DOI: http://dx.doi.org/10.7551/978-0-262-31050-5-ch049

Automatically Designing and Printing 3-D Objects with EvoFab 0.2

Tim Kuehn[1] and John Rieffel[1]

[1]Union College, NY 12308
rieffelj@union.edu

Abstract

Although Evolutionary Design has had great success in creating virtual objects, very few of these evolved designs have been manufactured. Standing in the way is the *fabrication gap* caused by a reliance on prescriptive rather than descriptive representations of evolved objects. Evolutionary Fabrication describes an alternative process which evolves *how* rather than *what* to build. In this paper we describe EvoFab 0.2, a completely automated physically embodied machine which implements Evolutionary Fabrication and evolves three dimensional objects. We describe the mechanism and underlying algorithms in detail, and show how it can be used to create novel structures.

Introduction

Evolutionary algorithms have been used to design a wide variety of objects, from furniture (Funes and Pollack, 1998; Hornby and Pollack, 2001) to architectures (Hemberg and O'Reilly, 2004) to robots (Sims, 1994). Evolved designs are often characterized by the novelty of their solutions, enabled by a process which operates orthogonally to human design methodologies and biases. Koza has justly described genetic algorithms as "automated invention machines" capable of human-competitive patentable designs (2003).

A historically valuable aspect of the patent process is the "working model", a physical prototype of the design submitted to the patent office. And yet most evolved objects are never physically manufactured, relegated instead to the virtual drawing board. Those few exceptions which have been manufactured – most notably Lohn *et al.*'s antennae(2005), and Pollack *et al.*'s robots (2001) – were done so with considerable human effort and interaction (Funes' LEGO structures, for instance, often had to be assembled sideways on a flat surface before being tilted into position.) The goal of our research into Evolutionary Fabrication is to automate the entire process of design and manufacture, leading to the possibility of a real "automated invention machine".

One significant source of the gap between evolved design and manufactured object – what we call the *Fabrication Gap* – is the fact that almost all evolved designs are *descriptive* rather than *prescriptive*. That is, much like a blueprint they specify what to build, but leave out the essential information of *how* to build it. Bridging the gap between evolved design and manufacture in a *post-hoc* manner requires considerable human input, and runs the risk of re-injecting human bias into a process whose success is otherwise greatly increased by the absence of such bias. Furthermore, purely descriptive evolutionary design runs the risk of generating *unbuildable* designs (Rieffel, 2006).

A second obstacle to the physical manifestation of evolved designs is the infamous Reality Gap (Jakobi et al., 1995). Evolutionary algorithms are experts at exploiting their substrates. When the design of objects happens in simulation, successful candidates often achieve high fitness by exploiting bugs in the simulator (see for instance Sim's seminal work on evolved artificial agents (1994)). Moreover, many complex systems, including the deposition of viscous materials at the heart of rapid prototyping, cannot be simulated with any degree of transferable verisimilitude even by advanced techniques such as computational fluid dynamics.

The solution we propose lies in evolving *how* to build rather than *what* to build. In Evolutionary Fabrication, the evolving genotype explicitly describes a *process* of manufacture rather than an *object* of design. Furthermore, guided by Rodney Brooks's sage advice that "the world is its own best model" (1990), we eschew simulation entirely, and evolve objects exclusively in the real world. In this sense we are in the company of others who have performed the evolution exclusively in the real world (Watson et al., 1999; Thompson, 1996; Zykov et al., 2004).

In 2010 we introduced EvoFab 0.1, a machine which implemented many aspects of Evolutionary Fabrication, but featured an interactive GA, and therefore required substantial human subjective interaction (Rieffel and Sayles, 2010). In this paper we describe a significant leap in the state of the art with EvoFab 0.2 (pictured in Figure 1), the first machine capable of closed-loop, fully automated Evolutionary Fabrication. EvoFab operates by embedding a genetic algorithm directly within an off-the-shelf rapid prototyping machine. The genotypes of the system are linear strings of printer in-

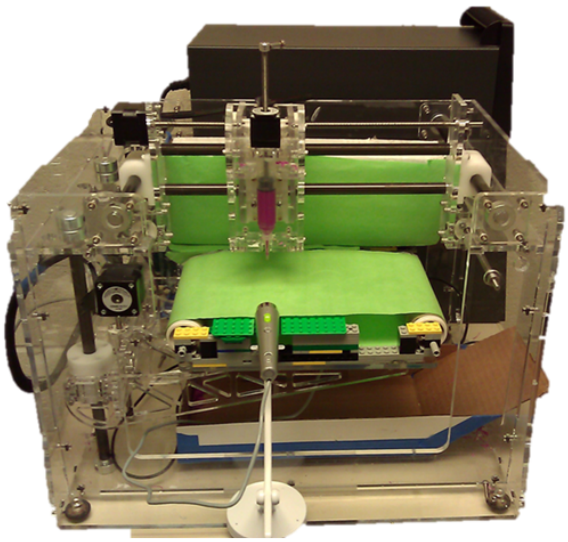

Figure 1: The "EvoFab 0.2" consists of a Fab@Home printer, computer vision software to determine fitness, and a conveyor belt, all controlled by an evolutionary algorithm.

struction primitives and phenotypes are the objects which result from printing. Fitness is determined by using machine vision algorithms to measuring physical properties of the printed objects. The process is further automated with the addition of a conveyor belt to discard objects once they have been evaluated.

EvoFab 0.2

As illustrated by Figure 2, EvoFab 0.2 is a three-stage process . First an object is printed by extruding material onto a platform. Next, the object is then evaluated using machine vision techniques. Finally, the, the object is moved off the printing platform to begin the process anew.

The Fab@Home Printer

At the heart of EvoFab 0.2 lies Fab@Home, an open-source 3D printer (Malone and Lipson, 2007) and appealing because of its low cost and relative ease of use. The most current model of the Fab@Home is Model 2 (Lipton et al., 2009), however we used the Model 1 as the basis for EvoFab because it allows greater access to the underlying API.

The Fab@Home operates by extruding material through a syringe and onto a platform. The carriage that holds the syringe is free to move along the X- and Y-axes. The platform upon which the material is deposited is free to move along the Z-axis. Fab@Home normally builds its products by interfacing via USB to a program that contains STL-based blueprints of objects. However, it also allows for direct control of print functions via serial port.

Printer Commands as Genotypes

Conventionally, when operating as a pure 3-D printer, Fab@Home constructs objects via additive manufacturing: depositing material layer-by-layer in a of rastering process. We place no such constraint upon the operation of the printer, however. Instead, evolved genotypes are purely *prescriptive*, consisting only of a linear sequence of primitive instructions sent to the printer.

The specific instructions available as components of the genome are as follows:

- **extrude** – This command causes a small amount of material to be deposited onto the print platform.
- **beginExtrude** – This command, rather than send a command directly to the printer, controls the action of the other commands. When activated, all other commands except endExtrude will send their command coupled with an extrude command. Effectively, all other commands say "do this while extruding" when beginExtrude is activated.
- **endExtrude** – This command deactivates beginExtrude.
- **goUp** – Raises the print platform.
- **goDown** – Lowers the print platform.
- **goLeft** – Causes the print carriage to move left.
- **goRight** – Causes the print carriage to move right.
- **goIn** – Causes the print carriage to move toward the back of the Fab@Home.
- **goOut** – Causes the print carriage to move away from the back of the Fab@Home.

Print Media

While the Fab@Home is able to extrude plastic from long spools, allowing for long print durations without refilling material, plastic printing involves incredibly high temperatures, necessitating caution and constant vigilance by the user. Because this negates the ability of EvoFab to act autonomously, we chose instead to use other materials.

EvoFab 0.1 used silicone bath caulk as a print material. With new goals, however, come new requirements, and after attempts with plastic and silicone caulk, we settled on a brand of modeling compound similar to Play-Doh. Silicone caulk is easily extrudable, readily available, and comes in many colors, which is useful in allowing computer vision software to easily differentiate a printed object from its background. However, it is also sticky when first printed, and its cure time of approximately thirty minutes for faster-drying variants is too long to wait between prints. Thus, the material would inevitably stick to the print platform, making automation difficult.

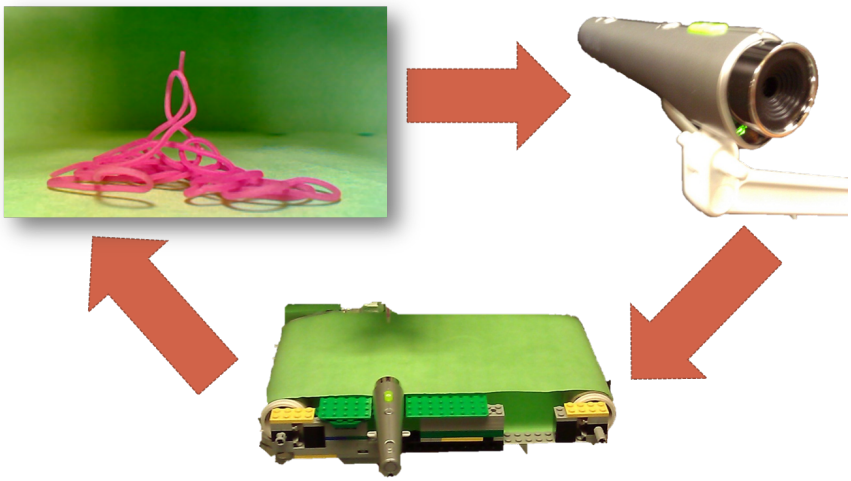

Figure 2: A graphical representation of the three-stage process: print, evaluate, recycle.

Play-Doh has the same benefit of being readily available in many colors and easily extrudable without the drawback of stickiness upon first being extruded. This lack of stickiness comes with its own set of problems: when printing, if the material is not extruded quickly enough, it will not stick to the platform, causing the print carriage to drag the thread of material around instead. This has led to a certain degree of unpredictability, but it has proven to be the best option that has been tried thus far.

From 0.1 to 0.2: Full Automation

Our earlier system, EvoFab 0.1, was the first to instantiate Evolutionary Fabrication, but suffered from several drawbacks, most notably its reliance upon subjective human input for fitness evaluation, and its reliance on human effort to clear the build platform between generations. As a result of this human involvement, a single generation of a GA could take several hours.

EvoFab 0.1 ran an interactive GA, or blind-watchmaker algorithm. The process began with the fabber printing four objects onto a piece of wax paper lying on top of the print platform. Then, a human operator would inspect the four objects and choose the one they deemed to be best-fit. Because of the complexity of the evaluation task, fitness criteria were relatively simple, such as the object's similarity to a desired 2-D letter shape ("O" or "A"). The user would then remove the wax paper containing the objects, input their fitness into a computer, and begin the cycle anew with the printing of four more objects (videos of this process are on the authors website.) While workable, there are a variety of ways in which this method was restrictive.

Automating Evaluation In an interactive GA such as the one used for EvoFab 0.1, a person's opinion on which object is best-fit is both highly subjective and prone to error. Especially with early generations, it may be very difficult for a person to choose between four seemingly shapeless masses.

To address this in EvoFab 0.2, we developed a completely automated evaluation process. Using openCV wrapped in Python, we have created computer vision software that works in tandem with a camera affixed to the front of the printing platform. This allows for the unbiased and consistent evaluation of fitness, and further allows us to evaluate more three-dimensional objects.

Cycling Another issue that arose in EvoFab 0.1 is that comparing more than four genotypes becomes unwieldy. The print platform was originally divided into quadrants, allowing one object to be printed in each quadrant. Because the printer was set up to only print four objects at a time, it would require replacing the print platform wax paper after every four prints.

To automate this process in EvoFab 0.2, we developed a belt based upon simple lego motors driven by a USB interface. Once an object has been evaluated, it is moved off of the platform by the conveyor and deposited into a disposal container. Thanks to this improvement, EvoFab can now run unattended for approximately thirty minutes before requiring a refilled syringe, and excluding these refills, EvoFab can in principle run unattended indefinitely.

Evolving Arches

With these pieces in place, as a proof of concept we can demonstrate how the new EvoFab 0.2 can be used to evolve objects in a closed-loop manner.

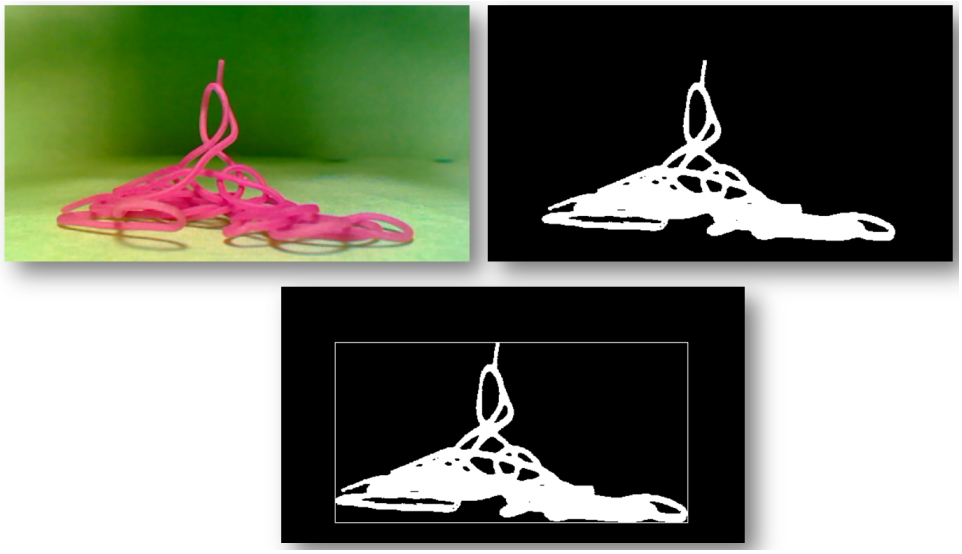

Figure 3: Evaluating ”archiness”: Fitness is determined by machine vision. First the image is thresholded into black and white. Second, a bounding box is calculated around the object. Finally, the percentage of overhanging mass is determined by columnwise counting the number of black pixels “shaded” beneath white pixels, and normalizing by bounding box area.

Fitness Criterion

In order to provide EvoFab with a challenge, we chose a fitness criterion which is deliberately difficult for 3-D printers to produce: overhangs. Since rapid prototypers conventionally print objects by rastering upwards layer by layer, higher layers require support from lower layers, and only very modest overhangs are allowed. As a consequence they cannot construct an objects with a large degree of cantilevering. Consider, for example, an arch, whose supporting columns are relatively easy to produce, but whose middle section can be a challenge, since it cannot deposit material onto mid-air. Our interest is therefore in discovering how a evolutionary algorithm, faced with this task, might arrive at a solution.

For our purposes, the “archiness“of a printed object can be calculated by the degree of overhang present. Figure 3 shows how such a fitness is evaluated: an image is captured by a camera that views the printing stage. Then, the image is thresholded so that the printed object is white and the background is black. This is made simple by printing in a color negative to that of the background, in this case pink being the negative of green. Then, a bounding box is drawn around the contours of the white image. For all pixels contained within the bounding box, fitness increases for every black pixel that is vertically below a white pixel in its column. Fitness is then normalized by total pixels within the bounding box to account for different sized objects, returning the percentage of overhanging mass in the image.

Evolvability of Linear Encodings

The printing of 3-D objects by extruding material from a print-head is an explicitly linear and serialized process. This has significant consequences in terms of the evolvability of any encoding of any such process.

In a conventional GA, mutation can occur anywhere in the genome with equal probability, and at least in principle the effects of a mutation are largely independent of where along the the genome a mutation occurs. This is no longer the case in linear encodings such as ours: the effect of a mutation is highly sensitive to where in the process the mutation occurs.

Consider a simple set of instructions to draw the letter “L” in our printer language:

```
beginExtrude
goOut
goOut
goOut
goRight
goRight
goRight
endExtrude
```

A change early on in the sequence, for instance changing the first goOut to a goUp, would result in the entire shape being printed on a higher plane, which, depending upon what was underneath might drastically affect the shape, whereas changing the last goRight to a goUp would have very little effect. This dependence on context is compounded when you take into account the full three-dimensional nature of the objects being printed.

This context-sensitivity is even more pronounced when considering the effects of crossover. The building block hypothesis (Goldberg, 1989) holds that crossover aids evolution by finding and duplicating useful regions of the genome.

In a serialized encoding such as ours, however, the context dependency means that a sequence of instructions which is highly fit in one context is unlikely to be as fit in a different context. For instance, the sequence of three goRight commands in the example above may draw a nice straight line on a flat surface, but would have a significantly varied phenotypic consequence if executed in mid-air or over pre-existing structure.

We will elaborate on future alternatives to linear encodings in our discussion section below.

Algorithmic Details

Given the challenges to evolvability imposed by the nature of our serial encoding, we have chosen to eliminate crossover and implement a 1+4 Random Mutation Hill-climber (RHMC) (Mitchell, 1996) rather than a more canonical GA.

Initial Genome length was 350 instructions, with a 10% mutation rate. Mutation was capable of changing the operation at a locus (i.e. from goLeft to goRight), inserting a new random instruction, or deleting the current instruction.

Results and Discussion

A video of the EvoFab 0.2 in action can be found on YouTube, tagged with "evofab" and "alife13".

Figures 4 and 5 show typical results of evolution achieved after ten generations. Raw images are in the left hand column, thresholded and bounded images are shown in the middle column, and fitness values are shown in the right hand column.

Quantitatively we can show that fitness according to our metric has increased over the course of evolution. The highest fitness individual from the original population, as measured by normalized overshadowed pixels, is 0.17, whereas the fitness of the best individual from generation 10 is 0.34.

Qualitatively, of course, the results are more equivocal. While the last figure may have a fitness measured at 0.34, it doesn't exhibit many features which we could describe as truly arch-like. These results therefore, while promising, and clearly proof of the concept, highlight several improvements required of our system.

The most obvious problem lies in our vision-based fitness function. As written, parts of the structure which are not overhanging at all, for instance the long stretches of material on the right hand side of the middle images in Figures 4 and 5, are awarded fitness by exploiting a trick of perspective view. Regions of an object which are printed further back on the surface of the arena contribute more toward this ersatz fitness than those printed toward the front. This perspective issue can be resolved largely by changing the angle of the camera and the field of view of the vision algorithm, such that the fitness function can only measure parts of the structure which are truly overhanging.

Secondly, since our camera can only view the X-Y projection of objects, it would be worth removing or reducing the Z-axis degree freedom for the print head. An arch is an arch so long as it is upright – its thickness is largely irrelevant.

Ultimately, we expect future progress of EvoFab to hinge on the matter of genotypic representation. One possibility would be to use a grammatical encoding such as an L-System (1990) to indirectly "grow" a linear encoding - this could allow for increased modularity and reuse in the phenotype, although the resulting linear representation might still be susceptible to the context-sensitivity demonstrated by our current encoding. Alternatively we could use a a developmental approach, such as the CPPNs used to by Clune and Lipson (2011) and Auerbach and Bongard (2010). While these particular efforts used CPPNS to create descriptive 3-D blueprints rather than prescriptive instructions, it would not be unreasonable to use a CPPN to directly control a 3-D printer.

Conclusion

We have described the world's first completely closed loop system capable of automated design and fabrication and proven the concept by demonstrated its application to a complex design task. While our methods would benefit from some modification, we are confident that the approach shows promise.

In the near term, there are more immediate and practical applications of Evolutionary Fabrication than the invention of objects. For instance, every new material used by a 3-D printer requires careful calibration of flow rates and printer head speed in order to produce a consistent "bead" of material. Currently, this calibration process is entirely driven by human trial and error across a set of more than twenty parameters. We envision a simple GA being able to more quickly and effectively arrive at these calibration settings, and perhaps being able to find particularly efficient bead characteristics. Secondly, sharp corners are very difficult to produce on 3-D printers, and small errors on the corners of tall objects can quickly add up in a deleterious manner. EvoFab could be well used to discover and optimize new methods of producing corners which are less susceptible to these errors.

In the long term, the automated design of 3-D objects has valuable applications in fields ranging from biomedical applications (for instance the development of compliant soft grippers) to soft robotics, not to mention the much-vaunted "automated invention machine". We look forward to making progress toward these goals with our upcoming EvoFab 0.3.

References

Auerbach, J. E. and Bongard, J. C. (2010). Evolving cppns to grow three-dimensional physical structures. In *Proceedings of the 12th annual conference on Genetic and evolutionary compu-*

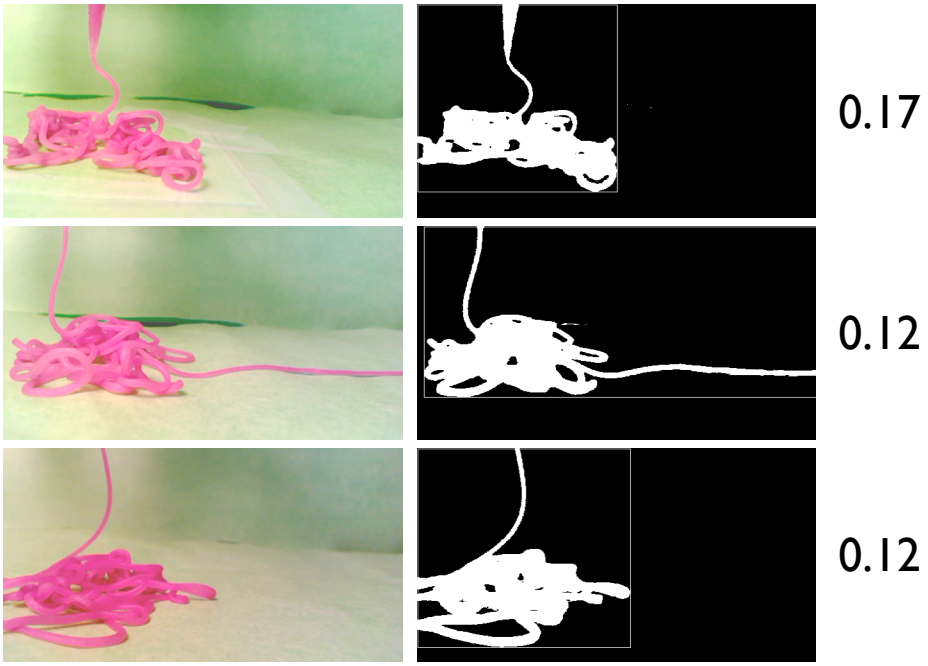

Figure 4: Example phenotypes from Generation 0. Raw images are in the left hand column, thresholded and bounded images are shown in the middle column, and fitness values are shown in the right hand column.

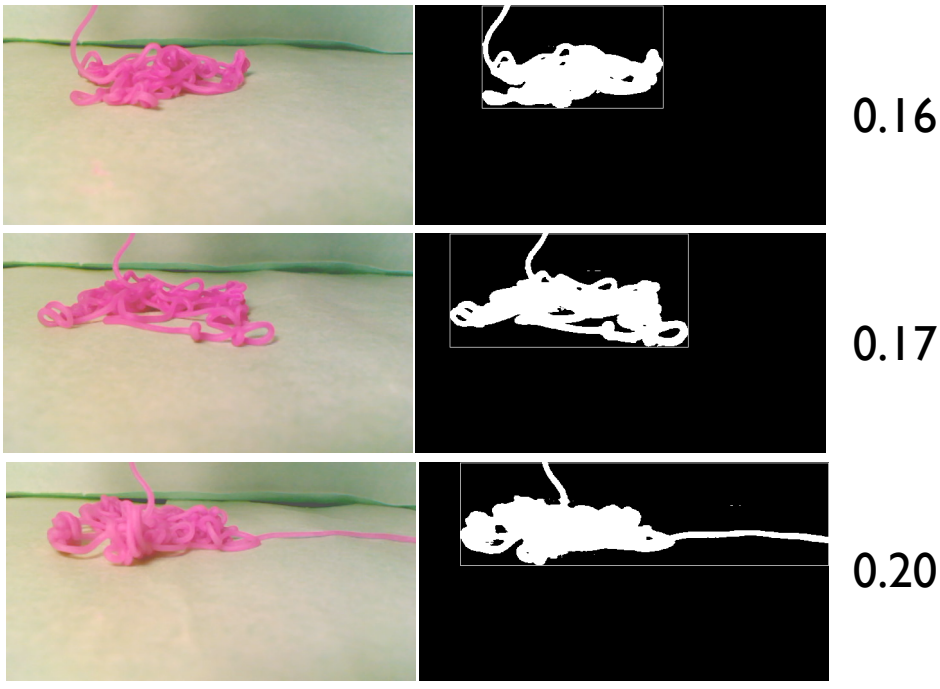

Figure 5: Example phenotypes from Generation 10.

tation, GECCO '10, pages 627–634, New York, NY, USA. ACM.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

Clune, J. and Lipson, H. (2011). Evolving 3d objects with a generative encoding inspired by developmental biology. *SIGEVOlution*, 5(4):2–12.

Funes, P. and Pollack, J. B. (1998). Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 4(4):337–357.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, Upper Saddle River,NJ, USA.

Hemberg, M. and O'Reilly, U.-M. (2004). Extending grammatical evolution to evolve digital surfaces with genr8. In *EuroGP*.

Hornby, G. S. and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 600–607, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proc. of the Third European Conference on Artificial Life (ECAL'95)*, pages 704–720, Granada, Spain.

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., and Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

Lipton, J. I., Cohen, D., Heinz, M., Lobovsky, M., Parad, W., Bernstein, G., Li, T., Quartiere, J., Washington, K., Umaru, A.-A., Masanoff, R., Granstein, J., Whitney, J., and Lipson, H. (2009). Fab@home model 2: Towards ubiquitous personal fabrication devices. In *Solid Freeform Fabrication Symposium*, pages 70–81.

Lohn, J. D., Hornby, G. S., and Linden, D. S. (2005). An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission. In O'Reilly, U.-M., Riolo, R. L., Yu, T., and Worzel, B., editors, *Genetic Programming Theory and Practice II*. Kluwer.

Malone, E. and Lipson, H. (2007). Fab@home: The personal desktop fabricator kit. *Rapid Prototyping Journal*, 13(4):245–255.

Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA.

Pollack, J. B., Lipson, H., Hornby, G., and Funes, P. (2001). Three generations of automatically designed robots. *Artifial Life*, 7(3):215–223.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer–Verlag, New York, USA.

Rieffel, J. (2006). *Evolutionary Fabrication: the co-evolution of form and formation*. PhD thesis, Brandeis University.

Rieffel, J. and Sayles, D. (2010). Evofab: a fully embodied evolutionary fabricator. In *Proceedings of the 9th international conference on Evolvable systems: from biology to hardware*, ICES'10, pages 372–380, Berlin, Heidelberg. Springer-Verlag.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press.

Thompson, A. (1996). Silicon evolution. In *Stanford University*, pages 444–452. MIT Press.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Zykov, V., Bongard, J., and Lipson, H. (2004). Evolving dynamic gaits on a physical robot. In *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO'04*.