# Evolving Processing Pipelines for Industrial Imaging with Cartesian Genetic Programming

Andreas Margraf
*Digital Production and AI*
*Fraunhofer IGCV*
Augsburg, Germany
andreas.margraf@igcv.fraunhofer.de

Henning Cui
*Organic Computing Group*
*University of Augsburg*
Augsburg, Germany
henning.cui@uni-a.de

Anthony Stein
*AI in Agricultural Engineering*
*University of Hohenheim*
Stuttgart, Germany
anthony.stein@uni-hohenheim.de

Jörg Hähner
*Organic Computing Group*
*University of Augsburg*
Augsburg, Germany
joerg.haehner@uni-a.de

***Abstract*—The reconfiguration of machine vision systems heavily depends on the collection and availability of large datasets, rendering them inflexible and vulnerable to even minor changes in the data. This paper proposes a refinement of Miller's Cartesian Genetic Programming methodology, aimed at generating filter pipelines for image processing tasks. The approach is based on CGP-IP, but specifically adapted for image processing in industrial monitoring applications. The suggested method allows for retraining of filter pipelines using small datasets; this concept of self-adaptivity renders high-precision machine vision more resilient to faulty machine settings or changes in the environment and provides compact programs. A dependency graph is introduced to rule out invalid pipeline solutions. Furthermore, we suggest to not only generate pipelines from scratch, but store and reapply previous solutions and re-adjust filter parameters. Our modifications are designed to increase the likelihood of early convergence and improvement in the fitness indicators. This form of self-adaptivity allows for a more resource-efficient configuration of image filter pipelines with small datasets.**

***Index Terms*—cgp, image filters, monitoring, segmentation**

## I. Introduction

In recent years, image processing has been dominated by convolutional neural networks (CNNs), achieving high accuracy in object detection and recognition, even in industrial applications for process monitoring. Most neural networks, however, entail a training overhead that requires extensive datasets with predefined labels, which might be challenging to obtain. On the other hand, traditional image processing with problem-tailored filters offers reproducibility, interpretability, and stability but may struggle with complex recognition tasks due to limited flexibility. To address this limitation, we propose an organic computing (OC) approach using Cartesian Genetic Programming (CGP) to evolve image filters for image processing tasks. CGP uses filter functions and parameters from common programming libraries to bridge the gap for applications in which low complexity and transparency outweigh generalizability.

### *A. The Role of Machine Learning in Industrial Monitoring*

The reliable discovery of anomalies is considered crucial in real-world applications like inline monitoring or scrap detection systems. Machine vision systems play a key role in industrial quality assurance, especially in manufacturing environments, using optical monitoring sensors to scan product surfaces during an ongoing process. This study examines surface patterns of high-performance textiles, glass, metal, wood, electronics, plastics, and fluids, including defects like fiber misalignments, aggregations of foreign structure and manufacturing-induced surface anomalies. Image-based monitoring systems aim to maintain production within a tolerance window. Detecting deviations from product-specific requirements is a challenging task that state-of-the-art machine learning models may not always meet.

### *B. Organic Computing Principles in Image Segmentation*

State-of-the-art computer systems have become more complex and ubiquitous, which is why managing their complexity and unpredictable behavior is challenging. With the rise of information technologies and edge computing, we are surrounded by intelligent interactive systems [1]. The design paradigm of OC addresses large computing systems with so-called *self-x* properties. OC systems are characterized by productive and organic parts responsible for system (re-)configuration and adaptation. Building such systems remains a challenge, but our proposed methodology integrates OC principles using biology and nature-inspired algorithms to provide *self-configuration*, *self-learning*, and *self-optimization*. Additionally, the approach is deemed to generate *interpretable* and *human-readable* programs, transforming logic into *compact*, *robust*, *transparent*, and *self-adaptive* pipelines for industrial image filter problems.

### *C. Related Work*

The following section will give an overview on prior work in related fields of research:
Inspired by Genetic Programming (GP) applications for circuit generation and boolean function learning, Miller et al. introduced CGP [2] to evolve efficient programs. The current state of CGP and open research questions have been elaborated by Miller in a recent publication [3]. Furthermore, the evolution of image filters has been applied to FPGA programming [4] as well as GPU architectures [5]. In a comparable context, Kowaliw et al. [6] investigated the use of GP for image transforms. Leitner et al. applied CGP-based image processing to demonstrate its potential for humanoid robot machine vision [7]. Subsequently, Sekanina et al. and Harding et al. published extensive surveys on the theory and application of CGP on

image processing tasks [8], [9]. Moreover, Goldman et al. proposed an approach to minimize wasted evaluations in CGP [10]. Additionally, CGP enhanced image filtering has been applied to industrial monitoring problems for different types of carbon fiber [11], [12] and for biomedical image analysis in a framework *Katezio* as presented by Cortacero et al. [13]. To the best of our knowledge, this paper is the first to study the impact of dependency graphs on CGP evolved programs in the context of industrial image segmentation.

### D. Structure

The remainder of this paper is structured as follows: In Section I, we present an overview of evolutionary algorithms in industrial imaging and open questions. Section II outlines modifications to CGP and the dependency graph for industrial image segmentation. Section III covers the experimental design, application environments, and evaluation methods, while Section IV critically analyzes the results. In Section V, we conclude the study and provide a summary of future research.

## II. Methodology

In traditional computer vision, image processing pipelines are manually engineered based on sample images with pre-labeled areas known as *regions of interest (ROI)*. These pipelines use cascading filters or function operators to extract features that approximate specific elements in the image based on a reference dataset to solve segmentation tasks. To reduce the effort of domain-specific algorithm configuration, we propose a CGP approach for image processing operators.

### A. Cartesian Genetic Programming

*CGP* is a specialized version of *GP* that arranges all nodes $F$ on a two-dimensional cartesian grid for genotype representation. These nodes consist of three operator types: *input nodes*, *function nodes*, and *output nodes*. CGP's configuration prevents genetic bloat, resulting in small, resource-efficient programs. The many-to-one phenotype-genotype mapping creates non-coding genes that do not directly affect the final solution, which benefits redundancy and genetic diversity. In each iteration, the mutation of edges affects inactive genes, leading to *neutral genetic drift*, which is favored by a high number of inactive genes in the genotype [14]. The configuration of CGP is specified by a program $P$:

$$P = \{F, r, c, n, l, i, o\} \tag{1}$$

$P$ uses a set of function nodes $F$ which includes a variety of image filter operations. The nodes are positioned on a cartesian grid. The size of this cartesian grid is defined by the *number of nodes* $n$, *rows* $r$ and *columns* $c$. Furthermore, the *levels back* parameter $l$ indicates the number of columns from which each node may accept an input. The parameters $i$ and $o$ define the number of input and output nodes, respectively.

### B. Modifications by Node Type Dependencies

In order to avoid invalid solutions, we introduce a *dependency graph* $DG$. In contrast to traditional CGP, our approach reduces the combinatorial complexity by only allowing connections between nodes that produce an executable solution. The rationale is to prevent the occurrence of invalid, non-executable individuals. Conversely, this means that only connections between operator nodes are formed that are allowed by graph dependency. We introduce the directed acyclic *dependency graph* $DG(T, E_{DG})$, consisting of operator types $T$ and relationships $E_{DG}$ among operator types $T_i \in T$ with $T = \{T_1, T_2, \ldots, T_t\}$. $DG$ defines the structure of the operator pipeline on the cartesian grid to ensure that any filter combination is executable. Thus, data processed by one node and passed on to another carries enough information for reasonable filtering. Once a filter is applied, it restricts the number of possible successors. $DG$ is monotonic, therefore: $\forall T_i, T_j : i < j \rightarrow T_i < T_j$. Also, it is acyclic, ergo: $\forall E_{i,j} = (T_i, T_j) \in E : i < j$. Our approach extends the methodology proposed by Miller et al. [15] by defining a pipeline solution as a composition of edges $E_{DG}$. Thereby, an edge $E_{a,b}$ connects operators $OP_a$ and $OP_b$ only if the types $T$ of both operators are connected by edge $E_G$ in $DG$:

$$\forall E_{a,b} = (OP_a, OP_b) \in E_C : (T(OP_a), T(OP_b)) \in E_G \tag{2}$$

In contrast to earlier implementations, the restrictions on $DG$ are reduced, allowing the search space to be explored more efficiently (cf. [12]). We introduce a program $P'$ which is derived from $P$ and adjusted it as follows:

$$P' = \{F, T, G, n_r\} \tag{3}$$

$F$ represents the set of function nodes and $T$ the set of operator types. While $r$ represents the number of rows, the number of layers contained in graph $G$ reflects the number of columns $c$, therefore $P'$ is not directly dependent on it. The height of $G$ equals $c$ and its size, thereby: $c = |G| = height(G)$. Unlike earlier implementations (cf. [16]), the arity $a$ of each function varies, allowing to mutate each parameter of a function, therefore $a = max(\#Inputs(F_i))$. Each node $N_{i,j}$ belongs to a layer which an operator type $T_i$ is assigned to, therefore: $\forall N_{i,j}\, with\, i \in [0, n_r) : T(N_{i,j}) = T_i$
The image processing operators are divided into 6 groups as presented in Figure 1: *RegionToRegion, ImageAndRegionToRegion, EdgeAmpAndRegionToRegion, ImageToRegion, EdgeAmplitude* and *ImageToImage*. *InputImage* is mandatory to be appended to the end of the pipeline, therefore is not to be classified as an operator type in the strict sense. Operators representing the category *ImageAndRegionToRegion* would force the program to fail because filters of the type *ImageAndRegionToRegion* only accept tuples of image and region object. This applies to the HALCON library and equivalent implementations such as OpenCV[1]. CGP training returns a sequence of filters, arithmetic, or functions. This *pipeline* is

[1]cf. documentations of HALCON: (https://www.mvtec.com/doc/halcon/13/en/index.htm), and OpenCV (https://docs.opencv.org/)

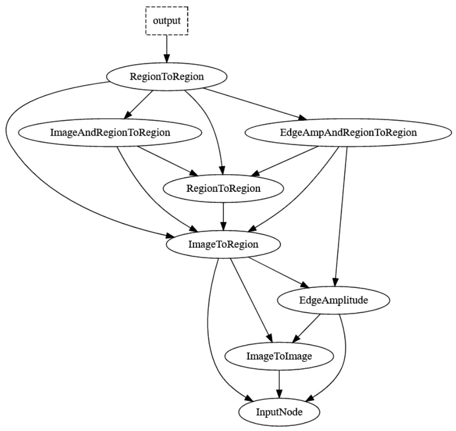

Fig. 1: Illustration of operator types and the dependencies serving as a constraint to the evolutionary process; all filter pipelines depend from the input image serving as root node.

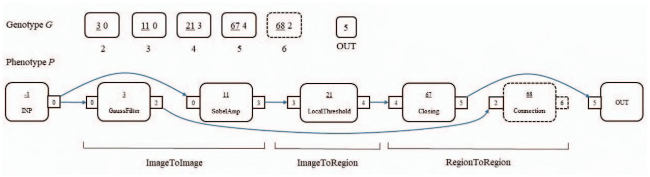

Fig. 2: Configuration of the CGP algorithm, employing image filter operators for image segmentation

represented by a DAG defining the data flow from input to output nodes. CGP applies a $(\mu + \lambda)$ ES with $\mu = 1$ and $\lambda = 4$ for combinatorial optimization problems with elitist selection for fitness rank-based individual selection in each cycle. The termination criterion is set at a maximum of 150 generations. By default, an input RGB image per pipeline is connected to one result image, ergo $i = 1$, $o = 1$. The grid is defined by $r = 10$, with $c = 7$ derived from the $DG$ (cf. Figure 1). The fitness is computed using Matthew's correlation coefficient (MCC) for the intersection between ground truth and prediction in each image-label pair and can be formally stated as follows:

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4}$$

We managed to expand our operator set to a total of 69 operators which are distributed as follows: 30 *ImageToImage*, 13 *ImageToRegion* and 16 *RegionToRegion*, 7 *ImageAndRegionToRegion*, 2 *EdgeAmplitude* and 1 *EdgeAmpAndRegionToRegion*, yielding a total number of $n_f = 69$.

Figure 2 illustrates a sample pipeline with assigned node types as generated for the *Tile_Crack* dataset. Due to $DG$, we only allow filter operators in the first instance connected to the image that can process an RGB image. Other configurations allow more inputs [7], [9] or use 3 separate channels [13]. All function nodes have been implemented using the HALCON

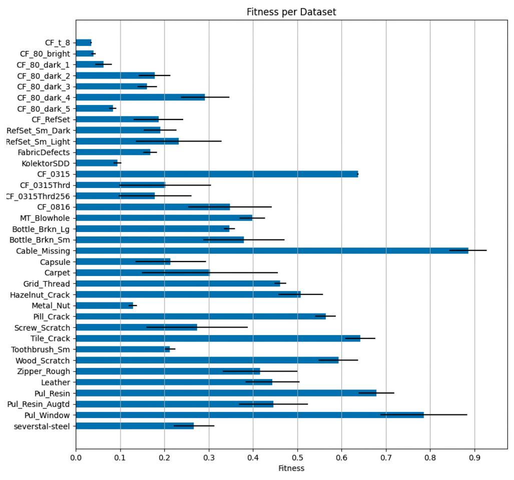

Fig. 3: Average $MCC$ fitness represented by blue bars and standard deviation (black) per dataset

framework. The function nodes depicted in Figure 2 form an active pipeline that comprises:

*SobelAmp(FilterType=y, MaskSize=7)*, *LocalThreshold(Method=adapted_std_deviation, MaskSize=31)*, *Closing(A=30, B=16, C=-0.393, StructElementType=Circle)* and *Dilation(Iterations=14, StructElement=Rectangle, A=4, B=3)*. The node connections *INP* $\rightarrow$ *GaussFilter(MaskSize=3)* $\rightarrow$ *Connection(Neighbourhood=8)* represent a common structure found in CGP genotypes. The graph shown in Figure 2 is a representative example of a solution generated from CGP training: The final node (*Connection*) is not connected to an output node making it a *non-coding gene*.

## C. Measures for Image Complexity

The complexity of an image reflects the amount of information contained in an image. Furthermore, it describes the processability of this information or parts of the image data and the level of difficulty associated with the segmentation task. Inspired by Mishra et al. [17], we define the following metrics to evaluate different aspects of image complexity: *Histogram Entropy (HE)*, *Rectangle Size (RS)*, *JPEG Complexity (JD)*, *Texture Features (TF)*, *Edge Density (ED)* and *Number of Superpixels (SP)*. The rectangle size (RS) is computed for the image frame (I) or the equivalent rectangular bounding box of the label: $RS(I) = H(I) \cdot W(I)$. The entropy of the histogram, denoted as $H(X)$, quantifies the amount of non-randomness in the distribution and is computed as: $H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$. According to [17], the level of compression achieved by the JPEG image compressor can implicitly determine the amount of information contained in an image. Mishra et al. define the JPEG-based complexity (JR) as the inverse of the compression ratio: $JR = \frac{size(image)}{size(compressed\_image)}$. As texture features (TF) we calculated the properties *contrast*, *homogeneity*, *correlation* and *energy* of a Grey Level Co-occurrence Matrix (GLCM) using *scikit-image*[2]. For the edge density (ED), we denote the length and width of a set of edges $E$ detected after applying the canny edge operator as $L$ and $W$, and the area of the image as $S(I) = H(I) \cdot W(I)$. The mathematical definition of edge density (ED) can be expressed as: $ED = |E|/S(I)$. To compute the number of superpixels (SP) with $S = \{s_1, s_2, ..., s_N\}$ of Image $I$ we minimize the following energy function: $E(S) = E_{data}(S) + \lambda \cdot E_{smooth}(S)$. Minimizing $E(S)$ iteratively allows determining the optimal set of superpixels $S$ for partitioning the image. In addition, we log the number of labels per frame (*label_count_per_image*) which potentially increases the difficulty of a segmentation task rendering it a noteworthy parameter.

## III. Empirical Investigation

This section describes the experimental algorithm configuration: By using a $(1+4)$ ES, 4 offspring individuals are derived from each parent to undergo mutation. For the experimental setup, $n_r$ is set to 10, while $c$ is derived from $DG$. For each node in $DG$, one column $c$ is generated. As the number of columns is quantified as $c = 7$, it yields a total of $n = 70$. This configuration is within the recommended range for $n$ [14].

### A. Experimental Setup

We conducted experiments on 36 datasets from various industrial monitoring applications, each representing a distinct segmentation problem. The evaluation is set to evaluate CGP performance on industrial datasets without directly comparing it to related segmentation models. While we acknowledge the potential of well-designed CNN architectures, they cannot be directly related to CGP due to differences in inference complexity and hardware requirements. Nonetheless, we encourage further research in this area.

### B. Datasets from Industrial Applications

All 36 reference sets from 6 publishers contain scan data from material surfaces of 12 different kinds. The images show objects on moving conveyor belts, continuous textile manufacturing (cf. fibers, films, or papers), and testing islands, some of which have been discussed by Margraf et al. [18].

The datasets used in this study include: *Electrical Commutator/KollektorSSD*, consisting of 400 microscopic images of defected electrical commutators captured in a controlled environment; *Carbon Fiber Processing*, representing industrial textiles with aligned filaments used in aircraft and wind energy parts; *Pultrusion Resin Injection*, comprising images of resin flow fronts used for impregnating carbon fibers; *Steel Welding*, containing images of flat steel surfaces for fault detection; *Magnetic Tiles*, with top-down views of magnetic tiles showcasing surface defects; *Fabric AITEX*, consisting of fabric images with pattern anomalies; and the *MVTec Anomaly Detection* dataset, which contains high-resolution images of industry-related objects and textures. Table I presents details on the datasets while Figure 4 displays image samples with segmentation results as overlays, indicating ground truth (green), false positives (red), and true positives (yellow).

[2]cf. https://scikit-image.org

| ID | Dataset Name | Material | Publisher | $\overline{MCC_i}$ | $\sigma_i$ | # Runs | # Images |
|---|---|---|---|---|---|---|---|
| 1 | CF_t_8 | Textile (CF) | Fraunhofer | 0.035 | 0.002 | 3 | 24 |
| 2 | CF_80_bright | Textile (CF) | Fraunhofer | 0.040 | 0.005 | 6 | 32 |
| 3 | CF_80_dark_1 | Textile (CF) | Fraunhofer | 0.063 | 0.019 | 8 | 16 |
| 4 | CF_80_dark_2 | Textile (CF) | Fraunhofer | 0.178 | 0.036 | 6 | 16 |
| 5 | CF_80_dark_3 | Textile (CF) | Fraunhofer | 0.161 | 0.022 | 9 | 16 |
| 6 | CF_80_dark_4 | Textile (CF) | Fraunhofer | 0.292 | 0.055 | 6 | 8 |
| 7 | CF_80_dark_5 | Textile (CF) | Fraunhofer | 0.084 | 0.008 | 6 | 16 |
| 8 | CF_RefSet | Textile (CF) | Fraunhofer | 0.187 | 0.056 | 12 | 34 |
| 9 | CF_RefSet_Sm_Dark | Textile (CF) | Fraunhofer | 0.191 | 0.037 | 5 | 6 |
| 10 | CF_RefSet_Sm_Light | Textile (CF) | Fraunhofer | 0.233 | 0.097 | 3 | 7 |
| 11 | FabricDefectsAITEX | Textile | AITEX | 0.168 | 0.016 | 7 | 20 |
| 12 | KolektorSDD | Electronics | Kolektor | 0.094 | 0.009 | 9 | 8 |
| 13 | CF_Sp0-0315 | Textile (CF) | Fraunhofer | 0.637 | 0.002 | 3 | 105 |
| 14 | CF_Sp0-0315Thrd | Textile (CF) | Fraunhofer | 0.202 | 0.104 | 6 | 29 |
| 15 | CF_Sp0-0315Thrd256 | Textile (CF) | Fraunhofer | 0.179 | 0.082 | 9 | 29 |
| 16 | CF_Sp2-0816 | Textile (CF) | Kolektor | 0.348 | 0.094 | 15 | 88 |
| 17 | MT_Blowhole_train | Metal | Chinese A.o.S. | 0.399 | 0.029 | 6 | 57 |
| 18 | Bottle_Brkn_Lg | Glas | MVTec | 0.347 | 0.012 | 3 | 10 |
| 19 | Bottle_Brkn_Sm | Glas | MVTec | 0.380 | 0.092 | 6 | 11 |
| **20** | **Cable_Missing** | **Electronics** | **MVTec** | **0.886** | **0.042** | **4** | **6** |
| 21 | Capsule | Plastic | MVTec | 0.214 | 0.080 | 3 | 10 |
| 22 | Carpet | Textile | MVTec | 0.303 | 0.154 | 6 | 10 |
| 23 | Grid_Thread | Metal | MVTec | 0.462 | 0.013 | 3 | 10 |
| 24 | Hazelnut_Crack | Food | MVTec | 0.508 | 0.050 | 3 | 9 |
| 25 | Metal_Nut | Metal | MVTec | 0.129 | 0.009 | 3 | 11 |
| 26 | Pill_Crack | Plastic | MVTec | 0.564 | 0.023 | 3 | 13 |
| 27 | Screw_Scratch | Metal | MVTec | 0.274 | 0.114 | 2 | 12 |
| 28 | Tile_Crack | Tile | MVTec | 0.643 | 0.034 | 8 | 8 |
| 29 | Toothbrush_Sm | Plastic | MVTec | 0.213 | 0.012 | 3 | 12 |
| 30 | Wood_Scratch | Wood | MVTec | 0.593 | 0.045 | 3 | 9 |
| 31 | Zipper_Rough | Clothing | MVTec | 0.416 | 0.084 | 3 | 8 |
| 32 | MVTec_D_Leather | Clothing | MVTec | 0.444 | 0.061 | 6 | 10 |
| 33 | Pul_Resin | Fluid Resin | Fraunhofer | 0.679 | 0.040 | 3 | 20 |
| 34 | Pul_Resin_Augtd | Fluid Resin | Fraunhofer | 0.446 | 0.078 | 3 | 20 |
| **35** | **Pul_Window** | **Fluid Resin** | **Fraunhofer** | **0.786** | **0.098** | **3** | **20** |
| 36 | severstal-steel | Metal | Severstal | 0.267 | 0.046 | 10 | 15 |

TABLE I: List of datasets with name, publisher, material surface, number of training images, mean $MCC_i$ fitness, standard deviation $\sigma_i$, CGP runs and number of images; highest values are bold, the lowest values are underlined.

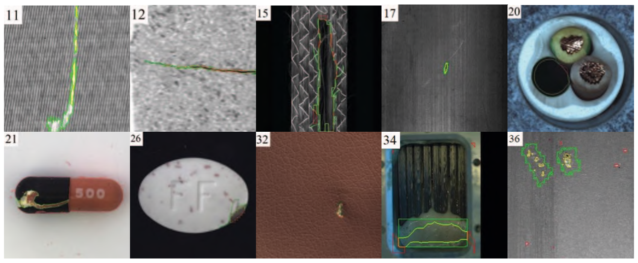

Fig. 4: Segmentation results evolved by means of CGP for different monitoring scenarios (cf. referenced ID); the colored overlays are encoded as follows: *red* represents false negatives, *yellow* all false positives, and *green* all true positives.

Although the majority of datasets we selected for experimentation have been collected for research purposes, 15 of the 36 use cases were collected at Fraunhofer's lab and had to be specifically prepared. The data was cleaned to avoid inconsistencies and effects of noise or overfitting. From the public datasets, only a smaller portion of images was used during training as this study is focussed on the training stage to examine the improvements on CGP generated pipelines.

## IV. Discussion of Results

This section analyzes the results of CGP experiments on industrial image datasets. We noticed that the fitness tends to decrease or remain constant for larger training sets. We make the assumption that datasets containing a high number of images hold a larger variety of features contained in the data. CGP, however, is designed to perform well on data with limited variance.

## A. Analysis of Runs with Different Datasets

As can be seen in Figure 3, the mean fitness values range from $MCC = 0.04$ for *CF_80_bright* and to $MCC = 0.886$ for *MVTec Cable Missing*. The datasets labeled *MVTec missing cable* and *Pultrusion Window* attained the highest fitness while the third-best were achieved for *pultrusion resin* and *pultrusion resin window*. The lowest $MCC$ values were returned for *CF_t_80_bright*, *CF_t_80_dark_5*, *KollektorSDD* and *CF_80_dark_1*. Upon visually examining the datasets depicted in Figure 4 it becomes apparent that the surface structures are less intricate and stand out from the background. This suggests that the CGP pipeline is better suited for structures with clear contrast and problems of lower complexity. While recent developments in AI-driven image segmentation relies heavily on CNN models, which perform well on large datasets, the authors of CGP-IP and its modifications are aware of this trend and have designed their approach to be complementary. CGP allows evolving effective pipelines on small datasets, therefore can quickly generate a solution with little computational resources. Also, simplicity keeps pipelines nearly human-readable. Since preprocessing pipelines address a narrow and specific objective, they are by nature determined to perform efficiently. Compact solutions allow improving the performance of existing complex models, which can lead to better results overall.

## B. Critical Reflection on Pipeline Performance

For better (pre-)selection of operators, we propose to analyze the segmentation task before filter pipelines are applied. Thereby, we examine how complexity and difficulty of a dataset and its annotation relate to the fitness achieved by CGP. The results suggest conducting a more detailed examination of the underlying dependency of data structure and *processability*. Visual examination suggets that pipelines yield high MCC values for structures of lower complexity (cf. Figure 5). We collected a number of high-level image descriptors and explained how they are linked to filter results.
Studies have shown that as the complexity of an image and its segmentation task increases, more complex models are required [**?**]. Our experiments suggest that for low complexity and less difficult segmentation tasks, CGP pipelines yield comparably high fitness levels. Therefore, we examined the correlation between the fitness achieved on the datasets. According to Table II, the *histogram entropy* of images and *edge_density* of labels indicate a moderate positive correlation. $JPEG\_complexity$ and $hist\_entropy$ achieve a weak but measurable correlation of $0.2 < \rho < 0.4$. Weak correlations appear for $num\_superpixels$ and $edge\_density$. Two metrics show essentially no correlation at $\rho \sim 0.0$: $edge\_density$ and $label\_count\_per\_image$. A moderate positive correlation between *histogram entropy*, *texture features* and the fitness achieved using CGP has been found. It is worth noting that

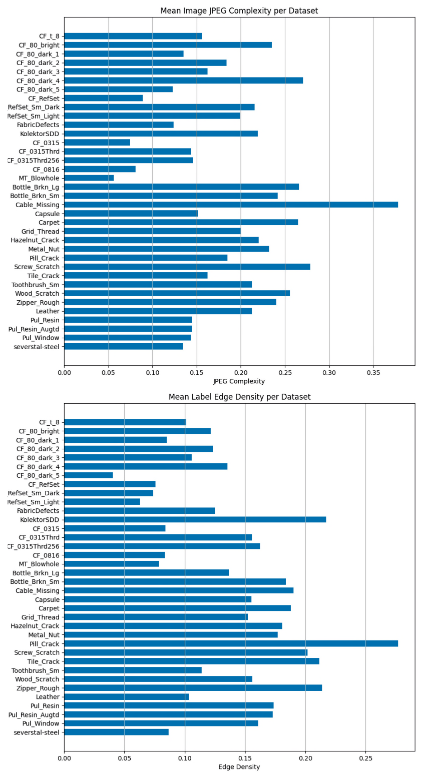

Fig. 5: Mean *JPEG Complexity* values of full image frames (top) and mean *edge density* computed from labels (bottom)

TABLE II: $\rho$ between complexity metrics $C_i$ and mean fitness $\overline{MCC}$ for dataset $i$ for image frames (left) and labels (right); highest values are bold, lowest values are underlined.

| **Metric** | $\rho_{\overline{MCC_i},C_i}$ |
|---|---|
| image_size | 0.174 |
| **hist_entropy** | **0.239** |
| **jpeg_complexity** | **0.248** |
| texture_features | 0.174 |
| edge_density | -0.059 |
| <u>num_superpixels</u> | <u>-0.090</u> |

| **Metric** | $\rho_{\overline{MCC_i},C_i}$ |
|---|---|
| label_count_per_image | -0.052 |
| label_size | -0.097 |
| <u>relative_label_size</u> | <u>-0.208</u> |
| lbl_hist_entropy | -0.116 |
| lbl_texture_features | 0.207 |
| **lbl_edge_density** | **0.277** |
| **lbl_num_superpixels** | **0.219** |

the obtained results are specific to the context of industrial surface monitoring.
In summary, the dependency graph avoids invalid candidates and eliminates exception handling, which yields early convergence and high fitness. The approach is helpful for highly-efficient filtering, image preprocessing and FPGA implementations. It allows for reproducible and high-volume data handling in real-time or high-resolution environments.

## V. Conclusion and Future Work

This study confirms the importance of data pre-processing in self-learning systems. Pipelines are set to be used within an OC system's observer and integrated in a knowledge base to incrementally build upon industrial filter problems. Imaging solutions rely on parameters such as brightness, contrast or surface characteristics, so the level of automation correlates with the quality of image filters used to extract information. The proposed concept can enhance the detection quality of imaging systems by obtaining more individualized configurations with less configuration effort. While data-driven ML models struggle with noisy data, CGP allows to automate parameter configuration for transparent, human-readable pipelines.

While industrial settings mostly have in common highly controlled production conditions, for the agricultural sector and crop production is but one example where such assumptions do not hold. For instance, crop production is characterized by high degrees of uncertainty regarding both the *objects* to be produced as well as the *environment* [19]. Agricultural production is also dependent on weather and micro-climatic conditions. Climate change substantially increases this environmental uncertainty leading to higher degrees of variability. Modern agricultural technology is increasingly equipped with sensors, actuators and computational units to process data in situ during the operations, which is referred to as *smart farming* or *digital farming*. Image analysis is important here due to its non-destructive way of sensing. As mentioned, alternative algorithmic approaches allow a) to *quickly reconfigure and adapt* to changing conditions reflected in the images, and, b) *executing highly efficient* image processing on specialized hardware on computationally limited machines essentially used in future intelligent agricultural technology. Examples of recent research on image segmentation in agricultural production focus e.g., on segmenting vegetables from ground for informing yield predictions [20] and novel ways for more efficient training of the underlying deep learning models [21], [22]. Further examples of image analysis applications are targeted to the pixel-wise classification of remaining biomass and soil cover from ground images [23] or weed detection [24]. Since this study does not compare the CGP approach to state-of-the-art CNN models, future work should be continued in this direction using metrics beyond accuracy, e.g., complexity and efficiency metrics. While CNN models are known for generalizing better with increasing complexity, CGP aims to provide efficient and transparent programs that interact with other ML models. Finally, we encourage investigating applications beyond controlled industrial monitoring, as in sports, food or environmental monitoring.

## ACKNOWLEDGEMENTS

The authors like to thank the *German Federal Ministry for Economic Affairs and Climate Action (BMWK)* for funding *AirCarbon III* within Lufo-IV and the *German Federal Ministry of Education and Research (BMBF)* for supporting *SaMoA* within VIP+.

## REFERENCES

[1] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif, and U. Richter, "Adaptivity and self-organization in organic computing systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 5, no. 3, pp. 1–32, sep 2010.

[2] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Proc. of the Third European Conference on Genetic Programming (EuroGP2000).*, vol. 1802, 2000, pp. 121–132.

[3] J. F. Miller, "Cartesian genetic programming: its status and future," *Genetic Programming and Evolvable Machines*, pp. 1–40, 2019.

[4] Z. Vasicek and L. Sekanina, "Evaluation of a new platform for image filter evolution," in *NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007)*. IEEE, 2007, 0-7695-2866-X.

[5] S. L. Harding, "Evolution of image filters on graphics processor units using cartesian genetic programming," in *IEEE Congress on Evolutionary Computation (CEC 2008)*, 2008, 978-1-4244-1823-7.

[6] T. Kowaliw, W. Banzhaf, N. Kharma, and S. Harding, "Evolving novel image features using genetic programming-based image transforms," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, May 2009.

[7] J. Leitner, P. Chandrashekhariah, S. Harding, M. Frank, G. Spina, A. Förster, J. Triesch, and J. Schmidhuber, "Autonomous learning of robust visual object detection and identification on a humanoid," 11 2012.

[8] L. Sekanina, S. L. Harding, W. Banzhaf, and T. Kowaliw, *Cartesian genetic programming*, ser. Natural Computing Series. Springer Science & Business Media, 2011, ch. Image Processing and CGP, pp. 181–215.

[9] S. Harding, J. Leitner, and J. Schmidhuber, *Genetic programming theory and practice x*. Kluwer Academic Publishers, 2013, ch. Cartesian Genetic Programming for Image Processing, pp. 1–17.

[10] B. W. Goldman and W. F. Punch, "Reducing wasted evaluations in cartesian genetic programming," in *European Conference on Genetic Programming*. Springer, 2013, pp. 61–72.

[11] A. Margraf, A. Stein, L. Engstler, S. Geinitz, and J. Hähner, "An evolutionary learning approach to self-configuring image pipelines in the context of carbon fiber fault detection," in *Proc. of 16th IEEE International Conference for Machine Learning and Applications*, 2017.

[12] A. Margraf, S. Geinitz, A. Wedel, and L. Engstler, "Detection of surface defects on carbon fiber rovings using line sensors and image processing algorithms," in *Proc. of the SAMPE Europe Conference 2017 Stuttgart*, 2017.

[13] K. Cortacero, B. McKenzie, S. Müller, R. Khazen, F. Lafouresse, G. Corsaut, N. V. Acker, F.-X. Frenois, L. Lamant, N. Meyer, B. Vergier, D. G. Wilson, H. Luga, O. Staufer, M. L. Dustin, S. Valitutti, and S. Cussat-Blanc, "Kartezio: Evolutionary design of explainable pipelines for biomedical image analysis."

[14] A. J. Turner and J. F. Miller, *Genetic programming and evolvable machines*. Springer Science and Business Media, May 2015, vol. 16, no. 4, ch. Neutral genetic drift: an investigation using Cartesian Genetic Programming, pp. 531–558.

[15] J. F. Miller, *Cartesian Genetic Programming*, ser. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 2011, ch. Cartesian Genetic Programming, pp. 17–34.

[16] J. Leitner, S. L. Harding, and A. Förster, "Humanoid learns to detect its own hands," Jun. 2013.

[17] S. Mishra, D. Z. Chen, and X. S. Hu, "Image complexity guided network compression for biomedical image segmentation," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 2, pp. 1–23, dec 2021.

[18] A. Margraf, H. Cui, S. Heimbach, J. Hähner, and S. Geinitz, "Model-driven optimisation of monitoring system configurations for batch production," in *Proc. of MODELSWARD 2023*, in press 2023.

[19] A. Bechar and C. Vigneault, "Agricultural robots for field operations: Concepts and components," *Biosystems Engineering*, vol. 149, pp. 94–111, 2016.

[20] N. Lüling, D. Reiser, J. Straub, A. Stana, and H. W. Griepentrog, "Fruit volume and leaf-area determination of cabbage by a neural-network-based instance segmentation for different growth stages," *Sensors*, vol. 23, no. 1, 2023.

[21] J. Boysen and A. Stein, "AI-supported data annotation in the context of uav-based weed detection in sugar beet fields using deep neural networks." in *GIL Jahrestagung*, 2022, pp. 63–68.

[22] N. Lüling, J. Boysen, H. Kuper, and A. Stein, "A context aware and self-improving monitoring system for field vegetables," in *35th Int. Conf. on Architecture of Computing Systems*. Springer, September 2022, pp. 226–240.

[23] P. Riegler-Nurscher, J. Prankl, T. Bauer, P. Strauss, and H. Prankl, "A machine learning approach for pixel wise classification of residue and vegetation cover under field conditions," *Biosystems Engineering*, vol. 169, pp. 188–198, 2018.

[24] N. Genze, R. Ajekwe, Z. Güreli, F. Haselbeck, M. Grieb, and D. G. Grimm, "Deep learning-based early weed segmentation using motion blurred uav images of sorghum fields," *Computers and Electronics in Agriculture*, vol. 202, p. 107388, 2022.