply to pay people to perform the interactive fitness test. Utilities like Amazon Mechanical Turk [2] provide infrastructure to *crowdsource* small tasks like these which require human perception or judgement. Another approach is to entice people to participate voluntarily by casting the task as a game—a "game with a purpose" like the Google Image Labeler [16] and other examples at gwap.com. Several techniques have been identified [35] to transform a mundane task into a game, such as assigning scores, imposing time limits, posting leaderboards, and providing live real-time competition against other human players. One more

intriguing approach to crowdsourcing this simulation was suggested by Dennis Allison: to install an interactive kiosk in a public place like a science museum and invite visitors to walk up and play a few rounds of the camouflage game.

The second new research direction is to investigate *synthetic predators* to allow evolving camouflage without a human in the loop. Using techniques from machine vision and machine learning, the goal would be to train an agent to break camouflage. It would need to analyze an image, identify unusual salient regions [18], and classify them as being either part of the background or potential camouflaged prey. Such an agent could then be coupled with the texture synthesis and evolutionary computation components of the current work to form a closed cooptimization loop (see [37] for a similar proposal). Camouflaged prey would demonstrate fitness by avoiding detection, while predator vision agents would demonstrate fitness by correctly detecting camouflage prey. Such a system would provide a useful computation model of the coevolution of prey camouflage and predator vision.

Acknowledgments

This research was made possible by the generous support of my employer, Sony Computer Entertainment, and particularly my manager: Dominic Mallinson, Vice President, US R&D. I owe special thanks to my remote friend and collaborator Bjoern Knafla for writing the Cocoa application that served as the GUI for these experiments. Christian Gagné helped extensively with the interface to Open BEAGLE. Thanks to readers of early versions of this article and my research proposal: Daniel Weinreb, Iztok Lebar Bajec, and again Bjoern Knafla. Others have contributed helpful suggestions and discussions: James O'Brien, Lance Williams, Ken Perlin, Michael Wahrman, Andy Kopra, and Karen Liu. Thanks to many coworkers at SCE US R&D, my patient and supportive wife Lisa, my son Eric, and my daughter Dana, who frequently contributed feedback and suggestions about this research.

References

1. Abbott, K. (2010). Background evolution in camouflage systems: A predator-prey/pollinator-flower game. *Journal of Theoretical Biology*, *262*(4), 662–678.
2. Amazon Mechanical Turk. (2005). http://www.mturk.com/ (accessed 2010).
3. Beddard, F. E. (1895). *Animal coloration. An account of the principal facts and theories relating to the colours and marking of animals*. London: Swan Sonnenschein & Co.
4. Bond, A. B., & Kamil, A. C. (2002). Visual predators select for crypticity and polymorphism in virtual prey. *Nature*, *415*(6872), 609–613.
5. Chu, H.-K., Hsu, W.-H., Mitra, N. J., Cohen-Or, D., Wong, T.-T., & Lee, T.-Y. (2010). Camouflage images. *ACM Transactions on Graphics*, *29*(4), article 51.
6. Cott, H. B. (1940). *Adaptive coloration in animals*. London: Methuen & Co.
7. Cuthill, I. C., Stevens, M., Sheppard, J., Maddocks, T., Parraga, C. A., & Troscianko, T. S. (2005). Disruptive coloration and background pattern matching. *Nature*, *434*(7029), 72–74.
8. Cuthill, I. C., Hiby, E., & Lloyd, E. (2006). The predation costs of symmetrical cryptic coloration. *Proceedings of the Royal Society B*, *273*(1591), 1267–1271.
9. Darwin, C. (1859). *On the origin of species by means of natural selection*. London: John Murray.
10. Dawkins, R. (1986). *The blind watchmaker*. New York: W. W. Norton & Co.
11. Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (1994). *Texturing and modeling: A procedural approach*. Boston: AP Professional.
12. Eizirik, E., David, V., Buckley-Beason, V., Roelke, M., Schaffer, A., Hannah, S., Narfstrom, K., O'Brien, S., & Menotti-Raymond, M. (2010). Defining and mapping mammalian coat pattern genes: Multiple genomic regions implicated in domestic cat stripes and spots. *Genetics*, *184*(1), 267–275.
13. Endler, J. A. (1980). Natural selection on color patterns in *Poecilia reticulata*. *Evolution*, *34*(1), 76–91.
14. Funes, P., Sklar, E., Juillé, H., & Pollack, J. (1998). Animal-animat coevolution: Using the animal population as fitness function. In R. Pfeifer, B. Blumberg, J.-A. Meyer, & S. W. Wilson (Eds.),

From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (pp. 525–533). Cambridge, MA: MIT Press.

15. Gagné, C., & Parizeau, M. (2006). Genericity in evolutionary computation software tools: Principles and case-study. *International Journal on Artificial Intelligence Tools*, *15*(2), 173–194.
16. Google Image Labeler. (2006). http://images.google.com/imagelabeler/ (accessed 2010).
17. Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
18. Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11), 1254–1259.
19. Kashtan, N., Noor, E., & Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences of the U.S.A.*, *104*(34), 13711–13716.
20. Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.
21. Merilaita, S. (2003). Visual background complexity facilitates the evolution of camouflage. *Evolution*, *57*(6), 1248–1254.
22. Montana, D. J. (1995). Strongly typed genetic programming. *Evolutionary Computation*, *3*(2), 199–230.
23. Open BEAGLE. (2002) Open BEAGLE evolutionary computation library, version 3.0.3. Web site for code and documentation, http://beagle.gel.ulaval.ca/ (accessed 2010).
24. Perlin, K. (1985). An image synthesizer. In B. A. Barsky (Ed.), *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH '85* (pp. 287–296). New York: ACM.
25. Reynolds, C. (2009) Texture synthesis diary. Blog/lab notebook: http://www.red3d.com/cwr/texsyn/diary.html (accessed 2010).
26. Schaefer, H. M., & Stobbe, N. (2006). Disruptive coloration provides camouflage independent of background matching. *Proceedings of the Royal Society B*, *273*(1600), 2427–2432.
27. Sims, K. (1991). Artificial evolution for computer graphics. In T. W. Sederberg (Ed.), *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH '91* (pp. 319–328). New York: ACM.
28. Sherratt, T. N., Pollitt, D., & Wilkinson, D. M. (2007). The evolution of crypsis in replicating populations of web-based prey. *Oikos*, *116*(3), 449–460.
29. Stevens, M., & Cuthill, I. C. (2006). Disruptive coloration, crypsis and edge detection in early visual processing. *Proceedings of the Royal Society B*, *273*(1598), 2141–2147.
30. Stevens, M., & Merilaita, S. (2009). Animal camouflage: Current issues and new perspectives. *Proceedings of the Royal Society B*, *364*(1516), 423–427.
31. Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, *89*(9), 1275–1296.
32. Togelius, J., Yannakakis, G., Stanley, K., & Browne, C. (2010). Search-based procedural content generation. In C. Di Chio et al. (Eds.), *Proceedings of 2nd European Event on Bio-inspired Algorithms in Games, EvoGAMES 2010* (pp. 141–150). Berlin: Springer LNCS.
33. Thayer, G. H. (1909). *Concealing-coloration in the animal kingdom: An exposition of the laws of disguise through color and pattern: Being a summary of Abbott H. Thayer's discoveries*. New York: Macmillan.
34. Unemi, T. (2003). Simulated breeding—A framework of breeding artifacts on the computer. *Kybernetes*, *32*(1/2), 203–220.
35. von Ahn, L., & Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM*, *51*(8), 58–67.
36. Wei, L.-Y., Lefebvre, S., Kwatra, V., & Turk, G. (2009). State of the art in example-based texture synthesis. In *Eurographics '09 State of the Art Reports (STARs)*. Goslar: Eurographics Association.
37. Wilson, S. W. (2009). *Coevolution of pattern generators and recognizers*. TR 2009006. Illinois Genetic Algorithms Laboratory.

Appendix 1: Texture Synthesis Details

One input to strongly typed genetic programming [22] is a description of a set of functions and the types associated with their inputs and outputs. The texture synthesis library used in this work included types for procedural textures, 2D Cartesian vectors, RGB colors, and numbers. There are five numeric types, all floating point, with unique ranges (including or excluding negative or zero values). Random constants (GP calls them *ephemeral constants*) are generated according to these types.

The texture synthesis library contained 52 texture-producing elements. Some of the names are self-descriptive; for the others, and for description of parameter types for each, see [25]. *Texture generators:* UniformColor, SoftEdgeSpot, Gradation, SineGrating, TriangleWaveGrating, SoftEdgedSquareWaveGrating, RadialGrad, Noise, ColorNoise, Brownian, Turbulence, Furbulence, Wrapulence, and NoiseDiffClip. *Texture operators:* Scale, Translate, Rotate, Mirror, Add, Subtract, Multiply, Max, Min, SoftMatte, ExpAbsDiff, Row, Array, Invert, Tint, Stretch, StretchSpot, Wrap, Ring, Twist, VortexSpot, Blur, EdgeDetect, EdgeEnhance, SliceGrating, SliceToRadial, SliceShear, Colorize, Gamma, AdjustSaturation, AdjustHue, BrightnessToHue, BrightnessWrap, BrightnessSlice4, HueIfAny, SoftThreshold, SpotsInCircle, and ColoredSpotsInCircle.

Appendix 2: CC Background Image Sources

- "Bark" by Six Revisions: http://www.flickr.com/photos/31288116@N02/3752674533/
- "Cracked Wheat" by Sanjay Acharya: http://commons.wikimedia.org/wiki/File:Sa-cracked-wheat.jpg
- "Flowers" (*Lantana montevidensis* in our backyard) by Craig Reynolds: http://www.red3d.com/cwr/iec/
- "Granite Yosemite" by David Monniaux: http://commons.wikimedia.org/wiki/File:Granite_Yosemite_P1160483.jpg
- "Green Hedge" by Craig Reynolds: http://www.red3d.com/cwr/iec
- "Leaves" by Scott M. Liddell (www.scottliddell.net) with permission: http://www.morguefile.com/archive/display/90656
- "Lentils" by Daniel Kulinski: http://www.flickr.com/photos/didmyself/2126646787/
- "Mixed Berries" by Angelo Juan Ramos: http://commons.wikimedia.org/wiki/File:Summer_Fruits.jpg
- "Pebbles" by Sean Hattersley: http://commons.wikimedia.org/wiki/File:Pebbleswithquarzite.jpg
- "Peppers" by Elavats: http://www.flickr.com/photos/elavats/549041490/
- "Serpentine" by Kevin Walsh: http://commons.wikimedia.org/wiki/File:Serpentine-texture.jpg
- "Twisty Wires" by Clara Natoli (used with permission): http://www.morguefile.com/archive/display/10850
- "Yellow Lichen" by Neil Mallett (used with permission): http://www.flickr.com/photos/enigmatic/3364679113/