A Data-driven Approach for General Visual Quality Control in a Robotic Workcell

IOP Conf. Series: Journal of Physics: Conf. Series **1335** (2019) 012013 doi:10.1088/1742-6596/1335/1/012013

A Data-driven Approach for General Visual Quality Control in a Robotic Workcell

Simon Reich[1,a], Florian Teich[1,b], Minija Tamosiunaite[1,c], Florentin Wörgötter[1,d], and Tatyana Ivanovska[1,e]

[1] Third Institute of Physics - Biophysics, Georg-August-Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

E-mail: [a]simon.reich@phys.uni-goettingen.de, [b]florian.teich@phys.uni-goettingen.de, [c]minija.tamosiunaite@phys.uni-goettingen.de, [d]florentin.woergoetter@phys.uni-goettingen.de, [e]tiva@phys.uni-goettingen.de

Abstract. Setting up computer vision quality control tasks in a robot workcell is expensive, time consuming, and often requires expert knowledge. In this work, a highly adaptable approach to mitigate this issue is introduced. First, an ontology of atomic building blocks is defined, where each block represents one computer vision algorithm. Second, these blocks are used to generate a complex graph structure. The generation follows a set of rules, which can be shown to the inexperienced worker as a list of simple questions. Afterwards, this graph can be refined. Finally, the graph is integrated into the robotic assembly sequence, and the visual quality control procedure can be executed. Two industrial use cases show the feasibility of this approach. Thus, this work leverages computer vision aided quality control tasks in robot workcells.

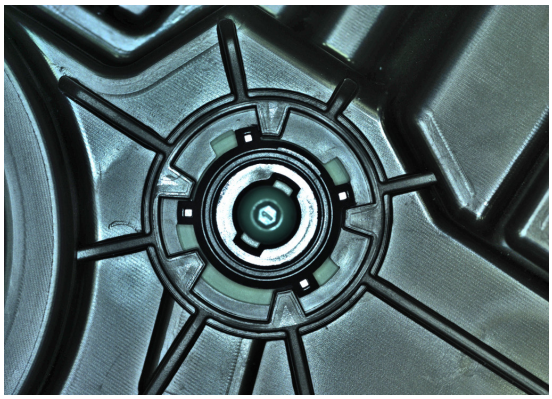
1. Introduction

Even in today's world real time computer vision remains a challenging task; This is especially true in industrial use cases. Here, a set of problems arises, which is usually not the scope of traditional robotics. For example, there are far from optimal environment conditions — meaning one deals with issues stemming from dust, noise, and ambient light. At the same time, the algorithms must perform with low false-positive and false-negative rates. But most important, fast setup times of the robotic hardware and software play an important role.

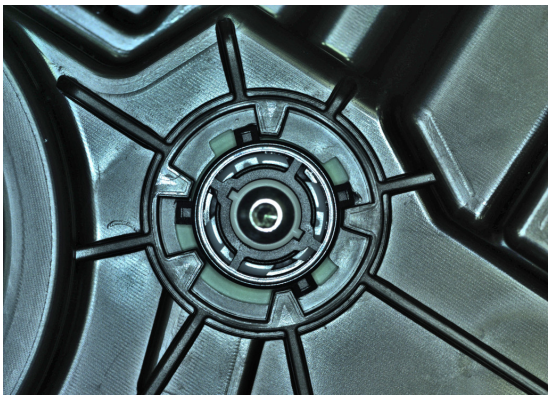
Next to the sensor problems mentioned above, the measurement domain of industrial use cases is large. While one company might want to measure the height of a screw, another might want to know, if a specific detail is present. A third company might want to perform optical character recognition on a specific image part. This leads to a large number of use cases. As current state-of-the art computer vision does not generalize well, this produces a large number of highly specific algorithms. They usually are able to solve one quality control task very well, but are unable to perform other measurement operations. An example from our industrial partners [1] is shown in figure 1.

There are some approaches to solve these issues for small or medium-size enterprises (SMEs) and few-of-a-kind productions. For example, in [2][3] a reconfigurable workcell is presented. The robot may grasp different tools for different steps of the production and is thus able to reconfigure itself. As programming is done via a graphical user interface (GUI), an inexperienced worker is

able to program the workcell. In [4] a concept of robot configuration optimization for workcells is introduced. Here, a simulation estimates the perfect configuration for each processing step. A concise overview on the state-of-the art of reconfigurable workcell robots is shown in [5][6].



(a) LWR drive type 1.



(b) LWR drive type 2.

Figure 1: Two sample RGB images with different LWR drive types (image courtesy of [1]); A LWR consists of a stepper motor, which regulates the height of an automotive's headlight, and the motor's casing. The quality control task asked is: "Is the motor inserted and rotated correctly?" and in traditional computer vision this does not generalize well from one drive type to another.

However, none of these approaches currently implement a modular and general way of quality control using computer vision: A non-expert user must be able to set up quality control tasks. This means, a graphical user interface must be available and source code programming must be avoided at all costs. There are several commercial software packages to develop solutions to machine vision tasks available (such as Halcon [7], Merlic [8], and Cognex VisionPro [9]). The main focus of these software packages is to provide the user with a set of tools to tackle computer vision problems in a generic way that can be automated. They often offer thousands of methods and operators to create pipelines for solving computer vision applications such as measuring, counting or checking for defects of objects inside given images.

For example, Halcon is a 3D machine vision library that is developed by MVTec and provides an IDE for programmers to create solutions for machine vision tasks such as position recognition or object identification [7][10]. However, as the IDE only supports the use of programming languages (as e.g. C, C++, C#), the software requires knowledge that the average inexperienced worker cannot meet.

Merlic, another product from MVTec, tackles this issue by introducing a GUI that enables inexperienced users and requires no programming skills [8]. This is implemented by visualizing the computer vision methods as blocks, which can be connected to each other and arranged to form a generic pipeline to solve the computer vision task at hand. This solution is far more attractive in scenarios where inexperienced workers have to design computer vision applications without the support of a programmer. This advantage, however, comes along with high license fees.

Another machine vision solution is Cognex VisionPro [9]. Similar to Halcon and Merlic, VisionPro offers methods as Optical Character Recognition (OCR), circle detection or blob analysis and incorporates tools to use peripheral hardware as a camera in the pipeline.

The application of commercial approaches often remain not straightforward though. First, these software solutions are expensive, which often is a key factor for small companies. Second,

these packages do not connect well with products from other companies. For instance, it is rather non-trivial to connect the commercial computer vision packages with the Robot Operating System (ROS) [11]. As alternative, many open source computer vision libraries exist, e. g. [12][13][14][15][16]. They offer a unified framework of handling computer vision tasks within each library, but rarely export well to other libraries. Furthermore, while offering great assistance to an experienced programmer, an inexperienced worker is hardly able to program computer vision tasks based on these libraries alone in an efficient manner. Thus, a framework is needed, which bundles these libraries in a coherent manner and allows to structure their methods based on input, which can be given by an inexperienced worker.

The contribution of this paper is as follows: First, an ontology of atomic computer vision building blocks is presented. The blocks are the smallest algorithmic entities used in this work, where each block has a predefined set of pre- and postconditions. Second, these blocks are used in a complex graph structure where each block is one node. The final result of one such graph is a highly adapted algorithm to one computer vision use case. Such an approach is rather flexible, since the algorithm can be easily adjusted for a new case by replacing some blocks with other ones, or by selecting new parameter values. This enables the pipeline presented here to produce highly adaptable computer vision algorithms given only a small amount of user input data. Finally, the pipeline proposed here is open source and can be adapted by experienced end-users and fully integrated into the company's processing flow.

This paper is structured as follows. In section 2, the used methods are presented: a thorough overview of the presented framework is followed by a detailed explanation of each step. Results are presented and discussed in detail in section 3. Section 4 concludes the paper and outlines the future work directions.

2. Methods

An illustrative example from the automotive industry is presented in figure 1. The assembly of the automotive light includes insertion of the LWR drive into the light housing. The quality control task consists of the check if the detail is inserted and rotated correctly and if no damages on the housing appeared. There are several generations of the LWR drive types, which have different appearance, but the same functionality are used interchangeably. The computer vision checking routine has to be partially rewritten, readjusted, and tested for each new LWR type. The resulting algorithms can be written implicitly in the source code or explicitly using a specialized software package (such as Merlic). The source code part can be written only by advanced workcell users and developers, i. e., the ones that have deep understanding of computer vision and can write and integrate the algorithms with the whole framework of the robotic cell.

This way, customized algorithms for certain quality control problems, for instance, for the automotive industry [17] can be designed and built into the general framework. The advantage of such a solution is that it can be highly optimized for a certain problem in terms of computational efficiency and general accuracy. However, this approach obviously lacks flexibility. For instance, if the imaging conditions, such as lighting, background, or distance between the camera and the object, or the detail appearance, change, the algorithms have to be readjusted and the complete framework must be recompiled and retested over again.

To assess the problems of visual quality control in SMEs, we propose a strategy shown in figure 2. The solutions for computer vision problems usually consist of multiple algorithmic steps. Traditionally, as shown in figure 2 on the left side, these steps are hard coded and fused by a programmer, which results in a highly specific quality control task. A more flexible alternative is to split the customized pipelines into smaller blocks, where each item represents one algorithm, and to build a graph of connected atomic blocks (see figure 2, right side). They are the smallest entity of algorithmic steps used here. These blocks have a predefined set of preconditions and postconditions and can therefore be used in a modular and structured way.

We developed a wizard, which guides an inexperienced user through a predefined set of questions and thus creates the initial graph. Afterwards, the graph can be extended.

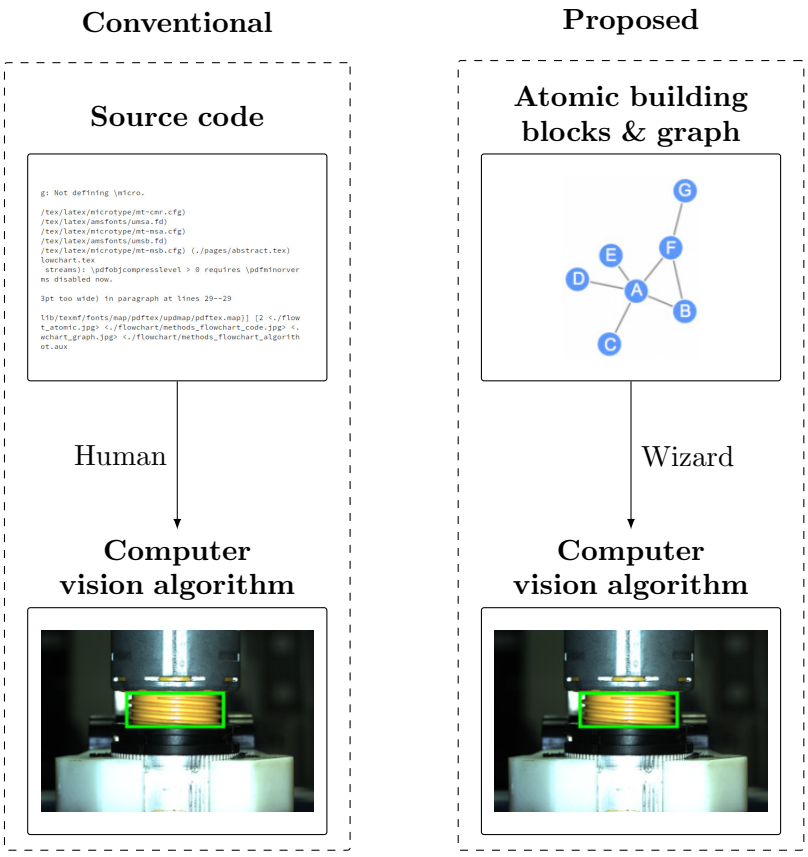


Figure 2: Flowchart of the methods in this chapter and how they relate. In the traditional way, a skilled programmer creates a computer vision algorithm by hard coding the algorithms (left side). This does not generalize and results in the solution of one specific measurement problem. In this novel approach, we propose a GUI, which places algorithmic steps in blocks (right side). These blocks can be concatenated to a graph, which results in an easily adaptable solution to different computer vision problems. Details are explained in section 2.

The list of blocks that we consider the most relevant is given in table 1. Of course, this list is non-exhaustive and can be further extended. As an alternative to the commercial packages, we propose an open source system for creating the graphs and executing the monitoring routines that is implemented as a ROS package [11] (i. e. integrated with the ROS framework in a straightforward way), and based on the data flow library [18] and the Open Source Computer Vision Library (OpenCV) [12].

The system consists of the following components. First, to help a completely inexperienced user, we developed a wizard, which guides the user over a predefined set of steps and builds the algorithmic graph in the background. Second, the diagram editor allows for building algorithmic pipelines from atomic blocks (see figure 3).

The diagram can either be built from scratch or the one from the wizard steps is utilized and further extended. The user can check the result of each pipeline step. When saved, the diagram is automatically integrated for monitoring. Third, the pipeline can be tested offline. Finally, a ROS service for actual quality control is provided.

Table 1: List of atomic building blocks developed in this work.

Name	Precondition	Postcondition	Description
Template Matching	Input and template images.	Found region of interest (ROI), which is similar to the template, and its location in the input image	Finds a region in a camera frame.
Region Cropping	Input image, input ROI location, Operation	Extracted ROI and its location	Extracts a region with respect to the input ROI location and the operation (e.g., below the input ROI)
Image Denoising	Input color or grayscale image, denoising algorithm with parameters	Denoised image	Applies a denoising algorithm with the given parameters to the input image.
Image Thresholding	Input image, thresholding algorithm	Black-and-white (BW) image with found regions	Applies a thresholding algorithm to an image. The algorithm can be automatic (e. g. Otsu's thresholding [19]) or manual, global or local.
Region Analysis	Black-and-white image and region selection rule	Selected regions	Applies connected component (CC) analysis and filters the CCs according to the defined rule
Image Morphology	Black-and-white image or grayscale, morphological kernel, and operator	Black-and-white or grayscale image	Applies the selected morphological operator (e. g. erosion, dilation, opening, closing) with the given kernel to the image.
Shape Detection	Grayscale or BW image, target shape properties	Found shapes and their locations	Applies Hough Transformation [20] to detect shapes.
Image Arithmetic	Two images and an operation	Resulting image	A binary operation (such as AND, OR, XOR) is applied to two images.
Classifier	Training examples, classifier parameters; Test image	Trained classifier; Prediction on the test image	In case of binary classification, positive and negative examples must be provided. The classifier is trained and can be applied to further test images.

The procedure of monitoring is shown in figure 4. Upon the request from the ROS system, an image is acquired from the camera and passed to the computer vision routine. Based on the results, the robot can continue the assembly. To allow the user to visually assess the monitoring process, we have also developed a visualization service, which communicates with the monitoring routine.

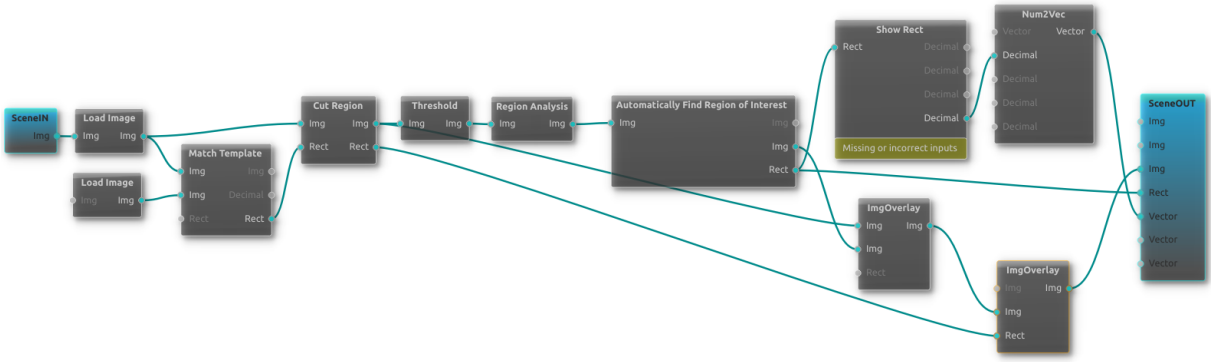


Figure 3: Graph view of the algorithm diagram editor. Each block contains a "configure" dialogue, which allows setting properties of the block.

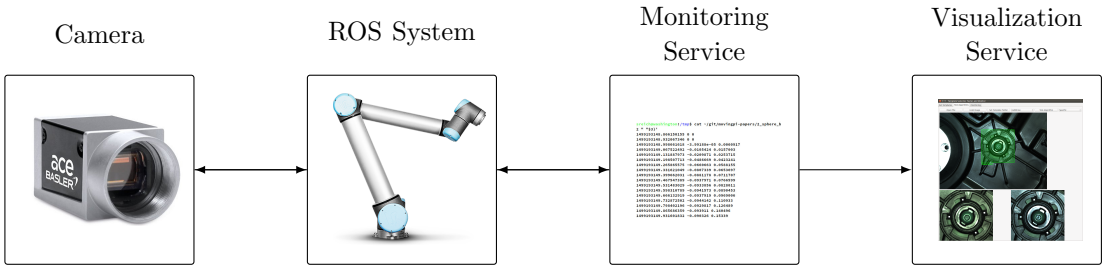


Figure 4: A schematic illustration of the monitoring service provided by our framework. The ROS system requests a new image from the camera and transfers it to the monitoring service. This service performs the quality control task based on the graph structure. A visualization service outputs each step.

These tools allow easy contribution of new blocks. Each block is agnostic of included computer vision libraries and thus an expert user can solve measurement problems at hand with any tool at disposal. The programmer only needs to handle the block's in- and outputs. Integration into the ROS system is performed via a wrapper; This means ROS can be accessed if needed.

3. Experiments

As a qualitative demonstration of the atomic block concept, the following subsections will each state a typical industrial use case, followed by the design of a generic pipeline consisting of such blocks to solve the task at hand.

3.1. Use Case 1: Measuring a Yellow Stator Gap

In this scenario, the assembly of a part of electronically adjustable furniture by LOGICDATA GmbH [21] is considered. The quality control task is to determine if the size of a detail corresponds to the specifications. It is determined by the adjustable yellow part (see figure 5), which must be measured for further assembly steps.

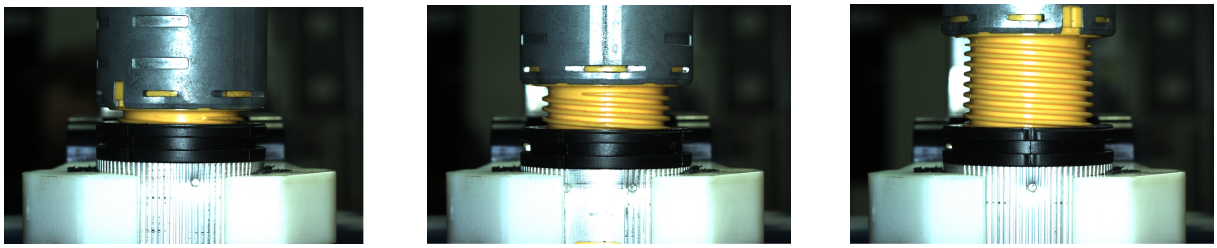


Figure 5: Three sample images with different stator sizes (image courtesy of [21]).

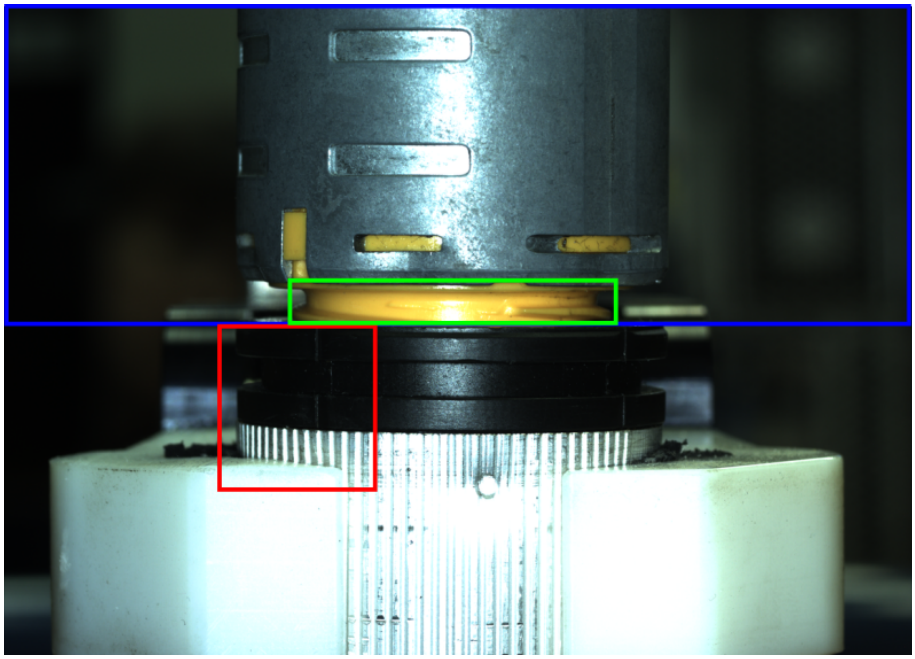


Figure 6: Processed example image from figure 5. Red box: Template bounding box; blue box: Cut-region bounding box; green box: Located stator bounding box.

The processing procedure can be split into the following computational steps. First, a template region can be detected in the image. We consider a non-changing region, which is present in every image, to be acceptable for this. In this case, the region of interest (ROI), i. e. the yellow stator part can not be utilized as a template, since its change is basically the detection goal (see figure 5). The template region is marked red in figure 6. The position of ROI relative to the found template is known in advance. Therefore, we reduce the search region by cropping the initial image, as it is demonstrated in figure 6 by the blue line. Thereafter, a color thresholding and region analysis steps can be applied. The bounding box of the biggest region is the measurement result, which is marked green in figure 6. The full graph is shown in figure 7.

3.2. Use Case 2: Detecting Damage of a Housing Part

In this use case, an assembly process of the automotive light is considered. The process is executed by the Company ELVEZ d.o.o. [1], a manufacturer of specialized products for automotive industry, electrical and mechanical engineering. Here, it must be assessed by computer vision methods, if the detail is damaged or not. Several examples are shown in figure 8. The part is considered defective, if significant portions of it are scratched or missing.

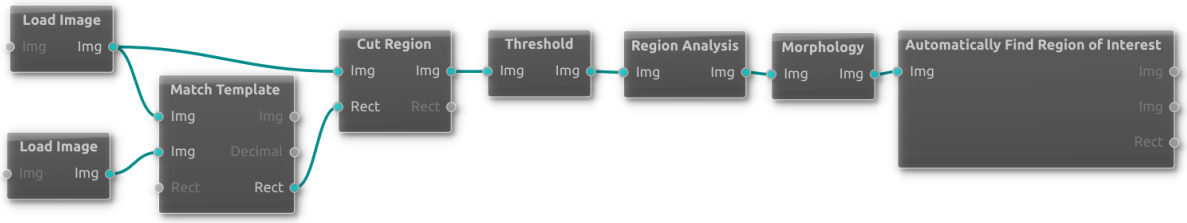


Figure 7: Underlying graph to solve the use case 1 "Yellow stator". Individual blocks represent the algorithms mentioned in table 1 and are linked by connections creating the resulting pipeline.

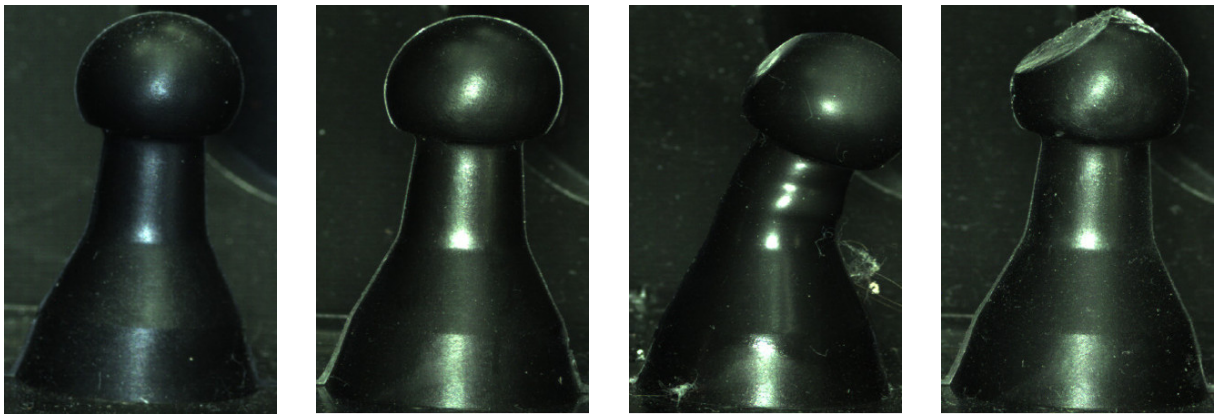


Figure 8: Four sample images (two positive and two negative ones) of the light housing part.

The algorithmic pipeline is as follows: To locate the detail itself, template-matching is applied. Afterwards, a binary classifier is applied, which analyzes whether the part has a defect. This classification can be implemented by a Support Vector Machine (SVM). For the SVM, a set of training images of both classes is required. The entire pipeline is visualized in figure 9.

4. Conclusions and future work

In this work, a computer vision pipeline is described, which offers a generalization of computer vision algorithms. The pipeline consists of three parts: First, atomic building blocks are introduced. Each block represents the smallest entity of one computer vision algorithm. Second, these blocks are placed into a complex graph via a set of rules. The rule's input is generated from questions, which can be answered by an untrained worker. Third, the graph can be further refined. Its output represents a highly specialized computer vision algorithm.

Currently, several parts of this remain as ongoing research topics. In this work, it is shown that the graph is created according to a limited set of questions. Of course, one can ask, whether the graph can be created using machine learning methods. However, the graph's state space, made up of *all* combinations of building blocks and their parameters, is simply too large, even for a small number of blocks and parameters.

In [22] a new method named Evolution Constructed Object Recognition (ECO) is developed, which is able to cope with such problems. It is significantly refined in [23]: First, a genetic algorithm composes new classifiers using crossover and mutation. Their output is passed to a Random Forest, which leads to a strong classifier making a decision. In this approach, the inexperienced worker would select the requested task from a predefined set. For instance, the user would select the following task: "measure the height of a screw".

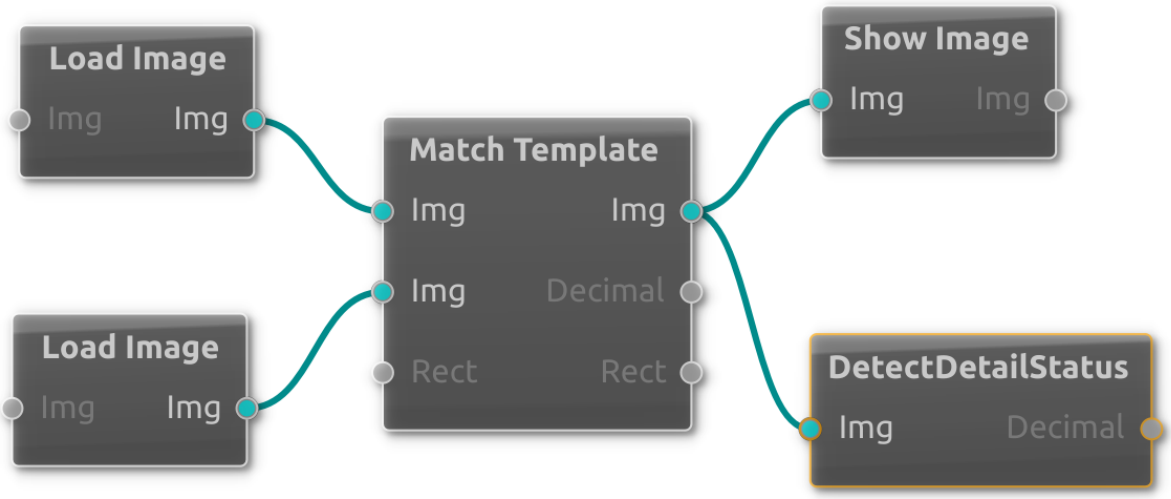


Figure 9: Underlying graph to solve use case 2 "Detail status". Individual blocks represent the algorithms mentioned in table 1 and are linked by connections creating the resulting pipeline.

This input would be used as fitness function for the first stage. The genetic algorithm tries different permutations of the atomic building blocks, such as edge detection, thresholding, measure linear dimensions etc. The final image is handed to one strong classifier, which is trained to measure object lengths. Permutations are tried until a reasonable result is obtained.

Lastly, deep learning algorithms seem to offer a good way to generalize computer vision. Currently, it is analyzed, if deep learning methods can be used to generalize measurement tasks. Here, the algorithm extracts its own set of features, which it deems important for the task, and uses these to perform the task.

Acknowledgement
The research leading to these results has received funding from the European Communitys Horizon 2020 Programme under grant agreement no. 680431, ReconCell (A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs).

References

[1] Elvez d.o.o. Elvez. `http://www.elvez.si/`, 2019. [Online; accessed 24-January-2019].
[2] Timotej Gaspar, Barry Ridge, Robert Bevec, Martin Bem, Igor Kovač, Aleš Ude, and Žiga Gosar. Rapid hardware and software reconfiguration in a robotic workcell. In *18th International Conference on Advanced Robotics (ICAR)*, pages 229–236. IEEE, 2017.
[3] Martin Bem, Miha Deniša, Timotej Gašpar, Jaka Jereb, Robert Bevec, Igor Kovač, and Aleš Ude. Reconfigurable fixture evaluation for use in automotive light assembly. In *18th International Conference on Advanced Robotics (ICAR)*, pages 61–67. IEEE, 2017.
[4] I-Ming. Chen. A rapidly reconfigurable robotics workcell and its applictions for tissue engineering. innovation in manufacturing science and technology program. In *Singapore-MIT Alliance Annual Symposium*. MIT Alliance, Singapore, 2003.
[5] ZM Bi, Sherman YT Lang, M Verner, and P Orban. Development of reconfigurable machines. *The International Journal of Advanced Manufacturing Technology*, 39(11-12):1227–1251, 2008.
[6] Mircea Fulea, Sorin Popescu, Emilia Brad, Bogdan Mocan, and Mircea Murar. A literature survey on reconfigurable industrial robotic work cells. *Applied Mechanics & Materials*, 762, 2015.
[7] MVTec Software GmbH. HALCON - The power of machine vision. `https://www.mvtec.com/products/halcon/`, 2019. [Online; accessed 24-January-2019].

[8] MVTec Software GmbH. MERLIC - A New Generation of Machine Vision. `https://www.mvtec.com/products/merlic/`, 2019. [Online; accessed 24-January-2019].
[9] Cognex Corporation. Cognex VisionPro. `https://www.cognex.com/products/machine-vision/vision-software/visionpro`, 2019. [Online; accessed 24-January-2019].
[10] Wolfgang Eckstein and Carsten Steger. The halcon vision system: an example for flexible software architecture. In *Proceedings of 3rd Japanese Conference on Practical Applications of Real-Time Image Processing*, pages 18–23. Technical Committe of Image Processing Applications, Japanese Society for Precision Engineering, 1999.
[11] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
[12] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.
[13] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 5 2011.
[14] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.
[15] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM.
[16] Yusuke Niitani, Toru Ogawa, Shunta Saito, and Masaki Saito. Chainercv: a library for deep learning in computer vision. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1217–1220. ACM, 2017.
[17] Tatyana Ivanovska, Simon Reich, Robert Bevec, Ziga Gosar, Minija Tamosiunaite, Ales Ude, and Florentin Wörgötter. Visual inspection and error detection in a reconfigurable robot workcell: An automotive light assembly example. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP): Visapp (VCEA)*, volume 5, pages 607–615. INSTICC, SciTePress, 1 2018.
[18] Dmitry Pinaev. Qt5 node editor. `https://github.com/paceholder/nodeeditor`, 2017.
[19] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.
[20] Paul VC Hough. Method and means for recognizing complex patterns, 12 1962. US Patent 3,069,654.
[21] LOGICDATA Electronic & Software Entwicklungs GmbH. LOGICDATA. `http://www.logicdata.net/`, 2019. [Online; accessed 24-January-2019].
[22] Kirt Lillywhite, Dah-Jye Lee, Beau Tippetts, and James Archibald. A feature construction method for general object recognition. *Pattern Recognition*, 46(12):3300 – 3314, 2013.
[23] Muhammad H. Zayyan, Mohammed F. AlRahmawy, and Samir Elmougy. A new framework to enhance evolution-constructed object recognition method. *Ain Shams Engineering Journal*, 2017.