Applied Soft Computing Journal 89 (2020) 106130

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Multi-objective Cartesian Genetic Programming optimization of morphological filters in navigation systems for Visually Impaired People

Antonio Miguel Batista Dourado [a,b,*], Emerson Carlos Pedrino [b]

[a] Federal Institute of São Paulo, Av. Zélia de Lima Rosa, 100, Boituva/SP, Brazil
[b] Federal University of São Carlos, Rod. Washington Luís, Km 235, P.O. 676, São Carlos/SP, Brazil

ARTICLE INFO

Article history:
Received 8 June 2019
Received in revised form 20 December 2019
Accepted 24 January 2020
Available online 29 January 2020

Keywords:
Genetic programming
Cartesian Genetic Programming
Multi-objective optimization
NSGA-II
Mathematical morphology

ABSTRACT

Navigation systems for Visually Impaired People (VIP) have improved in the last decade, incorporating many features to ensure navigation safety. Such systems often use grayscale depth images to segment obstacles and paths according to distances. However, this approach has the common problem of unknown distances. While this can be solved with good quality morphological filters, these might be too complex and power demanding. Considering navigation systems for VIP rely on limited energy sources that have to run multiple tasks, fixing unknown distance areas without major impacts on power consumption is a definite concern. Multi-objective optimization algorithms might improve filters' energy efficiency and output quality, which can be accomplished by means of different quality vs. complexity trade-offs. This study presents NSGA2CGP, a multi-objective optimization method that employs the NSGA-II algorithm on top of Cartesian Genetic Programming to optimize morphological filters for incomplete depth images used by navigation systems for VIP. Its goal is to minimize output errors and structuring element complexity, presenting several feasible alternatives combining different levels of filter quality and complexity—both of which affect power consumption. NSGA2CGP-optimized filters were deployed into an actual embedded platform, so as to experimentally measure power consumption and execution time. We also propose two new fitness functions based on existing approaches from literature. Results showed improvements in visual quality, performance, speed and power consumption, thanks to our proposed error function, proving NSGA2CGP as a solid method for developing and evolving efficient morphological filters.

1. Introduction

Visual impairment is a common condition throughout the world. Lack of vision, from moderate impairment to blindness, affects 253 million people, among which 36 million are fully blind, according to the World Health Organization [1]. Significant advances spanning several research areas are helping Visually Impaired People/Person (VIP) accomplish most daily tasks, such as using computers, smartphones, walk safely, etc. [2,3].

Vision-based navigation systems use cameras to gather images of the VIP's path and process them by means of computer vision operations [4]. The most common cameras used in VIP navigation systems are known as RGB-D cameras, which have built-in modules to generate environmental depth maps, by means of well-known methods such as structured light, time-of-flight and stereoscopy [5–7]. All of these methods, however, share the common problem of unknown pixel distance, which leads to areas of the depth map being set to zero distance, resulting in several navigation system issues, such as wrong segmentation. Depth maps can also be represented as grayscale images, with pixel colors defined according to distance, i.e., lighter grays for objects that are farther, darker grays for objects that are nearer (or vice-versa), and unknown distances simply displayed as black pixels.

Mathematical morphology and its operations can alleviate this problem by employing basic operations, such as dilation and erosion, with several shapes of structuring elements to handle a broad variety of images, containing several types of noises and/or imperfections [8,9]. Morphological filters may be generated through genetic programming algorithms, which use the main concepts of evolution theory – chromosomes, genes, alleles, mutations etc.– to evolve individuals of a population (in this case, filters) and achieve optimal results. Cartesian Genetic Programming(CGP) is a particular type of Graph-Based Genetic Programming where chromosomes are represented by a 2D array

* Corresponding author at: Federal University of São Carlos, Rod. Washington Luís, Km 235, P.O. 676, São Carlos/SP, Brazil.
E-mail addresses: dourado@ifsp.edu.br (A.M.B. Dourado), emerson@dc.ufscar.br (E.C. Pedrino).

https://doi.org/10.1016/j.asoc.2020.106130
1568-4946/

of integers (genes). Each integer may reference other genes or terminals (raw inputs), but it also may reference a function to process those inputs, resulting in one or more outputs [10].

Hardware setups for navigation systems must be portable, imposing an important constraint: power. Since the equipment cannot remain permanently connected to a power outlet, navigation systems need to rely on limited power supplies such as batteries [9,11,12]. Thus, an efficient use of available energy resources is required. However, the complexity of morphological operations might be a concern, due to the fact that personal navigation systems have to rely on small batteries for power, and complex sets of morphological operations might consume more energy and it adds an additional constraint to morphological filters: they must fix all the missing depth pixels whilst consuming as minimum of energy as possible, turning the creation of morphological filters into a problem with two conflicting objectives.

Multi-Objective Optimization (MOO) algorithms, when applied to morphological filters, should be able to evaluate and rank every filter according its quality (low error rate) and complexity (low complexity) within a feasible trade-off between them. The literature describes many evolutionary multi-objective optimization algorithms, such as the Nondominated Sorting Genetic Algorithm II (NSGA-II). NSGA-II employs the principles of nondominated fronts and crowding-distance [13] by using an elitist approach to create multiple nondominated fronts of parents and offsprings, and sort them by means of crowding-distance computation. Since NSGA-II has an evolutionary approach, it is possible to adapt it to work with a variety of genetic programming algorithms like CGP and evolve optimized sets of solutions, thereby being a good choice for our approach. Also, NSGA-II was chosen due its popularity among MOO algorithms as it also was reported to have superior performance with large populations, providing high diversity fronts [14–18].

The lack of depth estimation translates to black pixels in depth images and it has a direct impact on vision-based navigation systems, compromising (partially or totally) a depth image and the segmentation process, leading users to unsafe directions and/or presenting them wrong or misplaced obstacles. Even so, there is also the concern about how complex and power consuming is to fix depth images due the limited power sources that navigation systems rely on. Thereby, the main problem addressed by this work is the process to fix depth images containing incomplete depth data and overcome the challenge of doing so in a efficient manner in terms of complexity, a multi-objective problem. This paper proposes a method for multi-objective optimization of morphological filters, the NSGA2CGP Cartesian genetic programming MOO algorithm. The main goal of NSGA2CGP is to provide means to implement and generate sets of feasible and optimized morphological filters for depth images considering two objectives: minimum error and minimum complexity, aiming energy efficient filters. This method was devised and then experimentally applied to a specific use case, consisting of depth image scenes from university environments. In addition to NSGA2CGP we also proposed and applied two new fitness functions to evaluate images and calculate the error between them. The main contribution of this work is to provide a novel method to generate low-error x low-complexity optimized morphological filters to fix unknown distance areas of depth images in the context of vision-based navigation systems for VIP with multiple feasible solutions that fits different processing capabilities through varying trade-offs between error and complexity.

This paper is structured as follows: Section 2 presents literature approaches related to this work. The proposed method, NSGA2CGP, is presented in Section 3, followed by experiments in Section 4. Finally, results and conclusion are presented in Sections 5 and 6, respectively.

2. Related works

The literature presents a shortlist of approaches that combine CGP and NSGA-II. Most of them are based on theoretical digital circuit problems, jointly applying these techniques to improve circuit efficiency, reduce instruction delay, or minimize error [19,20]. They are improved for their own domain and were not conceived to work other domains such as images.

Kalkreuth and colleagues [21] proposed a modified version of the NSGA-II algorithm based on CGP to evolve small and efficient programs by minimizing both the age and size of each individual. However, their method was intended for image noise reduction, and the size objective was calculated simply as the gene count of each individual: this might not be ideal, considering that less operations do not necessarily mean less processing time, i.e., few heavy computational operations can be worse than many low-cost ones. Moreover, the condition of the resulting images was not made clear, since the authors only showed input images with two noise types and the desired output.

Kaufmann and colleagues [22] presented a modified form of NSGA-II and Cartesian Genetic Programming with a periodization approach that combines local and global search methods over a hybrid $\{\mu_{+}^{,}\lambda\}$ as evolutionary strategy named *hES*. It was applied to known digital circuit problems and results showed that it outperforms classical SPEA2 and NSGA-II algorithms on that domain. Even though this is a promising method with interesting results, authors stated that for the majority of generic benchmarks, their method was outperformed.

These few NSGA-II implementations with CGP found in literature are mostly based on digital circuits. Also, none of them presented a practical method to apply their results on real world scenario, such as to export the best individuals as deployable code or schematics. However, authors should further implement them in the future and modify the necessary parts to enable them to work in the domain of this paper, optimization of morphological filters.

The literature has even fewer studies on mathematical morphology filters evolved by CGP algorithms, all of them developed during the last decade by a single author's research group, [23–25], and focused on hardware architecture—describing the deployment of evolved filters on FPGAs. In any case, these previous efforts helped us identify the need for a multi-objective approach.

Many areas employ multi-objective optimization methods for several purposes. Xu and colleagues [26] used SPEA2 to optimize time consuming and load balance of edge computing operations on video surveillance server for a state-of-art application, Internet of Vehicles. Sajuna and Dinakaram [27] proposed a multi-objective algorithm to detect and extract text with the ideal size of bounding boxes plus low impact on processing time and the proposed method did perform well when compared to existing text detection methods. Srivas and colleagues [28] improved the classification of geographic information systems by combining an existing optimization algorithm (Dragon Fly) and a search algorithm (Cuckoo Search), using objective functions of both algorithms to optimize classification tasks.

As for image processing and computer vision tasks, Shi and colleagues [29] created a feasible MOO method to optimize image registration of mosaics, which refers to four classic computer vision tasks: feature extraction, feature description, feature matching and model estimation. Multi-objective approaches to optimize image segmentation were proposed based on different methods such as thresholding, bee colony algorithm, region growing, clustering and others [30–32]. All these works are very relevant and could be applied to vision-based navigation systems. However, a depth image containing unknown distance areas would not be properly segmented, since it is incomplete.

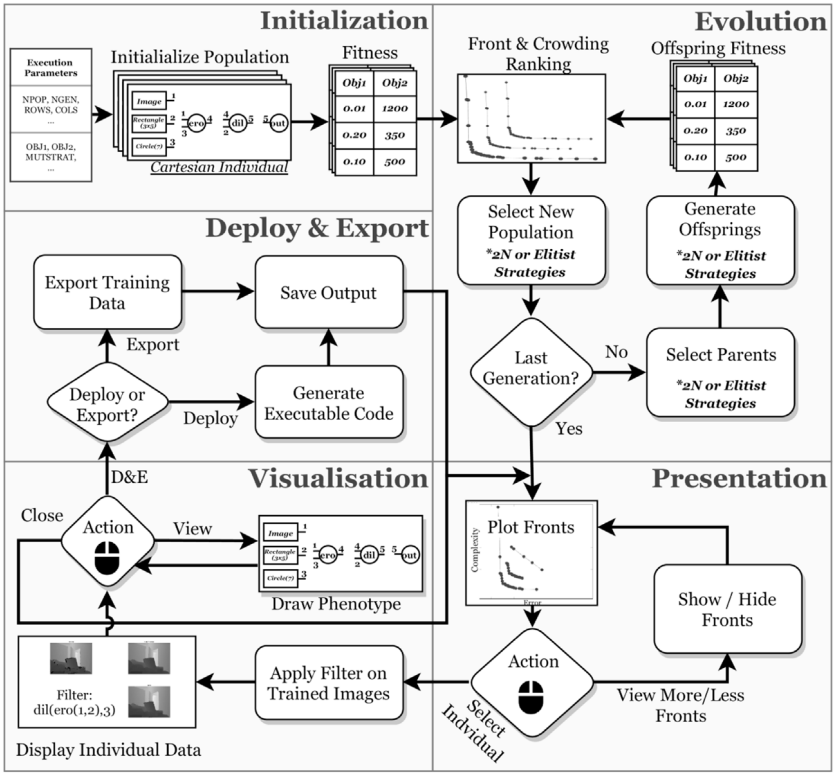


Fig. 1. Representation of the NSGA2CGP method and its five phases.

Multi-objective optimization algorithms have also been applied to optimize morphological operations. Dou and colleagues [33] used Particle Swarm Optimization to evolve anti-noise morphological filters, assessing quality by Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) objective functions. Their results were satisfactory but failed to address complex operations and their impacts over the performance of the system as well as power consumption. Koppen and Franke [34] used hypervolume operators to show the potential of multi-objective approaches combined to morphology, even when applied to color images. The MOO and MM combination was even employed to analyze the thermoelastic structures of porous materials [35,36].

There are several approaches to solving the unknown distance problem. Inpainting is a technique used for several image-processing functions, such as removing or replacing objects or restoring damaged areas [37,38]. Depth images can benefit from inpainting algorithms to fix unknown areas, generating good-quality final images [39,40]. However, this technique might not be ideal for VIP navigation systems, with some authors highlighting it as computationally and time-intensive, considering that navigation systems may have limited resources to deliver real-time feedback [41,42]. Median filters and bilateral filters can also used to improve depth maps, achieving good results, yet also at the cost of high processing time [43,44]. Some of those filters were tested using GPU parallelism to achieve faster processing times, but so far there are no tests in navigation systems with limited processing power, which differ greatly from desktop setups [45, 46].

Navigation systems deployed on embedded GPUs were recently reported, making use of parallelism via popular libraries such as NVIDIA's CUDA [47]. These studies also report power consumption and execution times on different embedded devices with distinct capabilities, demonstrating the significant performance gaps between them [9,48–50]. A less powerful device will naturally fall behind when executing the same filters. This work's goal is to enable the choice of morphological filters suitable to specific depth-image conditions, by generating and optimizing a set of feasible solutions depending on the image's quality and complexity. NSGA2CGP is an evolutionary method where a population of morphological filters is evolved via the combination of an optimization algorithm, NSGA-II, and a genetic programming algorithm, CGP.

As far as we were able to verify, the literature lacks a MOO method to optimize morphological filters in respect to error and complexity, in order to minimize both objectives and, in the specific context of navigation systems for visually impaired people, provide a wide range of solutions suitable to different target systems and their capabilities. Thus, this work advances previous studies on CGP-based evolved morphological filters by proposing NSGA2CGP, a new evolutionary multi-objective optimization method for morphological filters aimed at improving depth map images.

3. NSGA2CGP

NSGA2CGP is a modified version of the existing NSGA-II MOO algorithm. It uses Cartesian genetic programming – instead of a classic genetic algorithm – to optimize morphological filters for depth maps produced by VIP navigation systems. The method is intended to automatically generate and optimize filters based on training sets containing images from possible navigation areas, providing developers with the ability to choose the filter best-suited to the hardware specifications (battery capacity, processing power etc.)

NSGA2CGP was implemented in MATLAB using Windows 10 environment, but the method is independent of programming languages or operating systems. NSGA2CGP has five basic phases, which cover all the necessary tasks, from user input and parameter definitions to code generation and training data exportation. These are: Initialization, Evolution, Presentation, Visualization and Deploy/Export, as represented in Fig. 1.

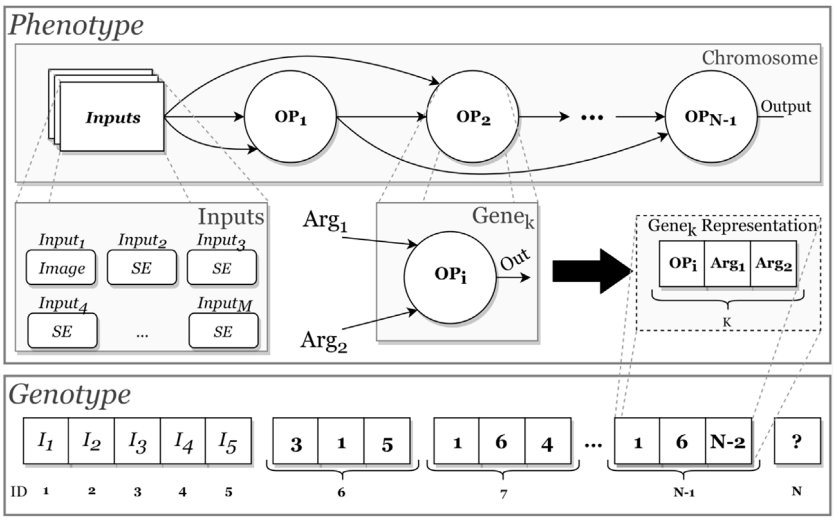


Fig. 2. Representation a CGP chromosome phenotype and genotype.

Table 1
NSGA2CGP basic parameters.

Parameter	Description	Parameter	Description
NGEN	(int) Total number of generations.	TRAINDATA	({Mat},{Mat}) Pair of arrays containing input and ground-truth images.
NPOP	(int) Population size.	STRELS	({SE1, …, SEn}) Array of structuring elements.
COLS	(int) Total columns of CGP chromosome.	ERROR	(string) Fitness function to evaluate chromosomes.
ROWS	(int) Total rows of CGP chromosome.	CODEWRPS	({@function}) Wrappers used to generate deployable code from individuals.
LBACK	(int) Gene levels-back.	USELOGICS	(logical) Allows logical functions such as AND/OR/NOT.
TMUT	(float) Mutation rate.	GENTYPE	(string) Defines how genes receive inputs.
MUTSTRAT	(string) Mutation strategy.	BPWEIGHT	(logical) Allows application of black pixel penalty.
INITPOP	(string) Use pre-trained population in a new train.	RESUTRAIN	(string) Resumes previously stopped training.
AUTOEXP	(logical) Automatically export training results.	SAVESTATE	(int) Save training state each *n* generation.

Execution parameters must be defined prior to running NSGA2CGP, since the evolutionary multi-objective approach should not demand user interaction during execution. NSGA2CGP employs several parameters to guide its execution, and each parameter carries default values in order to guarantee a basic application of the method. A list of NSGA2CGP parameters and basic types is presented in Table 1. Further details about each parameter are discussed later on.

Initialization

The initialization phase comprehends the task of preparing the population for evolution and optimization. NSGA2CGP receives user-defined execution parameters (shown in Table 1), parses them and initializes a parametric structure containing all of those settings. Also in this phase, each population individual is randomly generated, or previously trained population data can be imported by setting the *INITPOP* parameter to valid trained data. Interrupted training sessions can be resumed by setting the *RESUTRAIN* parameter to a valid training session state filename. Algorithm 1 describes the Initialization phase.

When a new population is created, a *P* number of individuals (chromosomes) is generated randomly according to the *NPOP* parameter. Chromosomes are arranged in a *COLS***ROWS*+1 1D array where each position corresponds to a gene, and the last position is a single output. Genes are represented by three numbers: one operation and two input arguments. Fig. 2 illustrates the representation of a chromosome as phenotype and genotype, which is an array of integers representing the phenotype.

There are two types of genes as defined by *GENTYPE*: *Morph-Pair*, which restricts genes to always receive the input image or output from other genes (first input) and the structuring elements

Algorithm 1: NSGA2CGP Initialization phase

```
Input: PARAMS
Result: {POP, FIT}
validateParams(PARAMS)
if INITPOP then
    POP ← loadTrainedPopulation(INITPOP)
else if RESUTRAIN then
    PARAMS ← loadExecutionState(RESUTRAIN)
    POP ← loadPopulationState(RESUTRAIN)
else
    POP ← generateRandomPop(PARAMS)
end
FIT ← calculateFitness(POP,PARAMS)
```

Table 2
'Free' gene type.

Input 1	Input 2	Dil & Ero	AND, OR etc.	NOT
IMG	SEL	Apply	Replicate SEL and apply	Apply to IMG
SEL	IMG	Apply	Replicate SEL and apply	Apply to IMG
IMG	IMG	Passthrough to 1 if odd gene, 2 otherwise	Apply	Apply to 1 if odd gene, 2 otherwise
SEL	SEL	Apply	Apply	Apply to 1 if odd gene, 2 otherwise

from *STRELS* (second input); and *Free*, which enables both gene inputs to receive either images (IMG) or structuring elements (SEL) in any order, applying gene function to them according to Table 2.

Evolution

The evolution phase starts after the initial population has been created and its individuals' fitness has been calculated, or after the previously interrupted training session has been loaded and its state variables have been set. This phase is repeated across *NGEN* generations, and the training state is saved each *SAVESTATE* number of generations.

The population and its fitness is submitted to a dominance matrix, resulting in nondominated fronts. The best theoretical solutions are placed in the first front, where each individual has the neighborhood's best-quality (or at least equal-equality) error/complexity relation. After front definition, the crowding-distance operation estimates the population density of all front individuals, for each front, assigning higher values to lesser-density individuals, so as to preserve diversity.

NSGA2CGP provides two mutation strategies that can be applied while the maximum number of generations has not been reached. The mutation strategy is defined by the *MUTSTRAT* parameter, set to either '2*N*' or '*ELITIST*'. Algorithm 2 describes the Evolution phase.

The 2*N* strategy is based on the classical NSGA-II mutation strategy, where *NPOP* best individuals (ranked by fronts and crowding-distance) are defined as the population and randomly selected to generate an offspring. In this strategy, every selected parent should have *TMUT%* of their genes mutated to a random value, respecting the constraints established by the *GENTYPE* parameter. All offsprings have their fitnesses calculated and are then appended to the existing population, resulting in a population with a 2*NPOP* size. Upon reaching the maximum number of generations, only *NPOP* best individuals are sent to the next phase.

The *ELITIST* strategy is a point-mutation operation which selects *TMUT%* best-ranked individuals and mutates *TMUT%* of their genes to produce offsprings. After fitness calculation, an offspring replaces its parent if it has better or equal fitness value, so as to preserve diversity.

For both mutation strategies, we propose an adapted *Single Active Mutation* technique, to be used during the selection of candidate genes for mutation [51]. This approach reduces wasted evaluations by only selecting active genes, making it less likely for inactive/unused genes to mutate. Instead of only a single active gene, as in the original technique, NSGA2CGP mutates *TMUT%* active genes, without disrespecting the active gene constraint.

Presentation

After it has been evolved and optimized during the Evolution phase, the population must be presented to the user. This task is handled by the Presentation phase, which provides a general overview of Pareto optimal solutions, divided into fronts. Algorithm 3 is an overview of the Presentation phase and the user actions that can be performed in it.

NSGA2CGP employs a classic graph form to display fronts. Each feasible-solutions set in the plotted graph is visually distinguishable from the previous and next set. Moreover, each individual solution leads users to the next phase, Visualization, which shows detailed information on how a certain solution was reached. Presentation allows the user to filter by number of displayed fronts, ranging from 1 to the total. Fig. 3 exemplifies how the Presentation phase delivers training results to users: two buttons at the top-right side of the graph are used to reduce or increase the number of fronts.

Algorithm 2: NSGA2CGP Evolution phase

```
   Input: {POP, FIT}
   Result: {POP, FIT, RANKEDPOP}
 1 GEN ← 0
 2 while GEN < NGEN do
 3     FRONTS ← organizeFronts(POP,FIT)
 4     RANKEDPOP ← crowdingDistance(POP,FIT,FRONTS)
 5     switch MUTSTRAT do
 6         case 2N
 7             POP ← POP[RANKEDPOP[1 → NPOP]]
 8             for i = 1 to NPOP do
 9                 PARENT ← selectRandomParent(POP)
10                 OFFSPRING ← mutateParent(PARENT)
11                 OFFSFIT ← calculateFitness(OFFSPRING)
12                 POP[NPOP + i] ← OFFSPRING
13                 FIT[NPOP + i] ← OFFSFIT
14             end
15         case ELITIST
16             for i = 1 to TMUT% do
17                 PARENT ← RANKEDPOP[i]
18                 OFFSPRING ← mutateParent(PARENT)
19                 OFFSFIT ← calculateFitness(OFFSPRING)
20                 if OSSFIT is better or equal fitness than
                   RANKEDPOP[i] then
21                     POP[RANKEDPOP[i]] ← OFFSPRING
22                     FIT[RANKEDPOP[i]] ← OFFSFIT
23                 end
24             end
25         end
26     end
27 end
28 if MUTSTRAT is 2N then ; /* Selects the best NPOP for
   next phase */
29
30     POP ← POP[RANKEDPOP[1 → NPOP]]
31     FIT ← FIT[RANKEDPOP[1 → NPOP]]
32     RANKEDPOP ← RANKEDPOP[1 → NPOP]
33 end
```

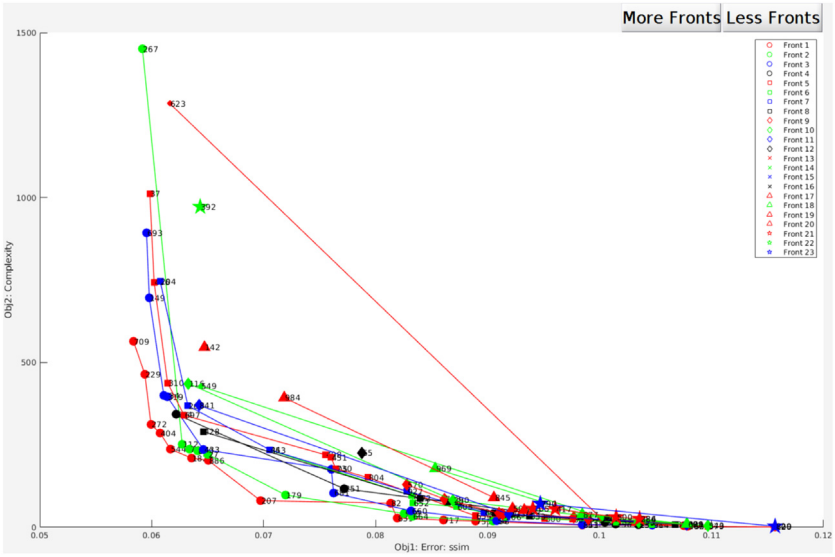


Fig. 3. Training results as displayed in the Presentation phase.

Algorithm 3: NSGA2CGP Presentation phase

```
   Input: {PARAMS, POP, FIT, RANKEDPOP}
   Result: {SAVED_TRAIN, FRONTS_PLOT}
1  MAXPLOT ← maxRank(RANKEDPOP)
2  for i = 1 to MAXPLOT do
3  |  plotFront(RANKEDPOP[FRONT == i])
4  end
5  while Presentation do
6  |  switch UserInput do
7  |  |  case Less Fronts AND Not at Minimum Front
8  |  |  |  deleteFront(MAXPLOT)
9  |  |  |  MAXPLOT ← MAXPLOT - 1
10 |  |  case More Fronts AND Not at Maximum Front
11 |  |  |  MAXPLOT ← MAXPLOT+1
12 |  |  |  plotFront(RANKEDPOP[FRONT == MAXPLOT])
13 |  |  case Visualize Individual
14 |  |  |  INDIVIDUAL ← userInput.parse()
15 |  |  |  visualization(INDIVIDUAL)
16 |  |  case Export Training
17 |  |  |  SAVED_TRAIN ← deployExport(Input)
18 |  |  end
19 |  end
20 end
```

Visualization

The Visualization phase is an extension of the Presentation phase. It allows the user to verify each solution individually, visualizing its results as objective values, output images, and chromosome structure. When an individual is under visualization, all training images are also shown, allowing for a side-by-side comparison with the individual's outputs, for an assessment of the achieved visual quality. The pseudocode in Algorithm 4 shows how the Visualization phase was implemented.

When visualizing a specific chromosome structure, users are not only able to understand each step of an individual's evolution, but also choose one specific gene and visualize its output, even if it is not an active gene. This allows structuring elements as well as other parameters to be fine-tuned, for future runs.

Deploy/Export

The last NSGA2CGP phase is Deploy/Export, described by Algorithm 5. This phase is responsible for generating and exporting a training session's deployable outputs, either in the form of programming language code or training data to be reused in future training.

Since NSGA2CGP is not attached to any single programming language, it is able to generate morphological filters in any language or library supporting image processing tasks, such as

Algorithm 4: NSGA2CGP Visualization phase

```
   Input: {INDIVIDUAL}
   Result: {CODE, PART_GENE, OUTPUT_IMGS}
1  plotCGPChromosome(INDIVIDUAL)
2  while Visualization do
3  |  switch UserInput do
4  |  |  case Visualize Gene Output
5  |  |  |  visualizeGene(UserInput.parse())
6  |  |  case Show Training Images
7  |  |  |  trainingImageOutput(INDIVIDUAL)
8  |  |  case Generate Deployable Code
9  |  |  |  CODE ← exportDeploy(INDIVIDUAL)
10 |  |  end
11 |  end
12 end
```

Algorithm 5: NSGA2CGP Export/Deploy phase

```
   Input: INDIVIDUAL OR {TRAINING_DATA}
   Result: CODE OR SAVED_TRAIN
1  switch Input do
2  |  case INDIVIDUAL
3  |  |  WRAPPER ← selectedCodeWrapper(CODEWRPS)
4  |  |  CODE ← @WRAPPER(INDIVIDUAL)
5  |  case {TRAINING_DATA}
6  |  |  SAVED_TRAIN ← saveTrainingData(TRAINING_DATA)
7  |  end
8  end
```

OpenCV. Thus, no matter which environment was used to implement the algorithm, the *CODEWRPS* parameter can be used to reference code wrappers for functions, allowing the user to export deployable code to different languages and platforms. As individuals are being exported, all wrappers appear as options.

4. Experiments

Once implemented, the NSGA2CGP method was submitted to an extensive series of experiments, in order to evaluate its behavior, fine-tune code and verify the quality of results. Thus, we carried out experiments to test if the method was able to optimize solutions to adequately filter grayscale depth images, reducing or eliminating black areas and minimizing their interference over the segmentation processes of a future navigation system.

Navigation systems developed for people with visual impairment use many data sources. Cameras are often found in those systems, enabling them to extract visual data from the environment through image processing and computer vision operations. Depth images from cameras are important sources of data on obstacle distances, and can be output in several formats, such as grayscale tones representing specific distances to camera. However, some areas may have undefined distances, resulting in a 'black pixel' or 'unknown distance' region, leading the obstacle segmentation process to either consider that area as non-existent or wrongly segment it as an object.

The experiments were part of an ongoing project to develop an embedded navigation system for visually impaired people. This system required a new method for adjusting input images, able to reduce the likelihood of system error and diminish computational costs (given the requirement for real-time image processing), since the prototype used LiPo batteries as a power source. Thus, the optimization problem of the experiments is defined in (1), where $\boldsymbol{X}$ is a vector of decision variables: $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})^T$. In this case, we have $n = 2$ as two objectives are being optimized.

$$\begin{aligned} &\textit{minimize} \quad F(\boldsymbol{X}) = (f_e(\boldsymbol{X}), f_c(\boldsymbol{X}))^T \\ &\textit{subject to} \quad g_1(\boldsymbol{X}) \equiv 0 \leq f_e(\boldsymbol{X}) \leq 1, \\ &\qquad\qquad g_2(\boldsymbol{X}) \equiv f_c(\boldsymbol{X}) > 0 \end{aligned} \tag{1}$$

The experiments had the following parameter values:

- Pop. Size: 100 and 1000
- Generations: 100 and 1000
- Columns: 100, 500 and 1000
- Levels-back: 2, Half columns and Free (Same no. of columns)
- Mutation Rate: 5%, 15 and 25%
- Gentype: Free and MorphPair
- UseLogical: No
- Colorscheme: Gray

Some parameters were empirically defined after preliminary runs, since by observing algorithm behavior we noticed that logical operators such as OR/AND / NOT did not have any positive impact over the results of any of the experiments. All experiments were performed in a desktop platform (CPU Intel Core i7-8700k @ 5 GHz, 32 GB Memory @ 3000 MHz) using MATLAB under Windows 10.

The set of functions used in the experiments were dilate, erode, and a custom function named "nop", a passthrough operator to send input 1 or input 2 as an output if the gene number is odd or even, respectively. As for structuring elements, a common set of the following shapes was used: square, disk, diamond, octagon, rectangle, and line. Each shape was generated with odd sizes varying from 1 to 7, while the line shape had angles of 0, 45, 90 and 135 degrees.

The first objective to be optimized is the error between the output images and the goal image. Objective functions for this objective must minimize error, so an error equal to zero means the output coincides exactly with the goal. Five fitness functions are used, as three of them are commonly used in literature approaches [43,52,53] and two were designed and proposed by the authors, based on these three first objectives. The first, f_1, concerns structural similarity (SSIM), and evaluates the quality of output images according to the desired output (Eq. (2)).

$$f_e(\boldsymbol{X}) = f_1 = SSIM(x, y) = 1 - \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2}$$

The second fitness function, f_2, calculates Root Mean Squared Error (RMSE) (Eq. (3)).

$$f_e(\boldsymbol{X}) = f_2 = RMSE(x, y) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N}} \tag{3}$$

The third fitness function is a normalized correlation coefficient (CCOEF) in [0, 1] range of output image x and goal image y (Eq. (4)).

$$f_e(\boldsymbol{X}) = f_3 = CCOEF(x, y) = 1 - \frac{(x \cdot y)}{\sqrt{x \cdot x}\sqrt{y \cdot y}} \tag{4}$$

We propose a fourth fitness function, f_4. Its value is the square root of correlation coefficient and root mean squared error (R^2MSECC) (Eq. (5)). *CCOEF* has achieved good results in black and white images according to literature. The proposal of this function is intended to extend *CCOEF* use on grayscale images by employing *RMSE* on the formula and checking whether it can be useful function or not.

$$f_e(\boldsymbol{X}) = f_4 = R^2MSECC(x, y) = \sqrt{RMSE(x, y) * CCOEF(x, y)} \tag{5}$$

Finally, we propose a fifth fitness function, f_5, comprised of the square root of root mean squared error and structural similarity ($R^2MSESSIM$) (Eq. (6)). This fitness function was designed after empirical observations of how *RMSE* and *SSIM* functions work on different scenarios and is expected to provide good results when used to evaluate sets of distinct images.

$$f_e(\boldsymbol{X}) = f_5 = R^2MSESSIM(x, y) = \sqrt{RMSE(x, y) * SSIM(x, y)} \tag{6}$$

Since the presence of black pixels indicates unknown distances, which need to be filled morphologically, if the *BPWEIGHT* parameter is set to true, NSGA2CGP applies a proportional penalty to individuals whose outputs still have black pixels present. Thus, the final error for each fitness function f_i is multiplied by the percentage of persisting black pixels, with N as the number of pixels of the image, and *BPerc* as the amount of persisting black pixels (Eq. (7)).

$$f_i(x) = f_i(x) * \left(\left(\frac{\sum_{j=1}^{N} x_j = 0}{N}\right) + 1\right) \tag{7}$$

The second objective to be optimized is complexity, *COMPLEX* (f_c), defined as the sum of all structuring element sizes (Eq. (8)), where S is the total number of structuring elements used by an individual x and *SE* is the reference to the structuring element itself.

$$f_c(\boldsymbol{X}) = COMPLEX(x) = \sum_{i=1}^{S} |SE(x_i)| \tag{8}$$

Complexity bears a significant weight on the selection of one individual for use on a functioning device. As previously mentioned, the power constraint has a important role in the trade-off between complexity and error. In this sense, an individual unable to resolve a few black pixels might still be feasible if its complexity/error relation is superior.

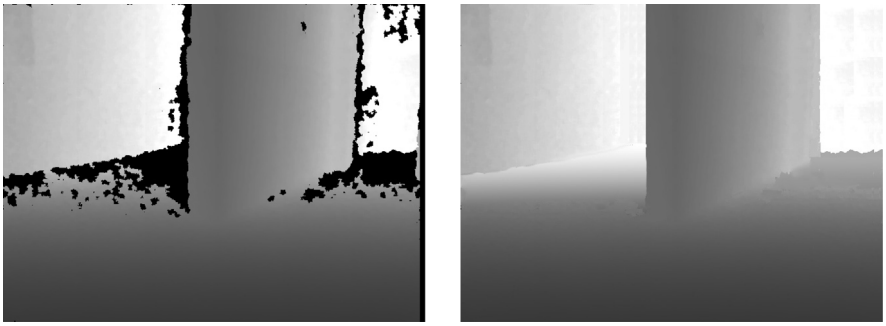
Table 3
Identification of training setups used to evaluate NSGA2CGP.

Train	Pop.	Gen.	Cols.	Mut.	Strat.
S1	100	1000	500	25%	2N
S2	1000	100	500	25%	Elitist
S3	100	1000	500	25%	Elitist
S4	1000	100	500	25%	2N

Table 4
Training results according to error function.

		f_1		f_2		f_3		f_4		f_5	
		O_1	O_2	O_1	O_2	O_1	O_2	O_1	O_2	O_1	O_2
S1	BO_1	0.0474	1387	0.0652	1561	0.1393	721	0.0899	904	0.0606	765
	BO_2	0.0930	10	0.1115	10	0.2753	10	0.1694	10	0.0997	10
	Fsb.	0.0588	265	0.0679	221	0.1550	235	0.0999	303	0.0618	251
S2	BO_1	0.0584	564	0.0718	702	0.1595	1043	0.0989	932	0.0639	1565
	BO_2	0.0889	19	0.0968	18	0.2555	13	0.1799	26	0.1018	12
	Fsb.	0.0616	237	0.0730	297	0.1715	237	0.1043	284	0.0669	248
S3	BO_1	0.0607	401	0.0719	432	0.1736	230	0.1034	826	0.0647	299
	BO_2	0.0961	19	0.1228	15	0.2555	13	0.1918	12	0.1048	10
	Fsb.	0.0607	401	0.0894	169	0.1917	131	0.1142	176	0.0682	183
S4	BO_1	0.0494	3200	0.0701	1009	0.1566	342	0.0797	5065	0.0631	1940
	BO_2	0.0930	10	0.1115	10	0.2753	10	0.1694	10	0.0997	11
	Fsb.	0.0578	410	0.0710	557	0.1733	110	0.0985	532	0.0639	404
RMF		0.0558	–	0.0858	–	0.2031	–	0.1320	–	0.0692	–
Inpainting		0.0662	–	0.0860	–	0.2592	–	0.0722	–	0.0647	–

*O_2 functions are the same for every f_n.



(a) Raw input image (b) Goal image

Fig. 4. Pair of training images used in the experiment.

A training set of ten 640 × 480 depth image pairs was used. Input images were gathered from Microsoft Kinect v1 and each goal image was produced manually. All images were either from house rooms (bedroom, office etc.) or public areas of the university, containing obstacles (walls, doors etc.) to put the system to the test. Examples of an environmental scene's raw input image and goal image are presented in Fig. 4.

Additionally, two other algorithms were implemented and ran with depth images to gather error, power consumption and execution time to fix incorrect depth estimation: inpainting and recursive median filter. Based on Telea's original paper, we implemented his idea of reconstruction of damaged areas of images considering wrong distance pixels as damaged areas [54]. Recursive median filter [55] can be used to fill unknown distance areas of images by applying median filters exclusively on black pixels while they are present in a depth image [44].

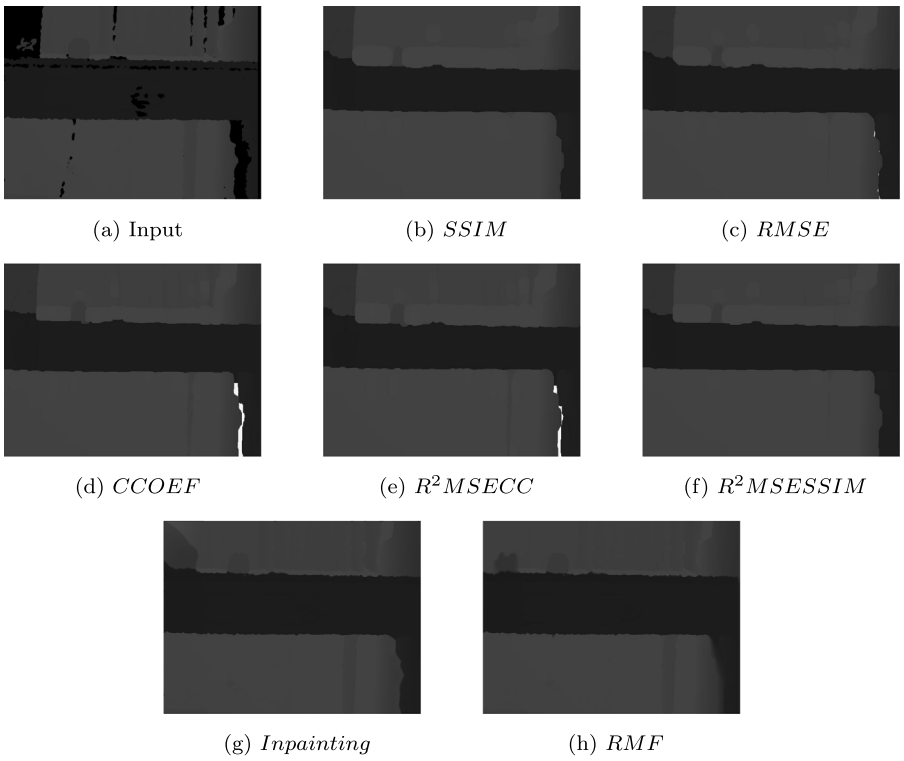
The performance of the evolved solutions was measured through two known evaluation metrics: Hypervolume (HV) [56] and Inverted Generational Distance (IGD) [57]. The first one calculates the hypervolume of non-dominated sets *PF* in respect of a reference point that is considered the worst solution. HV is defined by the sum of each individual hypervolume $v \in PF$(9). Since this is a minimization problem, the best non-dominated sets present higher values as they are more distributed through the objectives. IGD, on other hand, is a modification of the Generational Distance (GD) metric and calculates the average distance of objective vectors of a reference Pareto set P and objectives of an approximation Pareto set A(10). Small IGD values indicate better convergence and diversity.

$$HV = \sum_{i=1}^{|PF|} v_i \tag{9}$$

$$IGD(A,P) = \frac{\sum_{i=1}^{|A|} dst(A_i,P)}{|A|} \quad , where$$
$$dst(A_i,P) = min_{j=1}^{|P|} \sqrt{\sum_{m=1}^{M} \left(\frac{obj_m(A_i) - obj_m(P_j)}{obj_m^{max} - obj_m^{min}} \right)^2} \tag{10}$$

5. Results and discussion

As presented in the previous section, experiments were conducted to evaluate and validate NSGA2CGP using multiple indicators. Raw training results for execution time and energy consumption, among others, were analyzed first, in order to choose



(a) Input (b) $SSIM$ (c) $RMSE$

(d) $CCOEF$ (e) R^2MSECC (f) $R^2MSESSIM$

(g) $Inpainting$ (h) RMF

Fig. 5. Output images of the S4 setup after training. White areas indicate persisting unknown distances.

candidates for the next measurements. Due to the numerous possible parameter combinations, here we chose to present only the best settings. Thus, Table 3 identifies each training setup and its parameters. Particularities such as error function are discussed later.

Although Table 3 does not include this information, *GENTYPE* was defined as *Free*. Empirical data from training sessions demonstrated that neither the *MorphPair GENTYPE* nor a *LBACK* setting of 50% the number of columns were able to leverage any extra performance. It was also noted that the *ROWS* parameter provided no increase in individuals' quality or performance, thus being set to 1. Setting *COLS* to 1000 did not improve results compared to 500, so it was kept at 500. Finally, *TMUT* tests showed a percentage of 25% to be optimal.

As for objective values, the goal was to minimize both error and complexity. All four setups shown in Table 3 were submitted to error function $f_{(1-5)}$ and complexity function f_c, achieving superior results compared to any other setup. Table 4 shows those setups and their individual error minimization performance, together with the respective complexity values. Three individuals from each setup were included in Table 4: Best Objective 1 (Error) is shown as BO_1, Best Objective 2 (Complexity) as BO_2, and the selected feasible solution that fits both objectives appears as *Fsb*. Each result pair (error, complexity) is a column identified as $f_{(1-5)}$ and O_1 is the error value itself and O_2 is the complexity value.

The results in Table 4 show that Sample Correlation Coefficient f_3 is outside the scope of this work. Our investigation suggests that it is less forgiving when the output images still contain unknown distances.

Mutation strategies play a major role in minimizing error, with 2N producing better results than Elitist. S1 and S4 setups (2N) resulted in less errors than S2 and S3 (Elitist). Complexity varied throughout all setups, although S4's feasible solution achieved the same error as S2's best error in the f_5 function, while having 75.2% less complexity. This pattern (2N feasible solution better than Elitist's best error) occurred frequently for all f_i and their f_c complexities. Elitist strategies also were observed to deliver very small sets of best solutions (first front), creating situations where the only feasible solutions were the most complex ones, as in S3's *SSIM* error. As expected by authors, Recursive Median Filter and Telea's Inpainting methods achieved good results when submitted to each error function, although their complexity were not calculated for not being evolved morphological filters.

As for visual quality, Fig. 5 provides an example of raw input image that was not part of the training set, and resulting outputs after processing by the $f_{1:5}$ feasible trained solutions from the S4 setup. Both *SSIM* (Fig. 5*a*) and *R²MSESSIM* (Fig. 5*f*) were successful in resolving unknown distance pixels, while *RMSE* (Fig. 5*c*), *CCOEF* (Fig. 5*d*) and *R²MSECC* (Fig. 5*e*) presented flaws, highlighted in white. *Inpainting*(Fig. 5*g*) and *RMF*(Fig. 5*h*) also managed to eliminate unknown distance pixels as proposed by literature.

Since NSGA2CGP's main purpose is improving navigation systems for visually impaired people, the individuals presented in Table 4 were converted into C++ code using the NSGA2CGP Export/Deploy phase. A C++ OpenCV wrapper was developed in order to generate deployable morphological filters from trained individuals. These filters were then deployed to an embedded platform, the NVIDIA Jetson TX2 – the same used in our prototype VIP Navigation System – and executed as standalone code while gathering power measurements, shown in Table 5.

The power measurements listed in Table 5 complement the preliminary findings in Table 4. The correlation coefficient was originally included, but given the poor quality of its trained individuals, we opted not to consider it at all. Since *SSIM* and *R²MSESSIM* had similar visual quality, as shown in Fig. 5, their consumption data can be used to compare both functions in a real deployment platform. Elitist strategies were confirmed as inferior to 2N under the same feasible solution conditions, with S4 (2N) trainings consuming less power than S2 (Elitist), and S1 having similar results to S3. Individuals from error function *SSIM* in S3 performed similarly in respect to Best Objective 1 and its feasible solution, consuming the same power.

Table 5
Power consumption of generated filters and literature filters, Inpainting (Inpnt.) and Recursive Median Filter (RMF), in milliwatts (mW).

		Evolved				Others	
		f_1	f_2	f_4	f_5	Inpnt.	RMF
S1	BO_1	1114	699	947	**465**	353	5833
	BO_2	48	48	48	48		
	Fsb.	337	393	353	**289**		
S2	BO_1	721	**497**	767	1154		
	BO_2	48	48	48	48		
	Fsb.	425	392	257	**361**		
S3	BO_1	473	562	1177	**234**		
	BO_2	48	48	48	48		
	Fsb.	251	425	377	**226**		
S4	BO_1	1599	659	1410	**586**		
	BO_2	48	48	48	48		
	Fsb.	273	274	400	***97***		

Table 6
Execution times of generated filters and literature filters, Inpainting (Inpnt.) and Recursive Median Filter (RMF), in milliseconds (ms).

		Evolved				Others	
		f_1	f_2	f_4	f_5	Inpnt.	RMF
S1	BO_1	555	300	466	**236**	244	6530
	BO_2	4	4	5	4		
	Fsb.	199	156	155	**152**		
S2	BO_1	286	**234**	446	545		
	BO_2	4	4	4	4		
	Fsb.	181	177	380	**173**		
S3	BO_1	177	293	580	**151**		
	BO_2	4	4	4	5		
	Fsb.	6	188	166	**146**		
S4	BO_1	1245	**235**	933	537		
	BO_2	6	3	5	5		
	Fsb.	142	148	197	***67***		

As expected, Best Objective 2's solutions had very low power consumption, given their low complexity value, but also presented little to no progress in regards to unknown distance pixels. Individuals consumed a minimum value of 48 mW, the base consumption for running an entire filter's processing cycle. Complexities below 40, such as BO_2's, allow individuals to be run with minimal power consumption but deliver poor-quality outputs.

Author's proposed function $R^2MSESSIM$ achieved the best energy consumption throughout all training sessions, consuming 97 mW as a feasible solution in the S4 setup. $R^2MSESSIM$ also consumed less energy than all other error functions in the S1 and S3 setup. In the S2 setup, *RMSE* had the second lowest consumption in feasible solution and Best Objective 1, with 497 mW. The lowest consumption feasible solution is once again $R^2MSESSIM$. Fig. 6 presents a trained population using the S4 setup and the $R^2MSESSIM$ error function.

Inpainting method consumed low to moderate amount of energy, drawing 353 mW to perform its operations to fill missing pixels. Recursive Median Filter consumed a lot of energy, 5833 mW or 5.8 W, to recursively fill all the missing spots. Although they presented good visual and error results, their power consumption are not ideal to be used in a navigation system.

Another important concern for navigation systems is response time. The system should not only be power efficient but also capable of processing multiple tasks using minimal processing time, in order to send proper real-time feedback to VIPs. These systems have many other tasks beyond filtering depth images to perform, such as classification, segmentation and recognition, hence a depth image filter with fast execution will do better in this scenario. Table 6 shows the execution time (in milliseconds) for each solution deployed to the Jetson TX2.

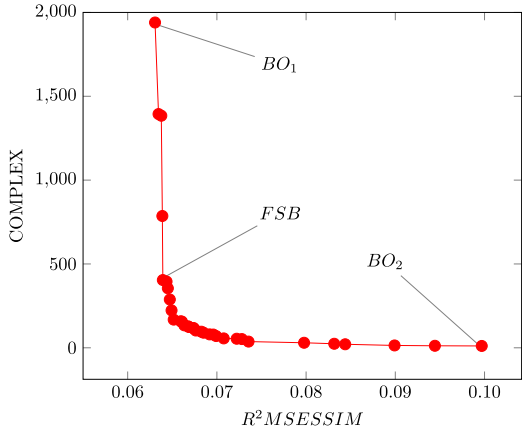


Fig. 6. Visualization of $R^2MSESSIM$ training for S4 setup.

Results from Table 6 provide important data for choosing an individual fit to navigation systems' previously discussed constraints. The individual should consume less power while maintaining good execution time and sufficient quality. Some good-quality results demanded not only more power, but also took longer and required impractical processing time. The top-quality individual employing the *SSIM* error function in the S4 setup took 1.245 s to filter an image, far from an ideal completion time, considering the purpose of this work.

Regarding feasibility, in every training setup, all best feasible solutions chosen by the authors ran faster than BO_1 in S1 to S4. Meanwhile, the *SSIM* error function was not faster in any

Table 7
Chosen feasible solution from S4 setup and f_5 fitness function. NodeID references the address of used* nodes while operation describes node's operation.

NodeID	Operation	NodeID	Operation
1	Input Image	45	nop(33, 41)
2	square(1 × 1)	51	ero(44, 34)
4	diamond(2)	53	nop(45, 2)
5	rectangle(1 × 3)	55	dil(53, 42)
12	square(3 × 3)	66	dil(45, 55)
20	line(3, 90°)	69	nop(42, 22)
22	square(5 × 5)	73	nop(51, 69)
28	line(5, 0°)	83	ero(20, 5)
32	square(7 × 7)	111	nop(20, 5)
33	disk(7, 9)	117	nop(111, 66)
34	diamond(8)	126	dil(32, 66)
41	line(7, 135°)	127	nop(83, 73)
42	ero(1, 28)	163	dil(126, 127)
43	ero(12, 4)	384	dil(117, 163)
44	dil(42, 43)	***Output***	***384***

*Nodes not used by output are omitted.

setup. Finally, $R^2MSESSIM$ had good power consumption results, and was also the fastest filter, considering feasible solutions, with an average execution time of 67 ms in S4 setup; it was also faster than the other functions in the S3 setup, with 146 ms average execution time for feasible solution. Both Inpainting and Recursive Median Filter did not present good execution times. Inpainting had an average time of 244 ms and RMF took over 6.5 s to fill the needed areas, thus they are definitely not suitable to be deployed on our system. Table 7 presents a list of used structuring elements, nodes and operations of chosen feasible solution (S4 setup and f_5 function) which best fits the needs of our navigation system.

Finally, the performances of S1, S2, S3 and S4 setups were calculated and their results are shown in Fig. 7. For better understanding, the results were normalized within [0, 1] interval.

As expected, S1 and S4 solutions had the best results when compared with S2 and S3 for both Hypervolume and Inverted Generational Distance metrics. For HV, the authors observed that S1 and S4 are widely distributed since their hypervolumes achieved higher values for every error function. S4 setup performed even better and presented average values over 0.9 for all functions. As for individual performances, the hypervolume of $R^2MSESSIM$ functions was better than all the other three in every setup scenario, delivering solutions very well distributed through their Pareto fronts. For IGD, S1 and S4 also performed better than S2 and S3, achieving close to zero values. However, the convergence of $R^2MSESSIM$ function was worse when compared to the other functions within the same setup. Even though the difference of the average IGD values for all functions in S1 and S4 setups is very low, authors should further investigate and improve it for the next works. S2 and S3, on other hand, had very poor convergence as it resulted in high IGD values over 0.7 for all functions.

6. Conclusion

This paper presented a multi-objective optimization method for depth image morphological filters, NSGA2CGP, using the NSGA-II algorithm and the Cartesian Genetic Programming technique. It was used to optimize error rates and structuring element complexity, so as to achieve a deployable, energy- and time-efficient feasible solution for VIP navigation systems.

The NSGA2CGP was constructed according to the specifications in Section 3 and was used to train and evolve multiple scenarios of parameters, varying the number of individuals, columns, rows, mutation rate etc. After hundreds of tests, we minimized it to four best setups and they evolved each one of the objective functions presented. By being a theoretical method, it can be implemented in any language capable of manipulating images such as C ++, Java and Python, and is capable of working with black and white images, not only grayscale.

We also proposed two error functions, $R^2MSESSIM$ and R^2MSECC (meaning squared root of *RMSE* times *SSIM*, and *RMSE* times Sample Correlation Coefficient, respectively). $R^2MSESSIM$ achieved better results in comparison to the other four functions, and provided the best feasible solution in terms of combining low power consumption and faster processing. It also prove itself better than other known approaches like Inpainting and Recursive Median Filter in execution time and power consumption. R^2MSECC did not prove to be useful, so it should not be used again in future works, unlike $R^2MSESSIM$. In terms of performance, $R^2MSESSIM$ presented good distribution of its non-dominated solutions by achieving high hypervolumes as well as good convergence due its very low IGD.

All trained experiments were deployed to a Jetson TX2 device, the same used in our prototype navigation system. Both power consumption and execution time were gathered by means of the NSGA2CGP Export/Deploy phase, generating a C++ OpenCV code wrapper ready for usage. NSGA2CGP was a good fit to the device's specs and managed to deliver good solutions in terms of quality, power consumption and execution time, enabling users to select and export any solution that fits their hardware constraints.

In the future, we plan to extend NSGA2CGP to be used in optimizing other parts of VIP navigation systems, and also tackle multi-objective optimization of GPU-ready morphological filter code, generating optimized kernels to further improve power consumption and speed. We also plan to implement and modify literature approaches proposed for digital circuits [20,22] and benchmark them in image processing domain to compare with NSGA2CGP.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106130.

CRediT authorship contribution statement

Antonio Miguel Batista Dourado: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Visualization, Project administration. **Emerson Carlos Pedrino:** Formal analysis, Writing - review & editing, Supervision.

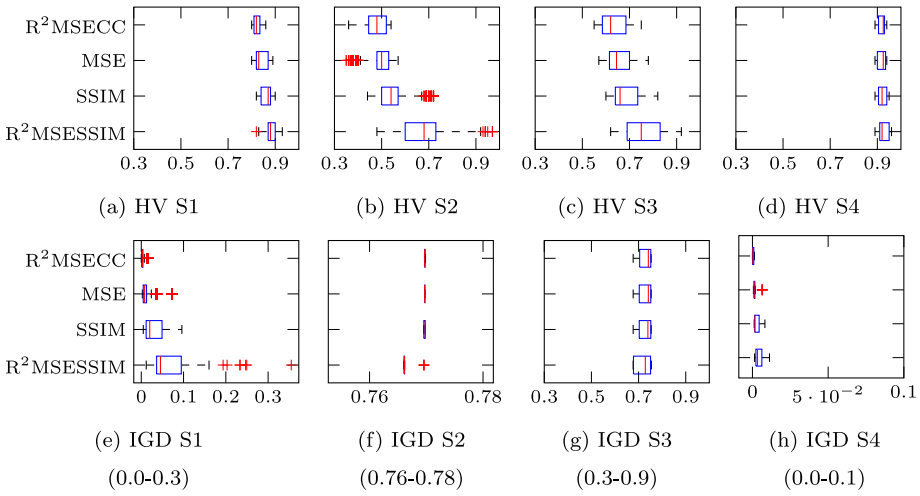


Fig. 7. Performance results of S1, S2, S3 and S4 setups according to their Hypervolumes (*a*–*d*) and Inverted Generational Distances (*e*–*h*).

Acknowledgments

This study was financed by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil – Finance Code 001, and by the São Paulo Research Foundation (FAPESP), Brazil – Project 2017/26421-3. We also thank the Federal Institute of São Paulo and the PARA-D.V. nongovernmental organization, whose contribution made this work possible.

References

[1] WHO, Blindness and vision impairment, in: Fact Sheet, 2018, p. 1, URL http://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment.

[2] B. Leporini, F. Paternò, Increasing usability when interacting through screen readers, Univ. Access Inf. Soc. 3 (1) (2004) 57–70, http://dx.doi.org/10.1007/s10209-003-0076-4.

[3] S.K. Kane, J.P. Bigham, J.O. Wobbrock, Slide rule : Making mobile touch screens accessible to blind people using multi-touch interaction techniques, Design 33 (2008) 73–80, http://dx.doi.org/10.1145/1414471.1414487.

[4] J. Zhang, S.K. Ong, A.Y. Nee, Navigation systems for individuals with visual impairment: A survey, in: I-CREATe 2008 - International Convention on Rehabilitation Engineering and Assistive Technology 2008, 2008, pp. 159–162.

[5] N. Silberman, R. Fergus, Indoor scene segmentation using a structured light sensor, in: 2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops, 2011, pp. 601–608, http://dx.doi.org/10.1109/ICCVW.2011.6130298.

[6] L. Li, Time-of-Flight Camera–An Introduction, Tech. Rep. January, Texas Instruments, Dallas, 2014, p. 10.

[7] S. Lee, S. Sharma, Real-time disparity estimation algorithm for stereo camera systems, IEEE Trans. Consum. Electron. 57 (3) (2011) 1018–1026, http://dx.doi.org/10.1109/TCE.2011.6018850.

[8] E.C. Pedrino, V.O. Roda, Real-time morphological pipeline architecture using high-capacity programmable logical devices, J. Electron. Imaging 16 (2) (2007) 023002, http://dx.doi.org/10.1117/1.2743084.

[9] A.M.B. Dourado, E.C. Pedrino, Embedded navigation and classification system for assisting visually impaired people, in: 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2018, Vol. 5, 2018, pp. 516–523.

[10] J.F. Miller, Cartesian genetic programming, in: Cartesian Genetic Programming, Springer, 2011, pp. 17–34, http://dx.doi.org/10.1007/978-3-642-17310-3_2.

[11] B.F.G. Katz, S. Kammoun, G.e. tan Parseihian, O. Gutierrez, A. Brilhault, M. Auvray, P. Truillet, M. Denis, S. Thorpe, C. Jouffrais, NAVIG: augmented reality guidance system for the visually impaired, Virtual Real. 16 (4) (2012) 253–269.

[12] A. Kumar, R. Patra, M. Manjunatha, J. Mukhopadhyay, A.K. Majumdar, An electronic travel aid for navigation of visually impaired persons, in: 2011 Third International Conference on Communication Systems and Networks, COMSNETS 2011, IEEE, 2011, pp. 1–5, http://dx.doi.org/10.1109/COMSNETS.2011.5716517.

[13] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[14] H.A. Shehadeh, M.Y.I. Idris, I. Ahmedy, H.R. Hassen, Optimal placement of near ground VHF/UHF radio communication network as a multi objective problem, Wirel. Pers. Commun. (0123456789) (2019) http://dx.doi.org/10.1007/s11277-019-06780-6.

[15] F. Habibi, F. Barzinpour, S.J. Sadjadi, A mathematical model for project scheduling and material ordering problem with sustainability considerations: A case study in Iran, Comput. Ind. Eng. 128 (February 2018) (2019) 690–710, http://dx.doi.org/10.1016/j.cie.2019.01.007.

[16] E.A. Kavunnikova, B.N. Starovoitova, S.V. Golovin, A.M. Krivtsov, Comparison of design optimization algorithms of a multiply fractured horizontal well, J. Phys. Conf. Ser. 1268 (1) (2019) 012029, http://dx.doi.org/10.1088/1742-6596/1268/1/012029.

[17] Z. Pooranian, N. Nikmehr, S. Najafi-Ravadanegh, H. Mahdin, J. Abawajy, Economical and environmental operation of smart networked microgrids under uncertainties using NSGA-II, in: 2016 24th International Conference on Software, Telecommunications and Computer Networks, SoftCOM, IEEE, 2016, pp. 1–6, http://dx.doi.org/10.1109/SOFTCOM.2016.7772136.

[18] S. Garcia, C.T. Trinh, Comparison of multi-objective evolutionary algorithms to solve the modular cell design problem for novel biocatalysis, Processes 7 (6) (2019) 361, http://dx.doi.org/10.3390/pr7060361.

[19] J. Hilder, J.A. Walker, A. Tyrrell, Use of a multi-objective fitness function to improve cartesian genetic programming circuits, in: 2010 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, 2010, pp. 179–185, http://dx.doi.org/10.1109/AHS.2010.5546262.

[20] R. Hrbacek, V. Mrazek, Z. Vasicek, Automatic design of approximate circuits by means of multi-objective evolutionary algorithms, in: 2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era, DTIS, IEEE, 2016, pp. 1–6, http://dx.doi.org/10.1109/DTIS.2016.7483885.

[21] R. Kalkreuth, G. Rudolph, J. Krone, More efficient evolution of small genetic programs in Cartesian Genetic Programming by using genotypie age, in: 2016 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2016, pp. 5052–5059, http://dx.doi.org/10.1109/CEC.2016.7748330.

[22] P. Kaufmann, M. Platzner, Combining local and global search: A multi-objective evolutionary algorithm for cartesian genetic programming, in: Complex. Comput., 2018, pp. 175–194, http://dx.doi.org/10.1007/978-3-319-67997-6_8.

[23] P. Paris, E. Pedrino, M. Nicoletti, Automatic learning of image filters using Cartesian genetic programming, Integr. Comput.-Aided Eng. 22 (2) (2015) 135–151, http://dx.doi.org/10.3233/ICA-150482.

[24] E.C. Pedrino, M.C. Nicoletti, J.H. Saito, L.M.V. Cura, V.O. Roda, A binary morphology-based clustering algorithm directed by genetic algorithm, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2013, pp. 409–414, http://dx.doi.org/10.1109/SMC.2013.76.

[25] E.C. Pedrino, J.H. Saito, V.O. Roda, A genetic programming approach to reconfigure a morphological image processing architecture, Int. J. Reconfig. Comput. 2011 (2) (2011) 1–10, http://dx.doi.org/10.1155/2011/712494.

[26] X. Xu, Q. Wu, C. He, S. Wan, L. Qi, H. Wang, Edge computing-enabled resource provisioning for video surveillance in internet of vehicles, in: Smart City and Informatization, Vol. 4, 2019, pp. 128–140, http://dx.doi.org/10.1007/978-981-15-1301-5_11.

[27] R.K. Sanjuna, K. Dinakaran, A multi-object feature selection based text detection and extraction using skeletonized region optical character recognition in-text images, Int. J. Eng. Technol. 7 (3.6) (2018) 386, http://dx.doi.org/10.14419/ijet.v7i3.6.16009.
[28] S. Srivas, P.G. Khot, Analysis and visualization of multidimensional GIS images using multi objective algorithm (MOA), Int. J. Comput. Sci. Eng. 6 (8) (2018) 460–464, http://dx.doi.org/10.26438/ijcse/v6i8.460464.
[29] M. Shi, Z. He, Z. Chen, H. Zhang, A novel multi-objective optimization-based image registration method, in: GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference, 2016, pp. 605–612, http://dx.doi.org/10.1145/2908812.2908872.
[30] M.A. El Aziz, A.A. Ewees, A.E. Hassanien, M. Mudhsh, S. Xiong, Multi-objective whale optimization algorithm for multilevel thresholding segmentation, Stud. Comput. Intell. 730 (2018) 23–39, http://dx.doi.org/10.1007/978-3-319-63754-9_2.
[31] T. Sağ, M. Çunka, Color image segmentation based on multiobjective artificial bee colony optimization, Appl. Soft Comput. J. 34 (2015) 389–401, http://dx.doi.org/10.1016/j.asoc.2015.05.016.
[32] C.W. Bong, M. Rajeswari, Multi-objective nature-inspired clustering and classification techniques for image segmentation, Appl. Soft Comput. J. 11 (4) (2011) 3271–3282, http://dx.doi.org/10.1016/j.asoc.2011.01.014.
[33] L. Dou, D. Xu, H. Chen, Y. Liu, in: C.M. Falco, X. Jiang (Eds.), Image de-Noising Based on Mathematical Morphology and Multi-Objective Particle Swarm Optimization, Vol. 10420, No. Icdip, 2017, http://dx.doi.org/10.1117/12.2281560, p. 104202I.
[34] M. Koppen, K. Franke, Pareto-dominated hypervolume measure: An alternative approach to color morphology, in: 7th International Conference on Hybrid Intelligent Systems, HIS 2007, IEEE, 2007, pp. 234–239, http://dx.doi.org/10.1109/ICHIS.2007.4344057.
[35] J. Deng, J. Yan, G. Cheng, Multi-objective concurrent topology optimization of thermoelastic structures composed of homogeneous porous material, Struct. Multidiscip. Optim. 47 (4) (2013) 583–597, http://dx.doi.org/10.1007/s00158-012-0849-6.
[36] H. Wang, G. Zhang, H. Qi, L. Ma, Multi-objective optimization on pore segmentation, in: 5th International Conference on Natural Computation, Vol. 4, ICNC 2009, 2009, pp. 613–617, http://dx.doi.org/10.1109/ICNC.2009.572.
[37] A. Criminisi, P. Perez, K. Toyama, Object removal by exemplar-based inpainting, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings, Vol. 2, IEEE Comput. Soc., 2003, pp. II–721–II–728, http://dx.doi.org/10.1109/CVPR.2003.1211538.
[38] I. Peterson, Filling in blanks, Sci. News 161 (19) (2002) 299, http://dx.doi.org/10.2307/4013521.
[39] A. Hervieu, N. Papadakis, A. Bugeau, P. Gargallo, V. Caselles, Stereoscopic image inpainting: Distinct depth maps and images inpainting, in: 2010 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 4101–4104, http://dx.doi.org/10.1109/ICPR.2010.997.
[40] F. Qi, J. Han, P. Wang, G. Shi, F. Li, Structure guided fusion for depth map inpainting, Pattern Recognit. Lett. 34 (1) (2013) 70–76, http://dx.doi.org/10.1016/j.patrec.2012.06.003.
[41] M.A. Garduño-Ramón, I.R. Terol-Villalobos, R.A. Osornio-Rios, L.A. Morales-Hernandez, A new method for inpainting of depth maps from time-of-flight sensors based on a modified closing by reconstruction algorithm, J. Vis. Commun. Image Represent. 47 (2017) 36–47, http://dx.doi.org/10.1016/j.jvcir.2017.05.003.
[42] R. Liu, Z. Deng, L. Yi, Z. Huang, D. Cao, M. Xu, R. Jia, Hole-filling based on disparity map and inpainting for depth-image-based rendering, Int. J. Hybrid Inf. Technol. 9 (5) (2016) 145–164, http://dx.doi.org/10.14257/ijhit.2016.9.5.12.
[43] L. Chen, H. Lin, S. Li, Depth image enhancement for Kinect using region growing and bilateral filter, in: Pattern Recognition (ICPR), 2012 21st International Conference on, No. Icpr, 2012, pp. 3070–3073.
[44] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view RGB-D object dataset, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 1817–1824, http://dx.doi.org/10.1109/ICRA.2011.5980382.
[45] A. Amamra, N. Aouf, GPU-based real-time RGBD data filtering, J. Real-Time Image Process. 14 (2) (2018) 323–340, http://dx.doi.org/10.1007/s11554-014-0453-7.
[46] A. Maimone, H. Fuchs, Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras, in: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, 2011, pp. 137–146, http://dx.doi.org/10.1109/ISMAR.2011.6092379.
[47] NVIDIA, CUDA C programming guide, in: Cuda Toolkit Documentation, 2018, p. 311.
[48] C. Brugger, L. Dal'Aqua, J.A. Varela, C. De Schryver, M. Sadri, N. Wehn, M. Klein, M. Siegrist, A quantitative cross-architecture study of morphological image processing on CPUs, GPUs, and FPGAs, in: ISCAIE 2015 - 2015 IEEE Symposium on Computer Applications and Industrial Electronics, 2015, pp. 201–206, http://dx.doi.org/10.1109/ISCAIE.2015.7298356.
[49] J.M. Castillo-Secilla, M. Saval-Calvo, L. Medina-Valdès, S. Cuenca-Asensi, A. Martínez-Álvarez, C. Sánchez, G. Cristóbal, Autofocus method for automated microscopy using embedded GPUs, Biomed. Opt. Express 8 (3) (2017) 1731–1740, http://dx.doi.org/10.7163/GPol.0025.
[50] L. Niu, C. Qian, J.R. Rizzo, T. Hudson, Z. Li, S. Enright, E. Sperling, K. Conti, E. Wong, Y. Fang, A wearable assistive technology for the visually impaired with door knob detection and real-time feedback for hand-to-handle manipulation, in: Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, Vol. 2018, ICCVW 2017, 2018, pp. 1500–1508, http://dx.doi.org/10.1109/ICCVW.2017.177.
[51] B.W. Goldman, W.F. Punch, Reducing wasted evaluations in cartesian genetic programming, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7831 LNCS (3), 2013, pp. 61–72, http://dx.doi.org/10.1007/978-3-642-37207-0_6.
[52] I. Yoda, K. Yamamoto, H. Yamada, Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms, Image Vis. Comput. 17 (10) (1999) 749–760, http://dx.doi.org/10.1016/S0262-8856(98)00151-6.
[53] M.I. Quintana, R. Poli, E. Claridge, Morphological algorithm design for binary images using genetic programming, Genet. Program. Mach. 7 (1) (2006) 81–102, http://dx.doi.org/10.1007/s10710-006-7012-3.
[54] A. Telea, An image inpainting technique based on the fast marching method, J. Graph. Tools 9 (1) (2004) 23–34, http://dx.doi.org/10.1080/10867651.2004.10487596.
[55] P. Varshney, A. Tyagi, An enhanced recursive median filter for noise reduction based on randomness in pixel values of an image, Int. J. Comput. Appl. 113 (8) (2015) 8–10, http://dx.doi.org/10.5120/19845-1706.
[56] Q. Liao, Z. Sheng, H. Shi, L. Zhang, L. Zhou, W. Ge, Z. Long, A comparative study on evolutionary multi-objective optimization algorithms estimating surface duct, Sensors 18 (12) (2018) http://dx.doi.org/10.3390/s18124428 (in Switzerland).
[57] L.C. Bezerra, M. López-Ibáñez, T. Stützle, An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10173 LNCS, 2017, pp. 31–45, http://dx.doi.org/10.1007/978-3-319-54157-0_3.