AN APPLICATION OF A ROBOT ARM FOR GARBAGE CLASSIFICATION USING IMAGE PROCESSING AND ARTIFICIAL INTELLIGENCE

Quang-Huy Pham [1], Cong-Hanh Nguyen [1,2,*], Hieu-Ngan Ly [1], Quoc-Cuong Bui [1], Nhat-Bang Bui [1], Gia-Nguyen Dang [1], Trieu-Khang Nguyen [1], Lam-Bao-Chan Le [1]

[1] Ho Chi Minh City University of Technology and Education (HCMUTE)
Vo Van Ngan Str., No. 01, Ho Chi Minh City (HCMC), Vietnam
[2] Reeco Science and Technology Company Limited
Room 202B, Building A, Software Technology Zone, Vo Truong Toan Street, HCMC, Vietnam
* Corresponding author. E-mail: conghanhnguyen@reecotech.com.vn

Abstract: Garbage classification is an important problem in big cities. By using a combination of free tools available from the open community, a platform can be created in laboratories for simplify this complicated systems into a simple model for students. However, this system still maintains rich features that help students in researching. In this paper, we present a platform that satisfies these requirements. Besides creating a mechanical robot arm Dobot M1, forward and reverse kinematic calculations are examined for this model. Thence, through image processing combination, we apply a Yolo neuron network (NN) structure. By using this network, based on a set of available data, we enrich the date by our new data. This structure is trained to classify kinds of garbage. Our experimental model works successfully through a survey to prove the high application of this research. A supervision platform is also presented for operators to control the process.

Keywords: robot arm, garbage classification, forward kinematics, reverse kinematic.

1. Introduction

Garbage process is a big urban problem in many countries [1]. By the increasing of population and consuming habits of people, the amount and complexity of garbage are larger day by day. Thence, creating a robot that use kinds of technological methods to solve this problem is important. This topic attracts researchers to follow this direction. Currently, the main trend in research on this topic is the integration of artificial intelligence (AI)[1][2]. Deep Learning technologies help robots accurately classify waste types (plastic, metal, paper, food, etc.). Robots not only classify but can also pre-process waste such as compressing, shredding or transferring to recycling areas, automating processes. Many recycling plants around the world have applied robotic systems to reduce labor costs and increase sorting efficiency, so they are widely used in industry. Most of the current research and applications focus on using AI to identify waste through computer vision [3], [4]. In those researches, only detection garbage by AI is focus, a total full research of actuators to develop a real robot is not operated yet.

Robot arms are developed to apply the actuators for operation. ABB's YuMi robot (Fig. 1) [5] is a flexible robot of ABB. It is a dual-arm robot combined with the IRB 1200 material handling robot. This system uses artificial AI and machine learning technology to classify waste into four groups, from hazardous waste to recyclable materials. After sorting, the waste is transferred to the IRB 1200 robot for collection and recycling. It was proven to work well in Shanghai second China International Import Expo (2019) [**6**]. AMP Robotics (Fig. 2) **[7]**[8]: In the US, AMP Robotics develops robotic arms that use computer vision technology to sort waste at recycling plants. The company has created a platform to expertize this solution.

Fig. 1. ABB's YuMi robot [6].

Fig. 2. AMP Robotics's robot sorting trash [8].

However, in condition of laboratory, a smaller model which is friendly with students. Based on that, the researching and training tasks can be applied for students. A requirement of developing a cheap and easy-to-use model still exists. Then, in this paper, we develop an arm

robot to satisfy the solution of garbage classification. The model of hardware is developed from Dobot M1 [9] due to its simplicity and population. The image processing is based on OpenCV [10]. We utilize the library with Yolo to classification objects. And, forward and reverse kinetic calculations are made to let robot work in real-time environment.

2. Experimental Model

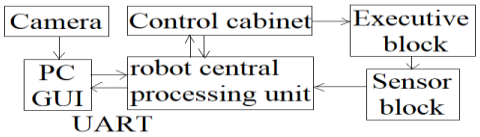


Fig. 3. System overview.

Description of blocks in Fig. 3:
- PC, GUI block: This is the important block of the system. The computer will receive signals from the camera, process AI to identify objects and send the coordinates of the object to the microcontroller to perform via the UART communication protocol. Besides, there is the GUI user interface.
- The main control block for the robot consists of 2 ESP32 microcontrollers communicating with each other via the UART communication standard. The first ESP32 is responsible for controlling the robot arm and communicating with the computer. Meanwhile, the second ESP32 will receive signals from the control cabinet and communicate with the first ESP32 to transmit and receive data
- Actuator block: Receives signals from the central processing block, converts them into voltage values to perform motor control
- Sensor block: Collects and processes information about the rotation angles of each joint, sends signals to the central block for processing.

We use ESP32 WROOM 32 Micro (Fig. 4) as control chip for this project.

Fig. 4. Pin description of ESP32 WROOM 32.

We choose step motor (Fig. 5) and servo motor (Fig. 6) as actuators of this project. The camera of this project is shown in Fig. 7.

Fig. 5. Step motor.

Fig. 6. DC servo motor.

Fig. 7. CMOS 2MP camera.

Thence, our designed robot is shown in Fig. 8 and the real model is created as in Fig. 9.

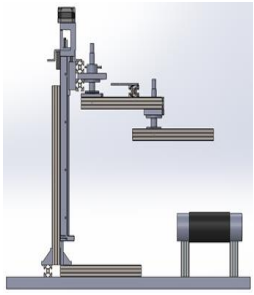


Fig. 8. Desined robot in SolidWorks.

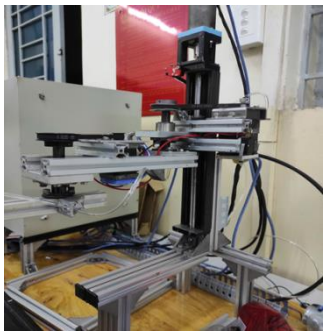


Fig. 9. Robot arm after completion.

3. Control Method

3.1. YOLO Network (You Only Look Once)

Fig. 10. OpenCV [11]

All these complex process in image processing can be solved easily by using OpenCV (Fig. 10). And, after utilizing image, we follow next to YOLO.

The YOLO model was introduced by Redmon in 2016 with the ability to directly predict bounding boxes and object class probabilities using a single network in a single evaluation. YOLO has been continuously improved over versions. In terms of accuracy, YOLO may not be the most accurate algorithm, but it is one of the fastest algorithms in object detection models. It allows for near real-time detection with a reasonable level of accuracy, without too much loss. YOLO is not simply an object detection and classification algorithm, but also has the ability to accurately determine the location of objects. The YOLOv8 version has overcome the shortcomings of previous versions, including errors in determining the location of objects, spatial constraints on bounding boxes, and the fact that each grid cell can only predict a very small number of bounding boxes.

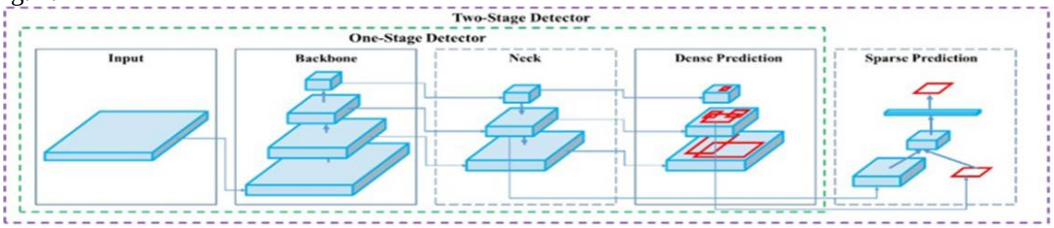


Fig. 11. YOLOv8 Object Recognition Architecture.

Processing of training and Application Process of YOLOv8 is:

- Preparing Training Data
- Doing steps to Create a Dataset
- Training
- Evaluating Model

Overall, model evaluation is an important step in machine learning that helps us understand the performance of a model and identify areas for improvement. By understanding the strengths and weaknesses of a model, we can iteratively improve it to achieve better results.

3.2. Combination YOLO and Image Processing

We use a NN model trained based on the YOLO library to solve the problem of object detection and information extraction in image frames. Structure of work is shown in Fig. 12.

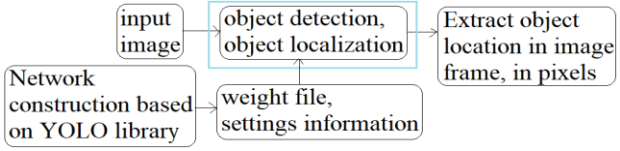


Fig. 12. Location of the network in the system

3.2.1. Creating Datasets with RoboFlow

RoboFlow is a framework for Computer Vision developers to collect better data for pre-processing and model training techniques. RoboFlow has public datasets available for users and also has access for users to upload their own custom data.

We use RoboFlow to label data as well as enrich and segment data.

The steps are as follows:

- **Collecting data**

Input images will be collected through devices such as cameras. This is the type of image that can be post-processed and processed. Image parameters such as resolution, quality, and storage capacity depend on the capture device, which also affects the subsequent image processing operations.

We collect images by taking photos from phones and webcams. After having a set of images, we will upload the data set to Roboflow software. Roboflow software is a framework for computer vision developers to collect better data for pre-processing and model training techniques.

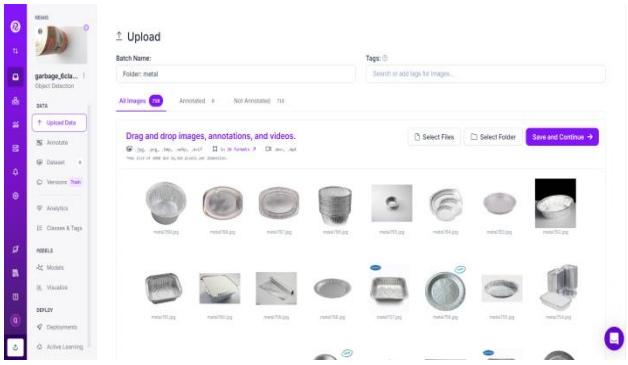


Fig. 13. Images from the dataset are available on Roboflow.

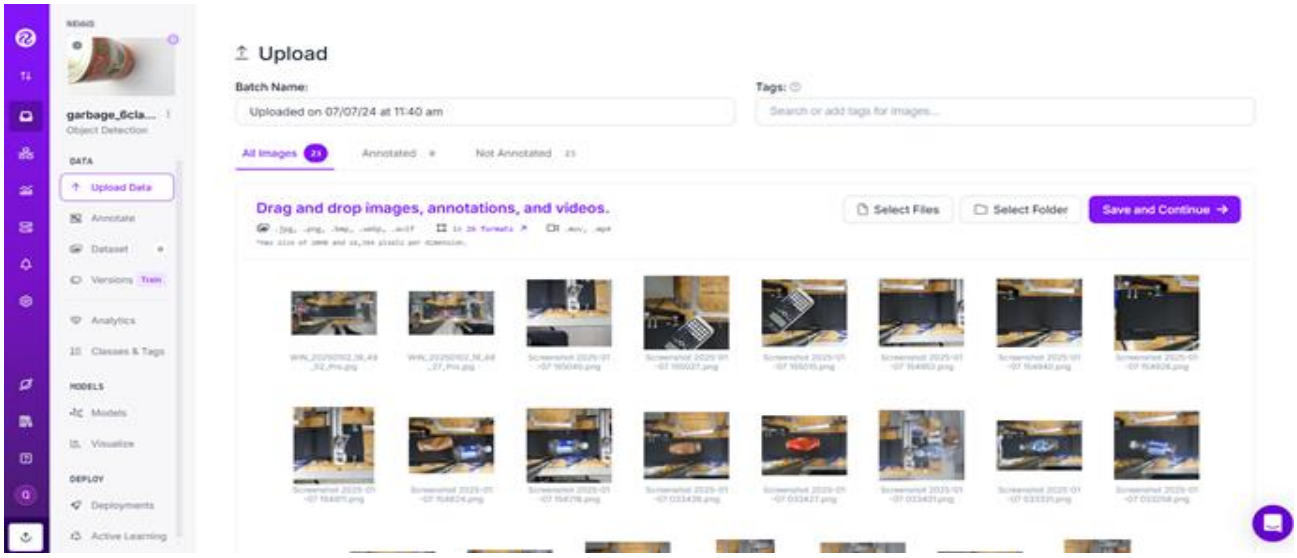


Fig. 14. Images added from reality.

- **Labeling**

After uploading the dataset to Roboflow, we manually label each image according to the Class created earlier. For this dataset, we create two Classes: plastic bottles (Plastic-Bottle) and objects that are not plastic bottles (Others).

There are two commonly used labeling methods in Roboflow: labeling according to the bounding box or embracing the entire object.

Fig. 15. Label according to bounding box.

Fig. 16. Label the entire object.

- **Devising dataset**

Training Set: is the training set.
Validation Set: is the validation data set, which helps to find the best model among the candidates trained from the training set. It also helps to limit overfitting by using the early stopping technique.
Testing Set: is the test set to evaluate the results obtained by the prediction model.

- **Preprocessing data**

We define steps that are performed on all images before they are fed into the model. For example, you can resize your images so they are all the same size or convert your images to grayscale.
We chose to resize the data to 416x416 for ease of training the network.

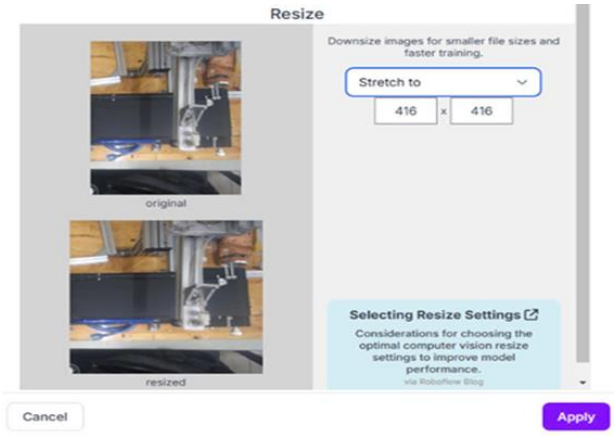


Fig. 17. Resize dataset to YOLO standard size.

- **Enriching data**

To add more images to the dataset, we use the tools available on Roboflow. For example, if we want to flip the image, rotate the image, blur the image, ... on Roboflow, each tool can help us double or triple the available dataset. Some data enhancement techniques that we used:

- Rotate 90°
- Increase or decrease brightness by 15%
- Blur image by 1.2px

At this step, the number of enhanced images depends on the user. We chooses 2x, which means that from the training image set, Roboflow will create 2 copies and randomly apply image enhancement techniques to the copies.

3.2.2. Model Training with Google Colab

Colaboratory, also known as Google Colab, is a product from Google Research, it allows Python execution on the cloud platform. It is especially suitable for Data analysis, Machine learning and education. It does not require installation or computer configuration. Everything can be run through the browser. You can use computer resources from high-speed CPUs and both GPUs and TPUs are provided. Using Google Colab has outstanding benefits such as: ready to run Python on any device with Internet connection without installation, easy sharing and teamwork, free use of GPUs for AI projects.
After completing the dataset, we proceeds with the training steps. In order to speed up training and share information faster. we use Google Colab, a customized Jupyter Notebook that allows Python execution on the cloud platform, provided by Google. Some advantages of Google Colab:

- Easily connect to data on the cloud
- Write and execute Python commands without installation
- Provide many popular libraries for DeepLearning such as Keras, TensorFlow, Pytorch and OpenCV
- Provide 12-25GB GPU for training data

Steps to train a model with Yolov8 on Google Colab are:
Step 1: Prepare a labeled dataset.
In this step, we used a dataset with 2766 images for "Train", 360 images for "Valid" and 83 images for "Test", with 2 classes to identify two objects: cans and plastic bottles
Step 2: Connect to Google Drive to save the Training process
Step 3: Install the ultralytics library

```
!pip install ultralytics
from ultralytics import YOLO
```

Step 4: Download the pretrain file with Yolo's available COCO dataset to your drive folder. We uses the model "Yolov8n.pt" to optimize speed

```
d /content/drive/MyDrive
!wget
https://github.com/ultralytics/assets/releases/downloa
d/ v8.2.0/yolov8n.pt
```

Step 5: Upload the folder containing the data created from Roboflow and save it to a folder in "My Drive"

```
%cd /content/drive/MyDrive/dataNew
!unzip /content/drive/MyDrive/newdataset_v3.zip
```

Step 6: After uploading the data to new folder, we open the "data.yaml" file and edit the path to the files containing the images in "Train", "Valid", "Test" sets
Step 7: Start the Training process with 100 epochs, the image size is 416x416px. The data during the Training process will be automatically saved to the folder on My Drive in the path "/content/drive/MyDrive/runs/detect/train"

```
!yolo task=detect mode=train
model=/content/drive/MyDrive/last.pt
data=/content/drive/MyDrive/dataNew/data.yaml
epochs=100 imgsz=416
```

After training, we need to pay attention to the parameters "box_loss", "cls_loss", "mAP50", "mAP50-95" to evaluate the quality of the model.

3.2.3. Object tracking algorithm with DeepSORT

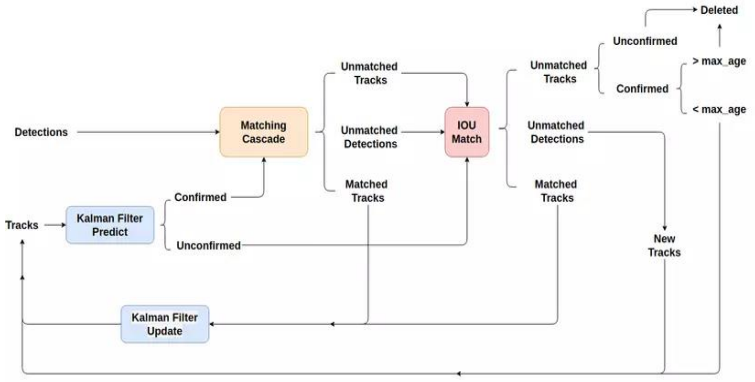


Fig. 18. Deep SORT processing flow.

Deep SORT was developed by Nicolai Wojke and Alex Bewley right after SORT to solve the shortcomings related to the high number of ID switches. The solution proposed by deep SORT is based on the use of deep learning to extract object features to increase the accuracy in the data linking process. In addition, a linking strategy called Matching Cascade is also built to help link objects after they have disappeared for a while more effectively. DeepSORT helps to track objects, predict paths, and automatically re-identify objects when they are obscured.

3.2.4. Calculating object location

After detecting and localizing the object in the image frame, the next step is to calculate and convert the object location from image coordinates to robot coordinates, in millimeters.

- **Find the internal camera parameter matrix**

The Camera Calibrator application allows you to estimate the internal, external, and lens distortion parameters of the camera. We use these camera parameters for various computer vision applications. These applications include of removing lens distortion from images, measuring flat objects, and reconstructing 3-D scenes from multiple cameras.

On Matlab, we uses "TOOLBOX_calib" downloaded from the Internet to find the Camera parameters

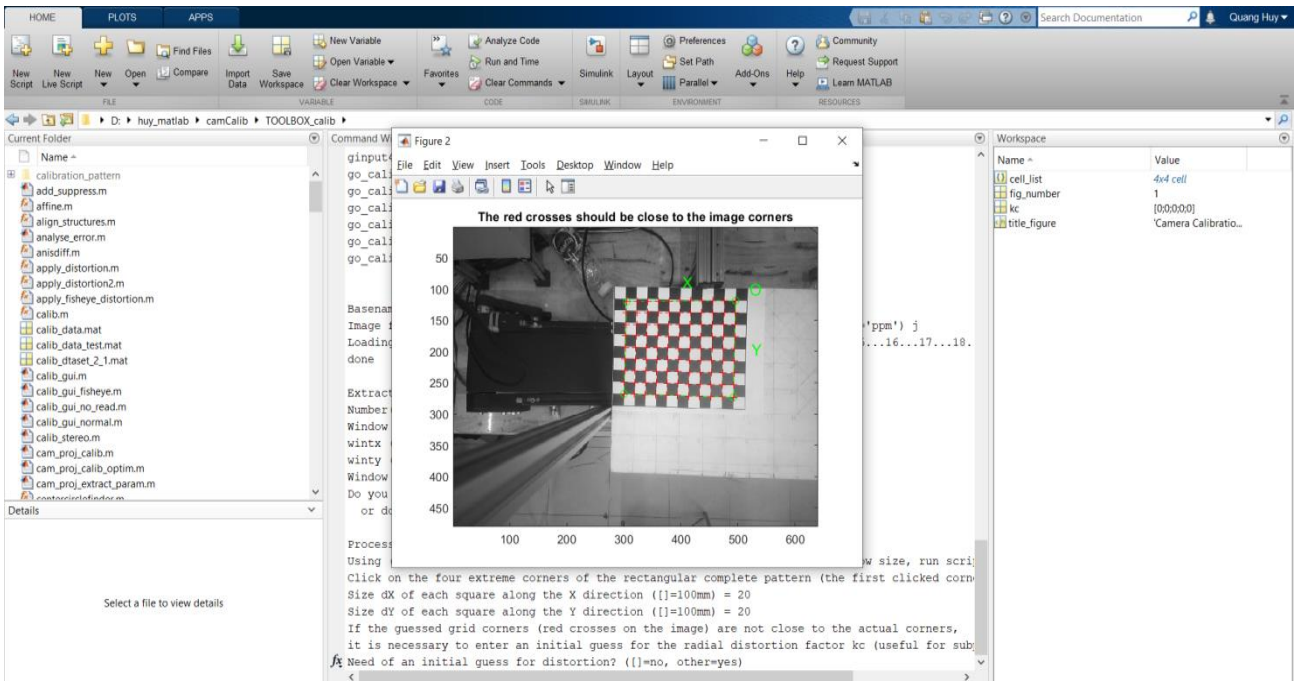


Fig. 19. Toolbox to find Camera parameters on Matlab.

The results obtained internal parameters of the camera:

$$camMatrix = \begin{bmatrix} 453.2517 & 0 & 326.2835 \\ 0 & 453.5607 & 242.9633 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Then, we calculate Extrinsic parameters matrix, obtain values such as Translation vector, Rotation vector, Rotation matrix.

From there, the coordinates of the points can be converted from px to mm.

- **Convert to Robot Coordinate System**

In OpenCV, the cv2.findHomography function is used to calculate the homography matrix between two sets of points. This homography matrix describes a geometric transformation between two planes.

Before that, we need to select the pixel points on the camera and the corresponding real coordinates of the robot, then use the cv2.findHomography function to calculate the homography transformation matrix

3.3. Kinetic Calculation

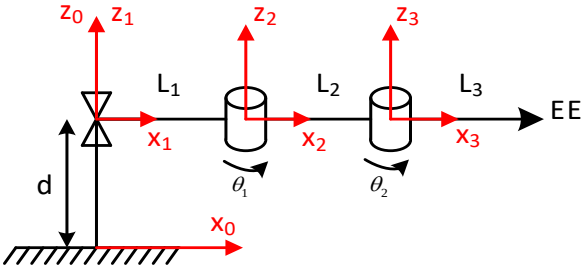


Fig. 20. Structure of M1 robot and axis placement for robot joints.

After calculation from Fig. 20, the coordinates of the end point relative to coordinate {0} is:

$$ {}^{0}P_{EE} = {}^{0}_{3}T\,{}^{0}P_{EE} = \begin{bmatrix} L_1 + L_3c_{12} + L_2c_1 \\ L_3s_{12} + L_2s_1 \\ d \\ 1 \end{bmatrix} \quad (2) $$

If end point is

$$ {}^{0}P_{EE} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \quad (3) $$

Then, after calculation, the angle of links are:

$$ \theta_2 = a\tan 2\left(-\sqrt{1-c_2{}^2}, s_2\right) \quad (4) $$

$$ \theta_1 = a\tan 2\left(s_1, c_1\right) \quad (5) $$

$$ = \frac{\left(L_3c_2 + L_2\right)Y - \left(X - L_1\right)L_3s_2}{\left(X - L_1\right)\left(L_3c_2 + L_2\right) + YL_3s_2} $$

Dynamic equations of robot arm are:

$$ \tau = M\left(\theta\right)\ddot{\theta} + V\left(\theta,\dot{\theta}\right) + G\left(\theta\right) \quad (6) $$

where: $M\left(\theta\right) = \begin{bmatrix} M_{11} & 0 & 0 \\ 0 & M_{22} & M_{23} \\ 0 & M_{32} & M_{33} \end{bmatrix}$; $M_{11} = m_1 + m_2 + m_3$;

$M_{22} = L_2{}^2m_2 + L_2{}^2m_3 + L_3{}^2m_3 + 2L_2L_3m_3c_2$;

$M_{32} = L_3m_3(L_3 + L_2c_2)$; $M_{32} = L_3m_3(L_3 + L_2c_2)$;

$M_{33} = L_3{}^2m_3$; $V\left(\theta,\dot{\theta}\right) = \begin{bmatrix} 0 \\ -L_2L_3\dot{\theta}_2m_3s_2(2\dot{\theta}_1 + \dot{\theta}_2) \\ L_2L_3\theta_1{}^2m_3s_2 \end{bmatrix}$;

$G\left(\theta\right) = \begin{bmatrix} g\left(m_1 + m_2 + m_3\right) \\ 0 \\ 0 \end{bmatrix}$; $\ddot{\theta} = \begin{bmatrix} \ddot{d} & \ddot{\theta}_1 & \ddot{\theta}_2 \end{bmatrix}^T$;;

$\tau = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 \end{bmatrix}^T$

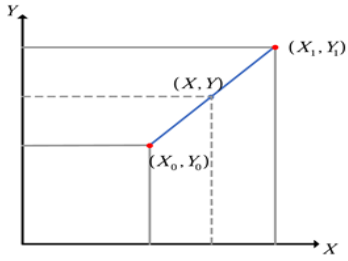


Fig. 21. Linear interpolation.

In this step, we choose to plan a straight-line point-to-point trajectory to provide uniformity in movement within the workspace and optimize system uptime (Fig. 21).

We do not use the direction of the last axis, so the workspace is only physically limited in terms of interaction with other project objectives such as the location of the Robot, the location of the camera and the conveyor. The result of simulation is shown in Fig. 22.

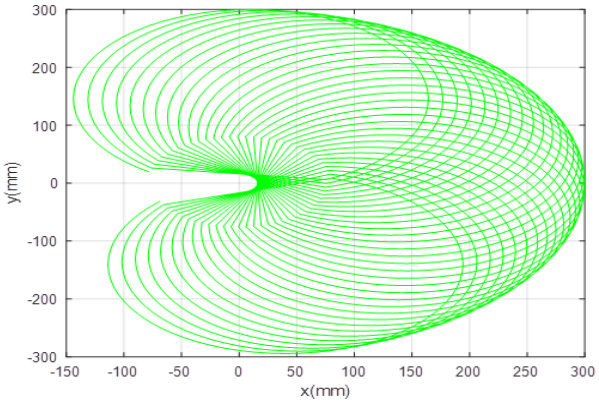


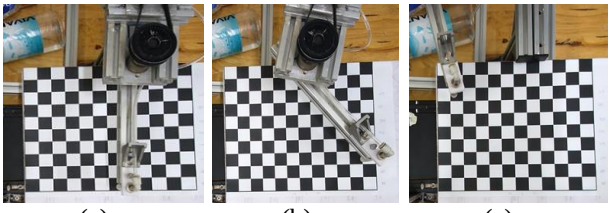
Fig. 22. Robot workspace in simulation.

In this research, PID controller is used for each link. Our robot PID control follow the structure of three PID controllers that follow the study [12].

4. Experimental Results

4.1. Testing Kinematic Motion

In this verification part, because the verification is point-to-point on a 2d plane, we ignore the z-axis. The end point of arm is controlled successful at selected positions (Fig. 23).



(a) (b) (c)

Fig. 23. Robot arm (up-down view) in 3 case
(a)- Home (x,y) = (300,0)
(b)- (x,y) = (260,80)
(c)- (x,y) = (160,-100)

4.2. Testing Yolo Model

After Yolo's training is complete, we get a folder containing images of the training results. Results are shown in Fig. 24. In Fig. 25, the parameters "recall",

"precision", "mAP50", "mAP50-95" are quite high, showing that the model has quite good results. From Fig. 26 to Fig. 29, all kinds of objects are classified exactly with corresspondi-ordinates.

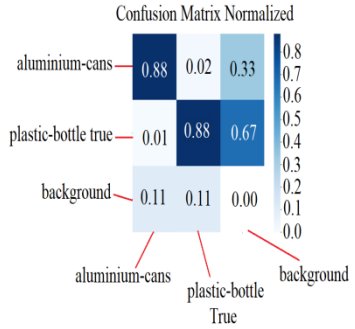


Fig. 24. Confusion Matrix.

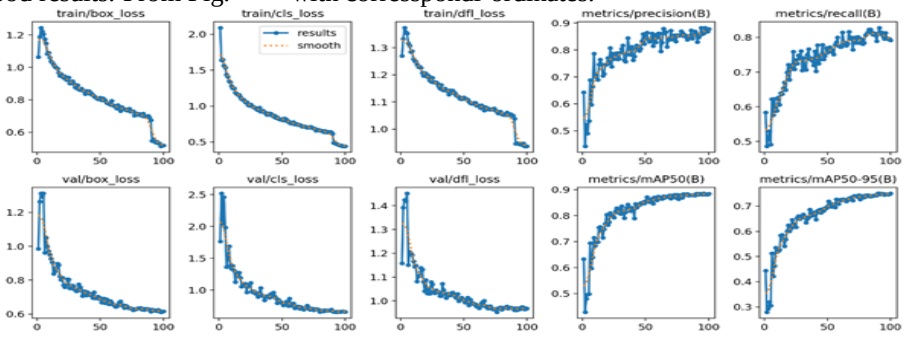


Fig. 25. Notable parameters after training.

Yolo model testing results after training are:

Fig. 26. Unobstructed case.

Fig. 27. Case covered by robot arm.

Fig. 28. Cans and plastic bottles are stacked against each other.

Fig. 29. In case there are objects other than cans, plastic bottles.

4.3. Verifing Position Transformation Matrix

To verify this content, we compare the coordinates after converting from pixels and the actual coordinates of the robot. As mentioned aboved, OpenCV has the function cv2.findHomography to help us do this more easily. Thence, we used Python combined with OpenCV to write a program that allows us to select 1 pixel on the camera, print out the coordinates with pixel units and coordinates according to the robot.

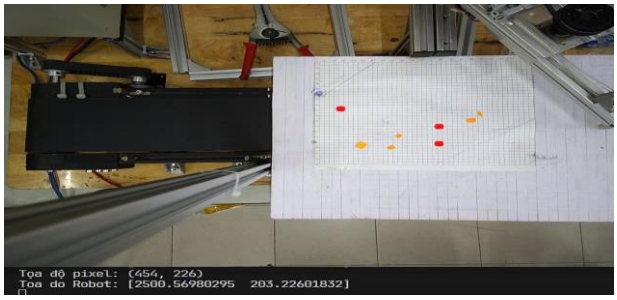


Fig. 30. Program to convert pixel coordinates to robot coordinates.

After verification, we get the results as shown in Tab. 1

Tab. 1. Error assessment after passing through the position transformation matrix

Pixel coordinates of the object	Data returned (mm)	Actual (mm)	Error (mm)
433, 276	302.52, 0.11	300, 0	<5
556, 222	241.95, 117.01	240, 120	<5
256, 24	36.85, -169.22	40, -180	<15
445, 268	288.17, 102.12	290, 100	<5
254, 212	235.98, -176.34	240, -180	<5

4.4. General Results

After the completion and experimental run, we obtained some parameters of the system as follows:

• Total number of cans and plastic bottles interspersed: 50, 25 of each type
  - Number of cans classified: 18 (72%)
  - Number of plastic bottles classified: 22 (90%)
  - Processing speed: 12 items/minute
• Only cans classified: 30
  - Number of cans classified: 25 (83%)
  - Processing speed: 12 items/minute

• Only cans classified: 30
  - Number of cans classified: 26 (86%)
  - Processing speed: 15 items/minute

4.5. GUI platform

GUI is presented in Fig. 31. The GUI allows connection to the microcontroller by selecting parameters such as connection port, transmission speed. In addition, the control interface has buttons to control the system on/off, buttons to control the robot arm operation, buttons to allow turning the camera on/off. In addition, there are also parameters displayed such as: current position of the robot's end mechanism, operating status of the robot and system, table displaying objects recognized by the camera.The camera is integrated with the Yolo model that has been trained to recognize previous objects and display on the object's bounding box, center coordinates, object name, as well as corresponding confidences.

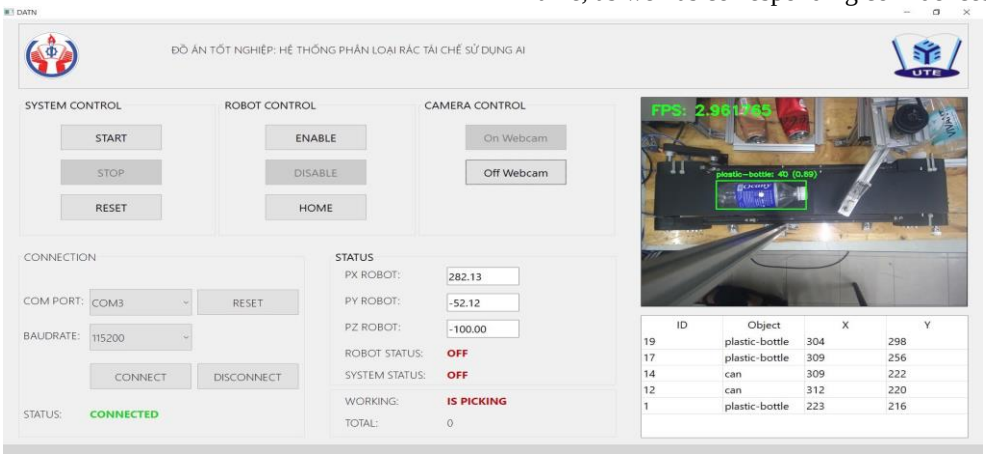


Fig. 31. GUI platform.

5. Conclusions

Through this research, we build an experimental hardware of Dobot M1 for garbage classification. Kinetic calculations, image processing, AI training through toolbox make system work well in realistic. The experimental results are examined through data from a GUI platform. This model is cheap and easy to use in our laboratory for training and researching for different tasks of robot field.

Acknowledgement

We want to give thanks to PhD. Vi-Do Tran (HCMUTE) due to his supervision for us and to PhD. Van-Dong-Hai Nguyen (HCMUTE) due to his help in fixing this paper. Operation of our experiment is shown in link:
https://www.youtube.com/watch?v=zrYJqBPZTkE

6. References

[1] Wang F. et al: "Impact of solid waste classification on urban ecology and economy in Beijing: A quantitative research", International Journal of Urban Sciences, 1–35, 2024.
[2] Bai H.: "Design of Garbage Classification System Based on Artificial Intelligence Technology", International Conference on Aviation Safety and Information Technology, Association for Computing Machinery, USA, pp. 167–170, 2022.
[3] Kishan P.S., Jaiswal J.: "Garbage Clssification and Detection for Urban Management", International Journal of Computer Trends and Technology, Vol. 69 Issue 9, pp. 17-26, 2021.
[4] Sharma A.et al: "Garbage Classification with Deep Learning Techniques," 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions, India, pp. 406-411, 2023.
[5] Zahavi A. et al: "A Dual-Arm Robot for Collaborative Vision-Based Object Classification", 17th Biennial Baltic Electronics Conference (BEC), Tallinn, Estonia, pp. 1-5, 2020.
[6] Link: https://www.ecns.cn/m/news/cns-wire/2019-11-04/detail-ifzqmsrp9611862.shtml
[7] Link: https://ampsortation.com/articles/amp-robotics-achieves-data-milestones-and-recycling-automation-breakthrough
[8] Link: https://spectrum.ieee.org/ai-guided-robots-are-ready-to-sort-your-recyclables
[9] Link: https://www.dobot-robots.com/products/dobot-series/m1-pro.html
[10] Sharma A. et al: "Object Detection using OpenCV and Python," *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2021, pp. 501-505
[11] Link: https://ieee-nitk.github.io/blog/opencv/
[12] Vo D.-H. et al, Position Control of 3-DOF Experimental Articulated Robot Arm using PID Controller, JFSC, vol. 3, no. 1, pp. 73–80, Mar. 2025.