Colour based sorting using a real time object detective pick and place robot

Ravinesh Perera
Department of Electrical Engineering
University of Moratuwa
Colombo, Sri Lanka
ravineshperera@gmail.com

Tharindie Pilapitiya
Department of Electrical Engineering
University of Moratuwa
Colombo, Sri Lanka
tharindie1018@gmail.com

Paramanathan Piragash
Depratment of Electrical Engineering
University of Moratuwa
Colombo, Sri Lanka
piragashparam36@gmail.com

***Abstract*—In the realm of automation and intelligent robotics, the ability to efficiently organize and sort items is a crucial challenge. In response to this, we present a pioneering initiative centered on a specialized robotic arm designed explicitly for the purpose of organizing a cluttered workspace littered with files. This report encompasses MATLAB codes focus on file detection, color identification, and Simulink simulations of the subsequent sorting process. In the latter segment of this report, the discussion extends beyond the virtual realm to explore the integration of these methodologies into the components of a physical robotic system.**

I. Introduction

Before moving into the era of robotics and automation sorting processes were performed by humans. The primary concern here lies in the sorting speed, notably affected by human limitations. The human eye's response time and the subsequent cognitive processing required for visual recognition contribute to a slower pace in color sorting. Yet, this setback can be mitigated with the integration of a computer system.

Additionally, accuracy poses a challenge in manual color sorting within the current industry setup. Operators handling numerous objects daily often succumb to errors due to fatigue, leading to compromised sorting precision. Conversely, machines consistently deliver accurate results, unaffected by repetitive tasks unless system faults occur.

Moreover, the cost of implementing a manual color sorting process is notably high. Scaling up to sort large product quantities demands a considerable workforce, along with additional expenses for shift arrangements and overtime. Consequently, the cumulative costs escalate substantially.

With the aim of mitigating above issues, this innovative solution leverages cutting-edge technology to detect, identify, and sort files based on their colors, streamlining the arduous task of file arrangement. The envisioned robot arm embodies a sophisticated system equipped with a high-resolution camera, enabling it to swiftly and accurately detect files strewn across a disordered table. Once identified, the arm employs color recognition algorithms to categorize these files based on their distinct hues. Through this process, each file's color becomes a pivotal factor dictating its placement into predetermined locations.

The following flowchart provides a concise overview of our process, offering a clear visual representation. This will serve as a foundation for a detailed discussion and deeper understanding of the processes as we progress further.

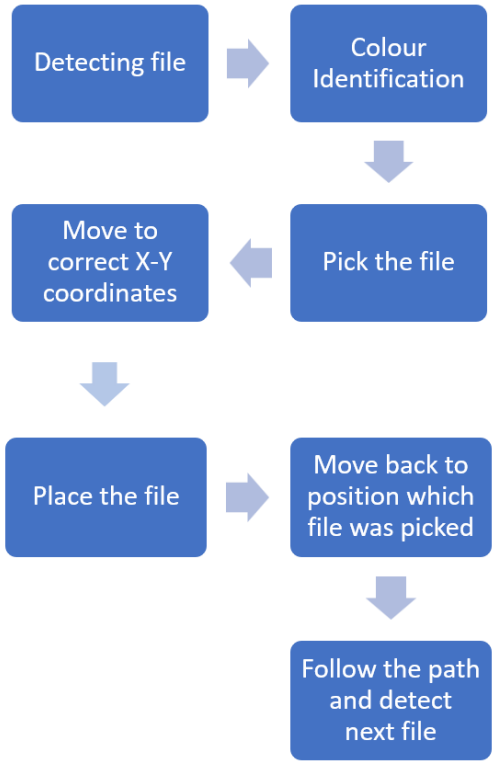

Fig. 1. Flowchart

II. Literature Review

Many research have been conducted on pick and place application. In industrial level, multi robot applications[1] are used for sorting purposes. Here the items will move on a conveyor belt and robots are placed alongside the conveyor belt. Each robot will monitor items which pass through and pick the relevant ones. Here a parameter like size, shape or colour can be used. This is efficient because each robot picks each set of items rather than one robot handling the

whole process. Since more than one robot operates energy consumption is high. Since our robot is designed for an office workspace rather than industrial environment, we do not need such huge investment.

For many pick and place applications, grippers made out of metals[2] are used. One reason for this is that it can handle large loads. In our case we only lift one file per time so there's no need of metal gripper. Also files might bend due to force applied from metal grippers so we thought of moving to a new gripper.

In wooden door companies vacuum area gripping system[3] is used for handling machinery. The components used here are a suction device with suction motor and a vacuum gripper which is a clamping tool with a suction pad. This is frequently employed in the creation of wall climbing robots that can adhere to smooth surfaces without endangering the item's surface. Though a complex mechanism is involved with the process, can be considered as effective.

Our robot application involves with colour identification. In many pick and place robots used in food industry like fruit picking, colour identification is involved. For the robots which are used in such applications artificial intelligence[4] is implemented to enhance image identification process. AI can be also used to calculate distance to the object which should be picked. Another method for image processing is reinforcement learning[5].

Regarding identifying the moment which the gripper should be activated and object should be picked up sensors play a major role. Sensor should have the capability either to calculate the distance towards the object so that gripper is activated after that distance is moved or give a signal to activate the gripper when it is touched on the object to be picked up. In first case ultrasonic sensors[6] are used while for the second scenario tactile sensors[7] or pressure sensors[8] have being used.

III. System Overview

A. Colour recognition

The initial part of this process involves detecting files and identifying its colour. Our robot should be capable to identify files scattered on the table. For that the whole surface should be checked. To make this task easy we can define a suitable path according to the workspace.

When going along this path camera will take a continuous video input. If it detects either red, blue, green or white, the colour will output and colour detection code halts so that pick and place can be executed.

Here we have assumed the surface of the table is not red, blue, green or white so that even if surface is detected by camera when files are not there, it won't disturb the functionality of the code by the surface being misidentified as a file. For the purpose of simulation, we also output the captured image so that we can verify the code. Figure 2 shows the matlab code which achieves the above task.

From onwards figure 3 up to figure 10 some captured images and output we got are included. First, we checked for green.

```
% Initialize webcam
cam = webcam;

% Capture loop
while true
    % Take snapshot
    img = snapshot(cam);

    % Convert image to HSV color space
    img_hsv = rgb2hsv(img);

    % Extract H (hue) channel
    hue = img_hsv(:,:,1);

    % Calculate histogram of hue values
    num_bins = 100;
    h_hist = imhist(hue, num_bins);

    % Find the bin with the maximum count
    [~, max_bin] = max(h_hist);

    % Map bin index to corresponding hue value
    max_hue = max_bin * (360 / num_bins);

    % Define hue ranges for red, green, blue, and white
    hue_ranges = [0, 30; 210, 270; 60, 150; 0, 360]; % Red, Blue, Green, White
    color_names = {'Red', 'Blue', 'Green', 'White'};

    % Check if the detected hue falls within any range
    color_detected = '';
    for i = 1:length(hue_ranges)
        if max_hue >= hue_ranges(i, 1) && max_hue <= hue_ranges(i, 2)
            color_detected = color_names{i};
            break;
        end
    end

    % Display the detected color
    disp(['Detected color: ' color_detected]);

    % Check if the detected color is red, green, blue, or white
    if strcmp(color_detected, 'Red') || strcmp(color_detected, 'Blue') || ...
            strcmp(color_detected, 'Green') || strcmp(color_detected, 'White')
        disp('Detected stop color. Stopping video input.');

        % Save the image
        imwrite(img, 'detected_color_image.jpg');
        break;
    end

    % Display the image
    imshow(img);
    drawnow;
end
```

Fig. 2. Matlab code

The code effectively identified the predominant colour of the surface as green, despite the presence of white letters. This ensure that letters or symbols printed on file surface won't disturb the functionality of code.

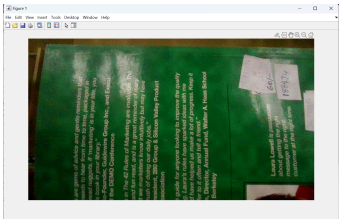

Fig. 3. Captured Image-Green

Then we checked for blue. The book we used had a light blue cover, which is not original blue. Still we got the output "blue" as we wished for, showcasing its capability to identify ranges of colours accurately.

The next target was to check if code is taking continuous input until one of the predefined four colours are detected. It remained consistent when directed to a wooden surface, until a red sheet is introduced producing the expected output.

Finally we wanted to check how light intensity affects our code. Here we identified light intensity is one of our

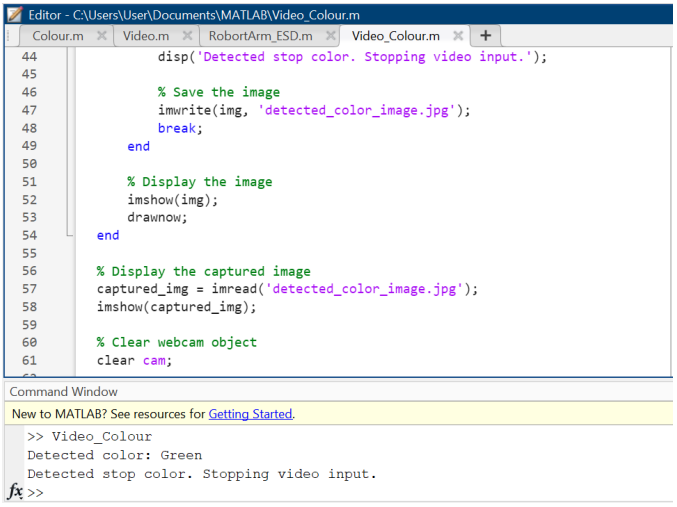

Fig. 4. Output-Green

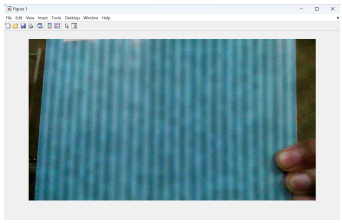

Fig. 5. Captured Image -Blue

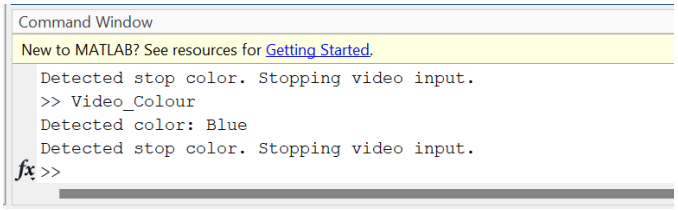

Fig. 6. Output-Blue

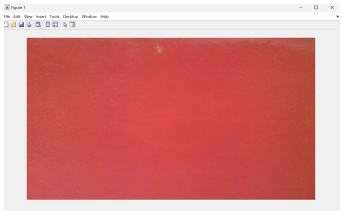

Fig. 7. Captured Image-Red

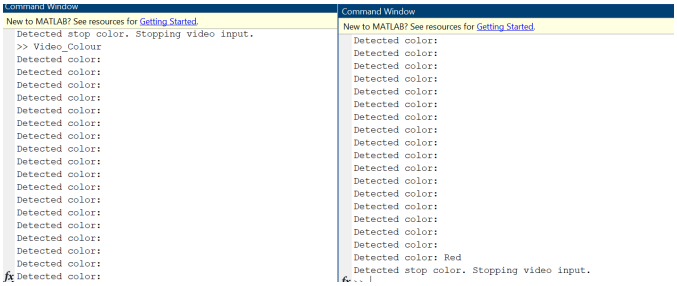

Fig. 8. Output-Red

limitations. Originally we wanted to detect yellow. But with lighting problems it always detected one of the RGB colours instead of yellow. Then code was adjusted to detect white.

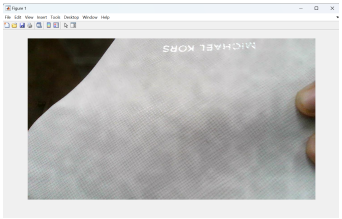

Fig. 9. Captured Image-White

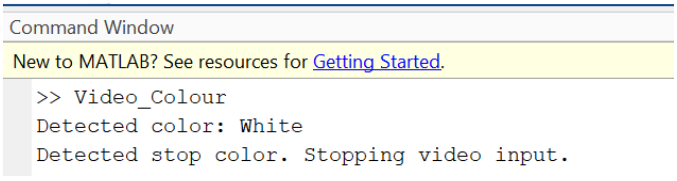

Fig. 10. Output-White

Our findings highlight that addressing lighting effects is critical for accurate color identification when dealing with more colors. This may necessitate extensive testing to develop the code to mitigate lighting influences or require an optimally lit workspace for ideal performance.

B. Robot arm movement

After colour is recognized pick and place task is executed. When the color of the file is provided as input, the file pick-and-place task is executed by manipulating the robot arm to predefined X-Y coordinates. These coordinates, currently set as samples for simulation purposes, can be tailored to suit specific workspace requirements. The robotic arm picks up the file, reaches the designated destination, places the file, and then returns to the position which it picked the file. Pre-defined path will be covered staring from this point and the process is repeated until whole path is covered. Figure 11, 12 and 13 shows the related simulink and matlab works.

Our robot arm, equipped with two degrees of freedom, functions as a tabletop robot. It has a cylindrical base, two links and a gripper which is marked by a cube, incorporates a soft, air suction-based gripper for file handling.

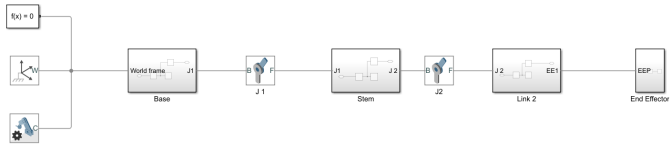

Fig. 11. Arm Simulink

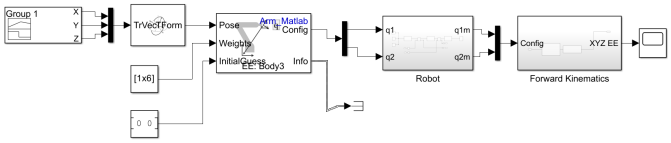

Fig. 12. Final Simulink

We also thought of designing a portable shelf to store files with cabins for each colour. When the robot is placed on the table, we can place the portable shelf accordingly. If the robot is taken to a new workspace, we can pull the shelf to the new place and locate accordingly. If not for this portable shelf, the table might have to locate near the storage spaces and even the robot have to be kept on surface in a specific angle only so that it can reach storage spaces. Likewise that limitation is mitigated. We can develop this portable shelf with wheels to be pulled. As a further development we can make the movement of the robot self-driven. If the shelf is able to track the movement of table robot, which will be probably carried by a human and placed on a new surface when moving to a new table, the shelf will follow the path and stay in place near the table accordingly. This can be achieved by using of sensors and codes which are not addressed in this report.

IV. System Design

Now we'll look into practical aspects of the robot arm.

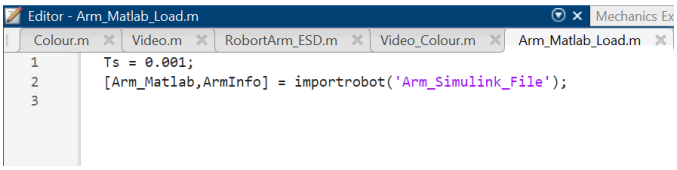

Fig. 13. Arm Matlab

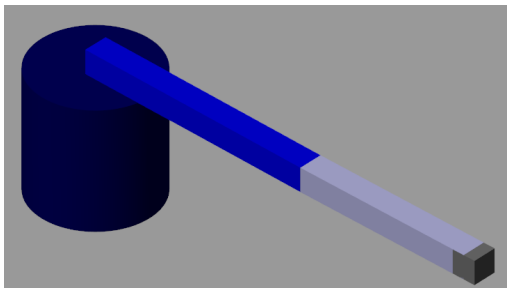

Fig. 14. Robot Arm

A. Camera

A robot equipped with machine vision that doesn't rely on depth or distance information typically opts for conventional 2D digital cameras. However, these cameras are sensitive to ambient light conditions, requiring careful attention to ensure proper lighting in the area where the files are situated. Moreover, 2D cameras might misinterpret a mirror image as a distinct object, introducing potential identification challenges. Additionally, the process of capturing images is inherently slow, demanding relative stillness of both the camera and the object.

B. Sensor

As the gripper approaches the file, the robot needs to identify the moment which it should activate the gripper. For this purpose, a sensor becomes crucial. In this scenario, a touch sensor comes into play. It possesses the capability to detect and register physical touch. Beyond touch, it can detect proximity without direct physical contact to a certain extent. Touch sensors operate akin to switches: upon being touched, pressed, or subjected to force, they activate and function as closed switches. Conversely, when the pressure or contact is removed, they revert to an open switch state.

C. Gripper

The Vacuum Area Gripping System is a tool designed to secure and move workpieces using a clamp vacuum based on suction. This system consists of a suction device connected directly to the suction valve and motor, with a flexible hose serving as the connecting component. An air cylinder is employed to mechanically manipulate the suction valve. The process involves the gantry handle descending to grip the file using suction, relocating it to the desired position, and eventually lowering to the lowest point to place the file.

To design this system, a formula is utilized to determine the necessary suction force required against the vacuum to lift files:

Vertical Suction Force (Fs) = (m/u) x (g + a) x S

Here, m represents the mass, u is the coefficient of friction, g is gravity's acceleration, a is the workpiece's acceleration, and S is a safety factor.

The Vacuum Gripper, a mechanical controller utilizing vacuum for grasping moving objects within the workspace, is adaptable and versatile in its design. This clamping tool incorporates a suction pad where compressed air is directed through the suction pump into the air connector hole. To reduce impact between the vacuum body and the object being suctioned, a dampening sponge is integrated into the vacuum chamber. Additionally, a membrane valve, situated atop the sponge, filters suction air to prevent debris from entering the vacuum chamber.

This system allows us to lift various types of sheets, not just hard-surfaced files. When handling paper with a hand-shaped gripper, there's a risk of bending or folding, but with this method, that risk is mitigated.

D. Base, links and motors

The foundation of the robot, crucial for supporting the entire assembly, is crafted from aluminum. Brass standoffs provide stability between the support elements of the robot assembly. Within the base, a motor is fixed and connected to Link 1, also made from aluminum, transmitting force to Link 2. The first end of Link 1 is linked to motor 2, while its other end is connected to Link 2, transferring power required for movement. Links are secured to each other using screws. The movement of Link 2 is responsible for controlling the gripper's motion.

E. NodeMCU

The NodeMCU functions as the controller and processor in this setup. It generates a PWM (Pulse Width Modulation) signal to drive the motor and operates on a 5-volt power supply sourced externally. With built-in Wi-Fi capabilities, the NodeMCU utilizes its digital pins, specifically from D0 to D3, to produce the PWM output signal.

F. Battery

Initially, rechargeable batteries often carry a higher upfront cost compared to disposable ones. However, their overall cost-effectiveness and reduced environmental footprint stem from their ability to be recharged multiple times at a low cost before requiring replacement. Certain rechargeable battery variants come in identical sizes and voltages as disposable options, allowing for interchangeable use between the two types

G. Converter

This converter should be able to convert a 12-volt input to a stable 5-volt output. We need this because battery supplies 12V while NodeMCU works with 5V power supply. Utilizing capacitors and the 7805 IC, it addresses fluctuations in voltage sources within circuits, ensuring a consistent output. The 7805 IC, belonging to the 78xx series of fixed linear voltage regulator ICs, stabilizes voltage variations. The "xx" in the series denotes the fixed output voltage, with the 7805 IC specifically providing a regulated +5 volts power supply and allowing for the addition of a heat sink if needed.

V. Results

A. Impact of light intensity towards colour detection

A new MATLAB code was developed to assess the influence of lighting on color detection in a live video feed. The experiment involved detecting red, blue, green, and yellow hues under varying lighting conditions.

The figure 16 illustrates color detection with the webcam directed at a wall picture under daylight with no additional lighting. It's evident that blue and yellow colors are not accurately detected in this setting.

Subsequently, the figure 17 demonstrates color detection under the same conditions but with a single light turned on. Although red and green object detection remains relatively consistent, there's a slight improvement in the detection of blue and yellow hues, indicating the impact of additional lighting on color recognition.

In the figure, with two lights turned on, the detection of blue significantly increases. However, due to limitations in the MATLAB code and webcam, the colors may not be detected with absolute precision. Nonetheless, the results vividly highlight the substantial impact of lighting conditions on object detection. These findings underscore the critical importance of ensuring adequate and consistent lighting when the robotic system operates. Proper lighting significantly enhances the accuracy and reliability of color detection, thereby optimizing

```
cam = webcam;
preview(cam)
% Define color ranges for red, blue, green, and yellow
redRange = [140, 255, 60, 140, 60, 120]; % Define red range in RGB format
blueRange = [0, 70, 100, 255, 190, 255]; % Define blue range in RGB format
greenRange = [0, 100, 0, 160, 0, 100]; % Define green range in RGB format
yellowRange = [150, 255, 150, 255, 0, 150]; % Define yellow range in RGB format

while true
    % Capture a frame from the webcam
    img = snapshot(cam);

    % Detect red objects
    redMask = (img(:,:,1) >= redRange(1) & img(:,:,1) <= redRange(2)) & ...
              (img(:,:,2) >= redRange(3) & img(:,:,2) <= redRange(4)) & ...
              (img(:,:,3) >= redRange(5) & img(:,:,3) <= redRange(6));
    redObjects = img .* uint8(redMask);

    % Detect blue objects
    blueMask = (img(:,:,1) >= blueRange(1) & img(:,:,1) <= blueRange(2)) & ...
               (img(:,:,2) >= blueRange(3) & img(:,:,2) <= blueRange(4)) & ...
               (img(:,:,3) >= blueRange(5) & img(:,:,3) <= blueRange(6));
    blueObjects = img .* uint8(blueMask);

    % Detect green objects
    greenMask = (img(:,:,1) >= greenRange(1) & img(:,:,1) <= greenRange(2)) & ...
                (img(:,:,2) >= greenRange(3) & img(:,:,2) <= greenRange(4)) & ...
                (img(:,:,3) >= greenRange(5) & img(:,:,3) <= greenRange(6));
    greenObjects = img .* uint8(greenMask);

    % Detect yellow objects
    yellowMask = (img(:,:,1) >= yellowRange(1) & img(:,:,1) <= yellowRange(2)) & ...
                 (img(:,:,2) >= yellowRange(3) & img(:,:,2) <= yellowRange(4)) & ...
                 (img(:,:,3) >= yellowRange(5) & img(:,:,3) <= yellowRange(6));
    yellowObjects = img .* uint8(yellowMask);

    % Display the detected objects
    subplot(2, 2, 1), imshow(redObjects), title('Red Objects');
    subplot(2, 2, 2), imshow(blueObjects), title('Blue Objects');
    subplot(2, 2, 3), imshow(greenObjects), title('Green Objects');
    subplot(2, 2, 4), imshow(yellowObjects), title('Yellow Objects');

    % You can add logic here to perform actions based on the detected objects

    pause(1); % Adjust the pause time as needed
end

% Clean up by closing the video preview and deleting the video object
closepreview(cam);
clear(cam);
```

Fig. 15. matlab code to detect colour from live video

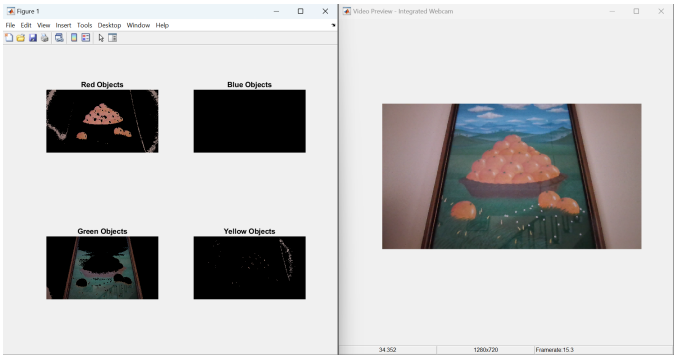

Fig. 16. No additional lighting

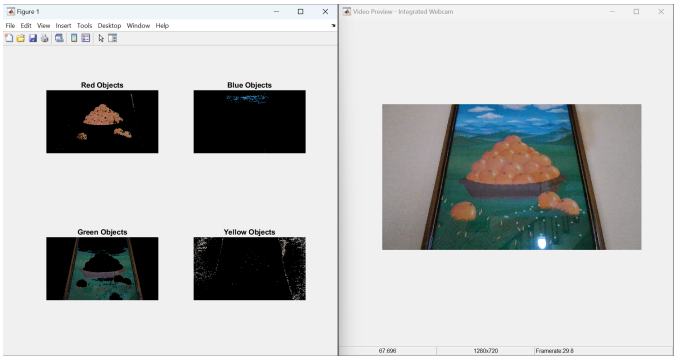

Fig. 17. Single light turned on

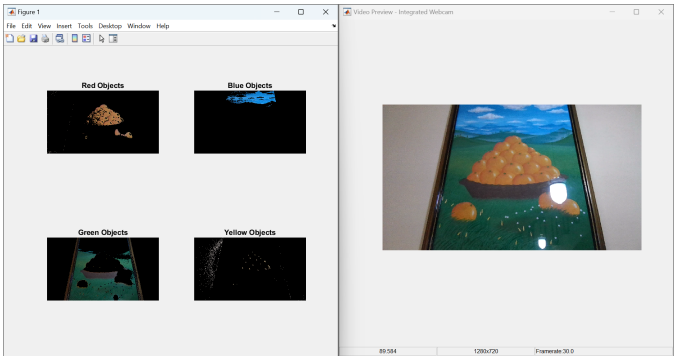

Fig. 18. Two light turned on

the system's performance during file organization tasks.

B. Robot arm movement

The figure 19 depicts the change of x-y coordinates of the robot arm with time.

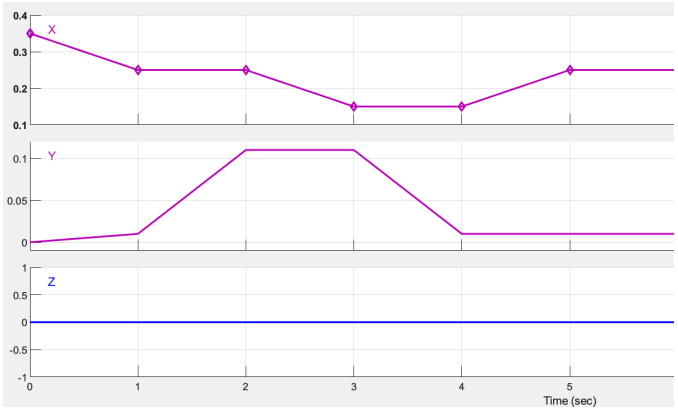

Fig. 19. X-Y-Z coordinates with time

C. Time and Energy consumption

The overall Energy Consumption (EC) during this operation is presented as a function of TET (Total Execution Time). Notably, the energy required for performing these operations increases concerning TET, deviating from a global optimum Topt, for both low and high TET values, representing slow or fast speeds, respectively.

Conventional scheduling optimization methods, like time-optimal point-to-point motions, typically assume that the robot functions at its maximum speed when permitted by scheduling constraints, remaining still otherwise. However, such a task planning approach might lead to higher energy consumption due to forceful accelerations and prolonged idle times, where energy is dissipated in counteracting gravitational forces. It's essential to note that achieving extremely low TET is unfeasible due to limitations in the nominal torque of the actuation system. Conversely, excessively high TET values are deemed unacceptable as they could detrimentally impact production rates.

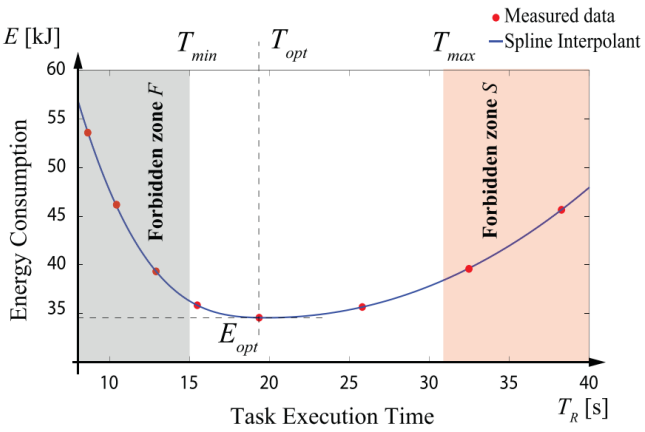

Fig. 20. Energy consumption with time[12]

VI. Discussion

The robot's capacity to aid office tasks is evident, yet there exists potential for additional advancements and expansions in its functionality.

A. Shape Detection Development

Expanding beyond color detection, our robot's capabilities can evolve to include shape recognition. Beyond files and papers, an office table often hosts various objects such as stationery, electronic devices, and debris. By incorporating both color and shape detection, our robot can advance to serve as a workspace organizer or table cleaner.

While this enhancement broadens the robot's identification abilities, it's important to note that workspace objects rarely conform solely to basic geometric shapes. Evaluating the soft gripper's capacity to handle these diverse objects remains crucial.

B. Mobility Enhancement

Presently, the robot necessitates human placement onto a table or workspace before commencement of its tasks. It lacks autonomous mobility and thus requires human intervention to relocate between tables or workspaces. To augment its functionality, we envision a development phase enabling the robot to autonomously transition between work areas.

This advancement would empower the robot to complete tasks at one location and proceed to the next, independently identifying objects that require sorting. This automated, continuous process eliminates the need for human intervention in its movement between tables, fostering efficiency and autonomy in its operations.

Conclusion

In conclusion, the development and exploration of a robotic arm designed for file organization have unveiled promising prospects in the realm of automation. The integration of sophisticated algorithms within MATLAB codes and Simulink simulations has showcased the viability of employing color-based detection and sorting methodologies. While the software-based algorithms serve as the cognitive backbone, the discussion on physical hardware integration elucidates the realization of these concepts in the real world. However, it's imperative to acknowledge the ongoing scope for refinements, particularly in mitigating the impact of external factors such as lighting variations on color recognition.

In essence, this report serves as a testament to the transformative capabilities of automation, underscoring the potential for innovation in streamlining everyday tasks and enhancing operational efficiency.

Acknowledgement

Special thanks are extended to Professor Buddhika Jayasekara and Dr. Ruwanthika for their invaluable insights, which proved immensely helpful in shaping this project. Additionally heartfelt appreciation is extended to all the instructors whose contribution and support were integral to the completion of this work.

Followings are the individual contribution of our group members.

Ravinesh Perera 210473M- Conceptual design of the robot arm

Tharindie Pilapitiya 210474R- File detection and colour recognition matlab code, new matlab code to identify colour from live video, report writing

Piragash Paramanathan 210476B- Robot arm design and movement(Simulation) using matlab and simulink

References

[1] K. Castelli, A. M. A. Zaki, and H. Giberti, "Development of a Practical Tool for Designing Multi-Robot Systems in Pick-and-Place Applications," *Robotics*, vol. 8, no. 3, p. 71, Aug. 2019, doi: https://doi.org/10.3390/robotics8030071.

[2] K. Ghadge, S. More, P. Gaikwad, and S. Chillal, "ROBOTIC ARM FOR PICK AND PLACE APPLICATION," *International Journal of Mechanical Engineering and Technology (IJMET)* , vol. 9, no. 1, pp. 125–133, Jan. 2018.

[3] Julfrianto and Edilla, "Implementation analysis of vacuum area gripping system in pick and place machinery in wooden door company ," *JTTM: Jurnal Terapan Teknik Mesin*, vol. 4, no. 2, pp. 172–183, Oct. 2023.

[4] N. L. Dung, N. H. Giap, P. H. Lam, and H. T. Nhien, "Apply artificial intelligence to the robot picking strawberries," *Journal of the Austrian Society of Agricultural Economics*, vol. 17, no. 9, pp. 585–593, Sep. 2021.

[5] N. M. Gomes, F. N. Martins, J. Lima, and H. Wörtche, "Reinforcement Learning for Collaborative Robots Pick-and-Place Applications: A Case Study," *Automation*, vol. 3, no. 1, pp. 223–241, Mar. 2022, doi: https://doi.org/10.3390/automation3010011.

[6] V. A. Zhmud, N. O. Kondratiev, K. A. Kuznetsov, V. G. Trubin, and L. V. Dimitrov, "Application of ultrasonic sensor for measuring distances in robotics," *Journal of Physics: Conference Series*, vol. 1015, p. 032189, May 2018, doi: https://doi.org/10.1088/1742-6596/1015/3/032189.

[7] R. S. and M. Valle, "Tactile Sensing for Robotic Applications," *Sensors: Focus on Tactile Force and Stress Sensors*, Dec. 2008, doi: https://doi.org/10.5772/6627.

[8] A. M. Almassri *et al.*, "Pressure Sensor: State of the Art, Design, and Application for Robotic Hand," *Journal of Sensors*, vol. 2015, pp. 1–12, 2015, doi: https://doi.org/10.1155/2015/846487.

[9] S. V. Chhaya, S. Khera, and P. Kumar S, "BASIC GEOMETRIC SHAPE AND PRIMARY COLOUR DETECTION USING IMAGE PROCESSING ON MATLAB," *IJRET: International Journal of Research in Engineering and Technology*, vol. 4, no. 5, pp. 505–509, May 2015.

[10] L. J. SHEN and I. HASSAN , "DESIGN AND DEVELOPMENT OF COLOUR SORTING ROBOT," *Journal of Engineering Science and Technology*, EURECA 2014, Special Issue, pp. 71–81, Jan. 2015.

[11] R. Mourya, A. Shelke, S. Satpute, S. Kakade, and M. Botre, "Design and Implementation of Pick and Place Robotic Arm," *International Journal of Recent Research in Civil and Mechanical Engineering (IJRRCME)*, vol. 2, no. 1, pp. 232–240, Sep. 2015.

[12] M. Pellicciari, G. Berselli, F. Leali, and A. Vergnano, "A method for reducing the energy consumption of pick-and-place industrial robots," *Mechatronics*, vol. 23, no. 3, pp. 326–334, Apr. 2013, doi: https://doi.org/10.1016/j.mechatronics.2013.01.013.