Interactive Evolution of Camouflage

Craig Reynolds

Sony Computer Entertainment, US R&D
craig_reynolds@playstation.sony.com

Abstract

This paper 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*. In these experiments, the predator's role is played by a human observer. They are shown a *cohort* of ten evolved textures overlaid on the background texture. They click 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,* a variant of the *genetic algorithm*. GP searches the space of texture description programs for those which appear least conspicuous to the predator.

Introduction

That animals often resemble their environment has been observed since ancient times. This sometimes incredible 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 by Means of Natural Selection*

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 lend support to this idea and point the way to more powerful human-computer hybrid systems as well as future simulation studies of the *coevolution* of prey camouflage and predator vision.

As defined in (Stevens and Merilaita, 2009) the term *camouflage* includes all strategies of *concealment*. To distinguish from *hiding*, this is taken to mean reducing the chance of recognizing an animal which is otherwise in plain sight. (Thayer, 1909) describes 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 homogenous environments like white snow or green leaves. Consider a color-matched and borderless photographic print of an environment, say the surface of a rock. If the print is placed on the rock it will not be perfectly cryptic. Discontinuities at the edge of the print stimulate low level edge detectors in the visual system, causing a strong perception of a rectangle. Moving the print to another location on the rock will also reveal subtle variations in color and texture which add additional contrast at the edge of the print.

Much recent work on camouflage (see next section) has focused on the importance of *disruptive camouflage*. While

Figure 1: camouflaged "prey" overlaid on the background image for which they were evolved
(a) tree bark, (b) twisty wire, (c) flowers, (d) serpentine, (e) Yosemite granite

Figure 2: these camouflaged prey are only partially or occasionally effective, features in this *peppers* background were too large to "solve"

these patterns often echo colors and textures from the environment, their effectiveness comes from their ability to visually disrupt the visual 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, see (Schaefer and Stobbe, 2006). Paradoxically, camouflage that does not match the background can be more effective through the use of strong visual features (false edges) that intersect the object's real edges (Stevens and Cuthill, 2006). 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 computation model of camouflage evolution could allow new types of theoretical experiments to be conducted in simulation which are not subject to constraints imposed by working in the field, or with live animals, and in general is not limited to examples found in Earth's biosphere.

Related Work

Over the last century several seminal works have surveyed the broad topic of camouflage in nature. These include (Beddard, 1895), (Thayer, 1909) and (Cott, 1940). The latter two continue to be widely cited today. Over the last 20-30 years there has been a significant renaissance in the study 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 (Stevens and Merilaita, 2009).

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" (Cuthill, et al. 2005) 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 with 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 moth. This technique has shown the key important of disruptive coloration (Schaefer and Stobbe, 2006), measured the disadvantage of symmetrical camouflage (Cuthill, et al. 2006), and several related topics.

Other experiments have used live captive birds (Bond and Kamil, 2002) and humans (Sherratt, et al. 2007) as predators of "virtual artificial prey" on a display screen. In both cases this predation was used to drive a evolutionary computation like in the work described here. In (Merilaita, 2003) 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 (Abbott, 2010).

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 (Dawkins, 1986). 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 (Sims, 1991). In (Funes, et al. 1998) and other papers, 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 (Togelius, et al. 2010). A deep survey of the whole field of *interactive evolutionary computation* is found in (Takagi, 2001).

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 (Wei, et al. 2009). Using this technique to generate camouflage image puzzles is described in (Chu, et al. 2010). In contrast, the synthesis of camouflage texture described in this paper does not "see" or otherwise access the input texture. Instead the background can only be *inferred* from the indirect evidence of predation, as it is in evolution of natural camouflage.

Texture Synthesis

In nature, patterns of surface coloration on plants and animal result from complex genetic and developmental processes collectively called *morphogenesis* (see for example, (Eizirik, et al. 2010)). In this simulation model, pattern formation is

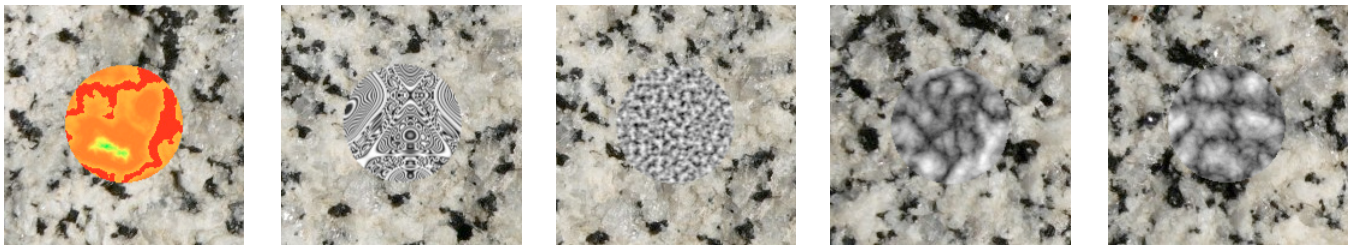


Figure 3: progression of camouflage patterns during a run with the granite environment

represented by procedural texture synthesis (Ebert, et al. 1994). 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 types (numbers, 2D vectors and RGB colors). Operators are similar but have one or more Texture parameters. These nested expressions look like composition of functions (see Figures 12 and 13) although in this implementation they are specifically 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 (Reynolds, 2009) brings together several preexisting techniques. Its generators include uniform colors and simple patterns like spots and color gradations. There are a collection of gratings (e.g. a sine wave grating) and an assortment of noise patterns such as *noise* and *turbulence* (Perlin, 1985) plus variations on these. The library's collection of texture operators include simple geometrical transformation (such as scale, rotate and translate), simple image processing operations (add, subtract, multiply, adjustment of intensity, hue, 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, ...). 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 compute heavy textures, awaiting a GPGPU implementation.

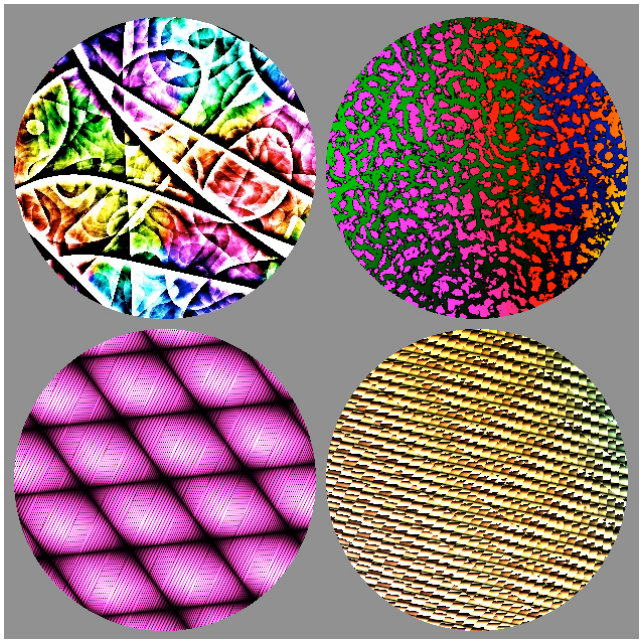


Figure 4: "random" textures automatically evolved with genetic programming using a non-interactive *ad hoc* fitness function

Evolutionary Computation

The texture synthesis library described in the previous section was designed for use with *genetic programming* (Koza, 1992). Like the closely related *genetic algorithm* (Holland, 1975), 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 a 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 a given grammar. In this work, each individual is a program that defines a procedural texture. These programs can be thought of as nested expressions of composed functions, or as a tree of functional nodes. 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 sub-node of one program with a sub-node 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. Some changes improve fitness and some reduce fitness, but 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 (Gagné and Parizeau, 2006), (Open BEAGLE, 2002). 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* or STGP (Montana, 1995). 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 10000 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 4 shows early tests (before the interactive camouflage experiments) of evolved textures using 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 4.

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 or trackpad to begin a "round" of the camouflage game. The window is redrawn to show a background texture on which is overlaid a *cohort* of circular prey objects, each with an evolved camouflage texture, see Figure 5. In these experiments a cohort contains ten individuals. Prey are placed on the background in random non-overlapping positions. They were allowed to extend partially outside the window, perhaps a poor choice.

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 ten. (Cohort size and the number "eaten" can be varied, 10 and 5 seemed to work well in these experiments.) The window returns to its blank state and awaits the next round.

In 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 (Unemi, 2003). 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 ten textured prey. These are drawn randomly from a deme population which is 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 old obsolete prey. Improvement during one run is shown in Figure 3.

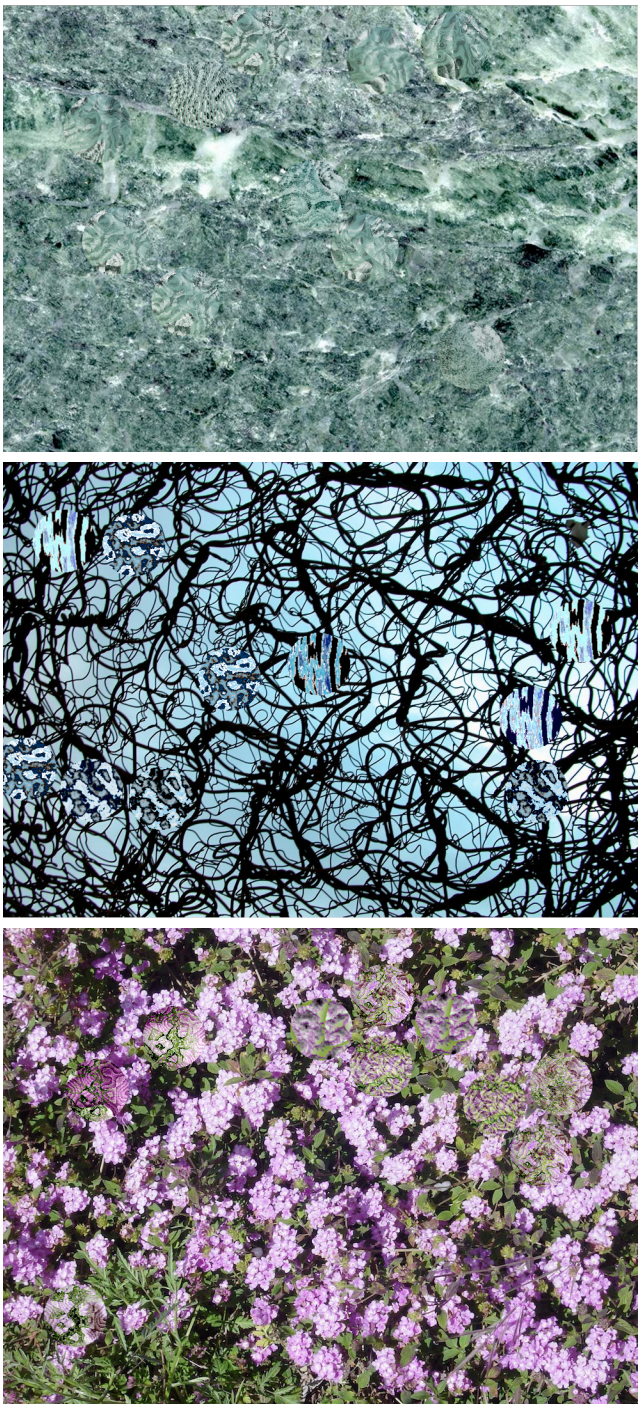


Figure 5: screen shots showing interactive sessions with "serpentine" (top) and "twisty wires" (middle) and "flowers" (bottom) environments. In all three, a new cohort of ten evolved textures is shown overlaid on the background.

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 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* detection seems to happen at a low level in the vision system and requires little or no abstract reasoning (Itti, et al. 1998).

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

Results

While not all evolutionary runs found convincing results, some produced effective camouflage. In fact some evolved camouflage was so effective that they were 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. That a jaded experimenter was actually fooled by evolved camouflage is a significant success. This happened with the "bark" background in Figure 1a. Similarly it was very hard to pick out some of the prey in the run with the "serpentine" background shown in Figure 5.

In these experiments, the evolving prey population usually moved toward matching the typical color or texture of the background. Matching on multiple characteristics was apparently harder. Sometimes a run would find the exact color but never really get the pattern right (see Figures 10(right) and 11) and vice versa (Figure 10(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 "berries" (Figure 6) and "peppers" (Figures 2 and 7)). Prey size implies an upper bound on the size of features (lower bound on spatial frequencies) that can be matched. In the extreme, an environment made up of large areas of uniform appearance allows no effective camouflage for small prey.

Evolutionary computation commonly produces a mix of successful and unsuccessful runs. Some variability is inevitable 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 trade-off between bigger populations and the limits of human endurance. In these experiments, a typical run has 1000 cohorts, so requires about 5000 mouse clicks. If the user can keep up a blistering pace of one evaluation and click per second, a run costs about 1.5 hours of mind numbing work. My rate is significantly slower, plus I cannot work steadily at it for more than

Figure 6: accidental "blue" berries

Figure 7: pattern on prey similar to stem on red pepper above it.




Figure 8: progression of camouflage patterns during a run with the pebbles environment
(nice color-matched texture, followed by better frequency matching, then something like feature matching)

Figure 9 lentils
(near feature size limit)

Figure 10: early results on "leaves" (left) and "cracked wheat" (middle) both based on the *Wrapulence* texture which features edges at many scales and so helps create disruptive camouflage. The right hand texture appears to be based on the cloud-like *Brownian* texture which is not appropriate for the "berries" background but managed to match three colors of the environment: red, white and blue.

15-30 minutes at a sitting. See Future Work 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 (Sims, 1991). 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 6). 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 7). These chance similarities do not say much about mimicry in nature, except that one can see how easily it can arise and then be amplified and refined by even a small survival advantage.

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

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. (Kashtan, et al. 2007) suggests that periodically changing evolutionary goals provides better results. For camouflage evolution, this might equate to periodically cycling between several related background images, perhaps several photographs of a similar environment.

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 (Amazon, 2005) provide infrastructure to *crowdsource* small tasks like these requiring human 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 (Google, 2006) and other examples at gwap.com. Several techniques have been identified to change a mundane task into a game, such as scores, time limits, leader-boards and live competition against other human players, see (von Ahn and Dabbish, 2008).

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 (Itti, 1998), 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 co-optimization loop (see (Wilson, 2009) for a similar proposal). Camouflaged prey would demonstrate fitness by avoiding detection while predator vision agents would demonstrate fitness by detecting camouflage prey. Such a system would provide a useful computation model of the coevolution of camouflage.

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

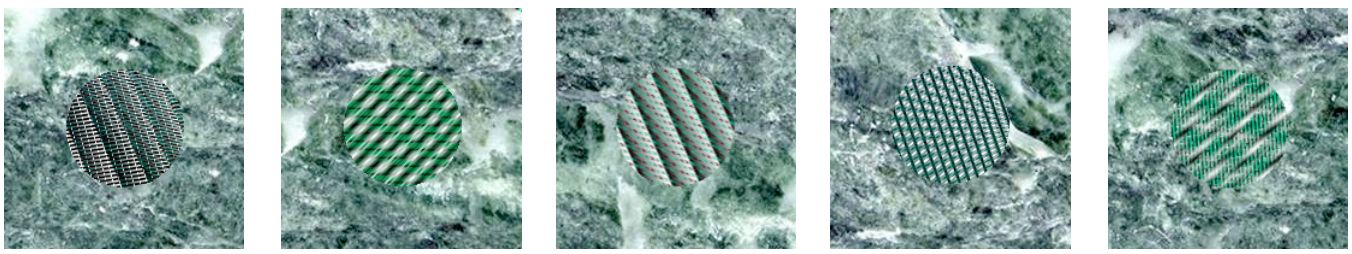
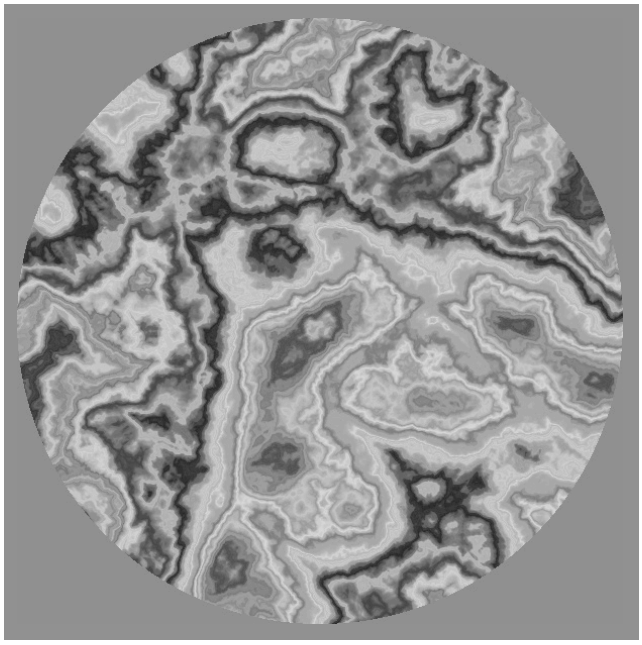


Figure 11: an unsuccessful early run using the "serpentine" background and a rank-based fitness scheme that was later abandoned



```
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)))
```

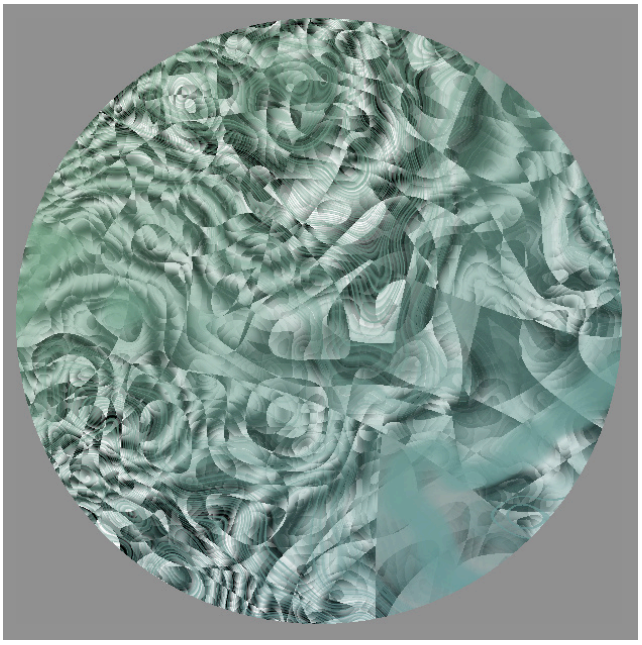
Figure 12: disruptive oak bark camouflage of Fig. 1(a), re-rendered at 600x600 resolution, with its evolved source code

owe special thanks to my remote friend and collaborator Bjoern Knafla for writing the Cocoa application that served as GUI for these experiments. Christian Gagné helped extensively with the interface to Open BEAGLE. Thanks to readers of early versions of this paper 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.

Appendix 1: Texture Synthesis Details

One input to *Strongly Typed Genetic Programming* (Montana, 1995) 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 (and so whether negative or zero values are included). 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 others, and for description of parameter types for each, see (Reynolds, 2009). **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.



```
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), SineGrating (17.4233, 0.477075)), Translate (Vec2
(1.3634, -3.05406), Colorize (SineGrating (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 13: camouflaged prey evolved on “serpentine” background with its evolved source code

References

Abbott, K. (2010). Background evolution in camouflage systems: A predator–prey/pollinator-flower game. *Journal of Theoretical Biology.* 262(4):662-678.

Amazon Mechanical Turk. Launched 2005, accessed 2010: http://www.mturk.com/

Beddard, F. E. (1895). *Animal Coloration. An Account of the Principal Facts and Theories Relating to the Colours and Marking of Animals.* Swan Sonnenschein & Co., London.

Bond, A. B. and Kamil, A. C. (2002). Visual predators select for crypticity and polymorphism in virtual prey. *Nature.* 415(6872): 609-613.

Chu, H-K, Hsu, W-H, Mitra, N. J., Cohen-Or, D., Wong, T-T, Lee, T-Y. (2010). Camouflage Images. To appear in *ACM Transactions on Graphics* 29(3).

Cott, H. B. (1940). *Adaptive Coloration in Animals.* Methuen and Co., London.

Cuthill, I. C., Stevens, M., Sheppard, J., Maddocks, T., Parraga, and C. A., Troscianko, T. S. (2005). Disruptive coloration and background pattern matching. *Nature.* 434(7029):72-74.

Cuthill, I. C., Hiby, E., and Lloyd, E. (2006). The predation costs of symmetrical cryptic coloration. *Proc. Biol. Sci.* 273(1591):1267-1271.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. John Murray, London.

Dawkins, R. (1986). *The Blind Watchmaker.* W. W. Norton & Company, Inc., New York, NY.

Ebert, D. S., Musgrave, F. K, Peachey, D., Perlin, K. and Worley, S. 1994. *Texturing and Modeling: A Procedural Approach.* AP Professional. ISBN 0-12-228760-6.

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.

Funes, P., Sklar, E., Juillé, H. and Pollack, J. (1998). Animal-animat coevolution: using the animal population as fitness function. In *From animals to animats 5*, 525-533. MIT Press, Cambridge, MA.

Gagné, C. and Parizeau, M. (2006). Genericity in Evolutionary Computation Software Tools: Principles and Case-Study. *International Journal on Artificial Intelligence Tools*, 15(2):173-194.

Google Image Labeler. Launched 2006, accessed 2010: http://images.google.com/imagelabeler/

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor.

Itti, L., Koch, C., and Niebur, E. (1998). A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 20(11):1254-1259.

Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences.* 104(34):13711-13716.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA. ISBN 0-262-11170-5

Merilaita, S. (2003). Visual background complexity facilitates the evolution of camouflage. *Evolution.* 57(6):1248-1254.

Montana, D. J. (1995). Strongly typed genetic programming. *Evolutionary Computation* 3(2):199-230.

Open BEAGLE. (2002). Open BEAGLE evolutionary computation library, version 3.0.3. Website for code and documentation, launched 2002, accessed 2010: http://beagle.gel.ulaval.ca/

Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.* 19(3):287-296.DOI=http://doi.acm.org/ 10.1145/325165.325247

Reynolds, C. (2009) Texture Synthesis Diary (blog/lab notebook), accessed 2010: http://www.red3d.com/cwr/texsyn/diary.html

Schaefer, H. M. and Stobbe, N. (2006). Disruptive coloration provides camouflage independent of background matching. *Proc. Biol. Sci.* 273(1600):2427-2432.

Sims, K. (1991). Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '91*. ACM, New York, 319-328. DOI= http://doi.acm.org/10.1145/122718.122752

Sherratt, T. N., Pollitt, D., and Wilkinson, D. M. (2007). The evolution of crypsis in replicating populations of web-based prey. *Oikos*. 116(3):449-460.

Stevens, M. and Cuthill, I. C. (2006). Disruptive coloration, crypsis and edge detection in early visual processing. *Proc. R. Soc. B.* 273(1598):2141-2147.

Stevens, M. and Merilaita, S. (2009). Animal camouflage: current issues and new perspectives. *Phil. Trans. R. Soc. B.* 364(1516): 423-427.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*. 89(9) 1275-1296.

Togelius, J., Yannakakis, G., Stanley, K., and Browne, C. (2010). Search-based Procedural Content Generation. *Proceedings of 2nd European event on Bio-inspired Algorithms in Games (EvoGAMES 2010)*.

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*. Macmillan, New York, NY.

Unemi, T. (2003). Simulated breeding – a framework of breeding artifacts on the computer. *Kybernetes*. 32(1/2) 203-220.

von Ahn, L. and Dabbish, L. (2008). Designing games with a purpose. *Commun. ACM* 51(8):58-67. DOI= http://doi.acm.org/10.1145/1378704.1378719

Wei, L-Y, Lefebvre, S., Kwatra, V., and Turk, G. (2009) State of the Art in Example-based Texture Synthesis, in *Eurographics '09 State of the Art Reports (STARs)*.

Wilson, S.W. (2009). Coevolution of Pattern Generators and Recognizers. Illinois Genetic Algorithms Laboratory TR 2009006.

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

Leaves by Scott M. Liddell (www.scottliddell.net) with permission:
http://www.morguefile.com/archive/display/90656

Lentils by Daniel Kulinski (Daniel*1977)
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