Hand Gesture Controlled Robot

Dávid Bárándi
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
barandidavid@stud.uni-obuda.hu

Károly Széll
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
szell.karoly@amk.uni-obuda.hu
ORCID: 0000-0001-7499-5643

***Abstract*— Robot manipulators in factories are usually used in automatic mode, running a pre-written program. My goal with this project was to be able to control the robot manipulator in real time with hand movements. This would allow in the future, by using augmented reality and a mobile robot manipulator, to work in places that would be dangerous for human life (e.g., chemical environment, extreme cold/hot environment.)**

Keywords—robotics, ROS, automation, industry 4.0, hand detecting and tracking, collaborative robot, OpenCV, Mediapipe

I. Introduction

Many examples in modern robotics show that, in some cases, seamless and real-time control is needed. In many cases it is difficult to control a robot with a keyboard or with the control panel of the robot itself. Gesture-based robot control is where a human can control the robot with hand movements and does not need to enter numerical values with a keyboard or control panel. This is suitable for controlling any robot in areas where human operation is not feasible for safety reasons [1]– [3]. So, you can control the robot safely in front of a TV or PC [4].

Thanks to the development of the Robotics Laboratory at the Alba Regia Faculty of Technology of Óbuda University, students will gain access to the most common industrial robots and collaborative robotics in industry. The collaborative robot in the lab can be used by any student to extend, develop, and test different ideas [5]– [8]. The collaborative robot is a Universal Robots UR5 e-series 6 degrees of freedom multifunctional co-robot.

II. Problem Description

I have taken the following aspects into account in the design:

- Implementation with minimum hardware requirements
- Plug n play with any webcam and universal robots
- 3D movement with a webcam
- Integrating OpenCV into ROS

The main problem was caused by the different Python versions. OpenCV and Mediapipe only works with Python3, while ROS Melodic uses Python2. The reason for choosing ROS Melodic over the newer ROS Noetic (which already supports Python3) is that the Universal Robots GitHub page for the package recommends Ubuntu 18.04, which only runs ROS Melodic.

III. Structure of the Enviroment

The project only required a Universal Robots UR5 e-series collaborative robot and a webcam. For image processing, I used the Open-Source Computer Vision Library (OpenCV) with Python3, and for hand recognition and tracking, I used Mediapipe developed by Google, also with Python3. Mediapipe was chosen because we wanted to track not only the hand itself, but also the hand points, to allow for the description and teaching of gestures in the future (see Figure 1.)

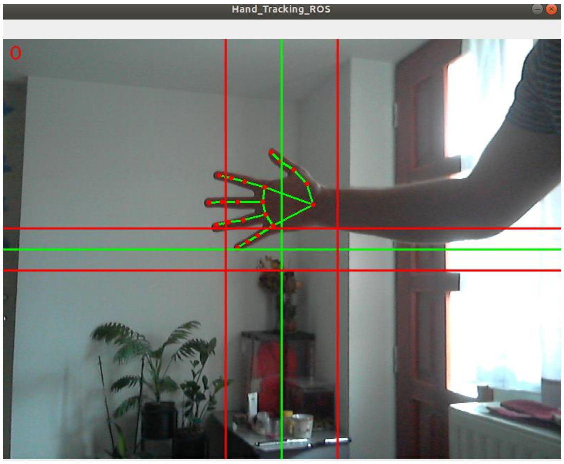

Figure 1. Hand recognition with OpenCV and Mediapipe

Universal Robots UR5e

Technical Data of UR5-e (see Figure 2.):

- Maximum reach: 850 mm
- Weight with cable: 20.6 kg
- Maximum payload: 5 kg
- Degrees of freedom: 6

Figure 2. UR5e

IV. STRUCTURE OF THE PROGRAM

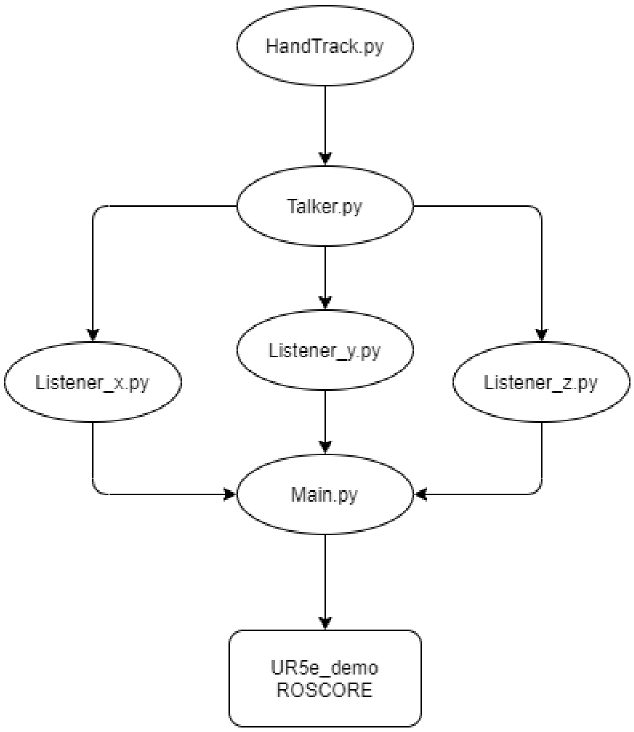

Figure 3. Structure of the program

The developed system uses live video stream for gesture recognition. It monitors in real time the position of the points on the hand and issues movement commands to the robot based on this information. The proposed technique Robot system operation is divided into four subsections:

- Setup ROS environment
- Hand detection with OpenCV and Mediapipe
- Processing data from hand detection
- Sending motion commands to the robot

A. *Setup ROS environment*

Robot Operating system (ROS) is a flexible framework for creating robot programs. Every ROS project needs a central core on which to hang other nodes that perform complex tasks. For this project, I used the Universal Robots factory package as the core. The package includes Moveit and Rviz (see Figure 5.), a visualization environment that allows me to easily design the path needed to move the robot and Rviz was used for one of the safety elements.

B. *Hand detection with OpenCV and MediaPipe*

The project is based on the HandTrack.py file (see Figure 3.). To detect the hand, I used the open-source computer vision library (OpenCV) and Mediapipe. OpenCV can be used to draw lines on the camera image so that I can draw the origin point and the threshold zone, where the robot stops moving when the tracked point is reached. For hand recognition and tracking I used Mediapipe, which is an open-source Python package developed by google which is perfect for hand recognition. It places a landmark of 21 points on the recognized hand, the points of which I used to control the robot (see Figure 4.).

C. *Processing data from hand detection*

The data from HandTrack.py is collected by the Talker.py (see Figure 6.) file and passed to the Listener files. The Talker.py file is the publisher node which is the transition between Python2 and Python3. The subscriber nodes are the Listener nodes. These files decide whether the robot should move in a positive or negative direction on a given axis, and they also define the size of the step per instruction and the threshold zone (marked with red lines, see Figure 1.). Listener_z.py is the file where the initial length of the segment between point 1 (WRIST) and point 12 (MIDDLE_FINGER_TIP) is defined (see Figure 4.). If the length of the segment increases by a factor of 1.2 during hand tracking, the robot moves in a positive direction, if it increases by a factor of 0.8, the robot moves in a negative direction on the axis. Here it is important that the hand is in front of the camera when starting the Main.py file, because only then can the length of the section be recorded, otherwise the robot will only move on 2 axes.

D. *Sending motion commands to the robot*

The Main.py file (see Figure 7.) is responsible for the movement commands. The starting position and the orientation of the TCP are defined here. In addition, the Main.py file collects the data returned by the Listener files and sends it to the robot, which executes it in the form of a motion. The program's motion, designed with state and executed with the real robot's state, is displayed in Rviz, so that we can even remotely control the robot manipulator after modeling the robot's environment.

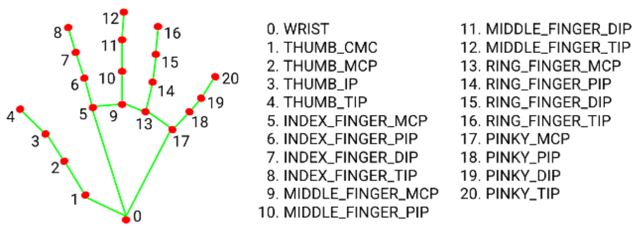

Figure 4. Mediapipe hand landmarks

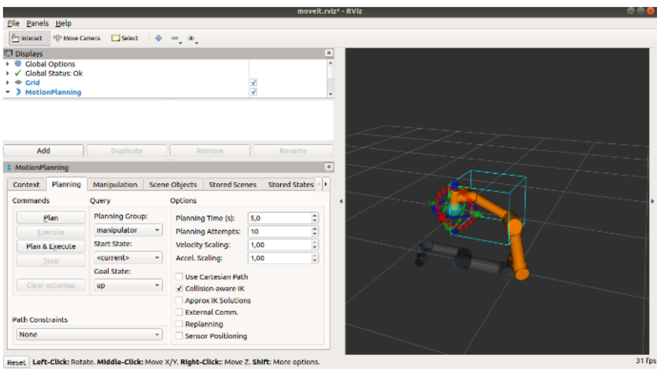

Figure 5. UR5e in Rviz

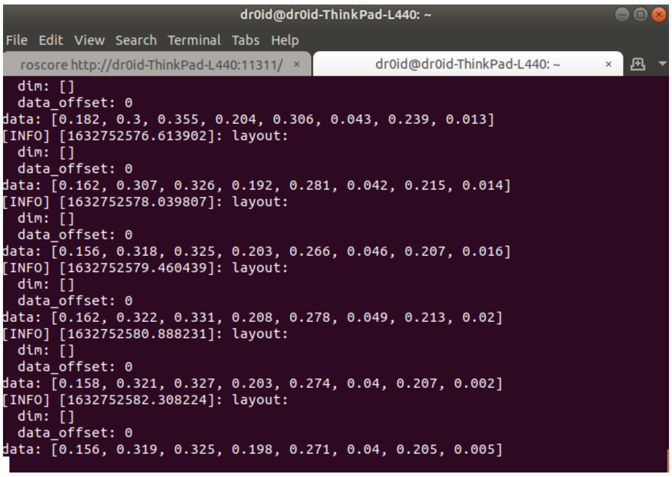

Figure 6. Talker.py during running

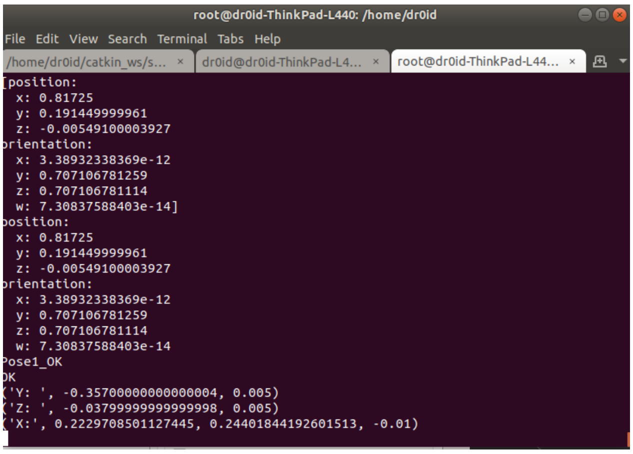

Figure 7. Main.py during running

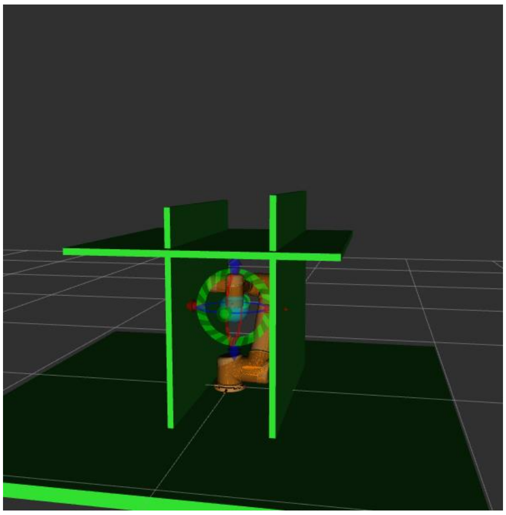

Figure 8. UR5e in Rviz with the safety cage

V. SAFETY FUNCTIONS

During testing, the robot follows unpredictable trajectories after losing track of the hand. Three main safety features were implemented to avoid unexpected trajectories. First, the robot's workspace was virtually constrained. This was solved within Rviz by creating a simple safety cage. We were able to limit the range of motion sufficiently so that the robot cannot damage its environment. When the robot reaches the green wall, it stops automatically (see Figure 8.). Secondly, when the program detects too much movement at the hand between two image refreshes, it stops immediately. This is necessary because if the detection of the hand is incorrect, the robot may receive incorrect data which can lead to unpredictable movement paths and possible collisions. Finally, when the camera loses hand tracking it sends a 0 value to the robot so that the robot stops moving, so there is no need to stop the program in case of signal loss.

VI. PROBLEMS DURING PROJECT AND OPPORTUNITUES DEVELOPMENTS

The project has encountered several problems. The main problem with 3D movement is that only the distance between 0 and 12 points is observed, so a fist strike is enough to move the robot. To solve this problem, we need to replace the webcam with a Microsoft Kinect V2 [5]– [9], which is capable of depth sensing, which would solve this problem. The other big problem is software optimization. The software currently works at a very low FPS. The consequence of this is that it is not possible to make fast movements with the hand because it loses tracking, this should be optimized in the future. One possible solution could be to rework the structure of the program and include Kinect Studio if Kinect V2 is used. This would allow faster movements and tracking on the robot side. With Kinect Studio, it would be possible to program and teach dedicated gestures [3], [14] and hand signals, which would facilitate working with a collaborative robot. Another solution could be to re-implement difficult computational parts/modules in C++, e.g., with python bindings. The use of Leap Motion for higher accuracy was considered, but we could not test and implement this because Leap Motion is not available at the university.

VII. CONCLUSION

The document presents a preliminary project for scientific studies and research. The robot can be controlled by hand gestures in the following ways. The robot will move along the x, y and z axes depending on the quarter of the camera image in which the tracked point on the hand (see Figure 4.), in this case point 0 WRIST, is located and the length of the segment between point 12 MIDDLE_FINGER_TIP and point 0 WRIST increases or decreases compared to the initial length recorded at startup. The experimental results show that with shorter steps and higher robot speed, a less skilled, more fluid and more accurate movement can be achieved. In the future, we would like to replace the webcam with a Kinect V2 or a Leap Motion sensor for more accurate hand detection. Furthermore, we would like to rewrite the more resource-intensive modules or, if necessary, the whole program in C++ to achieve higher performance.

ACKNOWLEDGMENT

The authors thankfully acknowledge the financial support of this work by the Howmet Aerospace Foundation, the Arconic Foundation, and the project no. 2019-1.3.1-KK-2019-00007 implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

REFERENCES

[1] "Gesture-controlled user interfaces, what have we done and what's next? | Request PDF." https://www.researchgate.net/publication/228815683_Gesture-controlled_user_interfaces_what_have_we_done_and_what's_next (accessed Oct. 15, 2021).

[2] "Gesture-controlled user interfaces, what have we done and what's next? | Request PDF." https://www.researchgate.net/publication/228815683_Gesture-controlled_user_interfaces_what_have_we_done_and_what's_next (accessed Oct. 15, 2021).

[3] J. L. Raheja, R. Shyam, U. Kumar, and P. B. Prasad, "Real-time robotic hand control using hand gestures," *ICMLC 2010 - The 2nd International Conference on Machine Learning and Computing*, pp. 12–16, 2010, doi: 10.1109/ICMLC.2010.12.

[4] D. Ionescu, B. Ionescu, C. Gadea, and S. Islam, "An intelligent gesture interface for controlling TV sets and set-top boxes," *SACI 2011 - 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pp. 159–164, 2011, doi: 10.1109/SACI.2011.5872992.

[5] M. Pogatsnik and R. B. Kendrovics: Communication and Reading Comprehension among Informatics and Engineering Students, 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2020, pp. 235-240, doi: 10.1109/SAMI48414.2020.9108764.

[6] Módné Takács Judit, Pogátsnik Monika: The online learning from the students' perspective, In: Szakál, Anikó (szerk.) 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Budapest, Magyarország : IEEE Hungary Section (2021) 507 p. pp. 00027-00032. , 6 p.

[7] György Györök, Bertalan Beszédes. Artificial Education Process Environment for Embedded Systems. In: Orosz, Gábor Tamás (szerk.) 9th International Symposium on Applied Informatics and Related Areas - AIS2014. Székesfehérvár, Magyarország : Óbudai Egyetem, (2014) pp. 37-42. , 6 p.

[8] Pogátsnik Monika: Dual education: connecting education and the labor market, Opus et Educatio Volume 8. Number 3., pp. 304-313, DOI: http://dx.doi.org/10.3311/ope.466

[9] Y. Liu, M. Dong, S. Bi, D. Gao, Y. Jing, and L. Li, "Gesture recognition based on Kinect," *6th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2016*, pp. 343–347, Sep. 2016, doi: 10.1109/CYBER.2016.7574847.

[10] M. R. L. Varshini and C. M. Vidhyapathi, "Dynamic fingure gesture recognition using KINECT," *Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*, pp. 212–216, Jan. 2017, doi: 10.1109/ICACCCT.2016.7831632.

[11] E. P. Ghonge and M. N. Kulkarni, "Gesture based control of IRB1520ID using Microsoft's Kinect," *Proceedings of the 2nd International Conference on Communication and Electronics Systems, ICCES 2017*, vol. 2018-January, pp. 355–358, Mar. 2018, doi: 10.1109/CESYS.2017.8321298.

[12] Y. Wang, C. Yang, X. Wu, S. Xu, and H. Li, "Kinect based dynamic hand gesture recognition algorithm research," *Proceedings of the 2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2012*, vol. 1, pp. 274–279, 2012, doi: 10.1109/IHMSC.2012.76.

[13] C. W. F. Cueva, S. H. M. Torres, and M. J. Kern, "7 DOF industrial robot controlled by hand gestures using microsoft kinect v2," *2017 IEEE 3rd Colombian Conference on Automatic Control, CCAC 2017 - Conference Proceedings*, vol. 2018-January, pp. 1–6, Jan. 2018, doi: 10.1109/CCAC.2017.8276455.

[14] R. C. Hsu, P. C. Su, J. le Hsu, and C. Y. Wang, "Real-Time Interaction System of Human-Robot with Hand Gestures," *2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020, ECICE 2020*, pp. 396–398, Oct. 2020, doi: 10.1109/ECICE50847.2020.9301957.

Dynamic Arc Stabilization Options for Gas Metal Arc MIG/MAG Robot Welding

István Elek Kiss
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
istvanelekkiss@gmail.com

Dr. Károly Széll
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
szell.karoly@amk.uni-obuda.hu
ORCID: 0000-0001-7499-5643

***Abstract* - As the number of welding experts and the cost of automation are reduced, the use of robotics is becoming increasingly important in this field. The purpose of this article is to introduce the automation of the production of a given component as a use case for university studies and experimental settings for further research. The part had previously been welded by hand, but the growing annual order number has necessitated robot welding. The article presents a new welding device that uses a robotic arm equipped with relevant sensors.**

Keywords—MIG/MAG welding, robot welding, welding sensors, digital education

I. INTRODUCTION

Nowadays, the number of skilled welders is declining. In order to perform quality work, the presence of welding robots is becoming more widespread, in both manufacturing and development areas. [1]

Improving production with a welding robot in most cases does not mean improving the equipment, but improving the production environment. For example, it is a common applicability advantage to repair a welded part device for proper quality production. [2] However, this solution is in most cases too expensive, time consuming and does not even provide adequate quality.

When using sensors for robotic welding [3](see Fig. 1.), it is advisable to use a non-contact welding seam tracking system. The intelligent laser sensor (SLS) must be mounted so that it faces the welding gun. The SLPr (Smart Laser Probe) (see Fig. 2.) is responsible for positioning the welding head and also for the welding wire to be in the correct connection. This intelligent laser probe, for most simple straight seams, offers significant benefits in terms of quality and productivity.

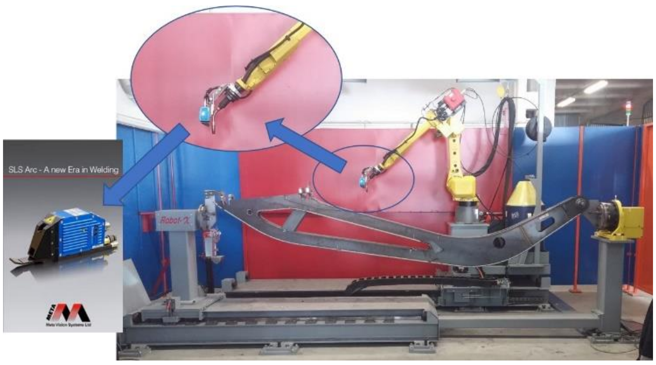

Fig. 1: Robot welding with sensor

The second section of the paper describes the basics of the problem of a welding robot like fixing the workpiece, the coordinate systems of the robot arm and welding.[4] The focus of the third section is the design of the welding jig mentioning those topics which might be important in education. The fourth section introduces a simulation environment for robot Smart Laser Probe which is a completely new and fairly effective approach to seam tracking. Leveraging the latest in electronics and digital technology, SLPr raises seam tracking to a better level of standard system performance.[5]

Fig. 2: Smart Laser Probe

II. TYPICAL APPLICATION

Smart Laser Probe is the tracking system of choice for machine welding applications such as:

- Welding lathes for air, water and all types of cylindrical tanks
- Linear machines, such as side beam, travel carriage and seam welders
- Gantry welders, such as railway wagon side and roof welders
- Simple column and boom applications

Key Features of SLPr include:

- Very high-quality, high-resolution image generation by exploiting several novel and unique design features

- Fully digital system in corporating a completely up-to-date control structure
- User friendly operator interface
- Wide range of standard sensors and interface options for different applications.[6]

If the sensor is used during welding, the programming time is significantly shortened, and the position of the seam can be determined more precisely. I would like to explain this by an example (see Fig. 3.). Welding must be performed on an arc marked in blue.[7]

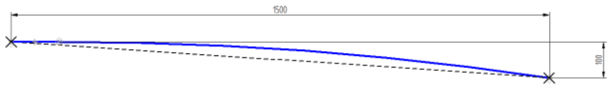

Fig. 3: Programming line by sensor

P1- the starting point of welding P2 is the end point of welding When programming, I record point P1 as the starting point where the robot will start welding. Point P2 is the end point where the welding is completed. If the sensor is switched off, the welding robot connects the two points in a straight line, so the robot would follow the dotted line. By switching on the sensor, the robot takes into account the “pattern” of the given welding radius, thus following the actual welding arc between the two points.

III. PROBLEM DESCRIPTION

As shown in Figure 4., the sensor is attached to the welding gun. One of the biggest disadvantages of Smart Laser Probe is that there is approx. 80 mm constant distance. [8]

Based on these, the robot will not actually start welding at point P1 (figure 3.), but 80 mm forward. We program a specific welding dimensions and determine the tolerance range.

As there is an 80 mm difference between the detection line and the actual welding point, the deformation due to welding is not detected by the robot within this section. This phenomenon is a problem because even in this range, a few tenths of a mm of deformation occurs due to heat input from welding. After determining the position of the workpiece, the next step was to design the fixation.

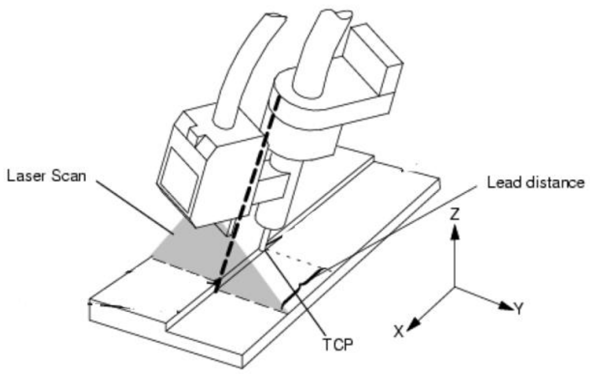

Fig. 4: Head with attached sensor

IV. PROPOSAL

The proposed solution to the aforementioned error is to use an arc sensor. [9]

Through Arc Seam Tracking (TAST) allows the robot to follow the welding seam vertically, the distance between the gun and the workpiece, observing changes in the welding current.

The information provided by TAST allows the system to adjust the robot trajectory to remain the center of the weld in the joint. The path of the robot can be set to the weaving plane and the vertical plane (z direction of the tool). You can use vertical tracking with or without subsequent subtracking, with or without weaving (see Fig. 5.). [10]

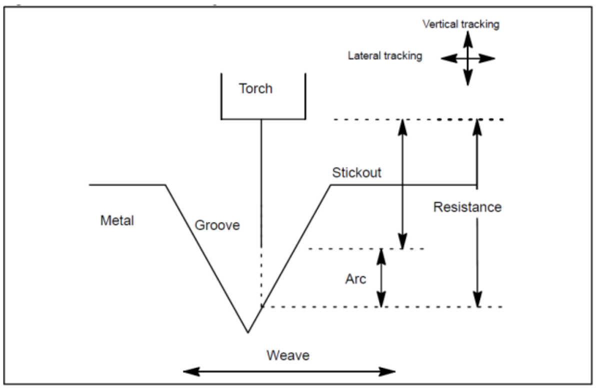

Fig. 5.: Sub tracking

The use of the six-point method for property racks are recommended to adjust the tool frame. When jogging in the tool, the z + coordinate should move along the gun nozzle and be away from work. The following requirements should be met for successful tracking:

- The material thickness should be greater than 2 mm.
- Grooves should have a consistent included angle of 90 degrees or less.
- Fillet joint scan have a maximum included angle of 90 degrees and must have at least 5 mm leg length.
- The minimum weave width must be three times the diameter of the electrode or greater.
- Tack weld, leg size, should be less than or equal than half the weld size, if possible, and concave in profile.
- The actual weld seam should deviate less than 15 degrees rotation from the taught weld seam.
- The torch must be positioned close to the center of the weld seam at the start of the weld (Touch Sensing might be necessary).
- Outside corner and lap joint fillets must use a weave width of 2 mm less than the base metal thickness.
- Fit up of the joint (gap) must be with in normal (blind) welding robot tolerances. Ideally, gaps should be consistent along the weld path
- Base metal must be ferrous or have a resistance greater than mild steel.
- TAST uses SINE type weaving only.