DEVELOPMENT OF CONTROL AND IMAGE PROCESSING FIRMWARE FOR ARTICULATED ROBOT ARM EQUIPPED WITH ADDITIONAL TRANLSATION AXIS

ANASTASIU Alexandru-Ioan

Faculty: Industrial Engineering and Robotics, Specialization: Robotics, Year of study: 1 ; E-mail: a.anastasiu@outlook.com

Scientific mentors: PhD Ing. Cozmin CRISTOIU, Conf. Florea Dorel ANANIA

SUMMARY: The project's end goal was to develop firmware equivalent to the industry standard for an educational 5 axis robot, replicating functionality of a Teach-Pendant and offering the possibility of programming, either by manually writing programs, or by utilizing third-party software, such as RoboDK. Additionally, steps were taken in order to fully utilize available hardware.

Keywords: Robot, Programming, Hardware instructions

1. Introduction

Thanks to the growing availability of such kits, it is becoming increasingly easy for students and hobbyists alike to gain access to the world of robotics. This is helpful in understanding the key concepts behind such machines, without the need to work inside an industrial environment or access cost-prohibitive hardware. By using one such robot kit, a scaled-down but otherwise analogous in function 5 axis robot arm was programmed.

2. Current stage

Current industries are moving more and more towards automation. This can perhaps best be seen in assembly lines, with factories all around the word utilizing industrial robots in order to streamline construction and reduce running costs. Historically, such robots have been kept closed source, with all aspects, hardware and software, intellectual property of the manufacturer. This makes it difficult for those who wish to learn exactly how to utilize such a robot (in the case of students, for example). By using an open platform (such a robot kit available for purchase), coupled with open-source software, students can gain access to these robots with ease.

3. Hardware

The hardware structure was chosen for its simplicity and robustness. The manipulator is constructed from pressed and bent aluminum sheets, alongside absolute position encoders (of the resistive type) mounted to servomotors. The controller is a raspberry pi, mounted to a driver board for the motors. Documentation for the kit was limited, but it was used as a starting point for the reverse-engineering process required to build the software. The firmware was written from the ground up, using no components from the original.

4. The Teach Pendant

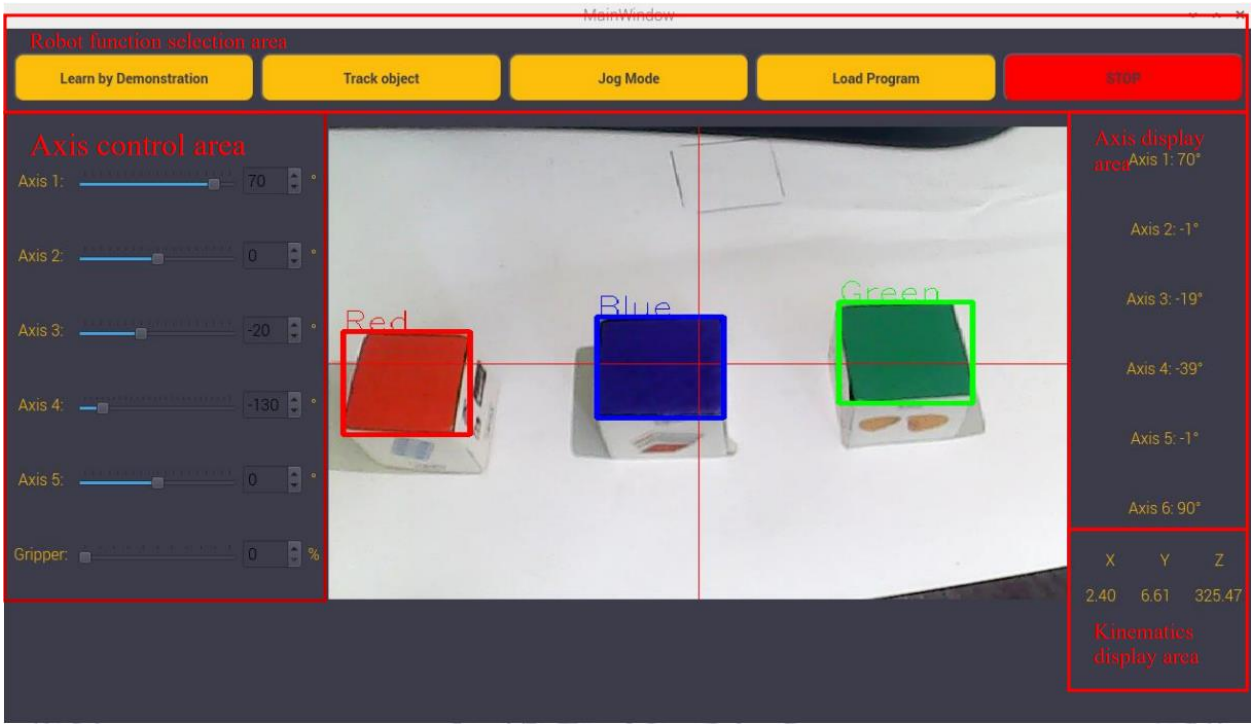

Fig. 1 The robot's graphical user interface

The Teach pendant is the main human-machine interface, allowing full control of the robot. It is designed to run on a computer attached to the robot, allowing a user to individually control the axes, to read information about current position of the robot, and to access the different operation modes. The position of the end-effector is calculated using a process called direct kinematics, transforming joint angles into XYZ coordinates, relative to the robot's base.

The different working modes can be divided into "Teach-in", or direct programming, "Object tracking", where the robot will automatically seek and follow an object of a selected color, "Free movement", or jog mode, where an operator can freely move the robot and "Program cycle", where the robot will follow a given program.

Teach-in allows the operator to program the robot directly, wherein the robot may be moved using the joystick, much like actual industrial robots. An operator can click a button to record a target once a specific position and orientation have been established. Upon running the cycle, it will subsequently proceed along the predetermined course while adhering as closely as possible to the coordinates specified.

Object tracking is done on a color basis, using an algorithm known as "Thresholding". This is done by filtering out every color that is not the one being searched for. This results in an image that only has non-black pixels in the areas colored with the value required. A bounding box can be drawn to contain all these pixels, resulting in a rectangle. By aligning the center of the viewfinder with the center of the box, tracking is achieved.

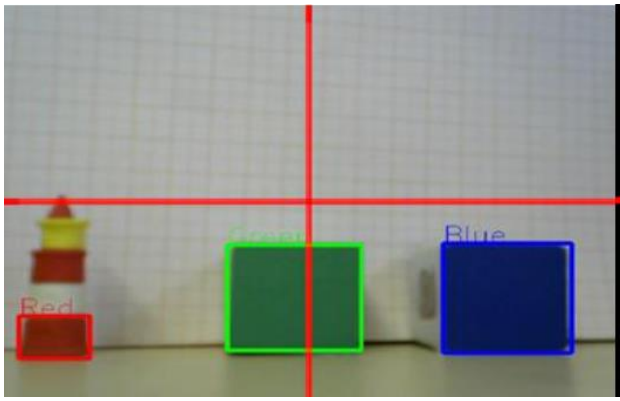

Fig. 2 Reference image

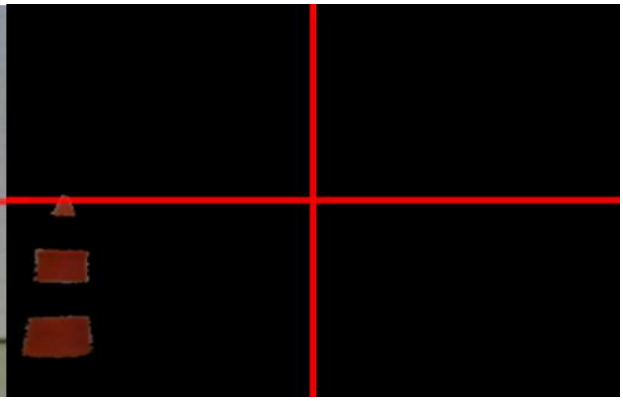

Fig. 3 Thresholding using red

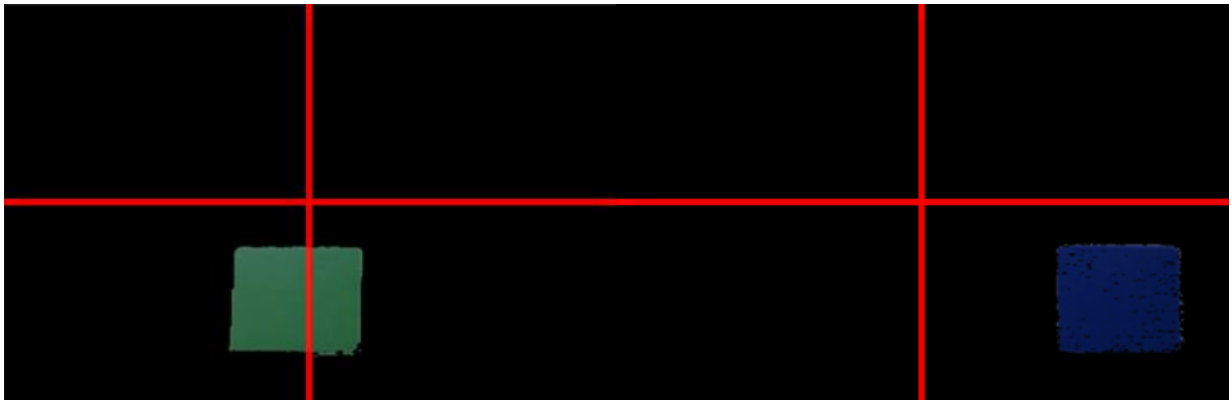

Fig. 4 Thresholding using green

Fig. 5 Thresholding using blue

5. Programmability

In order to fully achieve the goal of being a scaled-down industrial robot, the possibility of off-line programming must be added. This entails the need to create an instruction set, somewhat similar to G-Code. Branded as RASM (Robot Assembly), it has a vaguely ARM assembly-like syntax and can read and carry out 17 particular opcodes (instructions) pertaining to the manipulator's spatial movement. This makes it possible for the robot to be fully programmed, including logic and arithmetic operations (addition, subtraction, multiplication, with variable support, if, else, rudimentary branching support by means of goto-like instructions).

RASM is a hybrid language, a crossover between the statically typed, compiled languages such as C and the dynamically interpreted ones such as Python. This means that, while the code does not run directly on given hardware, necessitating an interpreter to translate generic instructions into robot-specific commands, preprocessing is done in order to greatly speed up execution. For example, while variable names are important for humans to read code, a machine is more likely to find a numeric value in an array than it is to find an alphanumeric (containing numbers and letters) one. This means that variable names can be optimized out and replaced with numbers, the optimization being offset at the assembler level.

Generally speaking, the program runs in the same manner as an x86 or arm program, with instructions being laid out in an array called the program stack. Traversing this stack is a program pointer, an index that increases, indicating which instruction should be executed. Jump instructions manipulate this program pointer, changing it to whatever value is necessary.

In order not to write code by hand (which would be difficult to maintain and do), a postprocessor for RoboDK was created, which is able to generate RASM instructions from a set of point coordinates and orientations.

```
NME Prog1
// Program generated by RoboDK v5.4.2 for My Mechanism on 12/05/
// Using nominal kinematics.
ANGS 0.000 -0.000 0.000 -0.000 -0.000
ANGS 0.000 59.182 32.491 -1.683 0.000
ANGS 0.000 60.025 37.162 -7.196 0.000
// Attach to TCP_CLESTE
ANG 50
ANGS 0.000 59.182 32.491 -1.683 0.000
ANGS 45.000 59.180 32.490 -1.680 -0.000
ANGS 45.000 67.931 43.366 -21.308 0.000
// Detach from TCP_CLESTE
ANG 0
ANGS 45.000 59.180 32.490 -1.680 -0.000
ANGS 0.000 59.182 32.491 -1.683 0.000
ANGS 0.000 67.933 43.364 -21.306 -0.000
// Attach to TCP_CLESTE
ANG 50
ANGS 0.000 59.182 32.491 -1.683 0.000
ANGS 45.000 59.180 32.490 -1.680 -0.000
ANGS 45.000 60.023 37.164 -7.197 0.000
// Detach from TCP_CLESTE
ANG 0
ANGS 45.000 27.427 86.403 -23.840 -0.000
ANGS 45.000 22.236 72.361 -4.606 -0.000
ANGS 0.000 -0.000 0.000 -0.000 -0.000
END
```

Fig. 6 Example code generated by RoboDK

6. Direct kinematics optimization using ARM NEON

Direct kinematics is the process by means of which the position and orientation of an end-effector are calculated. This is done using a method called “The universal approach”, by utilizing a rotation matrix coupled with a translation vector combined into a 4x4 matrix. Each joint can therefore be described by such a matrix. Multiplying them results in the final one being the position and orientation of the end-effector. In a conventional approach, this is done the same way it is taught in school, by multiplying and adding each element. This method has the main drawback of speed, as it entails a large number of calculations being necessary, resulting in the processor needing to utilize more clock cycles. A better approach is to utilize SIMD (single instruction, multiple data) built-in to the ARM architecture. This allows 4x4 matrix multiplication to effectively be done in 4 clock cycles, resulting in a speedup of almost 3x over the conventional way.

This leverages the fused-multiply-accumulate function, which, as the name implies, implements multiplication and addition in the same clock cycle. At a silicon level, processors work on what are called registers, small (64 bit) memory locations used to store numerical values, used in arithmetic and logical operations. ARM NEON allows combining 4 such registers into what are called quadword registers, wherein they behave like a vector, with the added benefit of being able to do arithmetic instructions on them directly (vector addition, for example, is done by element, in parallel). The end benefit is being able to spend less time processing kinematics, allowing for faster response times to an operator’s requests.

8. The base translation axis

In order to allow the robot to achieve global movement (from one machine tool to another, for example), it is equipped with a base translation axis. Using a screw and two guiding rails, a steel platform is driven by a stepper motor. Control is done in an open loop, using an additional Arduino Nano microcontroller. The robot sends position commands to the axis controller, which then replies only when it has finished travelling. Live reporting cannot be done, due to the reduced clock speed of the Arduino. In essence, stepper control is generated by quickly pulsating the “step” pin of a driver module, each pulse causing the motor to turn 1.8°. Each operation to send data to the Serial port (the communication method

between Arduino and Raspberry Pi) consumes a certain number of clock cycles, reducing the overall speed at which the step pin can be pulsed.

Calibration is done at power-up, with the axis automatically moving to the reference point, where a limit switch indicates that it has been reached.

8. References

[1] ARM Architecture Reference Manual, 2022 Arm Limited, https://developer.arm.com/documentation/ddi0487/latest

[2]. Basic Thresholding Operations, https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html

[3]. Cozmin CRISTOIU, Adrian NICOLESCU, "New approach for forward kinematics modeling *of industrial robots with closed kinematic chain", 2017*

[4]. Alexandru-Ioan ANASTASIU, Cozmin CRISTOIU, Florea Dorel ANANIA, "Educational 5 axis robot controller optimization using arm hardware instructions"

[5]. Dobrescu, Tiberiu Gabriel & Dorin, Alexandru & Nicoleta-Elisabeta, Pascu & Ivan, Ioana. (2011). CINEMATICA ROBOTILOR INDUSTRIALI.

[6]. Blanchette, Jasmin, and Mark Summerfield. C++ GUI programming with Qt 4. Prentice Hall Professional, 2006.