Development of hardware-software complex for adaptive sorting of solid domestic waste

Kirilenko Vladimir (✉ kivicode.dev@gmail.com)

ГБПОУ «Воробьевы горы», Центр Технического Образования, Москва, Россия

Research Article

Keywords: waste sorting, waste classification, computer vision, machine learning

Posted Date: February 18th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-250347/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Development of hardware-software complex for adaptive sorting of solid domestic waste

Kirilenko V. D.

ГБПОУ «Воробьевы горы», Центр Технического Образования, Москва, Россия

Abstract

This project presents an autonomous system that allows to classify various solid garbage as well as to control a manipulator-sorter of the waste. Sorting is performed on the basis of material, shape, or specific object class. The development was focused on the system adaptability and acceleration of training, which allows the system to adapt changes in incoming waste.

CONTENTS

1. Introduction	2
2. Description of the system, operating principles and technologies used	3
2.1 Computer vision system.....	3
2.2 Object recognition	4
2.2.1 Approach selection	4
2.2.1 Developed approach.....	6
2.2.1 Classifier selection	6
2.2.3 Implementation	8
2.3 Virtual simulation	8
2.3 Manipulator	9
3. Conclusion	11
4. References	13

1. Introduction

Pollution of the environment by household waste leads to a violation of the ecological balance on the whole planet. However, almost any garbage is suitable for recycling and reuse. One of the main problems of the waste recycling cycle is how to sort it into fractions that can be used for recycling. Robotisation of this process will both reduce the cost of the entire recycling cycle and reduce the error rate during sorting^[1].

To implement the solid waste classifier, the use of computer vision for searching and machine learning algorithms was chosen.

The goal of the project is to develop a hardware-software complex that allows sorting different objects in autonomous mode depending on their class. For correct work, it is necessary to achieve accuracy of at least 90% and retraining time should not exceed 2 minutes. The project developing process may be divided into the following tasks:

1. Creation of a computer vision system (hereinafter referred to as CVS), which allows using a camera fixed above the working area to calculate the parameters of a bounding rectangle for each sorted object
2. Evaluation of various approaches of classification of objects based on visual representation
3. Implementation of the classification algorithm
4. Development of a virtual environment allowing to test the algorithm of the real device
5. Development and creation of own manipulator capable of sorting objects with mass up to 1.5 kg and width up to 110 mm to check the performance of all systems in real conditions.

At the moment only a few companies, such as Sadoko and Max AI, are developing such solutions. These companies use deep neural networks in their developments. This approach allows to achieve good accuracy, but processing requires a lot of computing power^[2], and the retraining process may take up to several hours (see Table 1), which excludes the possibility of adaptive learning.

2. Description of the system, operating principles and technologies used

2.1 Computer vision system

Let us consider the resulting CVS. The task was divided into two subtasks:

- Removal of image defects and distortions resulting from the appearance of perspective
- Creation of a binary mask to separate the objects under consideration from the background

To solve the first problem, it was decided to place four ArUco^[3] markers at the corners of the working area. Such markers are minimally dependent on such external conditions as lighting, camera angle, etc. The core idea is to search for all 4 markers and, as a consequence, 4 corners of the working area of the manipulator (see Fig. 1.1), which allows to calculate the transformation matrix and use the perspective correction function, resulting in an image close to the orthogonal view from above (see Fig. 1.2). Such a result will make it possible to transform the object coordinates from the global reference system associated with the camera into the local Cartesian system of the working field. The transformation matrix is calculated by solving the following system:

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0 \cdot u_0 & -y_0 \cdot u_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot u_1 & -y_1 \cdot u_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 \cdot u_2 & -y_2 \cdot u_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 \cdot u_3 & -y_3 \cdot u_3 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0 \cdot v_0 & -y_0 \cdot v_0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot v_1 & -y_1 \cdot v_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 \cdot v_2 & -y_2 \cdot v_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 \cdot v_3 & -y_3 \cdot v_3 \end{bmatrix} \cdot \begin{bmatrix} c_{00} \\ c_{01} \\ c_{02} \\ c_{10} \\ c_{11} \\ c_{12} \\ c_{20} \\ c_{21} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

where (x_i, y_i) - the coordinates of the i -th point in the global reference system, (u_i, v_i) - the coordinates of the i -th point in the local reference system after the transformation, coefficients c_{ij} are used to calculate the local coordinates

$$\begin{cases} u_i = \frac{c_{00} \cdot x_i + c_{01} \cdot y_i + c_{02}}{c_{20} \cdot x_i + c_{21} \cdot y_i + c_{22}} \\ v_i = \frac{c_{10} \cdot x_i + c_{11} \cdot y_i + c_{12}}{c_{20} \cdot x_i + c_{21} \cdot y_i + c_{22}} \end{cases}.$$

Once the most convenient view of the image has been obtained, let us consider the second subtask. It has been solved by performing preliminary calibration of the average color value, converting the image to the HSV^[4] color space and applying an inverted algorithm of color search in a given range, which at the output gives a binary mask corresponding to the deviation of pixel color from the calibration standard (see Fig. 1.3). Such an algorithm is suitable for search of both opaque and transparent objects, which will be found anyway due to light refraction. Next, select individual objects on the resulting mask, using auxiliary morphological operations to improve accuracy. It is convenient to cut selected objects out of the original image for further processing and classification.

Fig. 1 — Object selection stages

2.2 Object recognition

2.2.1 Approach selection

Image recognition is a classic task in the field of machine vision and neural networks and has many solutions depending on the task. In this case, since the problem of object search is solved without the use of machine learning algorithms,

the recognition task is reduced to determining the object label for an input image, in other words, to classification.

Generally, several approaches may be used to solve the task. The most suitable of them are the following:

- Training and application of convolutional^[5] neural network (CNN) for classification
- Deep neural network (DNN) training and application for classification
- Use of one of the classification or clustering algorithms (e.g. k-nearest neighbours method).

As a result of the research of available solutions, the following was revealed:

- Direct usage of CNN classifier requires a lot of training data and shows an accuracy of ~30%
- Direct implementation of deep neural networks shows much higher accuracy in comparison with CNN, but it is still not accurate enough within the project and also requires too much time for training (see Table 1)
- Direct use of classifiers/clusterizers has minimal training time (in the range from 0.02 to 1 min.), but has insufficient accuracy to solve the problem (about 10%).

Model	CPU	GPU	Training time	Accuracy (<i>bAcc</i>)
Inception V3	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	13ч 24мин	~50 %
Inception V4	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	21ч 3мин	~60 %
Inception V5	Intel Core i7 6700k	Gigabyte GeForce GTX 1070Ti	12ч 11мин	~61 %
MobileNet	AMD Ryzen 3 3200G	MSI GeForce GTX 1050Ti	20мин	~30 %

Table 1 — Training of different models (8 classes with 100 samples each)

2.2.1 Developed approach

During the development, an approach was created. It can be trained in 11.2 seconds with an accuracy of 93.9%. Training and testing were conducted on Intel Core i5 2.4 GHz CPU and NVIDIA GeForce RTX 2070 SUPER. The idea is to combine of CNN ImageNet^[6] (which was later replaced with Inception V3), which allows converting a raster RGB image into a two-dimensional feature vector and one of the classification/clusterization algorithms. Such learning speed and classification accuracy allow you to add new classes and/or expand existing ones without long interruption of work.

2.2.1 Classifier selection

To select the most suitable classifier, a comparative analysis of the 5 most common algorithms was performed. Fig. 2 shows error matrices for the compared algorithms, testing was performed on the same equipment and training sample. The confusion matrix^[7] is a matrix M , which is calculated using the following formula:

$$M = \{m_{ij}\}_{i,j=0}^C, m_{ij} = \sum_{k=0}^N \mathbb{I}[a(x_k) = j] \mathbb{I}[y_k = i] \text{ for some sampling } x_i (i = 1, \dots, N, y_i$$

- i -th class label $y_i \in \{1, 2, \dots, C\}$), each object which belongs to one of C classes and a classifier a , which predicts these classes. This matrix shows how many objects of class i were classified as j . Table 2 shows the results of testing the algorithms under consideration. Figure 3 shows a graph that helps to compare the accuracy and training time of the algorithms.

Based on the data obtained, it was decided to choose the SVC algorithm^[8] for the classification process. SVC (or C-Support Vector Classification) is a type of support vector machine classifiers. The main idea is to transform the original vectors to a higher dimensional space and search for a separating hyperplane with the maximum gap in this space. Formally, the algorithm can be described in the following way:

points have the form $\{(x_1, c_1), (x_2, c_2), \dots, (x_n, c_n)\}$, where c_i is the index of the class that point x_i belongs to. A separating hyperplane is constructed, which has the form

$w \cdot x - b = 0$. Vector w - perpendicular to the separating plane, $\frac{b}{\|w\|}$ - distance from

the hyperplane to the origin. Next, the problem is reduced to minimising $\|w\|$ for expression $c_i \cdot (w \cdot x_i - b) \geq 1, 1 \leq i \leq n$.

Algorithm	Fitting time, ms	Accuracy, %
SVC	11900	93.9
Linear SVC	55200	91.3
KNearestNeighbors	418	89.6
KNearestNeighbors	6800	86.7
Gaussian Naive Bayes	118	85.8

Table 2 — Comparison of learning time and accuracy of classification algorithms

Fig. 2 — Confusion matrixes for different classifiers

Fig. 3 - Distribution of classifiers by time vs accuracy

2.2.3 Implementation

The final implementation of the CVS and classifier is performed on the high-level Python language using the following libraries:

- OpenCV - to capture video from the camera, and to process images
- TensorFlow 1.14 - to use and retrain ImageNet
- Scikit-learn - to process SVM
- OpenRV - proprietary library to accelerate development.

This choice is due to ease of use and speed of modelling, which allows minimising development time. Further, the CVS and the classifier will be combined under the name Visual Object Recognition System (VORS).

2.3 Virtual simulation

The simulation system must meet the following requirements:

- Ability to test algorithms of real device operation inside the simulation
- Ability to transfer data between the simulation and VORS via an API
- Ability to scale the simulation for multiple devices running in parallel
- Support for various manipulator kinematic systems
- Ability to synchronise the simulation and the real manipulator.

In the process of comparing the ready-made physical engines to create the simulation, the following SDKs were analysed:

1. Gazebo,
2. V-rep,
3. Webots,
4. MRS,
5. Unity 3D,
6. Unreal Engine

The best option among the listed is Gazebo, but it had to be abandoned due to the high complexity of recreating the linear movement of the system. For similar reasons, Webots, v-rep and MRS are not suitable for the project. Thus, the Unity 3D engine was chosen for the CM implementation as it is well documented, allows changing of the kinematics without much expense and has built-in support of serial connection for synchronisation with the real device, unlike Unreal Engine.

2.3 Manipulator

The process of creating a manipulator includes the following tasks:

- Kinematics system selection
- Determine the technical characteristics of the manipulator based on the weight of objects and the size of the working area
 - Maximum weight of the object to be moved - 1.5 kg
 - The working area size is 450mm × 900mm

The cost of manipulators is determined mainly by the price of electric actuators. The more powerful the actuator is the higher its price. Therefore, when choosing the kinematic scheme, the main attention was paid to reducing the load on the manipulator motors.

In most kinematic schemes, the drives perceive static and dynamic loads, both from the forces of the weight of the load being moved, and the forces of the weight of the links. This requires a significant increase in the power of electric motors for drives and additional braking devices, as well as counterweights.

In the SCARA kinematic scheme (Fig. 4) the links of the manipulator are mutually rotated in the horizontal plane, and the grip makes progressive movements up and down. Thus forces of weight of links and weight of cargo are perceived by bearings of kinematic pairs and influence only through friction forces in pairs. This makes it possible to use electric drives of low power and, accordingly, lower cost than in any other kinematic schemes. As a result, all this makes the construction cheaper. Taking

into account the above mentioned, the SCARA kinematic scheme was used in this project.

Рис. 4 — Кинематическая схема манипулятора SCARA

The manipulator is driven by three stepper motors Nema 21, which allows controlling the movement of all nodes with high precision. For manipulations with sorted objects, a combination of a pneumatic gripper with parallel jaws mounted on a servo-machine and the pneumatic suction cups is used, which replace each other depending on the object class (e.g. for bottle manipulation, a grapple gripper is used and a suction cup is used for a bag). The servo-machine is used for precise positioning that allows to efficiently capture such objects as bananas, bottles, etc.

To simplify the manipulator control, it is necessary to use inverse kinematics formulas that allow converting the object coordinates from the Cartesian field reference system into three angles for each of the manipulator links. Based on the illustration presented in Fig. 5, angle $q_2 = \cos^{-1} \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$,

$$q_1 = \tan^{-1} \frac{y}{x} - \tan^{-1} \frac{a_2 \sin q_2}{a_1 + a_2 \cos q_2} \text{ and } q_3 = \cos^{-1} \langle \vec{a_2}, \vec{L} \rangle.$$

Fig. 5 — Inverse kinematics

Low-level manipulator control is implemented on the Arduino Mega 2560 board, for communication with the control computer API for communication with VORS and/or the simulation is implemented.

3. Conclusion

As the result of this work, an automatic sorting system was developed that can recognise many classes as well as to classify such "featureless" objects as waste, and to control the sorting robot, as well as a virtual environment that allows to test the performance of software modules, including VORS. Various methods of classification of solid waste were analysed and the method suitable for the project conditions was developed. The use of adaptive learning allows to increase the number of classes of recognised objects without a long stop of work, which allows reconfiguring of the entire system to meet specific requirements.

It was decided to continue developing the project in the following directions:

- Reducing the percentage of errors in the classification of objects by working with the garbage sorting complex to collect a sufficient training sample
- Increase the localisation accuracy of the objects
- Modification of the manipulator, by increasing the accuracy of movements and reducing restrictions on objects (eg, the ability to take objects wider than 110mm and heavier than 1.5kg).

An example of the work of the resulting manipulator and the VORS can be found at the following link: https://youtu.be/P_LQVBXe5EY.

4. References

1. Research and development of scientific and technical solutions in the field of sorting operations in real time, with objects that have complex characteristics, using highly efficient robotic automation tools.
Access: http://fcpir.ru/upload/iblock/94b/corebofs000080000ldo38um1mqn7hvc_annotation.pdf
2. D. Jirak, S. Wermter Potentials and Limitations of Deep Neural Networks for Cognitive Robots. Access: <https://arxiv.org/pdf/1805.00777.pdf>
3. Detection of ArUco Markers. Access: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html
4. HSV color scheme. Access: https://en.wikipedia.org/wiki/HSL_and_HSV
5. Le Man Ha, A convolutional neural network for solving the classification problem // PROCEEDINGS OF MIPT. 2016. Volume 8, no. 3. 2016. - p. 91-97
6. ImageNet. Access: <http://image-net.org/index>
7. Confusion matrix. Access. [https://learnmachinelearning.wikia.org/ru/wiki/Матрица_ошибок_\(Confusion_matrix\)](https://learnmachinelearning.wikia.org/ru/wiki/Матрица_ошибок_(Confusion_matrix))
8. R. Berwick, An Idiot's guide to Support vector machines (SVMs) – p.5-28.
9. SCRA Robot Kinematics. Access: <http://www.deltatau.com/Common/technotes/SCARA%20Robot%20Kinematics.pdf>
10. Вапник В.Н. The nature of statistical learning theory – p.138-176

Figures

Figure 1

Object selection stages

Figure 2

Confusion matrixes for different classifiers

Figure 3

Distribution of classifiers by time vs accuracy

Figure 4

Кинематическая схема манипулятора SCARA

Figure 5

Inverse kinematics