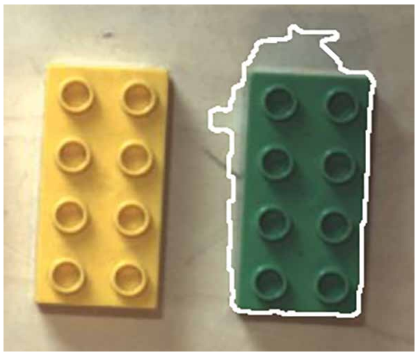
BUILDING SMART VISION BLOCKS

In manufacturing environments where collaborative robots are employed, conventional computer vision algorithms have trouble in the robust localisation and detection of products due to changing illumination conditions and shadows caused by a human sharing the workspace with the robotic system. In order to enhance the robustness of vision applications, machine learning with neural networks is explored. The performance of machine-learning algorithms versus conventional computer vision algorithms is studied by observing a generic user scenario for the manufacturing process: the assembly of a product by localisation, identification and manipulation of building blocks.

ROY DE KINKELDER AND BRIAN ARMSTRONG



Introduction

High-tech production factories in north-western Europe are characterised by high-mix, low-volume production. Assembly is becoming increasingly challenging due to market dynamics. Therefore, production automation, flexibilisation and optimisation are essential in the trend towards manufacturing smaller batches, while retaining the capability to deliver a large variety of products. Collaborative robotics is an essential element in this trend, with vision systems as an important factor that facilitates flexibility [1].

Vision systems are used in pick & place applications, quality checks, product localisations, flow monitoring, etc. Although vision-controlled robotics has shown great benefit regarding efficiency and yield, it has major disadvantages when changes in the production process arise; vision systems in particular are sensitive to unpredictable environmental changes.

The robustness of vision systems using conventional computer vision algorithms often deteriorates due to (minor) changes in product shape and colour, lighting conditions such as the influence of sunlight, relocation of the production system on the shop floor, and in cases of processing biological products and food, etc. In other words, the robustness of the vision systems depends on the production environment. Moreover, in the case of collaborative robotics, where humans share the same workspace, the illumination changes continuously, as a result of unwanted shadowing (by passing operators) in the camera field of view.

In these cases, the system fails to recognise the object and a computer vision specialist must evaluate the new situation, recalibrate the system and re-program the software. Machine learning using neural networks has the potential to overcome a considerable number of these problems as they have been proven to be considerably less sensitive to varying environments and lighting conditions.

The Saxion research group Mechatronics and the companies Benchmark Electronics, which specialises in electronics manufacturing, and Bronkhorst High-Tech, which specialises in mass-flow meters, are exploring the use of collaborative robots in their production process in the TechForFuture (TFF) RoboTAO project. The focus of the research is on the real collaboration instead of sequential task deployment. A vision system is used to recognise human handling of the product and the intended operator's interference in the production process. For the purpose of enhancing its robustness, machine learning with neural networks is explored more thoroughly.

Collaborative robot scenario

The regular assembly of a product comprising several building blocks (product housing, connectors and PCB boards) is represented by the assembly process of a Duplo® (Lego® group, Denmark) house at different levels of human-

AUTHORS' NOTE

Roy de Kinkelder is a project leader/researcher at the applied research group Mechatronics of the Saxion University of Applied Sciences in Enschede, the Netherlands. His research focuses on machine vision applications for robotics and industrial automation and he is responsible for the management of the TFF RoboTAO project. Brian Armstrong is a student researcher in the same group. He studies Mechatronics and focuses on computer vision and machine learning in industrial applications.

r.dekinkelder@saxion.nl
www.saxion.nl/mechatronics

1

Set-up with camera in the top and illumination of the Duplo blocks.

machine interaction. First, the cobot recognises human interference in the production process, and continue where the human stops. Later, the cobot recognises a human sharing the same workspace with a hand detection algorithm. Finally, the cobot interacts with the human to assemble the Duplo house by sharing the blocks. Currently, the project partners are working on the first stage of the human-machine interaction.

To compare conventional computer vision algorithms with machine learning, a user scenario was defined, in which four types of Duplo blocks have to be identified and localised in order to pick them up. The 4 types were distinguished by colour: red, green, blue and yellow. The set-up is shown in Figure 1.

The blocks were randomly placed in a predefined workspace (35 x 55 cm), but always with the circular studs facing upward. To observe the blocks, a CMOS camera (DFK 23UX174, The Imaging Source Europe) was placed 100 cm above the workspace, also to observe the build plate. A 16 mm fixed focal length lens was used to focus the light on the sensor (1,920 x 1,200 pixels). With an entire field of view of 76.5 x 47.0 cm, a spatial resolution of 0.39 mm/pixel could be achieved. To cancel out glare and unwanted specular reflections, a linear polarizer was placed in front of the lens.

A UR5 collaborative robotic arm (Universal Robots, Denmark) equipped with an RG2 gripper from OnRobot (Denmark) as an end-effector was used to manipulate the blocks and position the blocks on the build plate. There are a few methods to control the UR5. The most common methods are URscript, Matlab using the URControl, and Robot Operating System (ROS) using the URControl.

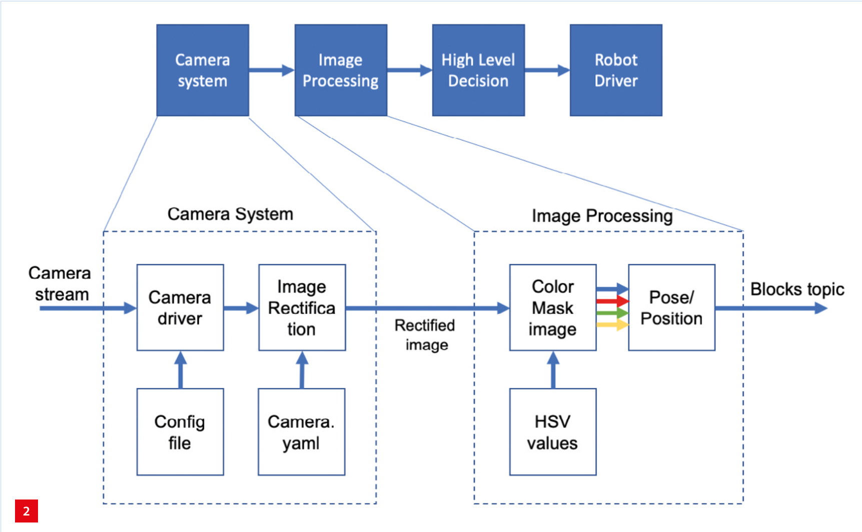
In the first method, the UR5 is programmed through the connected teach pendant along with the graphical PolyScope programming interface and URScript programming language. This method can be entered through the teach pendant and saved as a program to be executed on the robot. Hence, it is rather easy to program for pick & place tasks. However, using it for long and multi-tasking programs is complicated.

The second method is Matlab. Within the Matlab driver, the velocities in the joint space of the robot are controlled. This gives a good performance and is safe, but it is not open source. Also, there are limitations in the computational load for further developments and communication with other robots.

The last method is using the ROS programming environment, which is a set of software libraries and tools to build robot applications. This provides the services expected from an operating system, including hardware abstraction, and low-level device control. There are different packages in ROS that provide the capability of doing requests such as computing trajectory, connecting joystick and so forth. Specific packages can be added for many robotic applications. Furthermore, ROS has great simulation tools to show robot movements in offline or real operational mode. And above all, it is open source and it is relatively easy to communicate between Python and C++ programs.

Because of the aforementioned advantages and in consideration of further developments in the detection of and collaboration with humans, ROS was selected. ROS runs on Ubuntu and provides the capability to run different drivers or packages (including URdriver, camera, gripper, and additional sensors) within the ROS environment. This means that only one computer (here a NUC core i5) is enough to run the whole set-up.

The results and performance were analysed using two different computer vision approaches: one based on predefined colour spaces, also used in previous studies [2, 3]; and the other with machine learning, using multi-layer perceptrons [4].



Software flowchart of the computer vision script with predefined HSV (Hue, Saturation and Value) levels.

Conventional computer vision algorithms

In our ordinary computer vision script, we used the open source vision library OpenCV 3.0 in combination with ROS to acquire images from the camera. Images were asynchronously acquired with a maximum frame rate of 30 frames per second (fps). At start-up, white balancing was carried out and the gain and exposure time were set, see Figure 2. The image was then rectified using camera parameters that were determined after a camera calibration procedure [5]. This procedure must be carried out to correct for any distortions caused by the lens and remap the spatial sampling of the image.

Before the system can determine the orientation and position of the blocks, the rectified image must be masked using HSV spaces (Hue, Saturation and Value). The HSV spaces were determined by first adjusting the hue level until the correct colour (e.g. green) was shown in the masked image. Then, the saturation and value levels were adjusted to observe only that specific coloured block with the correct saturation and intensity value. The HSV spaces were saved for all four coloured blocks. Using the HSV levels, a 'find contour' algorithm was employed to find connected regions in the image, see Figure 3a. A minimum bounding rectangle was placed around the contour to determine the orientation of the block by using the corner values. The centre of the rectangle was used as position value for the block.

Field of view for the 'find contour' algorithm.
(a) Fixed top illumination showing correct detection of the green block.
(b) Environmental light causing errors in identifying the green block.

To demonstrate the problem with conventional computer vision algorithms, the set-up was placed close to a window to evaluate the performance under different and uncontrolled lighting conditions. These results are shown in Figure 3b. The altered illumination conditions caused the vision script to detect unwanted regions in the images; beforehand, the hard-coded HSV levels should have been recalibrated. In this case, the illustrated change in illumination was severe; even minor changes in intensity and spectrum of the light source can cause inaccurate determination of the position and orientation of the blocks.

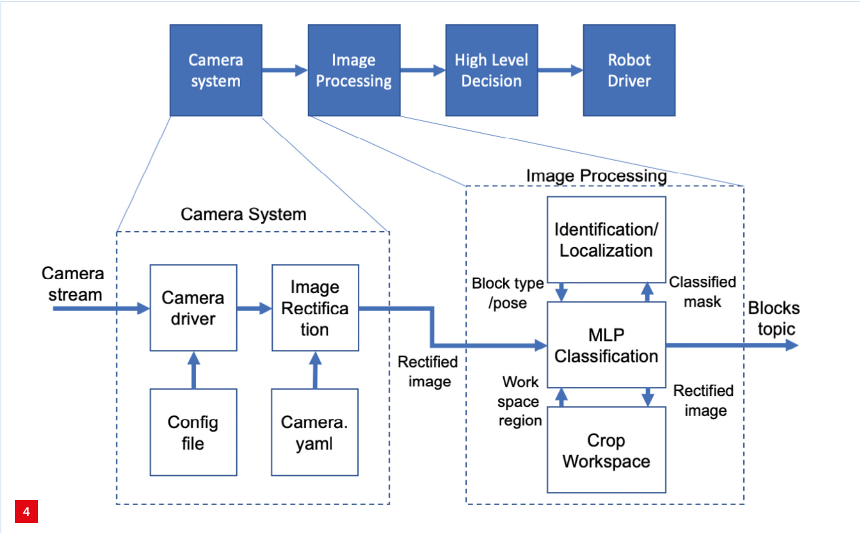
Blocks picked by the robot arm were first placed in a mechanical fixture to correct any alignment errors due to incorrect visual detection and errors caused by incorrect transfer frames (these frames convert camera pixels to real-world coordinates). After correction the blocks were placed on the build plate.

Machine-learning algorithm

In computer vision, the task of image segmentation is formulated as a classification problem where each distinct region of interest (ROI) is considered a class with distinct features. The six main classes in the image workspace are shadows, the background plate, and the four coloured blocks (red, green, blue and yellow). For each pixel in the image, the three primary colour channels Red, Green and Blue are considered the predicting features (please note that the latter group of colours are the camera colour channels and not the final classes).

Within each class definition, these features will vary due to the workspace lighting conditions not being constant. The relationship between features and classes is modelled using a Multilayer Perceptron (MLP), a popular type of Feed-forward Neural Network. The MLP has inputs and outputs that match the number of features and classes, 3 and 6 respectively.

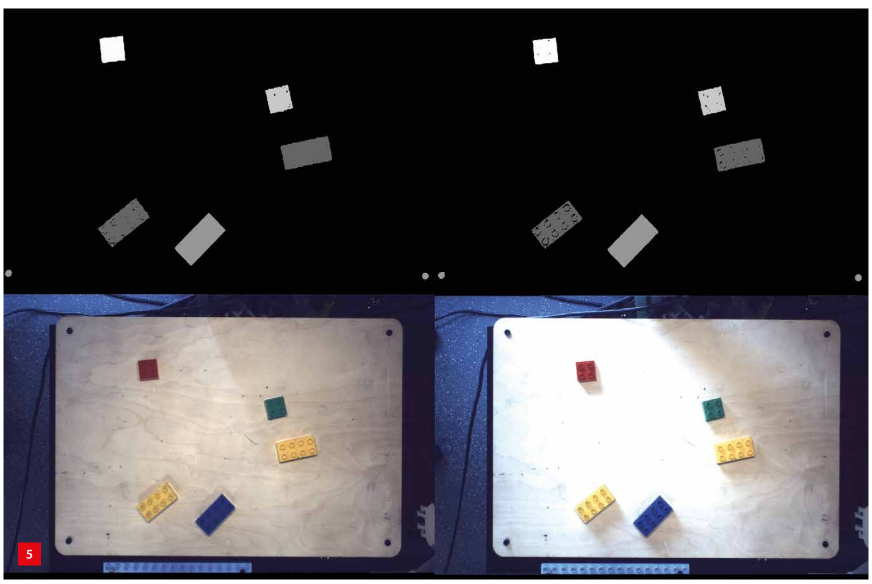
To train the MLP, supervised learning was used; this is a machine-learning method that can be used when the inputs and outputs of the network are known. The MLP was trained to map the class label to the input features. The training dataset consisted of paired features and class labels made using images that spanned a wide set of illumination conditions. The trained MLP was tested using novel data, achieving a prediction accuracy of 97.8%. Rectified images were sent to the block detection program,



Software flowchart of the machine learning script with MLP.

Figure 4. The bolts that fixate the workspace were used to determine the workspace location and the image reference frame was set. All new images were cropped to show only the workspace. Image classification was carried out by reshaping the 2D cropped input image into a 1D array containing all the pixels. The MLP carried out a batch prediction on this array and the resulting output was reshaped back into its original dimensions. The resulting output was a segmented image containing regions of interest and their respective class, as shown in Figure 5.

A block was localised in the image frame using the distinct round studs that line the top of each Duplo block. The position of a block's origin was considered to be the mean of the stud feature centre-point coordinates.



On the bottom, the raw images with changing illumination situations are shown. At the top, the labelled foreground masks are shown that were obtained with the machine-learning algorithm from the corresponding input images below.

To locate these features, the block was extracted from the original RGB image. Using the segmented image, the tightest convex polygon containing the block ROI was calculated. This calculated polygon was used to extract only the block from the RGB image. Centre-points of the studs were determined with a Hough circle transformation and the average of these feature coordinates was taken as the block's local origin. A minimum area rectangle was fit on the segmented block to determine its orientation. This process was performed iteratively for all blocks in the image and reached a high localisation precision 2σ: 0.321 mm, 0.233°.

Using the polygon area and the number of studs of each block, the block type could be identified. In the case of Figure 6, the block was identified as a blue 2 x 4 block. After identification, blocks were picked and directly placed on the build plate without the need for a mechanical fixture.

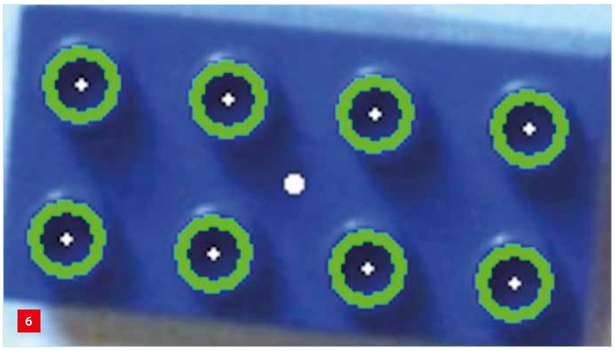
Comparison of results

In comparison to conventional computer vision algorithms, machine learning, and in particular the feedforward neural network, can be used to significantly improve robustness in the detection of colours and the identification of the Duplo blocks. We measured the physical position of the gripper with respect to the centre of the Duplo block with a caliper tool (mean error: 0.1 mm) to quantify the performance of the machine-learning method and the conventional computer vision method.

The four different coloured blocks were used and every block was measured twice under changing lighting conditions. The variation in lighting conditions was kept similar for both methods. The mean error in position estimation for the conventional vision algorithm was 4.6 mm (2σ: 7.3 mm), while the machine learning showed a mean error of 0.8 mm (2σ: 2.2 mm).

Identification and localisation is more robust due to the statistical nature of the classification. It determines the probability of a pixel belonging to a specific class (colour); therefore, changing the environment will have less influence when using machine-learning algorithms. This could be very beneficial in situations where product materials could vary between batches or human interaction is part of the production process, such as in collaborative robotics.

Computer vision script that uses the hard-coded threshold to differentiate between various colours can be unreliable with even the smallest changes in the environment. However, in the case of short processing times, computer vision algorithms are to be preferred because they are faster. In our situation, a maximum frame rate of 54 fps could be achieved. In comparison, the machine-learning algorithm



6

Duplo block with Hough circle detection to localise the centre (white dot) and the orientation of the block.

could be used with a maximum frame rate of 2 fps. By lowering the image resolution for classification, the frame rate can be increased to approximately 10 fps, but this is still significantly slower than the conventional methods.

Conclusion

Systems in manufacturing processes, or inspection lines using vision for pick & place applications and quality checks are more robust to changes in lighting conditions and product properties (such as material colours) when using machine-learning software algorithms.
This case study has shown that relatively simple feedforward neural networks like MLP can be used to identify products of interest. Furthermore, the need for mechanical alignment using a fixture can be prevented by using the machine learning algorithm. The mean error of 0.8 mm is sufficiently low to position the blocks on the build plate. With the 2σ deviation of 2.2 mm, however, the block is sometimes not aligned properly. This can be improved by a better calibration of the transfer frames of the robot to the build plate and the camera.
In the case when processing speed is a critical parameter, it is recommended to invest in proper hardware and stable (illumination) environments. The research is ongoing, but the preliminary results look very promising.

Acknowledgement

This project was supported by TechForFuture, Bronkhorst High-Tech and Benchmark Electronics. Special thanks go to Hanieh Esmaeeli, Mick Boe and Dirk Bekke.

REFERENCES

[1] J. Heilala, and P. Voho, "Modular reconfigurable flexible final assembly systems", *Assembly Automation*, Vol. 21 (1), pp. 20-30, 2001.
[2] A.Y. Mersha, R. de Kinkelder, and D. Bekke, "Affordable Modular Mobile Manipulator for Domestic Applications", *19th International Conference on Research and Education in Mechatronics (REM)*, Delft, pp. 141-146, 2018.
[3] R. de Kinkelder, and R. Zwikker, "Getting a grip on Medical and Modular Robotics", *Mikroniek* 57 (4), pp. 26-31, 2017.
[4] *scikit-learn.org/stable/modules/neural_networks_supervised.html*
[5] *wiki.ros.org/camera calibration*