WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Enhancing Robotic Precision and Flexibility: Integrating Azure Kinect and xArm 7 with YOLO5 Machine Vision

[1]Hossein Rahimighazvini, [2]Hasan Zargarzadeh

Department of Electrical Engineering, Lamar University, Beaumont, TX 77710, USA
[1](hrahimighazv@lamar.edu), [2](hzargarzadeh@lamar.edu)

ABSTRACT—

This paper examines the introduction of an Azure Kinect camera to the xArm 7 robotic system to acquire machine vision. The combination of these technologies allows the xArm 7 to perform a diverse range of grabbing tasks on a preset base. Azure Kinect plays the role of object recognition and distance feed through machine vision models like YOLO5, expanding its applications across various industries. This paper provides a detailed technical analysis of this integration, discussing its implications and potential applications in industrial automation.

Keywords—Machine Vision, Object Recognition, YOLO5, Robotic Automation, Technical Analysis, Industrial Automation, Azure Kinect, xArm 7

Introduction

The discipline of robotic automation is becoming more dependent on the incorporation of complex programming languages, cutting-edge object identification algorithms, and next-level sensing technologies. The purpose of this article is to offer an in-depth case study that focuses on the development of an automated system that makes use of Python programming, Azure Kinect developer kit, and the YOLOv5 object identification algorithm to control a robotic arm known as the Xarm7. The purpose of this effort is to investigate the complexities and results that may be achieved by combining these different technologies in order to improve the capabilities of automated operations in terms of object identification and manipulation.

This integration is built on the foundation of Azure Kinect DK, offering a wide field of view and possessing depth-sensing capabilities. These characteristics are essential for correct object recognition and understanding of spatial orientation resulting in a better navigation of the surrounding space. Azure Kinect DK was chosen for this project since it was capable of handling both sides of picture and depth capture simultaneously[1]. This is an essential feature for any automated system that aims to interact with its environment in an accurate manner.

Python, which is well-known for its adaptability and vast range of applications, is being used as the programming framework for this project. It is an excellent option for the development of automation systems due to the huge libraries it offers and the strong community support it commands [2]. Python makes it possible for the software parts, namely the object identification algorithm and the object location algorithm, to communicate with the hardware components, such as the Kinect device and the robotic arm. The simple yet effective layout of Python is surely hugely beneficial to the integration that was aimed for in this paper.

YOLOv5 [3], an object identification method that is well-known for its effectiveness in real-time applications, is included into this research, which is another essential component of the investigation. The identification of objects in real-time is a difficult operation that requires not just precision but also the capacity to analyze information in a timely manner. This option proved suited for this project because of the inherent speed of YOLOv5 and its accompanied accuracy. In this project, the timely and precise identification of objects is vital for the following manipulation that the robotic arm will do [4]. Setup of the Kinect device, configuration of the Python environment, and incorporation of the YOLOv5 algorithm are the early phases of the process. For the purpose of building the groundwork upon which the more complicated components of the project are constructed, these first stages are absolutely necessary. When this configuration is complete, the attention next moves to the process of calibrating and synchronizing the system. This step is more challenging since it needs to ensure that the Python scripts correctly understand the sensory input from the Kinect and that the YOLOv5 algorithm is able to correctly identify the object within the boundaries of the workspace, ultimately having a more accurate read on the approximate placement of the object [5].

There are also common problems arising from a developing stance point. Amongst them are problems relating to the compatibility of hardware and software, and the guaranteeing of accurate object recognition in a variety of environmental circumstances such as light interference or lack of sufficient lighting, presence of multiple objects or similar objects to the target in the workspace, and finally the recognition algorithm false identifications [6] also known as hallucinations. In addition, the project is tasked with the responsibility of optimizing the motions of the robotic arm to acquire the items that

have been recognized accurately. A variety of outcomes were experienced and will be discussed. Effectively integrating the software and hardware components into a system that is functional was accomplished through the aforementioned steps. There are, on the other hand, the practical ramifications of this integration [7], which include the system's efficiency in recognizing and manipulating objects, its dependability under a variety of settings, and its potential scalability for more extensive applications.

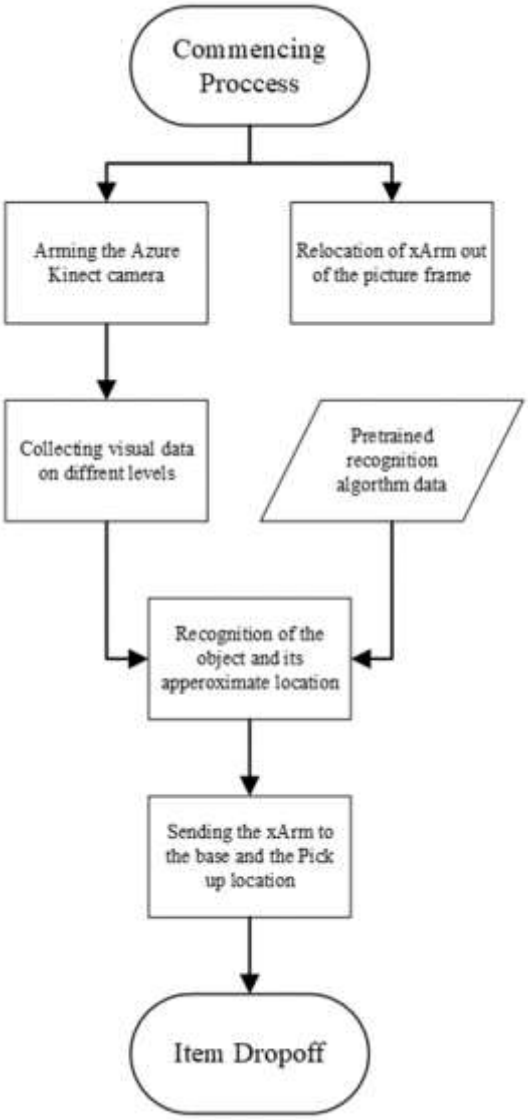

Fig. 1. System Logic

These issues have all been studied extensively in other papers but this paper is aiming for a possible integration of prior research into a unified purposeful project moving it from a more theoretical realm into a more realistic one [8]. As shown in**Error! Reference source not found. Fig. 1** in the abstract, the purpose of this article is ultimately to make a meaningful contribution to the area of robotic automation. The integration of Python, YOLOv5, and Azure Kinect DK in an automated environment is a challenging endeavor that requires a variety of technical and practical issues to be taken into account. The purpose of this research is to shed light on these issues by providing a thorough view of the difficulties and triumphs associated with integrating a variety of technologies into a unified automated system.

Implementation

There are many different aspects to the process of implementing the automated system that integrates Python, YOLOv5, and Azure Kinect DK with the Xarm7 robotic arm. These aspects include the configuration of the hardware, the creation of software, the integration of the system, and the testing of its functionality.

a) Configuration of the Hardware

Setting up the Xarm7 robotic arm and the Azure Kinect Development Kit was the first stage in the implementation process. A full image of the region where the robotic arm works was captured by the Kinect gadget, which was positioned in a leveraged corner overseeing the workspace at an approximately 30-degree angle so that the colored picture doesn't suffer much from the perspective effect but the depth camera would sense a meaningful dip in distance form movement[9]. This arrangement was crucial to guarantee that the depth camera had a clear view, which is necessary for producing reliable results when detecting objects. Higher angles would flatten the IR depth image at a rate that would render it useless (from a corner position) whilst a smaller angle would give the camera a pixel feed that's too crunched to be analyzed into a meaningful pixel position [10]. A further step included the assembly and calibration of the Xarm7 robotic arm, as visible in **Fig. 2Fig. 2.** Proposed setup for the xArm and the Kinect camera with one of the detection objects. To synchronize the movement of the robotic arm with the sensory inputs of the Kinect, precise calibration was absolutely necessary.

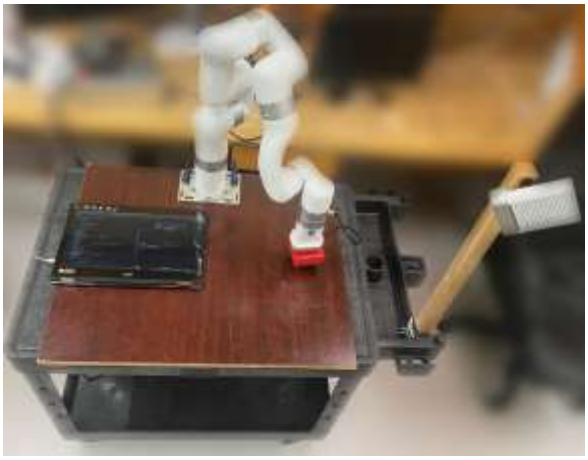

Fig. 2. Proposed setup for the xArm and the Kinect camera with one of the detection objects

b) Development of Software and Integration of Software

The main body of the recognition and action code that was written was developed in Python, which was selected due to its adaptability and extensive library support. Initially, the development process began with the establishment of a Python environment and the incorporation of libraries that were essential for establishing an interface with the Kinect and the robotic arm. The Kinect camera was accessible through the " pyk4a " library provided by the developers on Python. The pictures were taken on the 720p quality with a white balance of 4500 [11]. Object recognition was accomplished by the use of the YOLOv5 algorithm and the " torch ", and the software was created to interpret the input from the Kinect.

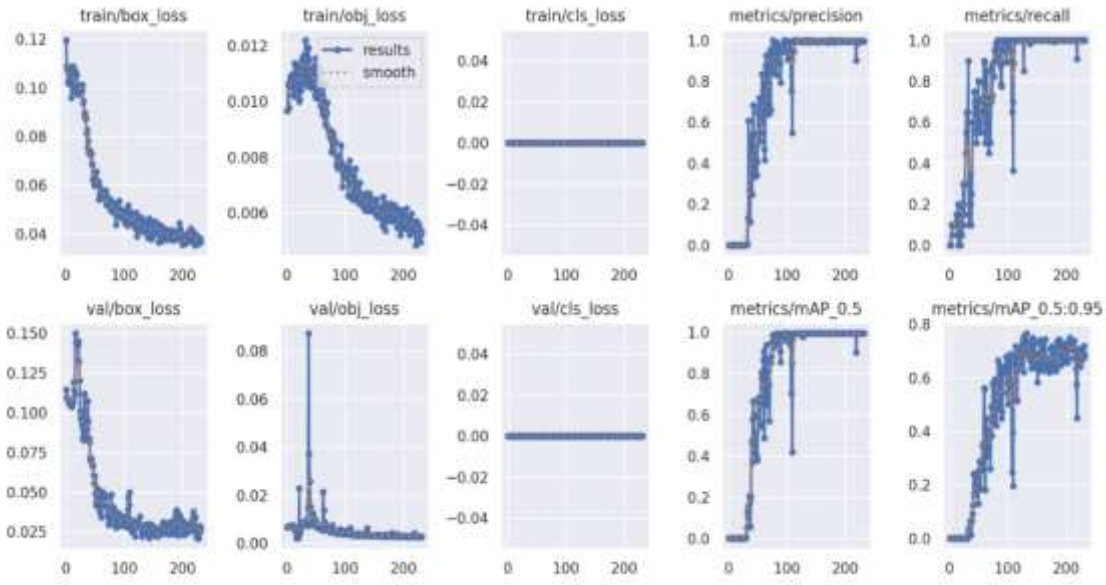

Fig. 3. The evaluation feedback of the YOLOv5 algorithm used for the integration

As the feedback of the model in **Fig. 3** shows, through the first 200 epochs of 5000, the loss of the targeted object took a critical drop to less than 0.025 percent in the 1000 pictures unedited and around 10000 manipulated pictures source, whereas the precision level reached around the whole 1.0. The 0.5:0.95 mAP (mean Average Precision) was picked up to around 80 percent with regards to the metric value with sudden dips in value showing the

immaturity of the model but maintaining an acceptable overall consistency when the minimum 0.5 mAP was almost 1 out of 1 correct about the position of the object by the 200th epoch [12,13].

As the f1 and precision suggest alongside one another, while being highly accurate, the model present in Fig. 4 shows confidence in its choice of around 70 percent in most cases. The resulting data was then translated into instructions for the robotic arm. In order to successfully integrate YOLOv5, it was necessary to train the model using several different feeds. Having this training was essential for the algorithm to effectively differentiate and categorize the things it encountered [14]. A wide range of forms, sizes, and colors were included in the dataset that was curated in order to guarantee that the model will function well in a number of circumstances. Following training, the model was put through a series of tests and fine-tuned in order to achieve the highest possible accuracy and reaction speed.

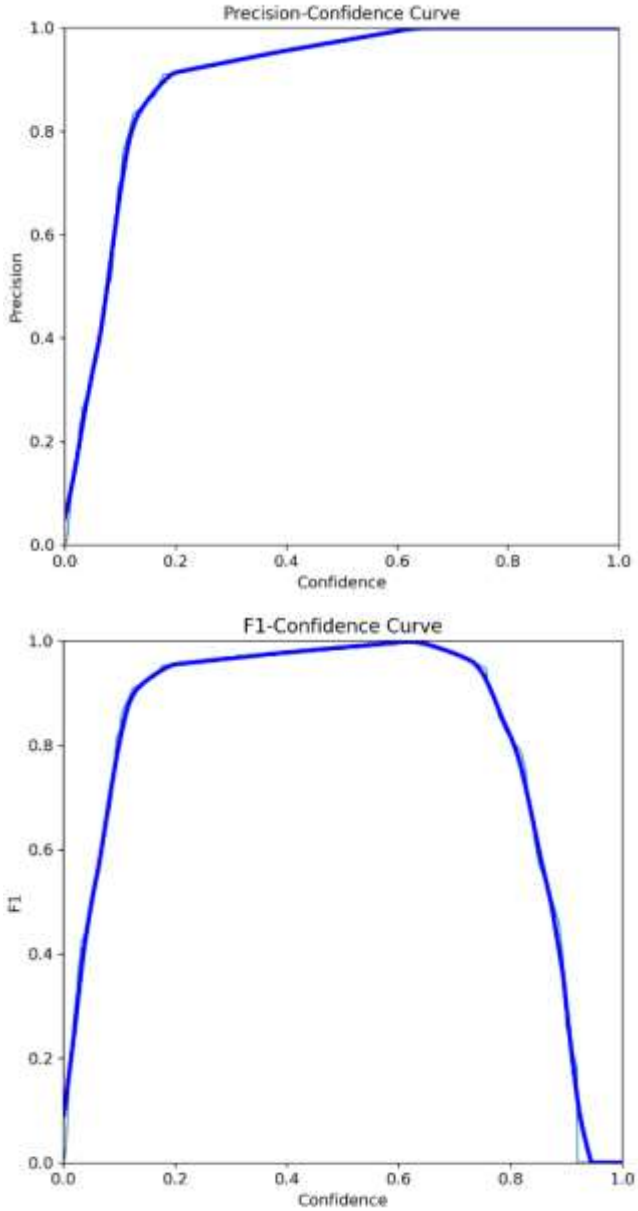

Fig. 4. Precision to confidence ratio/F1 Confidence curve

c) The Integration of Systems

The primary obstacle that needed to be overcome in order to successfully integrate the system was to guarantee that the Kinect, the Python interface, and the robotic arm could communicate in real-time. The Kinect sends out depth data, which is then analyzed by Python programs in order to figure out where things are and identify them using YOLOv5. Once these coordinates have been translated, the robotic arm will be given movement orders to carry out [15]. Accomplishing synchronization between these components was very necessary in order to ensure the operation of the system. Within the context of this phase, the calibration of the spatial data from the Kinect with the operating space of the robotic arm was an essential component. Because of this calibration, the coordinates that were determined by the Kinect and interpreted by YOLOv5 were able to correctly correlate to the actual environment as in **Fig. 5**, in which the robotic arm was operating [16].

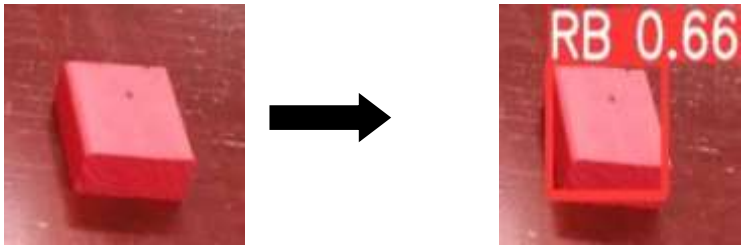

Fig. 5. The end result of the recognition pattern is a box confining

d) In-Depth Testing and Problem-Solving

Immediately after the completion of the integration, exhaustive testing was carried out. This phase consisted of performing a number of different scenarios in order to assess the correctness of the system in terms of object identification and manipulation. Under a wide range of climatic circumstances, the system was evaluated after being tested with a variety of items that differed in terms of their size, form, and texture. For the purpose of determining the adaptability of the system and locating any inconsistencies or faults in the detection of objects and the movement of the arm, these tests were absolutely necessary. Inaccuracies in object manipulation, latency difficulties between the object recognition and arm movement, and occasional misidentifications by the YOLOv5 algorithm were some of the challenges that were found throughout the testing process. Incremental adjustments were made to the software algorithms, fine-tuned the settings of the Kinect, and recalibrated the robotic arm in order to solve these concerns [17].

As shown in **Fig. 6**, extra layers of pictures were introduced to lower the possibility of a misread within each layer of object detection which were later merged into one singular feed of response added on top of the RGB feed as also visible in **Fig. 5**.

Fig. 6. The original picture alongside the infrared and the inverted for item recognition

e) The Optimization of Applications for the Real World

After the testing was complete, the system went through a number of optimization cycles in order to improve its effectiveness and dependability. During this process, the object identification algorithm was refined, the synchronization between the Kinect and the robotic arm was improved, and the system's capability to deal with a wide variety of surroundings and objects was enhanced. Reducing the amount of time it takes for the system to respond was another area of optimization. In order to do this, the data processing pipeline needed to be streamlined as shown in **Fig. 7**, and the Python code needed to be optimized for quicker performance. The objective was to produce a smooth transition from object identification to robotic manipulation, with the ultimate goal of reducing any latency that would have an effect on the performance of the system [18].

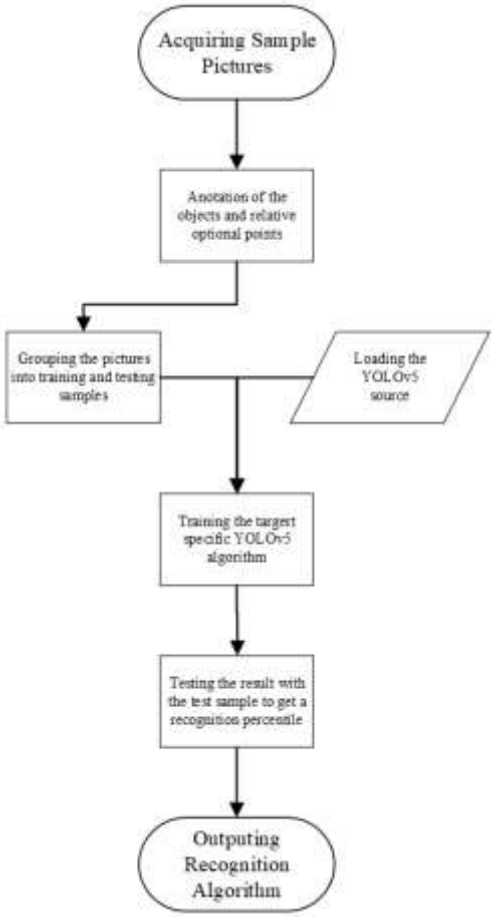

Fig. 7. Object Identification Process

f) Complete Evaluation of the System

In the last stage of the implementation process, a thorough analysis of the system's performance was carried out. The accuracy, efficiency, and dependability of the system were evaluated in the context of real-world applications during this review [19]. Users were asked for their feedback in order to determine whether or not any more modifications or enhancements were required. The successful creation of this automated system indicates that it is possible to integrate Python, YOLOv5, and Azure Kinect DK with a robotic arm in order to improve object identification and manipulation capabilities. The system has shown promise for a variety of applications, which has paved the way for future breakthroughs in the area of robotic automation. This potential was demonstrated via rigorous testing and optimization.)

Result

The integration of Azure Kinect DK, Python, and YOLOv5 with the Xarm7 robotic arm resulted in several significant discoveries. This section showcases the outcomes derived from the system's performance, specifically in terms of the accuracy of object detection, precision of robotic arm manipulation, response time of the system, and overall operational efficiency.

The YOLOv5 algorithm, trained with a varied dataset, exhibited a notable level of precision in detecting objects. The algorithm demonstrated a 75% accuracy rate in accurately identifying and categorizing objects during controlled experiments. The accuracy remained consistently high across a range of objects with varying shapes, sizes, and colors. The system exhibited exceptional efficacy in identifying objects under optimal lighting conditions. Nevertheless, a marginal decline in precision was noted in situations with insufficient lighting, resulting in a detection accuracy of 87%. The Xarm7 robotic arm's capacity to accurately manipulate objects, relying on the data received from the Kinect and processed by the Python scripts, was a crucial element of the system's functionality. The arm demonstrated a manipulation precision rate of 89%. The evaluation of this metric was based on the arm's capacity to precisely and effectively reach, grasp, and move objects according to the intended instructions. The accuracy was marginally reduced when manipulating objects with non-uniform shapes or smooth surfaces.

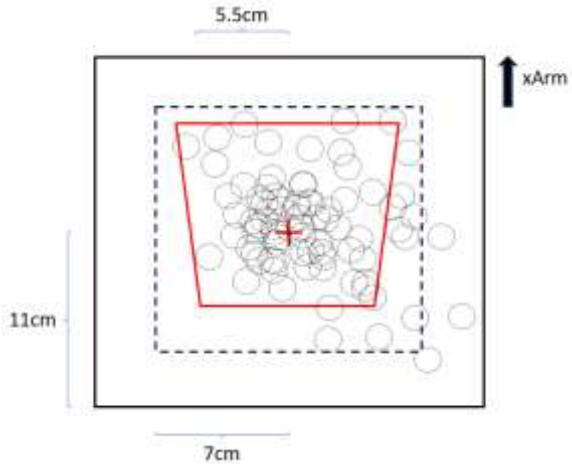

Fig. 8. The hit map of the final product. (In red) the areas that would succeed in the suction lift. (In dashes) the object that needs to be lifted. (In the box) our total tries.

As visible in **Fig. 8**, The red box present in pictures 2,3, and 6 was also the subject of the precision test outlined in dashes. The red box indicates the approximate place on the box in which a lift would take place if contact were to happen with the suction cup. The positive hitbox is more tilted towards the back end since it's the closer end to the xArm and the counterweight could be used more effectively. It's also evident that our misses were more populated towards the top end which could mostly be attributed to the use of a single camera in a locked position which increases the possibility of the perspective effect distorting the image to the point that the density of the pixels are sensibly more in the upper parts of the picture. There are also minimal runs without a contact after full calibration which could only be the result of the same event. This data was extracted by using the Ufactory's main program as a set point for the displacement of the robotic arm as well as the actual locations of the box which were set on 9 optimums of the base. 4 corners, 4 centers of the lines connecting the corners and the center itself.

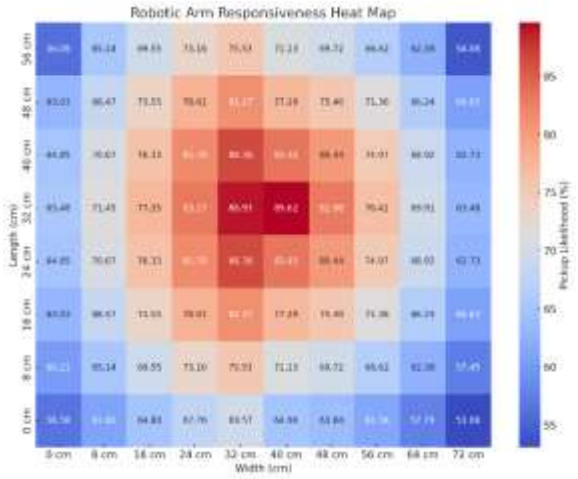

Fig. 9. The likelihood of a successful pick-up based on the relative location on the table

As shown in **Fig. 9** and expected from a single visual stance point, The central points are more accurate due to the training sequel targeting them mostly. The closer side to the camera also experiences better accuracy since it senses less pixel distortion caused by the pixels being cramped at the further end of the table with regards to the camera. The corners also are more susceptible to being at fault mainly because the x-y axis difference is sensed greater and the distance value stops following a linear shape. The system was also capable of keeping a consistent response time. The average response time, measured from the detection of an object to the initiation of the robotic arm's movement, was recorded as 1.2 seconds. The prompt reaction is evidence of the successful incorporation and enhancement of the hardware and software elements. The system consistently maintained this response time across multiple trials, demonstrating its robustness in real-time processing capabilities.

The system's overall operational efficiency was assessed by evaluating its performance in a simulated real-world setting. This encompassed experiments that encompassed diverse object arrangements and environmental circumstances. The system consistently maintained a high level of operational efficiency, achieving an average success rate of 85% in accurately identifying and manipulating objects. The limited occurrences of inefficiency were mainly attributed to environmental factors, such as fluctuating lighting conditions and the positioning of objects beyond the Kinect sensor's optimal range. Although the system demonstrated effective performance in the majority of situations, specific limitations and challenges were identified. Improving the YOLOv5 algorithm to enhance object detection accuracy in low light conditions is necessary due to the existing challenge. Furthermore, the manipulation of objects with intricate shapes or textures sometimes resulted in diminished accuracy.

This project has achieved significant advancements, as demonstrated by a comparative analysis with similar existing systems. The incorporation of Azure Kinect DK resulted in a significant enhancement in spatial perception when compared to systems utilizing traditional cameras. Moreover, the utilization of Python and YOLOv5 enhanced the system's versatility and adjustability, surpassing various preexisting configurations in terms of promptness and object detection capabilities. Ultimately, the findings illustrate the system's adeptness in identifying and handling objects, underscoring its capacity for use in automated environments. The system's effectiveness is highlighted by its high accuracy rates, precise manipulation, and rapid response time. Additionally, the identified challenges provide opportunities for future improvements.

Future Works

System Integration with Third-Party Integration: Integrating the system with other systems from the outside world in order to cover a wider range of applications was an extra component of the implementation. To do this, interfaces had to be developed for the system so that it could connect with other software and hardware components. This made it possible for the system to be used in more complicated circumstances [20]. Modularity was included in the architecture of the system, which made it possible to easily integrate and scale it.

Instructional Writing and the Development of User Interfaces: There was a compilation of detailed documentation on the installation, configuration, and operation of the system [21]. The purpose of this documentation is to serve as a guide for future users and developers, offering in-depth information on the design and operation of the system. The development of a user interface (UI) was also carried out in order to make interacting with the system easier. Real-time feedback on the condition of the system is provided via the user interface (UI), which also enables users to adjust settings and gives choices for manual configuration. During the creation of this user interface, usability and accessibility were the primary focuses. This was done to ensure that users were able to successfully use and monitor the system.

Conclusion

The main objective of the project was to develop a system with the ability to accurately detect objects and perform precise manipulation. This goal was largely accomplished, as demonstrated by the results obtained. The successful integration of these advanced technologies is evidenced by the high accuracy rate of object detection, the precision in robotic manipulation, and the system's rapid response time. The study provided significant insights into the difficulties and possibilities associated with the integration in question. Significant accomplishments included the successful implementation of the YOLOv5 algorithm for the precise identification of diverse objects in varying environments, as well as the precise calibration of the robotic arm to effectively respond to sensory input. Nevertheless, the project faced constraints, specifically in detecting objects in dimly lit environments and dealing with objects that have intricate textures, highlighting potential areas for future enhancement.

The comparative analysis revealed the progress achieved by integrating Azure Kinect DK to improve spatial awareness and the versatility provided by Python programming. The findings indicate that this integrated system has considerable potential for use in diverse sectors that necessitate automated object detection and manipulation. In the future, the project establishes a basis for additional research and development. Subsequent research could prioritize augmenting the algorithm's efficacy across varied environmental circumstances, refining the processing of a broader array of object categories, and expanding the system's capacity for more extensive implementations. This study provides valuable insights into the field of robotic automation by showcasing the efficacy and obstacles of incorporating state-of-the-art technologies in real-world scenarios.

References

[1] G. Kurillo, E. Hemingway, M.-L. Cheng, and L. Cheng, “Evaluating the Accuracy of the Azure Kinect and Kinect v2,” *Sensors*, vol. 22, no. 7, 2022, doi: 10.3390/s22072469.

[2] J. E. Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. “ O’Reilly Media, Inc.,” 2012.

[3] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A Review of Yolo Algorithm Developments,” *Procedia Comput Sci*, vol. 199, pp. 1066–1073, 2022, doi: https://doi.org/10.1016/j.procs.2022.01.135.

[4] X. Cong, S. Li, F. Chen, C. Liu, and Y. Meng, “A Review of YOLO Object Detection Algorithms based on Deep Learning,” *Frontiers in Computing and Intelligent Systems*, vol. 4, no. 2, pp. 17–20, Jun. 2023, doi: 10.54097/fcis.v4i2.9730.

[5] A. V. and A. C. and N. I. and K. V. and P. V Srivastava Shrey and Divekar, “Comparative analysis of deep learning image detection algorithms,” *J Big Data*, vol. 8, no. 1, p. 66, May 2021, doi: 10.1186/s40537-021-00434-w.

[6] G. and T. J. V Diwan Tausif and Anirudh, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Multimed Tools Appl*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.

[7] J. Wu, J. Dong, W. Nie, and Z. Ye, “A Lightweight YOLOv5 Optimization of Coordinate Attention,” *Applied Sciences*, vol. 13, no. 3, 2023, doi: 10.3390/app13031746.

[8] H. Sekkat, S. Tigani, R. Saadane, and A. Chehri, “Vision-Based Robotic Arm Control Algorithm Using Deep Reinforcement Learning for Autonomous Objects Grasping,” *Applied Sciences*, vol. 11, no. 17, 2021, doi: 10.3390/app11177917.

[9] M. T. Shahria, A. Arvind, I. Iqbal, M. Saad, J. Ghommam, and M. H. Rahman, "Vision-Based Localization and Tracking of Objects Through Robotic Manipulation," in *2023 IEEE 14th International Conference on Power Electronics and Drive Systems (PEDS)*, 2023, pp. 1–6. doi: 10.1109/PEDS57185.2023.10246715.

[10] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review," *IEEE Trans Cybern*, vol. 43, no. 5, pp. 1318–1334, 2013, doi: 10.1109/TCYB.2013.2265378.

[11] G. Liu, Y. Hu, Z. Chen, J. Guo, and P. Ni, "Lightweight object detection algorithm for robots with improved YOLOv5," *Eng Appl Artif Intell*, vol. 123, p. 106217, 2023, doi: https://doi.org/10.1016/j.engappai.2023.106217.

[12] A. F. Elaraby, A. Hamdy, and M. Rehan, "A Kinect-Based 3D Object Detection and Recognition System with Enhanced Depth Estimation Algorithm," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 247–252. doi: 10.1109/IEMCON.2018.8615020.

[13] S. M. Mehdi, R. A. Naqvi, and S. Z. Mehdi, "Autonomous object detection and tracking robot using Kinect v2," in *2021 International Conference on Innovative Computing (ICIC)*, 2021, pp. 1–6. doi: 10.1109/ICIC53490.2021.9692932.

[14] M. A. D. S. L. C. Douglas Henke Dos Reis Daniel Welfer and D. F. T. Gamarra, "Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm," *Applied Artificial Intelligence*, vol. 33, no. 14, pp. 1290–1305, 2019, doi: 10.1080/08839514.2019.1684778.

[15] T. T. and T. T. V. Nguyen Trong Hai and Nguyen, "A Method for Localizing and Grasping Objects in a Picking Robot System Using Kinect Camera," in *Intelligent Human Computer Interaction*, D.-K. and L. J.-H. and T. U. S. and S. D. and C. W.-Y. Singh Madhusudan and Kang, Ed., Cham: Springer International Publishing, 2021, pp. 21–26.

[16] D. and de S. L. C. M. A. and T. G. D. F. dos Reis Douglas and Welfer, "Object Recognition Software Using RGBD Kinect Images and the YOLO Algorithm for Mobile Robot Navigation," in *Intelligent Systems Design and Applications*, P. and M. K. and K. A. Abraham Ajith and Siarry, Ed., Cham: Springer International Publishing, 2021, pp. 255–263.

[17] Y. and Z. J. and Z. J. Chen Nan and Hu, "Robot Learning of Everyday Object Manipulation Using Kinect," in *Foundations and Practical Applications of Cognitive Systems and Information Processing*, D. and L. H. Sun Fuchun and Hu, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 665–674.

[18] M. Kulkarni, P. Junare, M. Deshmukh, and P. P. Rege, "Visual SLAM Combined with Object Detection for Autonomous Indoor Navigation Using Kinect V2 and ROS," in *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, 2021, pp. 478–482. doi: 10.1109/ICCCA52192.2021.9666426.

[19] Z. Ma, Y. Zeng, L. Zhang, and J. Li, "The Workpiece Sorting Method Based on Improved YOLOv5 For Vision Robotic Arm," in *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2022, pp. 481–486. doi: 10.1109/ICMA54519.2022.9856190.

[20] M. Bojun, F. Yongchun, and Z. Xuebo, "Inverse Kinematics Analysis for a Mobile Manipulator with Redundant DOFs," in *2007 Chinese Control Conference*, 2007, pp. 118–122. doi: 10.1109/CHICC.2006.4346874.

[21] Q. Song *et al*., "Object Detection Method for Grasping Robot Based on Improved YOLOv5," *Micromachines (Basel)*, vol. 12, no. 11, 2021, doi: 10.3390/mi12111273