# Inspiration-Triggered Search: Towards Higher Complexities by Mimicking Creative Processes

Milan Rybář
info@milanrybar.cz

Heiko Hamann
heiko.hamann@uni-paderborn.de

Heinz Nixdorf Institute, Department of Computer Science,
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany

## ABSTRACT

Open-ended evolution is still an unachieved goal in evolutionary computation. Evolution guided by objective functions can easily be trapped on local optima. Our approach is inspired by evolutionary paths along stepping stones, as observed in user behaviors of Picbreeder. We propose a general framework, inspiration-triggered search, which tries to roughly mimic the creative design process of a human being. Instead of using a fixed objective function, the search algorithm itself is free to switch between objectives within certain constraints but inspired by features of the currently evolved artifacts. The overall optimization task is to generate complex artifacts that cannot be generated by a direct optimization approach. In contrast to other approaches that make extensive use of external knowledge (e.g., Innovation Engines), we try to approach the ambitious goal of virtually bootstrapping a creative process from scratch. The proposed method is tested in the domain of images, that is to find complex and aesthetically pleasant images, and is compared to novelty search.

## Categories and Subject Descriptors

I.2.6 [**AI**]: Learning—*Connectionism and neural nets*

## Keywords

creative process, evolutionary art, non-objective search

## 1. INTRODUCTION

Getting trapped on local optima is one of the main challenges of stochastic optimization methods besides the difficulty of designing objective functions that do not bias the search in an undesired way. Novelty search [8, 9], for example, shows that the standard approach based on objective functions can be replaced by a search for novel behaviors. However, then the challenge is to design an appropriate behavioral distance measure and to limit the search space usefully [6]. The recently proposed Innovation Engine [13] produces an impressive variety and complexity of artifacts but leverages the classification capabilities of a Deep Neural Network (DNN) that has absorbed 1.3 million images. One could argue that the Innovation Engine is a sophisticated method to make the DNN release information about these images again. Here, we want to follow the even more ambitious goal of generating a creative process from scratch with little a priori knowledge. The question, whether it is overambitious to virtually ask a blind and deaf artist to produce something radically new, is fair but we take the challenge.

In this paper, we propose a novel algorithm called inspiration-triggered search (ITS) that operates without a fixed goal and tries to roughly mimic creative processes by changing the objective function on the fly and by using evolutionary algorithms. We try to generate complex artifacts that cannot be generated by a direct optimization approach. The rationale is to abstract creative processes of artists, researchers, or just users, for example, interacting with Picbreeder [18] (a website allowing users to collaboratively evolve images using artificial evolution). Picbreeder users were able to evolve images of particular complexity that could not be directly re-evolved from scratch using the desired image as an objective function [23]. We suppose that users have an abstract target artifact in mind and develop their works towards this goal. However, they are open to switch to another goal if they feel the opportunity (e.g., when eyes of a face remind them of a car's wheels [20]). In other words, they are inspired by the currently more interesting or promising alternative compared to their former goal. We test our approach in the grayscale image generation domain, that is, we try to find complex and aesthetically pleasant images. Obviously, we will run into hard problems of how to score complexity, aesthetics, and beauty which ultimately is, as we all know, in the eye of the beholder. Hence, we also propose two simple and practical methods to test for complexity. Both methods are used for post-evaluations only and do not bias our search algorithm. We compare our approach with a standard evolutionary algorithm and novelty search. We find that it is able to compete with the state of the art.

## 2. INSPIRATION-TRIGGERED SEARCH

ITS mimics creative processes by changing an objective function on the fly for stochastic optimization techniques using little a priori knowledge in the form of predefined feature functions. ITS is a general framework for stochastic optimization while here we focus on evolutionary computa-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*GECCO '16, July 20-24, 2016, Denver, CO, USA*

© 2016 ACM. ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908815

**Algorithm 1** General framework of ITS

```
1: procedure INSPIRATION-TRIGGERED SEARCH
2:     P ← ∅ is population, f ← ∅ is objective function
3:
4: Diversification phase:
5:     Diversify population P
6:     if inspiration criterion is met then
7:         goto Inspiration phase
8:     else
9:         Simplify objective function f
10:    goto Diversification phase
11:
12: Inspiration phase:
13:    Create new objective function f using population P
14:    goto Optimization phase
15:
16: Optimization phase:
17:    Optimization with f and P for at most k iterations
18:    if inspiration criterion is met then
19:        goto Inspiration phase
20:    else if convergence criterion is met then
21:        goto Diversification phase
22:    goto Optimization phase
```

tion. ITS has three phases: inspiration phase, optimization phase, and diversification phase (see Alg. 1).

In the inspiration phase, a new objective function is determined by interesting or promising properties of the current population. The optimization phase follows which is a standard optimization using the current objective function. If ITS is 'inspired' by a certain property in the current population, which is detected by an 'inspiration criterion' every $k$-th iteration, it switches to the inspiration phase to modify the objective function accordingly. Otherwise, if the optimization converges, ITS switches to the diversification phase to add diversity to the population. ITS diversifies the population until an interesting property is detected by the inspiration criterion which triggers the inspiration phase again. If the criterion is not met after diversifying the population, the objective function is reduced in complexity. ITS starts in the diversification phase to bootstrap an initial population and objective function. ITS can be considered as an approach towards open-ended automated incremental evolution without a predetermined objective. It starts with a simple objective and gradually tries to complexify it (sometimes interrupted by an interim simplification phase). In the following, we describe ITS in the context of our testing domain of generating images, which is the application example that we use in this paper.

## 2.1 Inspiration by parts of individuals

Next we specify how our algorithm can change its objective function triggered by features of the current population to implement the inspiration phase and to usefully increase the complexity in the population. In Picbreeder and other examples of interactive evolution the user, possibly an expert, selects interesting individuals. Such a selection can sometimes be motivated by interesting parts of individuals in the population (i.e., interesting regions of images) that have potential to solve the optimization problem (or just to satisfy the user's current objective) once they have been improved. Similarly, a teacher at an art school (i.e., an expert) may inspire a student by pointing to parts of a painting to focus on. We call such a focused part 'window' denoted by $W$ (i.e., a region of an image). This approach is applicable to any domain that can be logically split into parts.

In our approach, the figurative idea of both expert and student is automated in an algorithm. The expert part of the algorithm selects areas of interest following predefined criteria and the student part of the algorithm then tries to improve these areas using independent criteria. To formalize the concept of an expert, we define 'views' and 'descriptions'. We call a collection of windows a view denoted by $V = \{W_0, W_1, \dots\}$. A view $V$ represents an expert's viewpoint (i.e, regions of an image that have potential to be improved). For example, we use a view consisting of windows/regions defined by the detection of contours (later specified in Sec. 3.3), see Fig. 1. The expert assigns a description $D_I$ dedicated to each individual $I$ of a population. The expert's description is a set of views $V \in D_I$ (for simplicity we allow the notation $W \in D_I$). With a description the expert highlights interesting properties of individual $I$ via views $V$ and the respective windows $W$.

The student part of our algorithm selects an objective function that is composed of a predefined set of feature functions $\{F_0, \dots, F_N\}$ (later specified in detail in Sec. 3.2). Each is assigned and limited to only a part of an individual, for example, the 'symmetry' feature is computed only within a region of the image. Using the same concept as above, an objective function $f$ is defined as a set of views $f = \{V_0, \dots, V_N\}$ which are, in turn, sets of windows $W$ (for simplicity we allow the notation $W \in f$). Each view $V_i$ has an assigned feature function $F_i$. Later we use these views $V \in f$ to optimize their corresponding feature functions within parts of individuals defined by windows $W \in V$, while ignoring parts outside of any of these windows. We use the notation $F(W, I)$ to score an individual $I$ by feature function $F$ limited to an area defined by a window $W$.

The student part of our algorithm does not just directly adopt the areas of interest (windows $W$) defined by the expert's description $D_I$. Instead the description is used to inspire the selection of a few windows only. Furthermore, sometimes it is required to reduce the number of windows which also requires an appropriate selection mechanism. To implement that selection process and to connect the expert's description $D_I$ with the choice of an objective function $f$ we define the description metric $\delta(f, D_I) \to \mathbb{R}$. It assigns a score to pairs of objective functions $f$ and descriptions $D_I$ to formalize inspiration. The description metric is based on two functions: coverage $C(f, D_I) \to \mathbb{R}$ and extension $E(f, D_I) \to \mathbb{R}$ which are specified below. It is defined by

$$\delta(f, D_I) = C(f, D_I)^\alpha \cdot E(f, D_I)^\beta \tag{1}$$

with parameters $\alpha, \beta \in \mathbb{R}$. These two factors represent the typical tradeoff of exploitation (coverage) and exploration (extension). Coverage measures how well the current objective function $f$ represents the expert description $D_I$ (i.e., how much of the area the expert points to is covered by windows defined in the objective function). By focusing only on coverage, the search would mostly preserve what has been found. Extension measures how much the expert's description $D_I$ could extend the current objective function $f$. By focusing only on extension, the search would always try to add and explore new areas. Obviously, it is important to find the right balance between exploration and exploitation.

For the implementation of the process of adding and removing windows to and from the objective function, we define the description metric for our testing domain of generating images. First, we focus on a particular window $W_D \in D_I$ and want to estimate the utility of adding it to a particu-

lar view $V_f \in f$ in the objective function $f$. The extension $e(V_f, W_D)$ represents how much area in pixels is gained by extending $V_f$ with $W_D$. We compute the area of the complement between $W_D$ and $V_f$ and normalize it by the maximal area that could be gained (i.e., the complement of $V_f$ of the whole image) which represents the ratio of gained space. In addition, we consider the quality of window $W_D$ according to the respective feature function $F_{V_f}$ defined via the view $V_f$. We define

$$e(V_f, W_D) = \frac{\text{area}(W_D \setminus V_f)}{\text{area}(V_f^{\complement})} \cdot F_{V_f}(W_D, I) \quad (2)$$

with $V_f^{\complement}$ is the complement of $V_f$ and *area* gives the size of the argument in pixels. Windows with bigger $e$ are preferable. Now we extend this definition to extension $E(f, D_I)$ which is the average over all windows $W_D \in D_I$ combined with all views $V_f \in f$. We define

$$E(f, D_I) = \frac{1}{\|D_I\|_{\mathbb{W}} \|f\|_{\mathbb{V}}} \sum_{W_D \in D_I} \sum_{V_f \in f} e(V_f, W_D) \quad (3)$$

with $\|f\|_{\mathbb{V}}$ gives the total number of views in objective function $f$ and $\|D_I\|_{\mathbb{W}}$ is the total number of windows in description $D_I$. If no view of description $D_I$ contains a window, we define $E(f, D_I) = 0$.

The coverage $c(W_f, V_D)$ of a particular window $W_f \in V_f \in f$ from the objective function $f$ and the view $V_D \in D_I$ of the expert's description $D_I$ represents how much area of $W_f$ is preserved by replacing view $V_f$ with view $V_D$ in the objective function $f$. We compute the area in pixels of the intersection between $W_f$ and $V_D$ and normalize it by the size of the window $W_f$ which gives the ratio of preserved area. We also consider the quality of the window $W_f$ according to the respective feature function $F_{V_f}$ for view $V_f$. We define

$$c(W_f, V_D) = \frac{\text{area}(W_f \cap V_D)}{\text{area}(W_f)} \cdot F_{V_f}(W_f, I). \quad (4)$$

Now we extend this definition to coverage $C(f, D_I)$ which is the average over all windows $W_f \in f$ combined with each view $V_D \in D_I$. We define

$$C(f, D_I) = \frac{1}{\|f\|_{\mathbb{W}} \|D_I\|_{\mathbb{V}}} \sum_{W_f \in f} \sum_{V_D \in D_I} c(W_f, V_D) \quad (5)$$

with $\|D_I\|_{\mathbb{V}}$ gives the total number of views in description $D_I$ and $\|f\|_{\mathbb{W}}$ is the total number of windows in objective function $f$. If no view of the objective function $f$ contains a window, we set $C(f, D_I) = 1$.

## 2.2 Alternating inspiration and diversification

The inspiration criterion is met when the description metric value $\delta(f, D_I)$ is bigger than a threshold $I_c \in \mathbb{R}$ for any individual in the population. The inspiration phase is triggered when the inspiration criterion is met either during the optimization phase or the diversification phase. Once the inspiration phase is done, ITS continues with an optimization phase. During the inspiration phase, we modify the current objective function $f$ in the following way. First, we select the description $D_I$ of a particular individual $I$ with the highest value of description metric $\delta(f, D_I)$. Second, we modify the current objective function $f$ by adding a window $W_D$ from description $D_I$ to a view $V_f \in f$ or by removing a window from $V_f \in f$. A window is added by randomly selecting a view-window pair $(V_f, W_D)$ with probability defined by $e(V_f, W_D)$, whereas the window is chosen from the description $W_D \in D_I$ and the view is chosen from the objective function $V_f \in f$. That determines two actions in one: the actual choice of window $W_D$ (i.e., a region of the image) and the choice of view $V_f$ (i.e., the feature function assigned to that window). Later, the area defined by window $W_D$ is optimized using the respective feature function of view $V_f$. We remove a window $W_f$ from the objective function $f$ with probability $1 - c(W_f, D_I)$ if $1 - c(W_f, D_I)$ is above a threshold $R \in \mathbb{R}$.

The diversification phase is triggered when convergence is detected during the optimization phase. Once the diversification phase is done, ITS continues with an optimization phase. During the diversification phase, we first remove one window $W_f \in f$. Each window $W_f \in f$ is rated by the average coverage

$$\frac{1}{|P| \cdot \|D_I\|_{\mathbb{V}}} \sum_{I \in P} \sum_{V_D \in D_I} c(W_f, V_D) \quad (6)$$

where $|P|$ is the size of the population and $\|D_I\|_{\mathbb{V}}$ is the number of views in description $D_I$. We remove the window with the lowest value. We stay in the diversification phase and continue to remove windows until either the inspiration criterion is met or, in the worst case, we keep removing windows until the objective function is 'empty'. The description metric value $\delta$ is expected to increase when windows are removed because that simplifies the objective function.

We use a linear scalarization with unit weights to deal with our multi-objective optimization problem. The fitness score of an individual $I$, depending on the current objective function $f$ with respective feature functions $F_{V_f}$ for $V_f \in f$, is defined by

$$\sum_{V_f \in f} \sum_{W_f \in V_f} F_{V_f}(W_f, I). \quad (7)$$

# 3. TESTING DOMAIN: IMAGES

We test our approach in the grayscale image generation domain, that is to find 'complex' images and potentially even aesthetically pleasant images for humans. Many different approaches have been proposed to create interesting and aesthetically pleasant images (e.g, evolutionary art [15]). The main challenge is that the objective function is difficult to define formally because it is subjective. We chose this domain for two reasons. First, it was used before in Picbreeder [18] as a metaphor for the stepping stone challenge in creativity. Second, the problems of crafting an appropriate objective function and the challenge of premature convergence in the image domain are still similar to more complex domains such as evolutionary robotics.

## 3.1 Generating images with evolution

We encode images by compositional pattern producing networks (CPPN) [19], which are a variation of artificial neural networks differing in their set of activation functions. We use a similar setting as in [18, 19] but we only use feed-forward neural networks. The available CPPN activation functions are signed versions of sigmoid, Gaussian, sine, and linear functions. Network inputs are the Cartesian coordinates and the distance from the image's center for each pixel, linearly scaled to $[-1, 1]$. The network's output $x$ is transformed by $255|x|$ and determines the pixel's brightness.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| PopulationSize | 150 | MutateRemSimpleNeuronProb | 0.001 |
| DynamicCompatibility | True | RecurrentProb | 0.0 |
| MinSpecies | 10 | MutateWeightsProb | 0.9 |
| MaxSpecies | 15 | MutateWeightsSevereProb | 0.5 |
| YoungAgeTreshold | 15 | WeightMutationRate | 0.75 |
| YoungAgeFitnessBoost | 1.1 | WeightMutationMaxPower | 1.0 |
| SpeciesMaxStagnation | 15 | WeightReplacementMaxPower | 2.0 |
| OldAgeTreshold | 35 | MaxWeight | 8.0 |
| SurvivalRate | 0.2 | MutateNeuronActivationTypeProb | 0.02 |
| CrossoverRate | 0.75 | DisjointCoeff | 2.0 |
| OverallMutationRate | 0.25 | ExcessCoeff | 2.0 |
| InterspeciesCrossoverRate | 0.01 | WeightDiffCoeff | 1.0 |
| MultipointCrossoverRate | 0.6 | ActivationFunctionDiffCoeff | 1.0 |
| RouletteWheelSelection | False | CompatTreshold | 6.0 |
| MutateAddNeuronProb | 0.05 | MinCompatTreshold | 0.2 |
| MutateAddLinkProb | 0.04 | CompatTresholdModifier | 0.3 |
| MutateRemLinkProb | 0.04 | | |

**Table 1: Used parameters of NEAT.**

**Figure 1: Examples of selected windows based on detecting contours.**

Note, this image representation has in principle infinite resolution. We generate images with a resolution of $256^2$ pixels and we use the OpenCV library [2] to manipulate images.

We use NEAT [21, 19] to evolve CPPN. Its incremental complexification, that is incrementally adding new nodes over generations, is in principle similar to our approach (i.e., starting with simple images and complexifying them incrementally). Our implementation is based on MultiNEAT [3], which is a portable software library implementing NEAT. Table 1 shows NEAT parameters used in our experiments, which are based on original NEAT, Picbreeder, other similar experiments [21, 5], and preliminary experiments. CPPN starts without hidden nodes and with Gaussian as output activation function, which can be later changed by mutations.

## 3.2 Features

We use measures from the field of computational aesthetics [7, 12] and measures defining sharpness of images [14]. We provide our algorithm with 7 features to choose from.

Global contrast factor (GCF): proposed in [11], computes contrasts, the average difference between neighboring pixels, at various resolutions, and uses a perceptual luminance to build a human-perception-based method. Here, we only use super-pixels of sizes up to 50 due to the low image resolution.

Relaxed symmetry (RS): proposed in [4], calculates the difference in intensity of opposing pixels around horizontal, vertical, and diagonal axis.

Tenengrad (T): focus measure based on the magnitude of an image gradient, defined as $\sum_{(i,j)\in I}(G_x(i,j)^2+G_y(i,j)^2)$, where $G_x$ and $G_y$ are $X$ and $Y$ image gradients computed by convolving an image $I$ with Sobel operators [16, 22].

Normalized variance (NV): is defined as the variance divided by the mean of an image [16, 22].

Choppiness: computes the average standard deviation of pixels over all $5 \times 5$ windows within the image.

Image complexity by JPEG compression: compresses an image by JPEG compression with a quality setting of 75% and computes an image complexity defined by Machado and Cardoso [10] as $\frac{RMS}{CompressionRatio}$ where $RMS$ is the root mean square difference between the original and the compressed image. The compression ratio is the ratio between the original and the compressed image size.

Maximum of absolute Laplacian: defined as maximum of absolute values of the Laplacian of an image.

## 3.3 Used configuration of ITS

We use only one view in the expert's descriptions of images by detection of contours. The idea is to mimic a behavior similar to human vision. We detect edges by the Canny Edge detector and the result is used to find contours using a functionality provided by the OpenCV library. The window for the view is defined by an oriented bounding rectangle of a contour. We consider only windows with an area bigger than 2000 pixels$^2$ and smaller than a quarter of the image. In Fig. 1 we give example results of selected windows based on contour detection.

To diversify the population during the diversification phase, we apply mutations as implemented in NEAT to each individual twice. If the inspiration criterion is not met with the empty objective function (i.e., no windows in all views) in 10 tries, then NEAT is reset (i.e., ITS starts from scratch).

Here, extension values of the description metric have much smaller values in comparison with coverage values. Instead of tweaking parameters $\alpha$ and $\beta$, we scale the area of the complement by function $f(x) = \frac{2x}{x+\omega}$ where $\omega$ is the maximum value of $x$. We re-define eq. 2 by

$$e(V_f, W_D) = \frac{2\,\mathrm{area}(W_D \setminus V_f)}{\mathrm{area}(W_D \setminus V_f) + \mathrm{area}(V_f^{\complement})} \cdot F_{V_f}(W_D, I). \quad (8)$$

This is tailored for the chosen image size and allows a simple setting of $\alpha = 1$ and $\beta = 1$, which is supposed to be a well balance between exploitation and exploration.

The objective function is modified by removing a window followed by adding a new window. Then we reset the current information about the best fitness value and the number of stagnations in NEAT. This way, the new objective function is not influenced by previous objective functions and we do not require to normalize fitness across different objective functions. After a maximal number of 10 iterations of the optimization (here using NEAT) the inspiration criterion is checked, which can trigger a transition to the inspiration phase. The optimization is stopped by the convergence limit that triggers when there was no improvement for 20 iterations. The only difference between the experiments using different sets of feature functions are appropriately chosen inspiration thresholds $I_c$ for the inspiration criterion and thresholds $R$ for the removal of windows from the objective function (see Table 2).

## 3.4 Novelty search

For comparison we use novelty search [8, 9] with random archive selection of 6 individuals each generation. The behavioral distance between images is defined as the Euclidean distance (in pixel space). The number of nearest neighbors is $k = 15$. Novelty search is combined with a fitness function in order to compare it with ITS, which uses feature functions. The score of an individual is given by the weighted sum of the normalized novelty and fitness score. Fitness and novelty have the same weight. This setting is based on [6].

| | Complexity test by optimization | | ITS setting | |
|---|---|---|---|---|
| Set of features | Uninspired search | ITS | $I_c$ | $R$ |
| NV | 2×7+5×2 | 4×5 | 0.01 | 0.3 |
| RS | 2×7+8×2 | 3×5 | 0.01 | 0.4 |
| GCF | 5×3 | 3×5 | 0.1 | 0.4 |
| T | 5×3 | 3×5 | 0.04 | 0.4 |
| NV+RS | 3×3 | 3×5 | 0.04 | 0.3 |
| NV+RS+GCF | 3×3 | × | 0.04 | 0 |
| NV+RS+T | 3×3 | × | 0.1 | 0.5 |
| NV+RS+GCF+T | 4×3 | × | 0.1 | 0.5 |

**Table 2: Summary of experimental settings. Column 'Complexity test' shows number of chosen images for complexity test by optimization times a number of trials (e.g., 5×3 is 5 images, each image in 3 trials). Column 'ITS setting' shows ITS parameters used in our experiments: the inspiration threshold $I_c$, and the removal threshold $R$.**

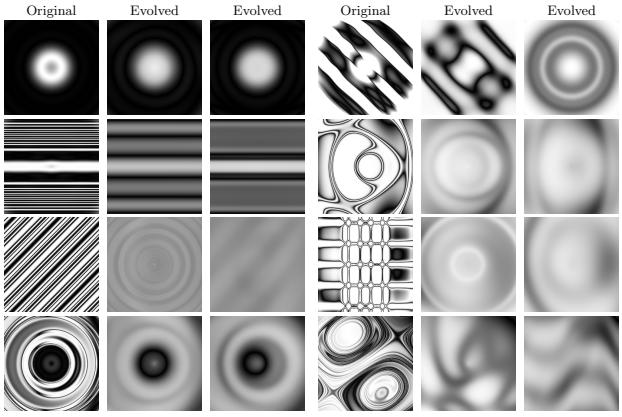



**Figure 2: Examples of images used for the complexity test by optimization. Shown evolved images are visually the most similar to the original image.**

# 4. RESULTS

Next we want to evaluate ITS and see whether it produces more complex and favorable images in comparison to a standard optimization technique, here represented by NEAT and called 'uninspired search' (US) for now, as well as novelty search using the same set of features. How to define complexity in the image domain is a challenge by itself. A human being could visually compare generated images, but that is subjective (see Fig. 5 for a selection of a few images generated by ITS, more example images for each algorithm and for all used sets of features and the source code are available online[1]). We have developed two different approaches for our evaluation: a complexity test by optimization and by a complexity classifier.

## 4.1 Complexity test by optimization

Our initial idea based on [23] is to test whether a particular target image can be found by directly optimizing using NEAT and pixel distance as an objective function. The optimization minimizes Euclidean distance in pixel space to the target image. We define a target image to be complex if it cannot be found by this approach (we test that with limited resources and hence cannot exclude that it could be found with more resources). We split the domain of pixel distances into two ranges for classification purposes based on our preliminary experiments. If the pixel distance is lower than $1.5 \times 10^4$ in the $600^{\text{th}}$ generation, the image is classified as simple, otherwise it is classified as complex image. In Fig. 2 we give a few example results and in Fig. 6 we give the statistical results of the complexity test by optimization. The green line indicates the complexity threshold.

For a single image, we take the pixel distance of the best individual of the last generation of each evolutionary run (different numbers of tested images and trials were used for the experiments, see Table 2) which represents the difficulty to find the image. To compare our approach, we select representative images from it and aggregate their results. Images produced by uninspired search were chosen from the best individuals of last generations. Images from ITS could be chosen from any generation. We selected images from ITS with different settings ($I_c$ and $R$, see Table 2) from generations that were considered as important (i.e., when the objective function was modified or when the optimization converged). The algorithm with higher values can be considered to generate more complex images. Table 2 shows the total number of tested images for each set of features.

[1]http://milanrybar.cz/inspiration-triggered-search/

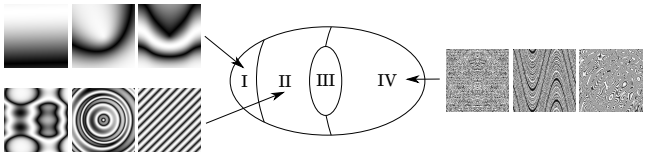
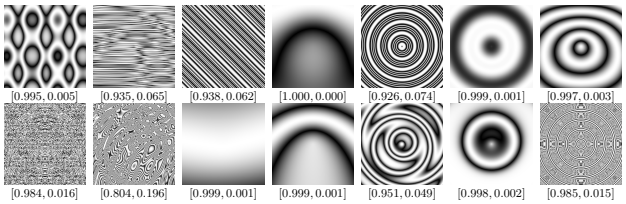
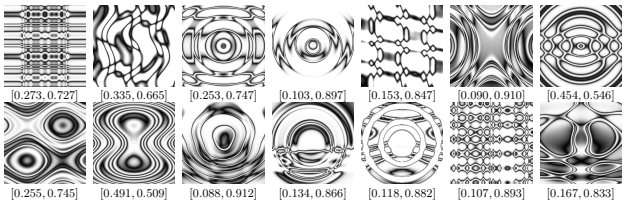



**Figure 3: Idea of splitting the image domain into 4 classes according to complexities. Class I: images with simple shapes and gradients (e.g., images at coordinates $[2,3]$, $[1,4]$ (row, column) in Fig. 4a). Class II: images with easily created patterns due to image representation by CPPN (e.g., images at $[2,5]$, $[1,1]$, $[2,6]$). Class III: 'true complex' images. Class IV: images with high spatial frequencies (e.g., images at $[2,7]$, $[2,1]$, $[2,2]$, $[1,2]$).**




(a) Simple images




(b) Complex images

**Figure 4: Example images from training set for complexity classifier with final complexity classifications (simplicity and complexity node).**

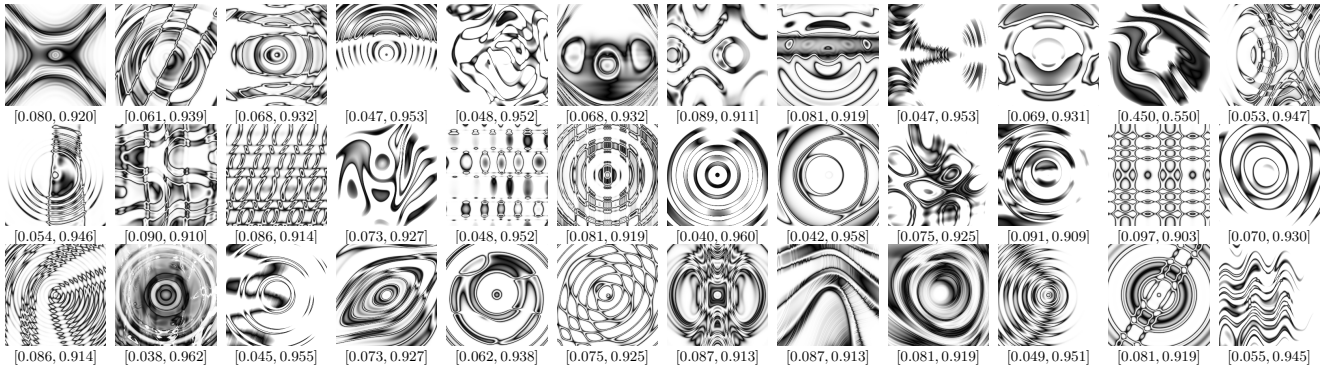



**Figure 5: Diversity of images from ITS with their classification by the complexity classifier.**

Our experiments revealed that the optimization (NEAT) is only able to find visually simple images. That is, using this test of complexity by optimization it is relatively easy to generate images that will be labeled as complex. For example, the images in Fig. 4a at coordinates $[1, 6]$, $[2, 3]$, $[2, 4]$ (row, column) could be found, but visually simple images at coordinates $[1, 7]$, $[1, 3]$, $[2, 5]$ were labeled as complex. However, intuitively and subjectively we would label these images as simple. Hence, the complexity test by optimization is of limited use to implement a thorough distinction between simple and complex images.

In direct analogy to an idea from the field of cellular automata [1], one can think of the image space as visualized in Fig. 3. Following that idea, there would be four classes of images showing different complexity levels; with rather simple images in classes I and II, noisy/random images in class IV, and interesting/complex images in class III. The complexity test by optimization is possibly only able to distinguish class I from the rest (II, III, and IV). For example, images generated by uninspired search used for the complexity test in Fig. 6 from NV and RS belong to the class I, and from GCF to the class II and IV. However, we would like to label an image as complex only when it is from the class III. Hence, we penalize images with high spatial frequencies (class IV) in all experiments containing the features T or GCF but only with limited success.

Results are shown in Fig. 6. Table 3 gives the classification and statistical comparison. The complexity of images from uninspired search is biased by the particular feature function. NV and RS produced simple images, even when combined. On the contrary, GCF and T produced complex images. When the set of features for uninspired search contained any of the latter, the images were found to be complex. Still, these images could be assigned to classes II and IV. When both algorithms produced images classified as complex, the statistical comparison is biased by the chosen images of the complexity test. Due to limited resources ITS was not tested with multiple features using GCF or T. These images would probably be classified as complex (GCF and T as single features already gave complex images).

## 4.2 Complexity classifier

In addition to the complexity test by optimization, we have also applied standard methods of machine learning to classify images as simple and complex. First, we created a data set of 4113 simple and 1343 complex, handpicked images from our preliminary experiments. We labeled an image as simple when it contained a simple gradient, blurred objects (class I and II), high spatial frequencies (class IV), or when it was easily created by mutations during initial populations in NEAT (class II). See Fig. 4a for examples. We labeled an image as complex when it was sharp or it included nontrivial shapes. See Fig. 4b for examples.

Second, we chose seven features providing addition information about an image (see Sec. 3.2). Note that this choice of features introduces no bias, because only few of them were used in combination as feature functions in our experiments. We randomly split the data set into training (75%) and testing data (25%). The mean and the standard deviation of each feature were used to scale the input values.

We trained a feed-forward neural network with two output nodes (10 nodes in one hidden layer) and a bias by back-propagation for 50 epochs using the PyBrain [17] library. One output node is used to classify the image as simple (simplicity node) and another output node for complex (complexity node). The node with a higher value wins the classification. The classification accuracy on the testing data is 91.5%. We define 'complexity difference' as the difference between the complexity and the simplicity node. Images are classified as complex if their classification difference values are positive and simple if negative.

To compare an algorithm, the complexity difference values of all individuals from the last generations of all trials are aggregated. The algorithm with higher values can be consider as producing more images classified as complex in the population (i.e., it has a higher probability to generate an image classified as complex). For the comparison, we use ITS with parameters shown in Table 2 based on our preliminary experiments. We ran each algorithm with each set of features for 2000 generations in 30 evolutionary runs.

Note that this unsophisticated technique of selecting images and aggregating results from the last generation underestimates the performance of both ITS and novelty search. ITS contains different phases and last populations that were generated by, for example, the diversification phase might give poor performance. Similarly for novelty search but possibly with limited impact because it does not have explicit, different phases of optimization and diversification. The results shown in Fig. 7 indicate a limited impact for both. However, if we are able to show that ITS creates more complex images in comparison to uninspired search even with this handicap, then our approach is useful.

Fig. 7 shows complexity difference values over generations for each algorithm (see the online material for evolved im-

| | Complexity test by optimization | | | | Complexity classifier | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Set of features | Uninsp. s. | C. | ITS | p-value | p-value | Uninsp. s. | C. | ITS | C. | Novelty s. | p-value |
| NV | simple | $<$ | complex | $< 0.001$ | $< 0.001$ | -0.312 | $<$ | 0.418 | $>$ | -0.002 | $< 0.001$ |
| RS | simple | $<$ | complex | $< 0.001$ | $< 0.001$ | 0.579 | $>$ | 0.538 | $>$ | 0.524 | $< 0.001$ |
| GCF | complex | $>$ | complex | $< 0.001$ | 0.014 | 0.465 | $<$ | 0.458 | $>$ | -0.043 | $< 0.001$ |
| T | complex | $>$ | complex | $< 0.001$ | $< 0.001$ | 0.023 | $<$ | 0.583 | $>$ | 0.039 | $< 0.001$ |
| NV+RS | simple | $<$ | complex | $< 0.001$ | $< 0.001$ | 0.007 | $<$ | 0.582 | $>$ | 0.484 | $< 0.001$ |
| NV+RS+T | complex | | × | | $< 0.001$ | -0.227 | $<$ | 0.103 | $<$ | 0.508 | $< 0.001$ |
| NV+RS+GCF | complex | | × | | $< 0.001$ | 0.254 | $<$ | 0.471 | $>$ | 0.327 | $< 0.001$ |
| NV+RS+GCF+T | complex | | × | | $< 0.001$ | -0.119 | $<$ | 0.112 | $<$ | 0.358 | $< 0.001$ |

**Table 3: Evaluation using complexity test by optimization and the complexity classifier, median of complexity difference over all runs (bigger is better); results of comparisons (C.) with p-values (Wilcoxon rank sum test); features: normalized variance (NV), relaxed symmetry (RS), global contrast factor (GCF), Tenengrad (T).**

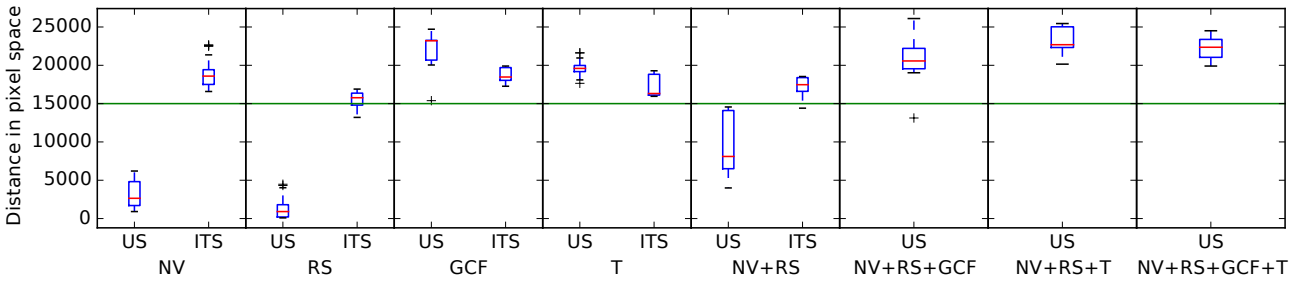



**Figure 6: Results according to complexity test by optimization (600$^{\text{th}}$ generation). Green line (distance of $1.5 \times 10^4$) indicates threshold for classification. US is uninspired search.**

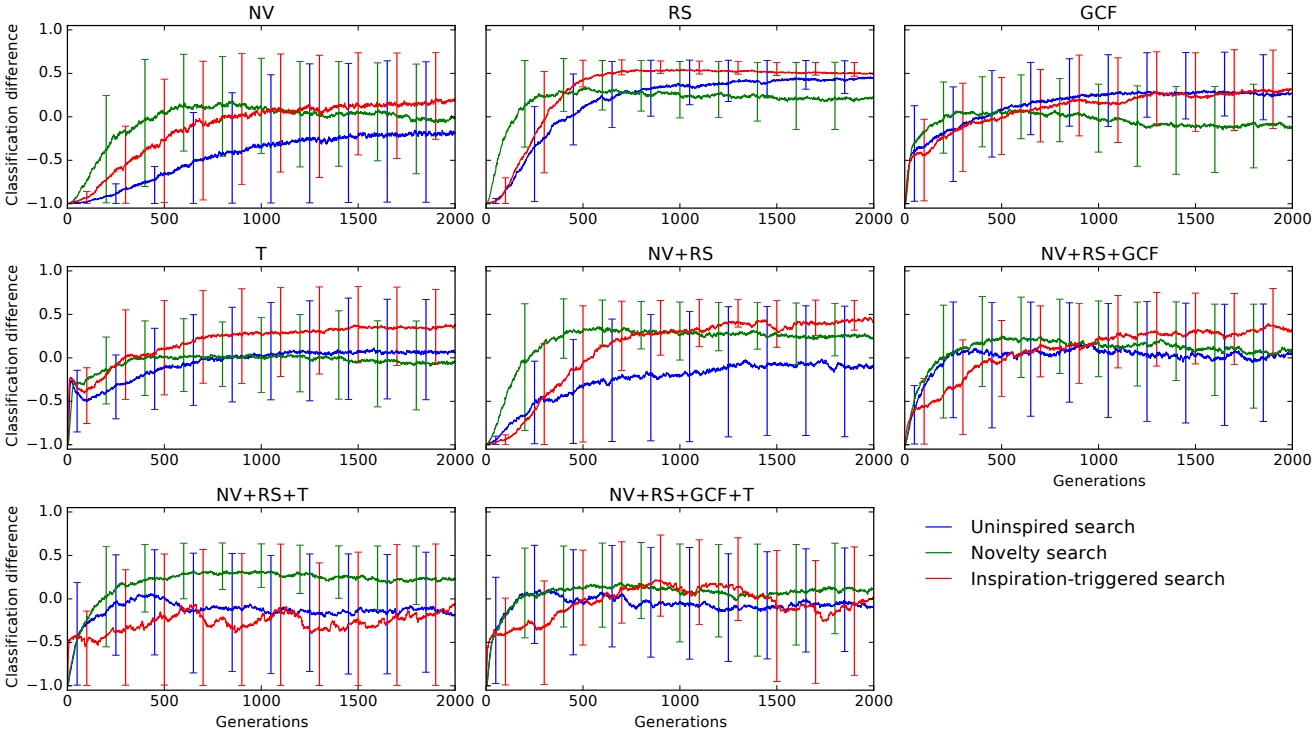



**Figure 7: Results according to complexity classifier (2000 generations in 30 trials). The bottom and the top of error bars are the 25$^{\text{th}}$ and the 75$^{\text{th}}$ percentiles.**

ages[2]). Table 3 contains the statistical comparisons. ITS is significantly better than uninspired search in all but one tested settings and ITS is significantly better than novelty search in 6 tested settings while novelty search is better for 2 settings (see Table 3). Similarly to the complexity test by optimization, the complexity of images from uninspired search is strongly influenced by the particular feature function. When a set of features for uninspired search contains GCF or T, the resulting images are more likely to be classified as complex. This is probably due to the data set we have chosen for the classifier which contains many sharp images. Images created by the features GCF and T are mainly sharp, hence they are classified as complex. However, they are typically rather simple according to our own subjective, visual evaluation.

# 5. DISCUSSION AND CONCLUSION

We have reported a novel algorithm called inspiration-triggered search that is allowed to change its objective function on the fly 'inspired' by features found within the evolved objects. Within the image domain, we have shown that our approach is able to compete with the state of the art.

For the evaluation of our method we had to approach the hard problem of quantifying complexity. We have reported two simple methods to compare the complexity of generated images in a post-evaluation. Both methods have limitations. First, evolving images that can, in turn, not be reproduced by trying to directly evolve them (i.e., using an objective function based on pixel distance to that target image) turned out to be easy to achieve and hence ineffective. Furthermore, this problem also raises questions about the efficiency of evolving images using CPPN and NEAT. Second, classifying images based on a trained ANN depends crucially on the training data. That data needs to be labeled by a human being who introduces a subjective view. Despite these weaknesses both methods allowed for an elementary comparison of the investigated approaches.

Existing approaches besides standard evolutionary algorithms can be categorized as interactive evolution using user input directly (e.g., Picbreeder [18]), purely novelty-driven approaches (e.g., novelty search [8]), and exploration-driven approaches that systematically explore a predefined feature space (e.g., Innovation Engine [13]). ITS usefully complements these existing approaches by trying to explore the feature space on the fly and triggered by features that already exist in the population possibly only as parts of individuals.

In future research, we plan to investigate ITS in different domains, to test different conditions that trigger the diversification and inspiration phases, to investigate more complex operators to modify the objective function, and to study a well-defined multi-objective optimization approach.

## Acknowledgment

Supported by EU-H2020 project 'flora robotica', no. 640959.

# 6. REFERENCES

[1] J. Avnet. Computation, dynamics and the phase-transition, 2000.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] P. Chervenski and S. Ryan. MultiNEAT.

[4] E. den Heijer. Evolving art using measures for symmetry, compositional balance and liveliness. In *Proc. of the 4th Int. Conf. on Comp. Int.*, pages 52–61, 2012.

[5] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural computation*, 22(7):1860–1898, 2010.

[6] J. Gomes, P. Mariano, and A. L. Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proc. Genetic and Evolutionary Comp. Conf. (GECCO'15)*, pages 943–950. ACM, 2015.

[7] G. Greenfield. On the origins of the term computational aesthetics. In *Proc. of the 1st Eurographics Conf. on Comp. Aesthetics in Graphics, Vis. and Imaging*, pages 9–12. Eurographics Association, 2005.

[8] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[9] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[10] P. Machado and A. Cardoso. Computing aesthetics. In *Advances in Artificial Intelligence*, pages 219–228. Springer, 1998.

[11] K. Matkovic, L. Neumann, A. Neumann, T. Psik, and W. Purgathofer. Global contrast factor-a new approach to image contrast. *Comp. Aesthetics*, 2005:159–168, 2005.

[12] L. Neumann, M. Sbert, B. Gooch, W. Purgathofer, et al. Defining computational aesthetics. *Comp. Aesthetics in Graphics, Vis. and Imaging*, pages 13–18, 2005.

[13] A. M. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proc. Conf. on Genetic and evolutionary Computation*, GECCO'15, pages 959–966, New York, NY, USA, 2015. ACM.

[14] S. Pertuz, D. Puig, and M. A. Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.

[15] J. J. Romero and P. Machado. *The art of artificial evolution*. Springer, 2007.

[16] A. Santos, C. Ortiz de Solorzano, J. J. Vaquero, J. Pena, N. Malpica, and F. Del Pozo. Evaluation of autofocus functions in molecular cytogenetic analysis. *Journal of microscopy*, 188(3):264–272, 1997.

[17] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

[18] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 1759–1768. ACM, 2008.

[19] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[20] K. O. Stanley and J. Lehman. The art of breeding art. In *Why Greatness Cannot Be Planned*, pages 21–28. Springer, 2015.

[21] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[22] Y. Sun, S. Duthaler, and B. J. Nelson. Autofocusing in computer microscopy: selecting the optimal focus algorithm. *Microscopy research and technique*, 65(3):139–149, 2004.

[23] B. G. Woolley and K. O. Stanley. On the deleterious effects of a priori objectives on evolution and representation. In *Proc. 13th Conf. on Genetic and evolutionary computation*, pages 957–964. ACM, 2011.

[2] http://milanrybar.cz/inspiration-triggered-search/