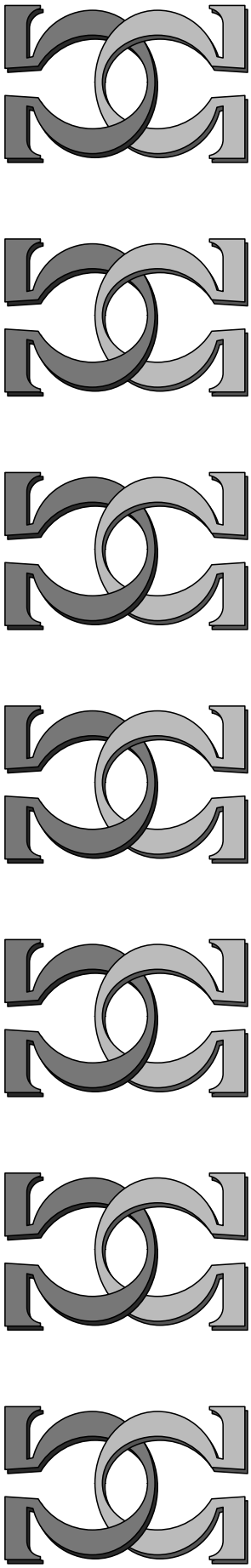

CDMTCS
Research
Report
Series

Is there a Universal Image Generator?

C. S. Calude[1] **and J. P. Lewis**[2]
[1]University of Auckland, NZ
[2]Massey University, NZ

CDMTCS-344
January 2009

Centre for Discrete Mathematics and
Theoretical Computer Science

Is there a Universal Image Generator?

Cristian S. Calude and J.P. Lewis
University of Auckland, NZ
www.cs.auckland.ac.nz/~cristian/
Weta Digital and Massey University, NZ
http://scribblethink.org

Abstract

Synthetic pattern generation procedures have various applications, and a number of approaches (fractals, L-systems, etc.) have been devised. A fundamental underlying question is: will new pattern generation algorithms continue to be invented, or is there some "universal" algorithm that can generate all (and only) the perceptually distinguishable images, or even all members of a restricted class of patterns such as logos or letterforms? In fact there are many complete algorithms that can generate all possible images, but most images are random and not perceptually distinguishable. Counting arguments show that the percentage of distinguishable images that will be generated by such complete algorithms is vanishingly small. In this paper we observe that perceptually distinguishable images are compressible. Using this observation it is evident that algorithmic complexity provides an appropriate framework for discussing the question of a universal image generator. We propose a natural Thesis for describing perceptually distinguishable images and argue its validity. Based on it, we show that there is no program that generates all (and only) these images. Although this is an abstract result, it may have import for several areas in graphics that deal with compressible signals. In essence, new representations and pattern generation algorithms will continue to be developed; there is no feasible "super algorithm" that is capable of all things.

1 Introduction

Is there a universal image construction, meaning a single algorithm that can generate all (and only) the possible perceptually distinguishable images? In an essay in *Metamagical Themas* [Hofstadter 1985] Hofstadter considered a similar but simpler question: is there an algorithm that can generate all possible letterforms. As it seems trivially possible to create any desired letterform with a spline drawing program such as Adobe Illustrator, the question needs to be explained. A universal generation algorithm for a class of images is an algorithm that can generate all members of that class and that never (or only very rarely) generates objects outside that class. In the case of Hofstadter's essay one imagines a master typeface program that will generate all possible letterforms as user-specified style parameters are varied, or that generates all possible letterforms automatically with no user input. The program's outputs should be predominantly images of the intended class— a small percentage of non-letterform patterns among the outputs of a letterform generator might be acceptable for some purposes, but a letterform generator that only rarely produces letterforms hardly justifies the name.

We adopt Hofstadter's terminology in describing the problem. Making an analogy to Gödel Incompleteness in mathematics, Hofstadter terms a letter forming algorithm *complete* if it can generate all possible letterforms, and *consistent* if it generates only letterforms and no other types of images. In these terms we define a *universal* algorithm to be one that is both consistent and complete. Hofstadter suggests that a universal (complete and consistent) letterform algorithm does not exist. The essay argues that a single algorithm cannot encompass the variety of unusual (but recognisable) letterforms that have been devised, as well as the infinity of letterforms that have yet to be designed, and points out that existing recognisable examples of the letter "A" do not even have any single feature in common (Fig. 1). Although it falls short of a proof, the argument and examples shown in the essay are quite convincing.

Figure 1: *The letter "A" in a variety of typefaces. Reproduced from [Hostadter 1995].*

The possibility of a universal image construction is of philosophical and some practical interest. Hofstadter framed the question philosophically, as an issue of whether the essence of something like a letter can be algorithmically defined. Practical graphics tasks such as designing a corporate logo could benefit from universal visual pattern generators. Creative activities can sometimes be decomposed into a creative phase of devising a number of candidate designs or ideas, followed by an "editorial" phase of selecting from these possibilities [Boden 2004]. The existence of a universal logo generator might allow graphic designers to skip the construction of candidates and merely select from a long list of possibilities generated by the program.

Although a program that can generate all possible letterforms, or all possible corporate logos, seems somewhat conceivable, a program

that generates all possible images (including for example a picture of you on a vacation that you will take in the future) initially seems more farfetched. As discussed in section 2, programs that enumerate all possible images do exist, but they are inconsistent and therefore are not useful. The possibility of a universal image generation algorithm remains an open question.

The existence of such an algorithm would be of fundamental interest to graphics researchers. Algorithmic construction and processing of images is intimately tied to the chosen representation, and existing representations are not universally advantageous. For example, spline outline representations are well suited for representing letterforms but are usually a poor choice for representing photographs. Various new approaches to signal analysis and representation periodically appear in graphics research—fractal image compression was a popular topic in the 1980s, and wavelets were equally popular a decade later. One may wonder whether new representations will continue to arise indefinitely, or on the other hand whether there is some "best" representation that will ultimately be discovered. A universal image construction presumably might embody some best approach to representing and building images.

Many researchers may find it intuitive that there is no universal approach to constructing images. In sections 4 and 5 we present arguments from algorithmic information theory [Calude 2002] that support this intuition. Before proceeding to the arguments, section 2 provides further background on our problem, showing that there are many universal (but not consistent) image algorithms, and that the lack of this second property makes them useless; attempts to fix this shortcoming are discussed. Section 3 defines some terms and concepts used in the following arguments.

2 Complete but Inconsistent Image Algorithms

All possible binary strings can be generated by a "British Museum" algorithm, i.e., by an exhaustive enumeration. In fact there are infinitely many algorithms to accomplish this, for example, the quasi-lexicographical enumeration, $0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots$ In a similar way, going from 1D to 2D, all possible images can be generated simply by enumerating all pixel combinations, i.e., for an image with n pixels, do the equivalent of n nested loops, each looping over all pixel values. An infinite number of other procedures exist—pick any complete image basis and enumerating all coefficient values in that basis will generate all possible images. Since a unitary transform is analogous to a rotation there are an infinite number of such algorithms even considering only linear orthogonal representations. In addition to exhaustive enumerations the use of a "reliable" source of randomness e.g. quantum generated randomness [Calude 2005] to pick pixel values or coefficients will eventually generate any possible picture.

Completeness vs. Consistency

Each of the above universal methods produces a complete set of images; are they consistent? Although in theory a set of pictures documenting all aspects of your life will appear in the output of these programs if you wait long enough, in fact the images generated by these approaches will look very homogeneous—almost all images will look like the "television noise" images in Fig. 2 (a counting argument in section 3 will justify this statement). A program that in practice only generates noise images is of little value as a universal image generator. The issue is not just that most images generated with these approaches will not look like natural images, but that one would have to wait a very long time to observe anything resembling a real-world image

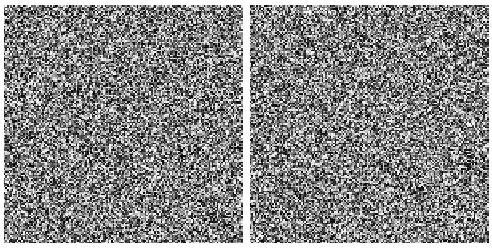

Figure 2: *"Random television noise" images are not memorable and easily distinguishable.*

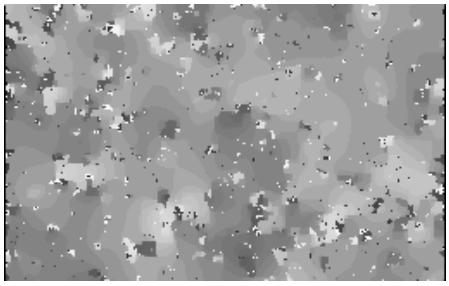

Figure 3: *Example of a random piecewise smooth approximation to a "natural-like" image. A collection of such images would not be particularly distinguishable.*

It is worth characterizing the magnitude of the quantities we are discussing. If we take a black-and-white 64^2 8-bit image as a minimal image representation, and generously assuming 8:1 lossless compression can be obtained, then the number of images is $8^{4096}/2^3 = 2^{12285}$. Recall that even 2^{64} is an intractable quantity: current processors are approaching speeds of 2^{32} instructions per second (four gigahertz clocks), but to apply some N instruction operation to each of 2^{64} objects using such a processor will take $2^{32} \cdot N$ seconds, or about $136 \cdot N$ years. Operations such as these, that are theoretically possible but practically impossible, will be labelled *infeasible*; they should be distinguished from both practically achievable operations and problems (such as the halting problem) that are theoretically uncomputable.

Perceptual Parameterization

A different approach to our issue is to observe that the parameterization of image space embodied in a pixel or Fourier basis is not matched to our perception. Most of the space in each of these representations maps to images that are indistinguishable, while the set of perceptually distinguishable images is a tiny subset of the space that random sampling will never locate in any feasible time (Fig. 5 and section 3).

This suggests that consistency means we wish to algorithmically generate all possible *perceptually distinguishable* members of a class of images. Since our visual systems are evolved to distinguish the types of images that arise in nature we will term these perceptually distinguishable images as *natural-like* (NL) images. Invented or hypothetical images that mimic the character of natural images are among the NL images. For example, computer graphics images of imaginary objects are usually easily distinguishable, as are many other graphic designs. On the other hand, the human visual system is not designed to distinguish pictures such as those in Fig. 2; random images are not NL.

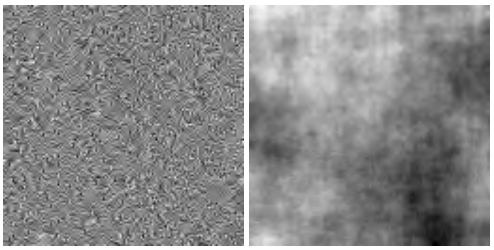

Figure 4: *a), b) Images from pseudo-randomly chosen DCT coefficients, (left) with uniform amplitude in 8x8 blocks; (right) a single 128^2 block with amplitude falling as $f^{-1.5}$ in the DCT frequency space.*

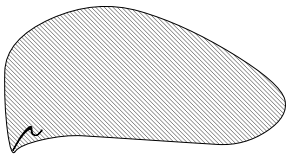

Figure 5: *Caricature of the space of all possible images. The space of natural images (black sliver), though vast, is a tiny subset that random sampling will not locate.*

Are the complete enumerations described above consistent? Certainly, not. Can they be adapted to satisfy consistency, i.e., to account for perception? In theory this might be accomplished by defining a perceptual metric. The space would be sampled as seen through this metric, so indistinguishable areas of the image space would not be repeatedly sampled. While in concept we could use pairwise similarity rankings in conjunction with multidimensional scaling (or other manifold learning procedures) to identify a perceptually parameterized space, there is a stubborn underlying problem—again, the space of images is infeasibly large. Consider a set of several thousand images that subjects are to rank for similarity (a boat, a dog, a car...). This is such a small subset of the space of possible images that extrapolation to other images is not reasonable, and simultaneously the distance between the individual images is so large that interpolation is also ill posed. Each pair of images will have significant differences in many thousands of coordinates; there is not enough information in the ranked distance to know what combination of coordinates is responsible for that distance.

Projecting onto the Subspace of Natural Images

Before giving up on the general idea of being able to generate all possible perceptually distinguishable images, consider an abstract approach that first characterises the subspace of NL images and then projects onto this subspace from a random location (image). As an approximation of this approach, take the characterisation of NL images to be those that are piecewise smooth. An attractive continuation method, *graduated non-convexity* (GNC), has been formulated for the problem of fitting a piecewise-smooth surface to data [Blake and Zisserman 1987]. The GNC 'weak membrane' model fits a smooth surface through the data but also allows discontinuities where the modelled surface changes by more than a threshold between pixels. In Fig. 3 we apply the weak membrane to an initially random (television noise) image, as an approximation of this idea of projecting on the NL manifold(s). Fig. 3 suggests that this characterisation of "natural-like" does not produce particularly

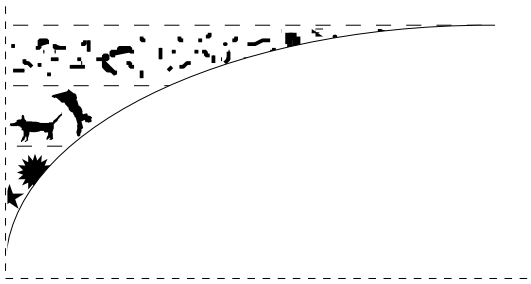

Figure 6: *Algorithmic complexity (vertical axis) versus number of objects (horizontal axis). Random objects (top) vastly outnumber natural-like objects (middle), which in turn outnumber simple procedural patterns such as fractals (bottom).*

distinguishable images and is too simplistic.

These ideas can be concretely illustrated by considering JPEG images. JPEG can represent all possible distinguishable pictures, and the representation is such that perceptually insignificant information is discarded. A JPEG decoder could be adapted to generate all distinguishable pictures by driving it with random discrete cosine transform (DCT) coefficients. Despite the fact that JPEG discards some perceptually indistinguishable data, however, an attempt to employ it as an image generator will result in noise images (Fig. 4).

It seems that the approaches mentioned in this section cannot be developed because of our incomplete understanding of human perception and the character of natural images. In fact however the abstract argument in section 4 indicates that the goal of algorithmically producing all distinguishable images is not possible any case.

3 Algorithmic Complexity

In this section we will briefly present some relevant results in Algorithmic Information Theory (AIT). Algorithmic complexity and entropy are related in some ways and have some similar theorems [Hammer et al. 2000], but whereas entropy is the complexity of describing particular objects from a population given their probability, algorithmic complexity deals with the complexity of a single object.

The algorithmic complexity $C(x)$ of a string (digital object) x is the length of the shortest computer program that generates that string (object). It formalises an intuitive notion of algorithmic complexity. Consider the three patterns:

```
111111111111111111111111111111
123123123123123123123123123123
992625800923176251906030137620
```

These strings (over the alphabet $\{0, 1, 2, \ldots, 9\}$) have of the same length 33, but the first two strings appear to be simpler than the third. This subjective ranking cannot be explained with a classical probability argument (all strings have uniform probability 10^{-33}), but is reflected in the length of the programs needed to produce these strings. For the first string the program is a few bytes in length,

```
for i:=1 to n print('1');
```

The program for the second string is slightly longer since it will

contain either nested loops or the literal `'123'`. As there is no (visible) pattern to the third string, the shortest program to produce it seems to the program that includes the whole string as literal data and prints it—this string seems incompressible or algorithmically random.[1] Fig. 2 is an illustration of an incompressible image.

Ideally the complexity of an object should be a property only of the object itself, but the choice of computer and programming language affects program lengths; however the resulting "uncertainty" is bounded by a fixed constant depending on the choice of the programming language. The choice of an inelegant language or machine adds only a constant amount to the algorithmic complexity, since a translator or simulator from any language or machine to any other is a fixed-size program. In the limit, for large objects this constant becomes insignificant. It should be emphasised that most AIT arguments, including those in section 4, similarly become precise in the limit of increasingly large objects. We imagine taking pictures of a natural scene with a succession of cameras of increasing resolution, or alternately running an image generation algorithm with an increasing series of output resolutions.

The Gödel Incompleteness Theorem states that every finitely-specified, sound theory which is strong enough to include arithmetic cannot be both consistent and complete. A theory is sound if it can prove only true statements. The flavour of algorithmic complexity reasoning is illustrated in the following alternative argument of mathematical incompleteness: proving complexity is beyond the power of standard mathematical theories.

Chaitin Algorithmic Incompleteness Theorem. *A consistent, finitely-specified, sound theory with N bits of axioms cannot prove statements of the form '$C(x) > t$' if t is much greater than N.*

The proof is by contradiction. Because of the arithmetic 'embedded' in the theory, a statement of the form '$C(x) > t$' can be formalised in the theory. If the statement $C(x) > t$ can be proved then it should be possible to extract from the proof the particular x that is used, and we assume that the extraction procedure is fairly simple (the theory is sound and well-specified). Then by appending this extraction algorithm to the proof sequence (of length $C(x) \approx N$) one can generate the string x using slightly over N bits. But the proof has shown that the algorithmic complexity of x is $C(x) > t \gg N$ resulting in contradiction.[2] An important result—that will be needed in section 4—follows directly from the above theorem: algorithmic complexity is uncomputable!

A string x is (algorithmically) m-random if $C(x) \geq |x| - m$, where $|x|$ is the string length and m is a number significantly smaller than $|x|$. So, a string is m-random if it cannot be compressed by more than m bits. Random strings defined in this way pass all traditional tests of randomness.

Thus, AIT defines randomness by incompressibility. A central question is, what proportion of all strings are incompressible? The question is answered in a standard counting argument that is the foundation of many algorithmic complexity results and provides a quantitative underpinning for some of the statements about images made in section 2. This argument observes that there are 2^n possible n-bit strings (or objects), but fewer than 2^n strings that are shorter than n bits. Thus, not all objects can be compressed regardless of the chosen compression algorithm—there are not enough short strings to uniquely represent each of the objects in compressed form. Algorithmic complexity *incompressibility* results are just an extension of this argument.

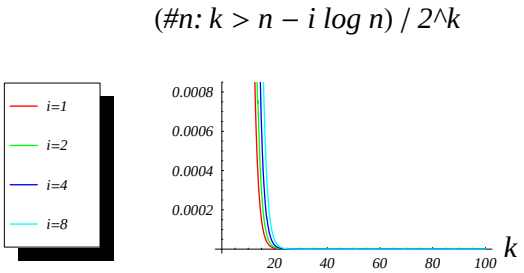

Figure 7: *Number of n that violate (1) for several i, plotted as a proportion of 2^k.*

As we said, there are 2^n such strings—how many of them are compressible by at most m bits, i.e., $C(x) \geq |x| - m$? There are at most $2^0 + 2^1 + 2^2 + \ldots + 2^{n-m} = 2^{n-m} - 1$ possible programs of size $n - m - 1$ or less (note the "at most"), so there are no more than $2^{n-m} - 1$ strings x with $C(x) \leq |x| - m$. There are at most $2^{n-1} - 1$ programs of size $n - 2$ or less, so at most only half of the strings x can be of algorithmic complexity $C(x) \leq n - 1$. Likewise, at most a quarter of all strings are compressible by two or more bits, and only $1/2^m$ of all strings have complexity less than $n - m$. This trend has been dubbed "exponentially few strings are exponentially compressible" (Fig. 6).

Fix two integers $k, i \geq 0$. With the exception of finitely many n, the proportion of n-bit strings x having algorithmic complexity $C(x) < k$ is smaller than n^{-i}, a quantity that (effectively) converges to 0 when n tends to infinity. Here is a counting argument proving this assertion. Take n such that

$$k \leq n - i \log n \tag{1}$$

and observe that only finitely many n do not satisfy (1); here $\log n$ is the integer part of the base 2 logarithm of n. Out of all 2^n n-bit strings, at most $2^{n - i \log n} - 1$ strings can have $C(x) < k \leq n - i \log n$, hence for each n satisfying (1) the proportion of n-bit strings x having algorithmic complexity $C(x) < k$ is smaller than $2^{n - i \log n}/2^n \leq n^{-i}$.

Incompressibility provides a quantitative answer to several questions raised in section 2 concerning the 'enumerate all coefficients' or 'British Museum' approach to complete image generation.

- Any complete image synthesis procedure of the 'British Museum' sort will not only generate random images such as Fig. 2, but these types of images will vastly outnumber the non-random ones. This is because almost all digital objects are incompressible.

- The fact that almost all objects are incompressible is true regardless of the basis used to represent the objects [Calude 2002; Calude and Jürgensen 1994; Staiger 2002]. The algorithmic complexity perspective provides an easy indirect proof of the fact that choosing random coefficients in some

[1] We have used the imprecise verb "seems" instead of the exact "is" because although most images are algorithmically random there is no way to prove that a specific images is indeed algorithmically random; of course, one can prove that a specific image is *not* algorithmically random. More details will be presented in the section 5.

[2] Actually, a stronger result is true [Calude and Jürgensen 2005] for a slightly different type of algorithmic complexity: the theorems of a consistent, finitely-specified, sound theory cannot be significantly more complex than the theory itself.

other basis (e.g. Fourier) nevertheless produces a noise image.

- The likelihood of encountering anything other than a noise image can be quantified. A 64^2 8-bit image is $8 \cdot 2^6 \cdot 2^6 = 32$k bits uncompressed. An image with half of the complexity (2:1 compression) might be sufficiently compressible that it would not look like another noise image. Such an image would be smaller by 16k bits; the proportion of strings that are 16k bits less complex than a random string is 2^{-16384} —an infeasible minority.

4 Characterising Images

Are NL Images Compressible?

The fact that the somewhat unnatural images represented by Figs. 2 are also not compressible suggests that perhaps the converse is also the case, i.e. that natural images are compressible. This is certainly true – lossless compression of about 2.5:1 is commonly achieved [Skodras et al. 2001], and lossy compression of 10:1 or more can be achieved with little or no perceptual degradation.

Is it accurate to conclude that most or all images that are potentially of interest are also compressible? Most images of the real world depict objects that have some formative process, and that process can be approximately or statistically modelled to obtain compression. Even some random timeseries that occur in nature are correlated ($1/f$ noise) or consist of intermittent events (radioactive decay) and are thus compressible, although this is not the case in general [Calude 2005]. Similarly in the realm of human creations, most designs reflect some internal logic rather than being completely random. Additional evidence is provided by the fact images typically have a power-law spectrum [Ruderman and Bialek 1994] and thus are compressible (it is argued that human vision takes advantage of this fact [Atick and Redlich 1992])

One way to model this evidence is to require that one can compress an NL image x by at least by the logarithm of its length:

$$C(x) \leq |x| - \log |x|. \tag{2}$$

(The exact form of the upper bound in (2) will be relaxed in section 5.)

How Compressible Are NL Images?

However, NL images are "not too simple" in the sense that their complexity is not too low. For example, the images in Figure 4 have low complexity (more precisely, their C complexity is about the logarithm of their length plus a constant: a pseudo-random string can be generated by a seed amplified by a short algorithm, hence one needs a constant plus the length of the output to produce it; the output length can be encoded in $\log |x|$ bits). To model this fact we require that one cannot compress an NL image x by more than the logarithm of its length plus a constant:

$$C(x) > \log |x| + \text{constant}. \tag{3}$$

Thesis

From the above discussion we conclude that NL images are compressible, but not too simple. These facts are expressed mathematically by (2) and (3), so we are led to formulate the following

> Thesis: Every natural-like image x satisfies (2) and (3).

i.e. NL images are an infinite subset of the set

$$\{x : \log |x| + \text{constant} < C(x) \leq |x| - \log |x|\}. \tag{4}$$

If this Thesis is accepted—more arguments supporting it will be discussed later in section 5—algorithmic complexity arguments will show that that there is no single program that can generate all NL images.

5 The Argument

First let us introduce two classical notions in Computability Theory[3]: computably enumerable (c.e.) and computable sets. A set is *computable* if its membership predicate is decidable, i.e., can by computed by algorithm. The set of primes or the set of syntactically correct programs are decidable. Every finite set is computable. A *c.e.* set is one whose members are output by an algorithm, possibly in arbitrary order and with repeats. The set of programs that halt is a standard example of a c.e. set. The following procedure is called *dovetailing*: programs are bit-strings, and all possible bit-strings can be enumerated; generate the first bit-string, execute it for one time step, generate the next potential program, add it to the pool of programs, run each one for one time step, and continue in this fashion, printing any programs that finish. If a program halts this procedure will eventually print that fact. But since the maximum runtime of programs of a particular size cannot be algorithmically computed (the maximum runtime rises faster than any computable function of the program length), one does not know how long to wait for an answer. In general, a computable enumeration of a set does not guarantee an answer to whether a particular object is in that set, although it may provide that answer. Every decidable set is c.e., but the converse implication is false: the set of halting programs is c.e. but not computable, [Calude 2002].

Fix $k \geq 0$ and consider a set of very simple images, represented by the bounded algorithmic complexity strings,

$$A_k = \{(x, k) : C(x) \leq k\}.$$

The set A_k is c.e. (generate all the strings of size less than k, consider them as potential programs, then run them dovetailed), finite (so computable), but not uniformly computable in k. Indeed, A_k has at most $2^k - 1$ elements, so is computable. Can membership in A_k be decided by an algorithm working with parameter k (i.e., uniformly in k)? The answer is negative because the function

$$f(k, x) = \begin{cases} 1, & \text{if } C(x) \leq k, \\ 0, & \text{otherwise}, \end{cases}$$

is not computable and the reason is that the set $A = \{(x, k) : C(x) \leq k\}$ is c.e., but not computable ([Calude 2002], Corollary 5.37). Note that this negative result holds true in spite of the set A_k being very small, as we have shown in the previous section.

Suppose the universal image construction program exists and further suppose that the time it requires to generate each image has some computable limit. This limit may be long, i.e., we allow the program to run for hours (or even centuries) to generate each image, but there is nevertheless some bound, perhaps one that is a function of the resolution and bit depth. Then we can adapt this program to

[3] See more details in [Calude 2002].

compute the algorithmic complexity of an arbitrary string, which is impossible.

The fact that the simple strings cannot be identified can also be seen as a consequence of algorithmic incompleteness. If a simple program could identify all the simple (complexity $\leq k$) strings of a particular size, it could be modified to print the first string in any enumeration of all strings of the chosen size that is not simple. If k is larger than the size of the program contradiction results.

There Is No Universal Image Generator

In view of Corollary 5.37 in [Calude 2002], the set of compressible strings

$$\{x : C(x) \leq |x| - \log |x|\}$$

is c.e. but not computable.

This set is also very small. Indeed, the counting argument in section 3 applies: the proportion of n-bit compressible strings in the set of a n-bit strings is smaller than $2^{n-\log n}/2^n \leq 1/n$.

There are sets much more uncomputable than c.e. (but not computable) sets. One such class is the *immune sets*, i.e. infinite sets that contain no infinite c.e. subset.

Now we can prove our main result:

> *If the Thesis is accepted then the set of NL images is not c.e., hence there is no program that generates all (and only all) NL images.*

Indeed, NL is an infinite subset of $\{x : \log |x| + \text{constant} < C(x) \leq |x| - \log |x|\}$, hence an infinite subset of the immune set $\{x : \log |x| + \text{constant} < C(x)\}$ (see Corollary 5.34 in [Calude 2002]), hence NL is not c.e.

Is the Thesis Valid?

Early in this section we have presented arguments in favour of the Thesis. Next we complement them with a mathematical analysis which shows the robustness of the Thesis.

The specific logarithmic functions used in (2) and (3) may seem rather arbitrarily chosen. Fortunately all our previous results hold true in a more general context. Consider the class of possible thresholds

$$T = \{f : f = \text{computable}, \lim_{n\to\infty} f(n) = \infty, \text{ and for all } c > 0,$$
$$\lim_{n\to\infty} n - cf(n) = \infty\}.$$

The threshold functions used in the Thesis, $f(n) = \log n + \text{constant}, g(n) = n - \log n$, are clearly in T. The functions $\log\log n, \sqrt{n}$ are in T, but the functions $\lfloor n/2 \rfloor, n-2$ are not in T because they grow too fast.

The class T has a few interesting properties: a) if $f \in T$ and a is a positive integer then $af \in T$, b) if $f, g \in T$ then $\max\{f, g\}, \min\{f, g\}, f + g \in T$, c) if $f, g \in T$ then $g(n) < n - f(n)$ for almost all n. For b) note that $f(n) + g(n) \leq 2\max\{f(n), g(n)\}$; c) follows from b).

With a very similar argument (and using the same results in [Calude 2002]) we can show that for all functions $f, g \in T$ the set $\{x : f(|x|) < C(x) \leq g(|x|)\}$ is immune.

6 Conclusion

In summary:

- We have argued that the goal of generating "all possible" images is poorly posed, since the set of all images is overwhelmingly composed of nearly indistinguishable noise images like Fig. 2,
- We proposed that generating all (and only all) compressible images is a reasonable restatement of this goal. The compressible images contain all the NL images, but the vast set of noise images is excluded.
- We proposed a natural Thesis for NL images and argued its validity.
- Assuming the Thesis we have shown that there is no program that generates all (and only all) NL images.

The argument in this paper may be applicable in other fields dealing with compressible data. Intuitively, a complex phenomenon has large algorithmic complexity, and there are many simple theories (with small algorithmic complexity) that do not completely describe it. On the other hand, as the complexity of a theory approaches that of the original phenomenon the theory looses its value as a compressed representation of some aspects of the phenomenon.

Acknowledgements

Doug Fidaleo, Dan Ruderman, John Schlag, and Ludwig Staiger provided helpful comments. Fig. 2 uses random bits from a physical source, obtained from `www.random.org`. Figs. 3, 4 use pseudorandom values.

References

ATICK, J., AND REDLICH, A. N. 1992. What does the retina know about natural scenes. *Neural Computation*, 4, 196–210.

BLAKE, A., AND ZISSERMAN, A. 1987. *Visual Reconstruction*. MIT Press, Cambridge, MA.

BODEN, M. 2004. *The Creative Mind*. Routledge.

CALUDE, C. S., AND JÜRGENSEN, H. 1994. Randomness as an invariant for number representations. In *Results and trends in theoretical computer science (Graz, 1994)*, vol. 812 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 44–66.

CALUDE, C. S., AND JÜRGENSEN, H. 2005. Is complexity a source of incompleteness? *Adv. in Appl. Math. 35*, 1, 1–15.

CALUDE, C. S. 2002. *Information and Randomness: An Algorithmic Perspective*. Springer-Verlag.

CALUDE, C. S. 2005. Algorithmic randomness, quantum physics, and incompleteness. In *Machines, computations, and universality*, vol. 3354 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 1–17.

HAMMER, D., ROMASHCHENKO, A. E., SHEN, A., AND VERESHCHAGIN, N. K. 2000. Inequalities for Shannon entropy and Kolmogorov complexity. *Journal of Computer and System Sciences 60*, 2, 442–464.

HOFSTADTER, D. 1985. Metafont, metamathematics, and metaphysics: Comments on Donald Knuth's article "The Concept of a Meta-Font". In *Metamagical Themas: Questing for the Essence of Mind and Pattern*. Basic Books, New York, NY, ch. 13, 260–296.

RUDERMAN, D., AND BIALEK, B. 1994. Statistics of natural images: scaling in the woods. *Physical Review Letters*, 73, 814–817.

SKODRAS, A., CHRISTOPOULOS, C., AND EBRAHIMI, T. 2001. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine 18*, 5, 36–58.

STAIGER, L. 2002. The Kolmogorov complexity of real numbers. *Theor. Comput. Sci. 284*, 2, 455–466.