UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

Surface Reconstruction from 3D Point Data with a Genetic Programming/Evolution Strategy hybrid

Robert E. Keller, Wolfgang Banzhaf, Jörn Mehnen,
and Klaus Weinert

No. CI-44/98

Technical Report ISSN 1433-3325 September 1998

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

Abstract

Surface reconstruction is a hard key problem in the industrial domain of computer-aided design (CAD) applications. A physical object, like a workpiece, must be represented in some standard CAD object description format such that its representation can be efficiently used in a CAD process like redesign. To that end, a digitizing process represents the object surface as a weakly-structured discrete and digitized set of 3D points. Surface reconstruction attempts to transform this representation into an efficient CAD representation. Certain classic approaches produce inefficient reconstructions of surface areas that do not correspond to construction logic. Here, a new reconstruction principle in form of a computational-intelligence-based software system is presented that yields logical and efficient representations.

Keywords: digitized point data, computer-aided design (CAD), surface reconstruction, constructive solid geometry (CSG), genetic programming (GP), evolution strategy (ES), computational intelligence (CI), multi-criteria optimization, structure evolution, incremental optimization, pattern recognition, interactive evolution

1 Introduction

Genetic programming (**GP**) (Banzhaf *et al.* 1998) is an evolutionary search process that generates structures of arbitrary shape and size. The most prominent special case of such a structure is the representation of an algorithm, for instance, as a computer program in a common language like LISP, C or the machine language of a certain processor. The present contribution, however, focuses on GP as an apt tool for the evolution of representations of three-dimensional objects.

The aim of advanced **surface reconstruction (SR)** is the transformation of a physical object – like a hand-modeled prototype of a machine part that shall be produced – into a data representation, like a computer-aided design (**CAD**) 3D representation, that meets the high technological requirements of a construction engineer. As an essential part of the associated production process, the engineer modifies this CAD representation with a CAD system so that the resulting representation can be used to produce a corresponding physical object, like a gear box.

Such a system is typically used in mechanical engineering. Here, a physical object – like a prototype of a forging die – often does not have a CAD representation. Especially, it often lacks an exact geometrical CAD representation as a CAD object, either because a corresponding CAD data base does not exist, or the physical object has been changed manually in the course of the production process.

In order to obtain a CAD object, optical or tactile **digitizing process**es can be used. For instance, a tactile sensor may systematically scan the physical object's surface. For a practically relevant physical object, a digitizing process generates several megabytes of **weakly-structured** discrete 3D point data at least. "Weakly-structured" means the 3D point data set does not allow for a trivial recognition of the represented physical object by an automated process.

Modern rapid-prototyping production processes, like high-speed cutting or stereotype lithography, accept surface- or volume-oriented CAD objects and manufacture the corresponding physical object. A surface-oriented CAD object is constructed by combining CAD surfaces – like a saddle surface –, while a volume-oriented object is constructed by combining CAD volumes, like a sphere.

Accordingly, there are surface- or volume-oriented CAD systems and hybrid systems used by a construction engineer for operating on such CAD objects. Many CAD objects mainly consist of primitive CAD objects like spheres, cylinders, cuboids or tori. Thus, a CAD system provides corresponding object libraries and supports the manipulation of such objects.

A prominent class of *volume*-oriented CAD systems employs **constructive solid geometry (CSG)**. The CSG principle is to construct complex CAD objects from primitive objects. The resulting CAD objects (**CSG objects**) represent physical – that is, solid – objects. Thus, CSG is especially apt for the construction of structured surfaces.

Unstructured – especially, curved – surfaces, like certain parts of car bodies, are typically represented by **triangulations**, that is a surface approximation by plane triangles. Another well-known representation uses **non-uniform rational B-splines (NURBS)** which are especially apt for the construction of curved surfaces by smoothly joining curved surface parts.

Surface reconstruction – that is the automatic construction of a CAD object from data, like a 3D point data set – is a hard and industrially relevant problem. With respect to a reconstruction from a given non-empty discrete 3D point data set – which is the task being considered in this contribution –, the problem core is that, in $\mathbb{R}^3$, the set represents infinitely many different *geometrical* surfaces: those and only those surfaces that have the set in common. However, the data set represents only one *physical* surface, which is the surface of that physical object from which a digitizing process generated the data set.

Thus, the surface-reconstruction system must reconstruct a CAD object that approximates the physical object. This corresponds to the task of recognizing a physical object in a 3D point set, which is a special case of *pattern recognition.* The system must perform this task such that a construction engineer can start working with the resulting CAD object without being forced to introduce an expensive manual modification to the representation.

Pattern recognition problems are classically covered by artificial intelligence and machine learning, especially computational intelligence methods. Here, a new and evolutionary approach using a genetic programming (GP)/evolution strategy (ES) hybrid (Banzhaf *et al.* 1998) is used to reconstruct a CSG object from a non-empty discrete 3D point data set. The approach is represented as the software system **SurREAl (Surface Reconstruction by Evolutionary Algorithms)**.

2 Classic context

2.1 Digitizing and preprocessing

A digitizing process generates a point-data set – which can be imagined as a **point cloud** – from a physical-object surface for use by a CAD system. Usually, such a cloud has too many points and is topologically structured too weakly for efficient handling by a CAD system.

Thus, **preprocessing** – for instance, data reduction by chordal deviation (Friedhoff 1996), depth-pass filtering (Müller and Mencl 1997), or mesh optimization (Hoppe *et al.* 1993) – is needed. The preprocessed point cloud represents an **approximated physical-object surface**.

Preprocessing generates topological information relevant to surface reconstruction. This information has to be calculated only once due to the static nature of the point cloud. Two prominent instances of such information, which will be explained below, are a normal vector and an approximation of the local Gaussian curvature.

Surface reconstruction consists of two key tasks: to *obtain topological information* from the preprocessed point cloud by analytic methods, and then to *construct an* **approximating geometrical surface** – that is, a surface that approximates the physical surface – from this point cloud by use of the topological information. Typically, a surface-reconstruction method assumes digitizing and preprocessing are topology-preserving, that is the approximated physical surface has a topology close to that of the physical surface.

Note that, subsequently, the approximated physical surface will be identified with the physical surface, which is legitimate since the former is the most direct computer-accessible representation of the latter.

2.2 Gridded representation and topological information

A classic surface-reconstruction method uses the idea of constructing an approximating surface with a usually very large number of plane pieces. **Triangulation**, for instance, yields an approximating surface consisting of triangles as plane pieces. In this context, an intuitive idea of smoothness of the approximating surface is used: the surface is considered "smooth" in a certain area if the angles between the area's plane pieces are not "too wide". The formalized search for a smooth triangulation of a point cloud is hard. Different methods yielding smooth triangulations can be found in (Müller and Mencl 1997, Weinert *et al.* 1997*b*, Schumaker 1993, Weinert *et al.* 1997*a*).

There are several topological properties of a surface that may be used by an advanced classic surface-reconstruction method. In order to determine the peculiar properties of a given surface, a **gridded triangulation** may be calculated from a smooth triangulation. A triangulation is called "gridded" if and only if its points – these are the points of the triangles – lie orthogonally over the points of a uniform 2D **grid**.

The grid represents a plane area on which the physical object lies during the digitizing process, while a grid point represents a point that is aimed at by a sensor, like a pin of a tactile digitizing hardware. The sensor aims at this grid point along a vector that is orthogonal to the plane area. For a gridded triangulation, the indicated topological properties can be easier determined than for a non-gridded triangulation. Two examples of such properties are a "normal vector" and "Gaussian curvature".

A **normal vector** in a surface point is orthogonal to the surface. Since the approximating geometrical surface differs from the approximated physical surface, the normal vector in a point of the approximating geometrical surface must be approximated itself. This approximation can be done, for instance, by help of differential geometrical considerations (Suk and Bhandarkar 1992) or simulation (Weinert *et al.* 1998).

The **Gaussian curvature** K, a certain topological description of the curvature of a surface, can be calculated to classify a surface. The cone, cylinder, plane, pseudosphere, and sphere, for instance, all have a *constant* Gaussian curvature (Gray 1993).

Thus, considering an $n \times m$ grid, the result of the first key task, i.e., obtaining topological information from the preprocessed point cloud, can be represented as an $n \times m$ matrix of vectors $s_{n.m} \in \mathbb{R}^{3+3+1}$. $s_{n.m}$ consists of three subvectors: the coordinate vector $c \in \mathbb{R}^3$ of $p_{n.m}$ – that is a physical surface point over the grid point $n.m$; the normal vector $n \in \mathbb{R}^3$ in $p_{n.m}$; the physical-object surface's Gaussian curvature $g \in \mathbb{R}$ in $p_{n.m}$.

The topological information is used by a typical classic surface-reconstruction method. It is also used by the evolutionary approach that will be presented next.

3 SurREAl– a Genetic programming/Evolution strategy hybrid

An **evolution strategy** is an evolutionary algorithm introduced by Rechenberg and Schwefel in the 60s (Rechenberg 1994, Schwefel 1995). Its original form and modern variants are powerful tools for solving hard parameter optimization problems as they are ubiquitous in engineering domains.

3.1 The approach

A CAD system supports one or more internationally standardized formats for CAD-data exchange, like STEP, VDA-FS, or IGES (Reed *et al.* 1991). A geometrical surface can be represented as a **CAD expression** that obeys one of these formats, while the expression represents an algorithm that constructs the surface. The basic idea of SurREAl is to have genetic programming evolve a CAD expression for a given preprocessed point cloud such that the expression represents the physical surface that underlies the cloud.

A significant advantage over classic approaches is that *curved* physical-surface parts can be approximated by curved geometrical surface parts instead of plane geometrical surface pieces like triangles. This is an important feature since a curved physical part, like a wind-channel-designed car bonnet, is frequently used in an industrial production process. Approximating such a part by plane pieces is very inefficient in terms of computing resources. Also, approximating curved physical parts by curved geometrical parts may yield a better approximation. Finally, the resulting approximation looks more natural than a plane-piece approximation that often has a "jagged" appearance.

3.2 Overview

When designing an evolutionary algorithm, the following major issues must be dealt with:

- algorithmic structure (generational vs. steady-state model, etc.)
- a problem-specific *genetic representation* of an individual
- a *quality measure* applicable to each evolvable individual
- a *search operator* set including creation operator(s) and genetic operators, like reproduction and variation operators like mutation and recombination
- a *selection of individuals as operands for the genetic operators*
- a *selection of individuals as members of the next generation*
- closure and completeness of the search process
- *parameters* like maximal run time, population size, operator application probabilities, etc.

These issues will be discussed subsequently with respect to SurREAl with the exception of the trivial reproduction operator.

3.3 Algorithmic structure

The algorithmic structure is a GP/ES hybrid. After creation of a population with fixed population size $\mu \in \mathbb{N}$, a generational-cycle model takes over. A parental generation is transformed into an offspring generation of λ individuals by application of genetic operators. After the generation of an offspring generation has been completed, a selection method selects μ individuals from the offspring generation into the next parental generation. Then, the cycle restarts.

Each genetic operator is tagged by a particular **operator-selection probability**, so that, for instance, mutation will be selected with 0.3 probability as next genetic operator to be applied to a genotype. The sum of all operator-selection probabilities must be 1.

3.4 Genetic representation

3.4.1 Constructive solid geometry

The CAD construction of an object is realized either via line- or surface- or volume-oriented (3D) construction methods. A 3D-CAD system has some advantages in comparison to a surface-oriented CAD system. For instance, a 3D-CAD system gives a realistic visual representation of a physical object, which is a major reason why an increasing acceptance of 3D-CAD systems can be expected (Spur 1997).

Constructive solid geometry (**CSG**) is a very prominent object-construction method used by several 3D-CAD systems (Spur 1997). A **CSG object** is either a **CSG primitive** – like a cube –, or it is a **CSG complex** which is represented as a **CSG sequence** of **construction operators** and CSG objects. Addition, subtraction, and intersection of CSG objects are prominent construction operators. A CSG primitive is characterized by its parameters, the essential ones being size, position, and orientation.

A CSG sequence is a word from a context-free language. For instance, a schematic CSG sequence looks like the infix expression "cube ∪ sphere" which represents a CSG complex obtained by placing a cube into a sphere such that the cube's and sphere's centers are identical. Depending on the sizes of both primitives, the constructed CSG complex may look like a sphere with the cube corners showing on the sphere surface. Actually, a concrete CSG sequence representation as it is used by a CAD system contains parameters like the sphere radius and the cube edge length.

A CSG sequence can be represented as a hierarchical structure, that is a **CSG tree**, which corresponds to a – linear – infix expression like the one shown above if the tree is being in-order traversed. For instance, the above infix expression can be represented as a tree with a ∪-labeled root node that has one cube-labeled and one sphere-labeled child node.

Note that, for each CSG object, there is an infinite number of representing CSG trees, which means the CSG-object representation is not unique. For instance, a sphere can be represented as "sphere", "sphere ∪ sphere", "sphere ∪ sphere ∪ sphere", etc., with all spheres having the same radius.

3.4.2 Terminal and function set

The terminal set contains the CSG primitives "box", "sphere" and "quadric". A **quadric** is a generic geometric object that can be instantiated as, for instance, a cone or a cylinder. As a special case of a finite quadric, the terminal set contains a cylinder. Other stereometrical primitives like a torus, an obelisc, barrel-shaped bodies, and primitives with non-trivial topologies can be included into the terminal set, if desired. The function set contains the binary construction operators "union", "subtraction", and "intersection".

An example of three CSG complexes can be seen in figure 1.

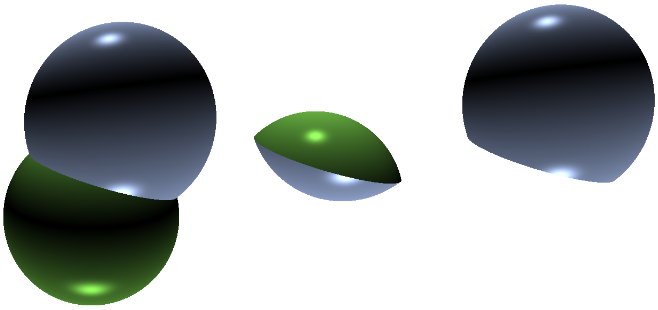

Figure 1: Union, intersection, and subtraction of two spheres.

3.4.3 Search space

Due to the genetic representation, the **search space** is the set of all CSG trees of a user-defined maximal depth. Note that this CSG-based genetic representation ensures that each CSG object can be represented as an individual. This ensures that almost any practically relevant physical object can be represented as an individual, since CSG has been designed for the representation of such physical objects.

Note also that an arbitrary point in search space – that is, a CSG tree –, can be generated by an apt sequence of variation operators. Thus, no potential solution is excluded from the search process access due to a variation operator bias. Finally, the genetic representation of an individual can be transformed – as it has been exemplified above – into a CSG sequence which can be directly processed by a corresponding CAD system.

3.5 Quality measure

Surface reconstruction is an instance of a *multi-criterion optimization* problem. In the context of this contribution, one obvious optimization criterion is the quality of the approximation of a physical object by a corresponding CSG object resulting from the reconstruction. Another criterion is the *parsimony* of the CSG tree that represents the approximating CSG object. Since the CSG-object representation is not unique, CSG trees of vastly different sizes represent the same CSG object. In order to save CAD system resources, the representing CSG tree should be small.

Further optimization criteria are related to topological properties of the CSG object, which will be explained subsequently. According to all of the optimization criteria mentioned above, *fitness criteria* have been introduced which will be discussed next.

3.5.1 Distance criterion DELTA

This criterion represents the idea that a well-approximating CSG-object surface should contain the points from the preprocessed point cloud. Subsequently, a **z-value** of a cloud point is the point's height over the corresponding grid point. If this value is greater or equal to a digitizing-hardware-dependent $\epsilon > 0$, this means the digitizing process detected a physical-object surface over that grid point. A **vanishing z-value** – $0 \leq z < \epsilon$ – means the digitizing process detected no physical-object surface over that point. The concept of a vanishing value must be introduced due to the unavoidable imprecision of the physical measuring a digitizing process must perform.

The criterion is realized via the sum σ over all grid points of the differences between a physical-object surface point's z-value and the corresponding CSG-object-surface point's z-value. The result has been normalized to the interval $]0,1]$ by use of

$$DELTA := (\sigma + 1)^{-1}.$$

Thus, a perfect CSG object has "one" as DELTA value, since, in this case, σ equals zero.

Note two special cases that come into existence because not each grid point is necessarily covered by a CSG-object. A grid point with no corresponding CSG-object surface point and a vanishing z-value means SurREAl models reality correctly in this grid point: no physical-object surface – no CSG-object surface. A grid point with a non-vanishing z-value with no corresponding CSG-object surface point means reality has not yet been grasped in this grid point: a physical-object surface – no CSG-object surface. Thus, the error in this grid point – represented as an addend of σ – equals the z-value.

The time complexity of the computation of DELTA is linear in the number of all $n \cdot m$ grid points.

3.5.2 Angle criterion ABN

This criterion represents the idea that the CSG-object surface should have the same spatial orientation as the physical-object surface. It is realized by taking into consideration the normal vectors of the CSG-object surface points and of the physical-object surface points over all grid points. The physical-object normal vectors have to be computed only once at the beginning of a SurREAl run since the physical-object representation – i.e., the preprocessed point cloud – is constant.

The criterion is realized as the normalized sum over all grid points of the absolute cosine value of the angle α_i between the CSG-object normal vector and the physical-object normal vector over the same grid point.

$$ABN = \frac{1}{n \cdot m} \sum_{i=1}^{n \cdot m} |cos\ \alpha_i|.$$

The cosine for each normal-vector pair yields "one" for parallel normal vectors and zero for orthogonal normal vectors, which implies that ABN is in $[0,1]$ and a perfect CSG object has ABN value "one".

The two above-mentioned special cases with respect to uncovered grid points are handled in the obvious analogous way. The time complexity of the computation of ABN is linear in the number of all $n \cdot m$ grid points.

3.5.3 Curvature-type criterion CTYPE

This criterion represents the idea that, in corresponding points, the CSG-object surface should have the same curvature type as the physical-object surface. The purpose of CTYPE is to guide the selection of an apt CSG-primitive for the approximation of a certain physical-object surface area such that primitive and area have identical curvature types. CTYPE is defined as the sum over all grid points of type matches – rated as "one" – and mismatches – rated as "zero" – divided by the number of grid points. Thus, CTYPE is in $[0, 1]$, and a perfect CSG object has CTYPE value "one", since each grid point represents a type match.

The two above-mentioned special cases with respect to uncovered grid points are handled in the obvious analogous way. The time complexity of the computation of CTYPE is linear in the number of all $n \cdot m$ grid points.

3.5.4 Primitive-number criterion PRIM

Evolution may lead to the generation of a CSG object that consists of a huge number of primitives ϕ. Such an object is inefficient in terms of computing resources, since each data representation of a primitive must be stored. Moreover, a CAD system must draw each primitive when drawing the CSG object, and redraw whenever the construction engineer translates – that is, moves along a vector – or rotates the object on-screen. The more realistic the visual rendering of the CSG object is – for instance, by application of hidden-line deletion and ray tracing – and the higher ϕ is, the more CPU intense the drawing gets. Even with powerful graphic hardware support – like hardwired rotational matrices – the engineer's work interrupts due to the redrawing process delay may become annoyingly long.

Thus, PRIM realizes a parsimony criterion. Note that the evolution of an empty tree is not allowed by SurREAl's definition. PRIM is defined as

$$PRIM := 1 - \frac{\phi}{c},$$

where $\phi \leq c \leq n \cdot m, c \in \mathbb{N}$. c is a constant value that defines the maximally legal number of leaves – that is, CSG primitives – of a CSG tree. Thus, PRIM is in $[0, 1 - 1/c]$, and a perfect CSG object has $1 - 1/c$ as PRIM value. c should be set as high as the maximum number of leaves expected necessary to build an acceptable CSG object.

The *quality measure* is defined as a weighted sum of the single criteria, and an instance is shown in the results section. Note that the quality measure is defined for an arbitrary individual.

3.6 Variation

3.6.1 Mutation

In **traditional genetic programming** – that is the variant introduced by Koza (Koza 1992) – mutation is a secondary operator. In SurREAl, however, mutation plays an important role, which reflects the major role of mutation in organic evolution. A single *mutation* has one of the following types:

- **primitive** It randomly modifies a **CSG-primitive parameter** – like position, size, and orientation – with normal distribution for small modifications. Its objective is to tune the evolved geometric surface with many smaller and few bigger changes. This reflects the principle of variation to be observed in organic evolution.

- **construction** It randomly replaces a construction operator – like intersection – by another construction operator. This mutation type usually introduces a major phenotypic change – see fig. 1 as a very simple example. Its objective is to introduce – by repeated application to different individuals in the course of evolution – several *topologically qualitatively* different phenotypes. This, hopefully, generates a phenotype that captures the characteristics of the physical object to be approximated, so that the corresponding genotype may then be tuned.

- **replacing** It randomly replaces a CSG primitive by another primitive. Its objective is to tune the evolved geometric surface by exchanging subareas. The parameter settings of the introduced primitives are defined by standard settings.

- **insertion** It randomly replaces a CSG primitive by a non-trivial random-generated CSG tree. Its objective is to introduce a significant change of the phenotypic shape. Especially, this operator is imperative for introducing CSG complexes into the population, since creation, as will be discussed later, only generates primitives.

- **deletion** It randomly deletes a non-trivial subtree that is not the entire individual. Its objective is identical the insertion objective and it counteracts the genotypic size increase introduced by insertion. This countermeasure is required due to limited computing resources.

It can be observed in typical runs that the intended objectives are being reached.

Mutation-type selection probability The probability of the mutation being an insertion or deletion is inversely proportional to the fitness of the mutant. Thus, a hi-fitness phenotypic shape is not likely to be destroyed by insertion or deletion. If neither insertion nor deletion have been selected, another mutation type is being equal-probability random-selected.

Mutation rate The **mutation rate** – that is, the number of mutations per time unit – is controlled by use of a mutation-rate adjustment function that yields exponentially decreasing rates over run time in order to have the search process "home in" on an acceptable local optimum. The run-time dependence is implemented by having the function take the current generation number as argument.

3.6.2 Recombination

Recombination is defined as a crossover of two parental CSG subtrees between two parents. The subtree root nodes are being equal-probability random-selected independent from their depths within the parental trees. Thus, the phenotypic difference between parents and offspring covers the full range from almost identical to very different in shape and size.

In the ideal case, a phenotype is as complex as necessary with respect to to approximation quality and as parsimonious as possible.

3.7 Creation

A trivial creation operator generates a user-defined number of individuals each of which being a randomly chosen CSG *primitive* with random parameter settings. This ensures – a reasonable population size assumed – the occurrence of all primitive types in the initial population. This **type diversity** is helpful for the synthesis of complex topologies as they are ubiquitous in practically relevant physical objects. Also, no *a-priori* knowledge about the physical object is introduced in order to demonstrate, by subsequent evolution, the learning ability of the system.

Note that **structural diversity**, especially diversity in sizes, does not have to be introduced by creation, since the – trivial – insertion operator described above quickly introduces this diversity type into the initial population. Thus, the implementation and application of one or more – relatively sophisticated – tree creation operators like those described in (Koza 1992) are unnecessary.

Note that the search space is closed under the search operators. Thus, the search process cannot leave the space.

3.8 Selection for genetic operands

With respect to recombination, two different parents are equal-probability random-selected – fitness plays no role here. Thus, with population size P, each individual has the same selection probability $1/P$. This purely random-based selection mechanism would result in a Monte-Carlo-like search process if it was the only such mechanism present in the system.

However, with respect to mutation, SurREAl offers four fitness-based selection strategies to the user who chooses one that will be used during a run:

- elitist selection
- ranking selection under an exponential distribution
- 2-tournament selection
- fitness-proportional selection

3.9 Selection for the next parental generation

With respect to selection, SurREAl follows the evolution strategy. The classic evolution strategy introduces a selection pressure on the genotypes via using the **plus** and the **comma selection**. These selection methods *deterministically* select the μ best individuals for the next generation. Plus-selection selects from $(\mu + \lambda)$ parental *and* offspring individuals, and comma selection selects only from $\lambda \geq \mu$ offspring individuals. Thus, in order for the comma selection to work, the genetic operators must generate $\lambda \geq \mu$ offspring from μ parents.

SurREAl offers the use of either the plus-selection or the comma-selection in a run. Let us call a SurREAl run a $(\mu + \lambda)$ run when it employs a plus-selection.

3.9.1 Selection pressure

An evolutionary algorithm using these selection methods allows for a simple and effective tuning of the *selection pressure* by adjusting the μ/λ ratio: the larger the ratio, the higher the selection pressure. A high selection pressure yields fast but unsafe search process convergence to an acceptable genotype, while a small pressure leads to the opposite: a Monte-Carlo-like slow search process with safe convergence, enough – that is, very much – run time given. A theory yielding a good choice for the selection pressure is currently unknown. A good rule of thumb is to choose a ratio of 5/7 with $\lambda >> 50$ for difficult problems.

3.9.2 Biases

Note that, simultaneously, there is beneficial and detrimental potential in both selection methods. Plus-selection may result in a very long existence of a good but still not acceptable individual. This may lead to a critical and fast collapse of genotypic diversity via iterated reproduction of this individual. However, it may also lead to an increase of hi-fitness genetic information which finally may result in the evolution of an acceptable genotype.

Comma-selection may result in the situation that the best genotype of the next parental generation is worse than that of the previous one (this event can be somewhat counteracted by using a large population. However, the risk of a population takeover by a good but not acceptable individual is lower than with plus-selection.

4 Results

4.1 Problem

Subsequently, SurREAl will be applied to the surface reconstruction of a simple but practically relevant physical object: a dowel. A dowel is a standardized (DIN 7) and ubiquitous part in mechanical engineering which is used to reduce the shearing load of screws that connect two parts. Fig. 2 shows a manually constructed CSG object representing a dowel. The CSG object can be constructed from a cylinder and a half-sphere at each cylinder end.

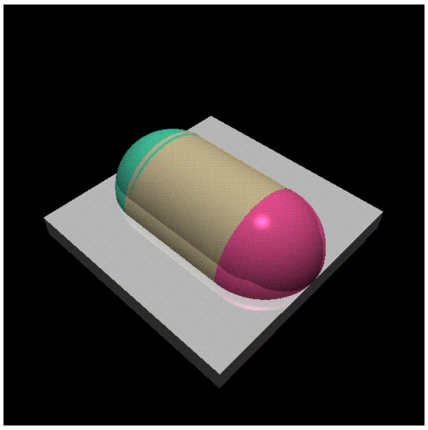

Figure 2: CAD reconstructed object

There are infinitely many CSG trees representing a dowel. A parsimonious one consists of 5 nodes: three leaves for the primitives “left half-sphere”, “right half-sphere”, “cylinder”, and two inner nodes for the construction operators “join”.

4.2 Parameters

4.2.1 Grid dimension

The required grid dimension, that is resolution, depends on the structural complexity of the physical object to be reconstructed. If the resolution is too low, a critical physical surface area may not be digitized. If the resolution is too high, redundant data is being generated during the digitizing process. This redundancy is detrimental, since it results in an unnecessary use of the limited computing resources. For the problem of the dowel reconstruction, the a grid dimension of 20×20, which appears to be beneficial to the search process, has been found experimentally.

4.2.2 CSG primitive number

The structural complexity of the physical object influences the maximum number of leaves a CSG tree should have in order to allow for a successful evolution. A number c – as introduced for the PRIM definition – of 220 is used for the problem.

4.2.3 Quality measure

The fitness function f is defined as

$$f = 1.5 \cdot CTYPE + 0.5 \cdot ABN + 0.5 \cdot DELTA + 0.5 \cdot PRIM.$$

Thus, an f value is in $]0, 3 - 0.5/c)]$, and a perfect CSG object scores the upper-interval-limit value.

4.2.4 Selection

Experiments with $(50 + 50)$, $(100 + 300)$, and $(500 + 800)$ runs all resulted in elitist strategies yielding best solutions, shortly followed by tournament selection strategies. Fitness-proportional selection and ranking selection schemes did not perform well.

4.2.5 Variation

Each selected individual is either mutated or recombined with a 0.5 probability. This effectively has mutation and recombination operate “in parallel” on the population during the entire run.

Mutation translates or resizes or rotates a CSG primitive by a normally distributed random value. A CSG primitive is changed with 0.1 probability. The inner nodes of a CSG tree are changed with 0.01 probability. These relatively low values are motivated by the phenomenon to be observed in nature that macromutations happen rarely.

The probability of an insertion or deletion is indirectly proportional to the fitness value of the individual. Thus, a well shaped individual is not likely to be changed significantly which would risk to destroy the shape, while a badly shaped individual has a chance to be transformed into a better shape.

4.3 Discussion

4.3.1 Incremental optimization

Note that, in the fitness function, CTYPE clearly has the largest weight, which is due to the experience that an adequate amplification of the curvature-type weight usually yields better results. This high weight forces an early optimization of the CSG-object shape and size, while orientation – evaluated by ABN – and position – evaluated by DELTA – are targets for final tuning.

Accordingly, when watching an animated picture sequence assembled from the best-in-generation phenotypes from the first to the last generation of a typical run, one often sees a corresponding structural evolution. This change is reflected by the genotypic size progression, which can be seen in fig. 3.

One can interpret the sequence of the best-in-generation phenotypes from the first to the last generation of a run as snapshots taken of the structural and spatial evolution of a **meta-individual**. Initially, this individual “morphs” into the topology of the physical object. During this period, the genotypic size increases rapidly, while the surface of the physical object is being approximated roughly by a set of primitives that have coarse curvature-type relations with physical-surface areas. In this phase, the CTYPE criterion implies topology optimization of the CSG object.

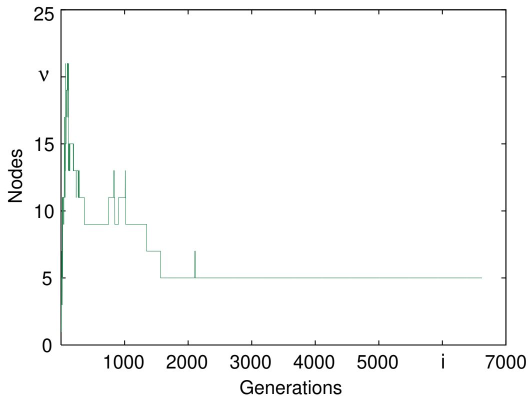

Figure 3: Progression of the genotypic size.

Then, the meta-individual rotates into the orientation the physical object had when it was scanned. Due to the PRIM criterion, the number of primitives is being reduced rapidly while the structure of the CSG object is being refined. In this phase, the ABN criterion implies orientation optimization of the CSG object. Finally, after the structure of the physical object has been recognized, the CSG object floats into the position the physical object had relative to the grid. The distance criterion DELTA implies position optimization.

A typical structure evolution of a meta-individual, evolved during a $(50 + 50)$-run with elitist selection, can be seen in figure 4. Due to the static and sparse nature of this sequence, watching it is far less impressive and instructive as watching the corresponding animation.

Note that, initially, a meta-individual is chaotically structured and has low structural complexity. During evolution, structural order and complexity increase clearly.

This type of structure evolution is an instance of incremental optimization, which is a powerful approach to many multi-criteria optimization problems. The idea is to water down the difficulty of the entire task by sequentially solving subproblems. This is effected by a quality measure that is composed of differently weighted elements. We return to this issue below in the context of fitness progression.

4.3.2 Fitness progression

The following discussion is concerned with an experiment that consists of 10 $(50 + 50)$-runs with elitist selection, initialized with different randomizer seeds. Each run lasted for, at least, $3,000$ generations and found a perfect CSG object. Fig. 5 shows the result of a typical run of this set that was terminated after $6,600$ generations.

Visually, no differences between the physical object and the CSG object can be detected. Especially, SurREAl has recognized the construction logic of a dowel and implemented it in the evolved CSG object: a cylinder, closed by half-spheres.

The visual impression is endorsed by the progressions of the single criterion values of the best-individual-in-generation fitness: 0.979363 (DELTA), 0.980912 (ABN), and 0.982 (CTYPE). The progressions are displayed in fig. 6.

Initially, rapidly increasing values of CTYPE and ABN reflect the structure evolution of the meta-individual during an exploring phase of the search process. Then, after the difficult topology and orientation optimization has been more or less completed, the easier position optimization is being dealt with, mirrored by the late start of the DELTA value increase.

4.3.3 Population size and convergence

Typical progressions of the best-individual-in-generation fitness of a $(100 + 300)$- and a $(500 + 800)$-run using elitist selection are displayed in fig. 7 with a logarithmic generations scale.

The progressions endorse the rule of thumb that larger populations strongly enhance convergence speed: both runs evolve the same genotypic optimum, but the $(500 + 800)$-run uses only 35% run time of the time consumed by the $(100 + 300)$-run.