Machine Vision and Artificial Intelligence in Robotics for Smart Factory

[1]Rajeev Kanth, [2]Jukka Heikkonen
[1]Savonia University of Applied Sciences, Kuopio Finland,
[2]University of Turku, Turku Finland
rajeev.kanth@savonia.fi, jukhei@utu.fi

2023 IEEE International Conference on Emerging Trends in Engineering, Sciences and Technology (ICES&T) | 978-1-6654-5560-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICEST56843.2023.10138859

***Abstract*— In this work, we have illustrated how machine vision and artificial intelligence could enhance operations in a smart factory along with a robot via several use cases. The first use case is to teach the robots with a leap motion sensor. In Industry 4.0 environment, the intelligent factories decide the subsequent actions based on the sensor data. The leap motion sensor can control a robot's movements and is beneficial, particularly in smart factory environments. We have demonstrated how a leap motion sensor could control picking and dropping off cylindrical objects. The second use case is to control a robot using a mouse and how the robot could play chess. For this use case, we taught the robot to grip a chess object and put it in the designated place. Similarly, we also use a machine vision camera for detecting several entities, such as wingnuts and bolts. The TensorFlow algorithm has been utilized for detecting different shapes and colors.**

***Index Terms*— Artificial Intelligence, Leap Motion Sensor, Niryo Robot, Smart Factory**

I. Robots for Education and Research

Smart factories have often utilized robots to improve automation, productivity, and safety. Niryo robot is a six-axis collaborative robot arm designed to reproduce all the movements required in the most advanced uses in industry 4.0, with a precision of 0.5mm and a repeatability of 0.5mm. Its aluminum structure makes it exemplary robust, allowing it to accomplish the movements required for the robotic functionalities with fluidity. This robot takes advantage of the capacities of the Raspberry Pi 4, with a 64-bit ARM V8 high-performance processor, 2GB of RAM, and improved connectivity. This is a collaborative robot based on Ubuntu 18.04 and compatible with Robot Operating System, an open-source solution created for robotics. The compatibility with the robotic operating system allows multiple libraries allowing us to conceive many programs, from the simplest to the most complex ones, responding then, in a flexible way, to our needs. The Niryo robot's components are shown in Fig. 1.

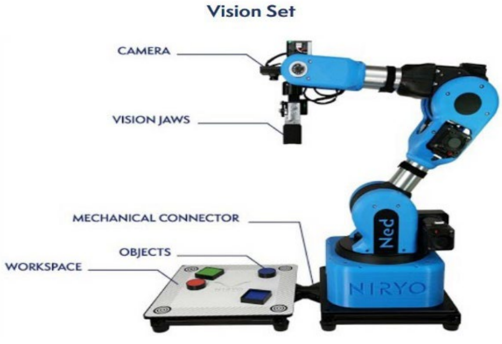

Fig. 1. Components of a Niryo Robot

This robot's entire educational and research package includes a customized and adaptive gripper allowing it to grab non-standard objects with complex shapes. The paper is organized as follows. Section II presents the control of a robot using a leap motion sensor. Sections III and IV, respectively, describe bolt recognition and mouse control.

II. Control of Robots with Leap Motion Sensor

In this work, we have accomplished the control of a Niryo [1] robot using an algorithm based on artificial intelligence and a Leap Motion Sensor to move the robot with simple hand gestures without a keyboard or mouse. The leap motion sensor is a new device developed for gesture interaction, and the mall device has a dimension of 0.5x1.2x3 inches by measurement. A user connects the leap motion to a computer by USB and then places their hand over the leap motion controller to control the Niryo Robot.

A. Methodologies

In order to control Niryo 6-axis collaborative robot with our hand movement, we need to configure the software for both the leap motion sensor and the robot. As shown in Fig. 2, the laptop's circular white portion is the hand gesture's working space. The functional team did the programming with Python with the compatible software for to leap motion

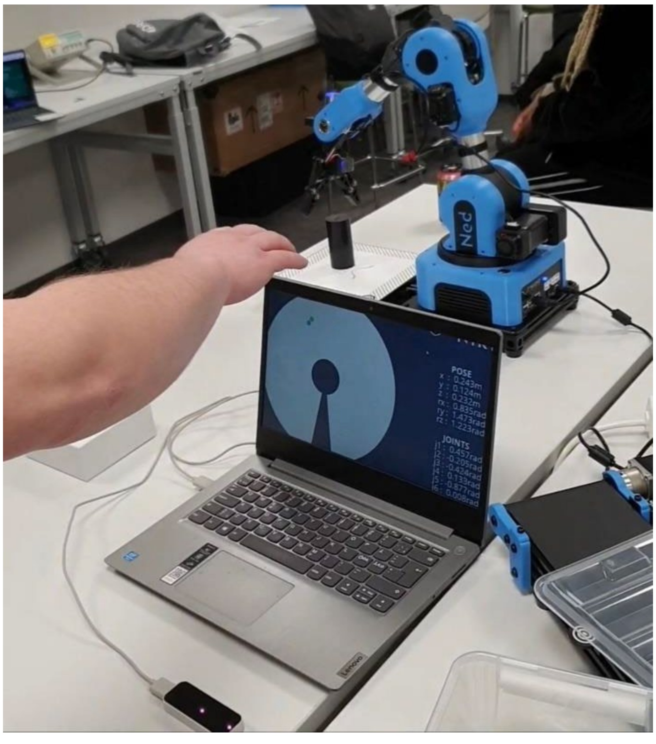

Fig. 2 Control of Grip of the Robot using Leap Motion Sensor

sensor and Niryo Ned. The programming aims to make the robot's system understand information coming from the motion sensor so that the robot can track hand movements made above the motion sensor.

To accomplish this work, the common motion used in available content and amusement was first classified as referred to in the scientific article [2] based on the classification of the motion lexicon (ML), which can be used universally as defined. Structural Motion Grammar (SMG) was determined through the combination of motion lexicon. The overall process of the leap motion that enables the free movement of hands and recognition used the same architecture as mentioned in the article [2].

So as the leap controller tracks hands in this view, it provides updates as a set or frame of data, each frame object representing a frame containing any hands, detailing the movement at any time. So, the hands provide the positioning and other characteristics of a detected hand. The leap motion controller provides information about each finger on a hand. All or part of a finger's characteristics is estimated based on recent observations and the anatomical model of the hand.

B. Results and Discussions

We successfully used the Niryo Ned 6-axis collaborative robot by hand movements using the leap motion sensor, as shown in Fig. 2. The calibration for the robot was done to adjust the movement recognition. Raspberry Pi powered the processing for the robot. However, the robot appeared a bit slow with its movement and responded slowly to hand movements, and it was pretty hard to use with the leap motion control. Finally, we could fully exploit the robot's accuracy because of inaccuracy and delay caused by the leap motion sensor. After the suitable installation and calibration of the Niryo robot, the programmed robotic arms started to perform more accurately and consistently than humans. The hand gesture movement needs practice so that the robot arm would work with better accuracy. The effective training of robot jaws depends on several issues, such as fast internet, the processing capabilities of the robot's processor, and the ethernet cable connection between the robots and the laptop.

A wide variety of tools are available for the robot, allowing many different features to be performed. In a smart factory environment, it is possible to move objects from place A to place B in accordance with the technical characteristics of the robot. You don't even need to be close to the robot; it still responds to hand movements because it is wirelessly connected to the laptop with WiFi. The robot works perfectly with an ethernet cable connection. It doesn't matter how big the object is, it just needs to fit in the gripper. The slippery object may not be nicely gripped within the robot's jaws. In order to respond to the robot instantly and follow the hand gesture in real-time, it needs a lot more accuracy. Once the robot is trained robustly, it can perform more accurately and consistently than human beings. It also can be used in, for example, in the hospital or working in a hazardous environment (chemical or nuclear industries), where human health is the number one priority.

In summary, the main aim of controlling the robot's arm using a leap motion sensor was accomplished after teaching the robot. This requires watching related online videos, studying several tutorial manuals[3], and finally setting up the necessary hardware specifications. The robot could respond to simple hand movements positioned between 25 to 600 mm above the sensor. The leap motion sensor is embedded with optical sensors and infrared light, allowing the controller to recognize and track hands, fingers, and finger-like tools.

III. Wing nut Vs. Bolt Recognition

In the innovative factory environment, it is essential to detect several object sizes and shapes and arrange them at the exact location. This is the second use case where we have demonstrated highly accurate recognition of wing nuts and bolt and picking up by the robot and keeping them at the conveyer belt. Object modeling, python programming, and testing the models are some of the critical steps for recognizing and placing identical objects in the factory.

A. Object Modeling and 3D Printing

We decided to develop our own objects to be printed using our 3D printing facilities and chose things from the actual manufacturing process. The initial idea was that the robot chooses and picks up a non-threaded bolt vs. a threaded bolt. However, due to the limited resolution of the embedded machine vision set, we concluded to teach the robot such that it is possible to detect wing nuts from the workspace. Solidworks software was used for modeling. We modeled both bolts and wing nuts, as shown in Fig. 3.

B. Python Programming and Results

First, according to the instructions of the robot manufacturer, familiarity with the parts and how to use the robotic arm was done [4]. Then the software required to connect the robot to the computer was installed and the robot was calibrated completely. Then, to use the TensorFlow platform, Anaconda software was installed as a platform for implementing the Python programming language, and the steps mentioned in the manufacturer's instructions [5] for using TensorFlow were followed step by step. The steps involved included installing PyGame, OpenCV, and PyNiryo, and calling the main TensorFlow platform implementation folder. In addition to the software and programming steps, the objects defined for the project, which included simple bolts and wing nuts, were designed and 3D printed by the mechanical team. The next step was to open the main software and introduce the bolts and nuts to the robot, which was done through photos taken by a camera mounted on the robotic arm and programming. The robot was

Fig. 3. Several Bolts and Wing nut modelled and 3d printed

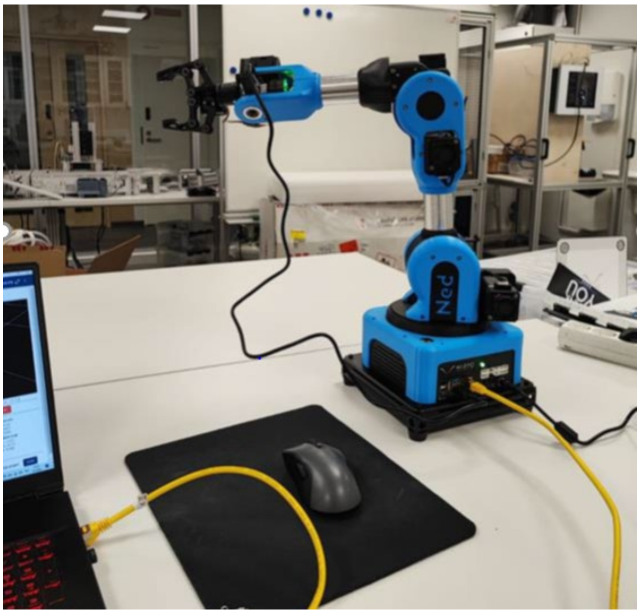

Fig. 4. Testing Niryo Robot

then tested to identify and fix possible errors, as shown in Fig. 4. The Niryo robot was set as planned. The Niryo studio was uploaded on the computer, and we could calibrate it and move different joints to different positions with the workspace set up and saved.

The TensorFlow program opened up, and the new camera was working correctly. It had the ability to recognize objects by their shapes and colors before it was replaced by the old one. We performed several experiments using different objects to evaluate the robotic arm prototype. As shown in Fig. 5, the Niryo Robot was programmed with the TensorFlow algorithm in such a way that it is able to detect the wingnut from the collection of wingnuts and bolts. Initially, the robot was trained in such a way that it should be able to recognize among threaded bolts versus unthreaded bolts. Due to the limited resolution capacity of the attached machine vision camera in the robot's jaw, it could not detect the unthreaded and threaded bolts. So, finally, the wingnut versus bolt was recognized successfully.

IV. Control of Robot Using Mouse and Arduino Uno

In this section, control of the robot with a mouse and Arduino Uno has been illustrated. Each sector required a little understanding of coding and perception to see how the robot would react to a mouse's movement. The primary coding language that was used in our project was Python.

A. Controlling Robots with Mouse

It is fair to say that controlling Ned with a mouse is the easiest part of the project. We only needed to install Niryostudio and Python, and pyNniryo and pyGame add-ons to it.

After downloading and installing these programs, one can launch the mouse control API with the command ROS launch on Python. The mouse-controlling application can also be used in simulation, i.e., RVIZ or Gazebo. When controlling the real robot, first, we have to be connected to it. For that part, the Niryostudio is needed. From there, we can establish a connection between the host computer and the robot, and

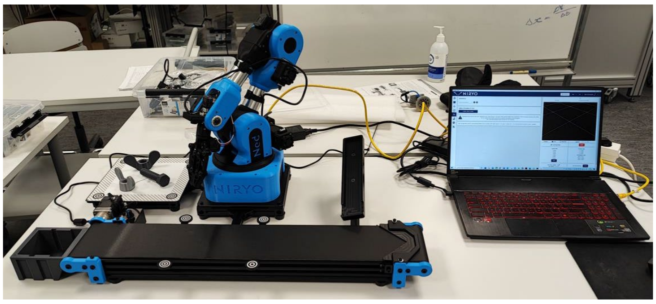

Fig. 5. Detecting the wingnuts and collecting them on the conveyor belt

you can also find the IP address for the robot. Before launching the mouse control API, it is crucial to check that the IP address is correct from the robot.ned.py file. If the connection to the robot has been established, and we have the valid IP address, you can now launch the mouse control panel.

From the mouse control API, as shown in Fig. 6, one can see the orientation of the robot's pose and joints. From there, the green indicator is the mouse, and the blue is Niryo following it. We found that slow and steady movement is crucial so Niryo can keep up with the activity and execute given tasks. Connecting with Niryo via ethernet cable is recommended because a wireless connection can cause minor glitches in the robot's response to movement. The mouse-controlling application can be helpful at times, but it is fair to say that other technologies have surpassed it.

B. Controlling Robots with Arduino Uno

Arduino is a platform and collection of microcontroller boards that contain a small piece of code called an Arduino bootloader. As they are open-sourced, anyone can utilize the programming environment with the Arduino board. All different Arduino models have other functionalities, like some of them have onboard WiFi, Bluetooth, strong processors, and more inputs/outputs. The Arduino board can connect to the computer using the USB port of the Arduino board. Once it is connected to the computer, write a code using the development environment. Then send the code through the USB to the board. Anytime the user powers the board, the code on there will run in a cycle. The output can be seen as light or any other signal.

We were using an Arduino Uno in our project. To set it up

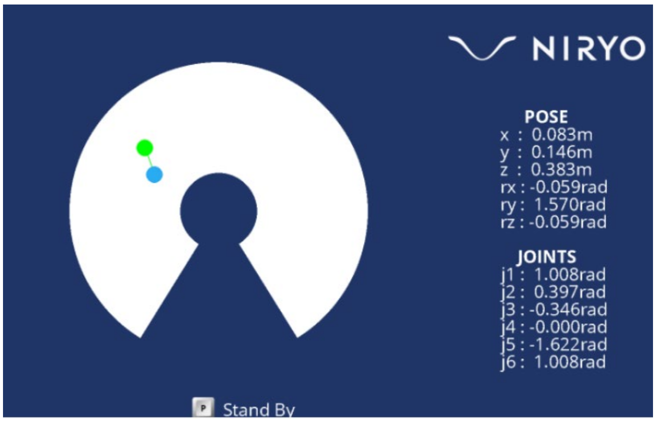

Fig. 6. Mouse Control API

with the robot, we downloaded applications found in the Niryo Ned Arduino tutorial. We downloaded an Arduino application where we can write code for the Arduino board. We also downloaded a tutorial code and modified it based on our requirements. Arduino is connected to the robot by connecting wires to the pins. It also needs to be connected to a computer with a cable. Arduino sends commands to the robot, and the robot works as the code demands. For example, Arduino can be used when the robot repeatedly needs to do the same movement. It also can be coded for the long trajectory that could be used in industry, for example.

V. Discussion and Conclusion

In this article, we have demonstrated the use of robots intelligently in the industrial and factory environment. Several control technologies, such as leap motion sensor and integration of machine vision set for detecting different shaped and sized objects, have also been presented. In this work, we also gathered the information for controlling the Niryo robot with a mouse and Arduino Uno microcontroller. Teaching a robot with a leap motion sensor was the most challenging task, as it required several trials. The resolution and sensitivity of the embedded machine vision set with the robot also played an essential role in the detection of objects. The key conclusions from this work are listed below:

- The critical configuration steps to integrate the robot, machine vision set, artificial intelligence algorithm, and related technologies are explained in this paper.
- Several challenges to the efficient performance of the robot have been studied well.
- Finally, we are able to implement the desired objectives for utilizing robots efficiently in the smart factory environment.

There are further many avenues for the future continuing applied research in the arena of an intelligent factory. We are also currently working on several learning modules that focus on content creation for industry 4.0, collaborative robots, sensors, data analysis, AR, VR, Mixed reality technologies, and data-driven manufacturing under a European Union VLEFACT project [6].

Acknowledgment

We would like to acknowledge the contribution of our Savonia University of Applied Sciences students, who have realized the actual implementation of this work. Some names are Mauri Alhonen, Cynthia Monkap, Niko Pitkänen, Queen Akwiwu, Tuomas Hakala, Omid Jarari, Antti Rompainen, and many others students. We also would like to thank the VLEFACT project, a joint consortium of four educational institutions from Germany, Denmark, and Finland.

REFERENCES

[1] The Niryo Robot documents for NED, vision set, conveyer belt, programming, and applications, last accessed on 19th October 2022, https://docs.niryo.com/en/index.html

[2] Byungseok Lee, Donghwe Lee, Seongah Chin, "Structural Motion Grammar for Universal Use of Leap Motion: Amusement and Functional Contents Focused," Journal of Sensors, vol. 2018, Article ID 6073786, 15 pages, 2018. https://doi.org/10.1155/2018/6073786

[3] The documents for NED, vision set and AI programming, and applications, last accessed on 19th October 2022 https://docs.niryo.com/applications/ned/v1.0.0/en/source/examples/vision_pick_artificial_intelligence_tensorflow.html

[4] The technical specification of the Niryo robot, last accessed on 19th October 2022, https://docs.niryo.com/product/ned/v3.1.1/en/source/hardware/technical_specifications.html

[5] PyNiryo Documentation, last accessed on 19th October 2022, https://docs.niryo.com/dev/pyniryo/v1.1.2/en/index.html

[6] The Consortium of four European HEIs under the Erasmus Mundus Project Virtual factory (VLEFACT), last accessed on 19th October 2022, https://vlefactory.eu/