Optimizing Trajectory Points for High Speed Robot Assembly Operations

Florin Anton, Silvia Anton, Silviu Raileanu and Theodor Borangiu

Abstract The work presented in this paper reports a research done in order to optimize the relation speed—trajectory length for a complex robotic assembly task. The assembly task consists in fixing an engine part with 8 screws, the screws being already inserted and pre-fixed but not tightened; in some cases the screws must be screwed for a length of 1 cm, and then tightened at 25 Nm. The operation duration (cycle time) should be of maximum 45 s measured from the time the pallet enters in the working area until the pallet exits the working area. Due to the conveyor operation which takes 12 s to place the pallet in the working position and to remove the pallet from the working area, only 33 s remain for the robot operation including the operation time of the screwdriver placed on the robot. The solution is based on developing an algorithm that uses the dynamics equations of the robot to compute the time needed to accomplish the task, based on the load of the robot and the stop points on the trajectory.

Keywords Robot dynamics · Industrial robot · Trajectory optimization · Assembly · Cycle time

F. Anton (✉) · S. Anton · S. Raileanu · T. Borangiu
Department of Automation and Industrial Informatics,
University Politehnica of Bucharest, Bucharest, Romania
e-mail: florin.anton@cimr.pub.ro

S. Anton
e-mail: silvia.anton@cimr.pub.ro

S. Raileanu
e-mail: silviu.raileanu@cimr.pub.ro

T. Borangiu
e-mail: theodor.borangiu@cimr.pub.ro

© Springer International Publishing Switzerland 2016
T. Borangiu (ed.), *Advances in Robot Design and Intelligent Control*,
Advances in Intelligent Systems and Computing 371,
DOI 10.1007/978-3-319-21290-6_13

1 Introduction

In manufacturing applications where industrial robots are involved, the *cycle time* represents an important production indicator. The cycle time represents the time required for a robot to repetitively accomplish its task; this means that the cycle time is the time spent to execute a production operation in a certain work station in the fabrication line. If the cycle time is shorter, then the productivity is increased, and this is one of the goals for every manufacturer.

The cycle time depends on different parameters, for example the complexity of the task, the trajectory length and shape, the payload of the robot and the tool performance.

This problem has been approached from multiple points of view, for example Carlson et al. [1] proposed a novel method for quality and throughput optimization based on a systematic search algorithm which exploits properties of the welding process in order to minimize the dimensional variation and robot travelling time in welding stations. Huang et al. [2] addresses the problem of realizing multi-robot coordination that is robust against pattern variation in pick-and-place tasks; therefore, they propose combining part-dispatching rules to coordinate robots, by integrating a greedy randomized adaptive search procedure (GRASP) and a Monte Carlo strategy (MCS).

Abdel-Malek and Li [3] addressed the problem of finding the robot location in a manufacturing cell that minimizes the execution time of its assigned tasks. Minimizing the robot cycle time leads to increased productivity in several industrial applications. In this approach, the robot geometric structure and specifications were considered in developing models using Inverse Kinematics to determine the travelling times between the different tasks of a particular manufacturing cell. Then, an optimization scheme is introduced to find the robot base location that minimizes its cycle time.

Nilakantan et al. [4] propose models with dual focus on time and energy to minimize the cycle time and total energy consumption simultaneously: one model (time-based model) with the primary focus to optimize cycle time and the other model (energy-based model) with the primary focus to optimize total energy consumption. The models proposed have a significant managerial implication in real assembly line systems. Depending upon the priorities of the management—primary focus on reducing cycle time or total energy consumption—suitable models could be selected. The proposed models are useful to reduce the total energy consumption and cycle time in robotic assembly lines. It can be observed that the computation time for control with time-based model is less compared to control using energy-based model.

Another approach is to reduce energy consumption without sacrificing cycle time. Björkenstam et al. [5] combined recent algorithms for collision free numerical optimal control and for optimal sequencing, and created an algorithm that was successfully applied to several industrial cases demonstrating that the proposed method can be used effectively in practical applications to find fast and energy efficient solutions. Energy optimization related to trajectory planning was also approached in [1, 6–8].

Our approach is to optimize the cycle time by minimizing the distance which the robot end effector should travel, and to maximize the robot speed to obtain a robot cycle time acceptable for the application and also avoid collisions.

2 Application Description

The application consists in an assembly task executed by an ABB IRB 2400, 6 d.o.f. industrial robot. The robot is mounted on the ceiling above a conveyor belt. On the conveyor belt, engine components are presented on pallets, the components being pre fixed in 8 screws. The time required for this operation is maximum 45 s, but the time taken by the conveyor belt to bring a pallet in the working position and to remove the pallet after the screw tightening operation is 12 s. In this case the time remaining for the robot operation is of 33 s.

The robot is equipped with a Desoutter EME38-20 J [9] automatic screwdriver with telescopic pipe wrench with a screw search feature. The entire configuration: screwdriver, wrench, telescope and robot mounting kit weights 9.3 kg. The weight was determined by executing the robot routine Load Identify from the Call Service Routine group. The screwdriver operation time is about 1 up to 1.5 s for each screw; depending on how well the screws are pre fixed (this operation is done by human operators and some time the screw must be screwed for about 1 cm by the robot before tightening). Figure 1 shows two images with the engine assembly and screw position for screws 1, 2, 3 and 4. The screws are positioned on two rows, and some of them are very close to the engine edges.

Because some of the screws are very close to the edge of the engine and because the pipe wrench has the search feature (that means that it wobbles with ±4 mm) there is the risk that the pipe wrench could enter in collision with the engine. In order to avoid this problem a constraint has been imposed regarding the way in which the wrench should approach the screw: the robot will move using a linear

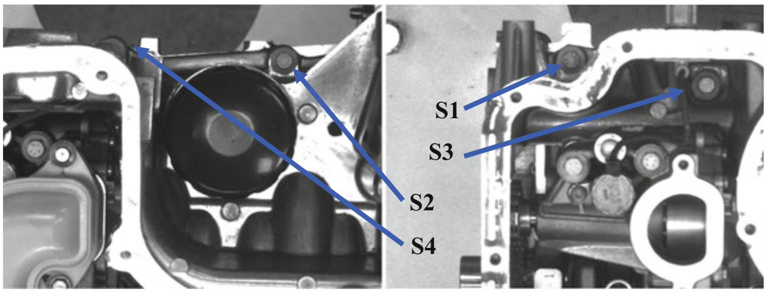

Fig. 1 The engine and screw positions

movement (in Cartesian space) from outside the engine to the screw. The approach point and the point where the wrench touches the screw define a line in space which has a distance of 12 mm from the engine edge in order to avoid collisions (the pipe wrench has a diameter of 12 mm and the wobble has 4 mm; this means that the minimum distance from the engine edge to the wrench edge will be 2 mm).

The sequence of operations in the robot assembly station is:

(a) The robot is available and the workspace does not contain any pallet.
(b) A new pallet is allowed to enter the workspace and the pallet is fixed in place.
(c) The pallet RFID tag is read and the type of engine part is identified (there are three types of engines).
(d) The robot tightens the screws (screw 1–8, and then the screws 1 and 2 are tightened again) and then moves to a safe position.
(e) The pallet is removed from the workspace.

If we compute the time required for all operations one can notice that:

(a) For conveyor operation an amount of 12 s is required.
(b) For screwdriver operation there is needed a minimum time of 1 s × 10 screws (10 s) and a maximum time of 1.5 s × 8 screws and 1 s × 2 screws (14 s).
(c) For robot movement a maximum time of 19 or 23 s remains.

Figure 2 illustrates the moment when the robot engages a screw; one can see the distance on which the screw should be screwed before tightening.

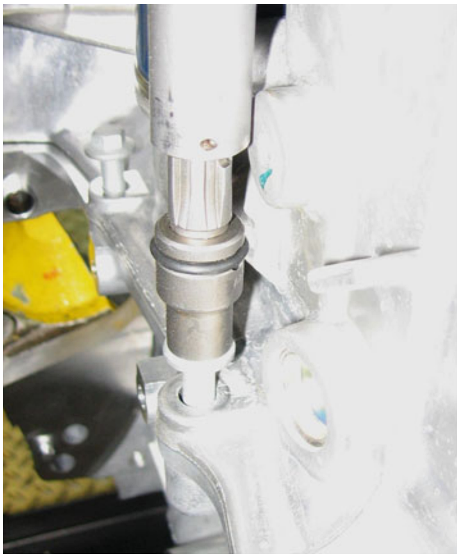
Fig. 2 The moment when the screw is engaged

3 Trajectory and Cycle Time Optimization

In order to optimize the trajectory and the cycle time we started from the constraints of the problem: the first constraint is represented by the points which are used to access the screws, which are fixed. Because the pipe wrench has a telescope with spring, after the screw tightening starts, the screw is moving down and the wrench follows without the need to move the robot.

The second constraint is the shape of the engine and its position in space. The shape of the engine was imported from a CAD file in Matlab and positioned relative to the robot base exactly in the same position as in the assembly station. The shape of the engine and its position were used in order to determine the collisions between the robot and the engine.

In Fig. 3 the optimization algorithm is schematically presented. The initial robot points are the points used to access the screws (extracted from the robot program and trained manually), see Fig. 4; the loop robot point (where the robot is waiting

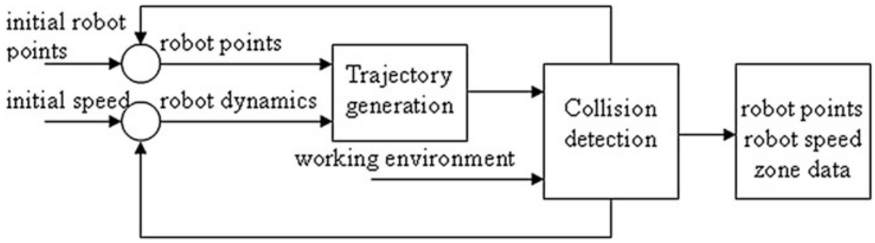

Fig. 3 The algorithm for trajectory and cycle time optimization

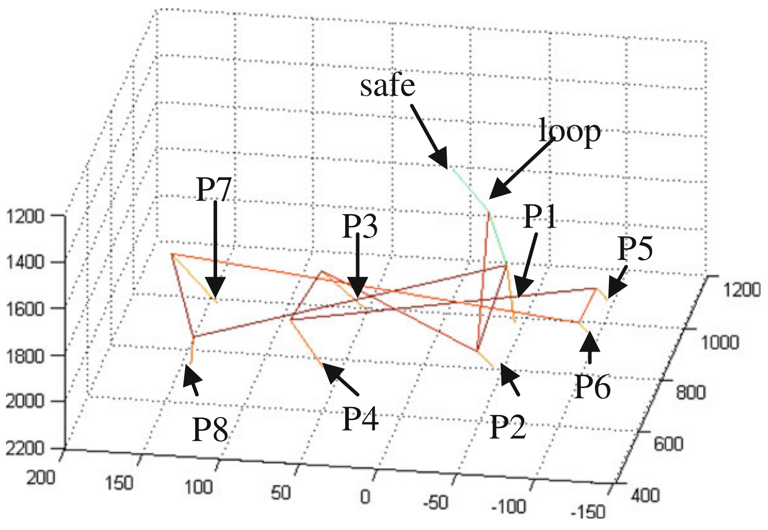

Fig. 4 The positions of the screws

the pallet with the engine) is also trained manually, and the approach points above the screws are used as intermediary points in the robot trajectory. These intermediary points are defined initially as described in Sect. 2 in a plane above the engine at 12 mm.

The initial speed is set to maximum (vmax), the robot dynamics is computed using Robotics Toolbox [10], the trajectory is computed by using the function `ctraj` and the tool weight is also added to the model.

The working environment is defined by the area where the engine is placed; the coordinates are imported in Matlab using the STL file [11] of the engine model.

The collision detection block is based on a function which computes the intersection between the engine model and the robot model (also STL) and the tool defined in the robot model.

The optimization is done in two steps; first the path is computed without taking into account the speed, then the speed and the time required to execute the motion are computed, and finally the trajectory is adjusted in order to maximize the speed.

4 Experimental Results

The trajectory is analysed in sequences: (loop, PA1, P1), (P1, PA1, PA2, P2), (P2, PA2, PA3, P3), (P3, PA3, PA4, P4), (P4, PA4, PA5, P5), (P5, PA5, PA6, P6), (P6, PA6, PA7, P7), (P7, PA7, PA8, P8), (P8, PA8, PA1, P1) where PA is the approach position for each point P1 to P8. For each sequence the zone data has been defined; for final points P1 to P8 the zone is *fine*, for the approach points the initial zone is *fine*, then the zone is increased. Once the zone is increased the PA points are translated on the segment (Px, PAx) in order to avoid the intersection of the zone with the engine. The trajectory is then generated taking into account the dynamics, the collisions are verified and the speed and time is also computed.

Figure 5 shows the simulation window where one can see the robot and the trajectory to be generated.

The zone data is increased until a zone increase is not generating a speed increase or a smaller execution time anymore. The speeds and accelerations are also plotted in order to have a visual check too. The simulation of the robot, the computation of speed and accelerations was executed using the Carlos Baraza [12] simulator.

In Fig. 6 the joint speeds and accelerations for a single execution are presented.

After executing the algorithm the zones are presented in a graphical fashion. Also, the end effector speed, the PA points and associated zone data are saved in a dedicated file. Figure 7 presents these zones.

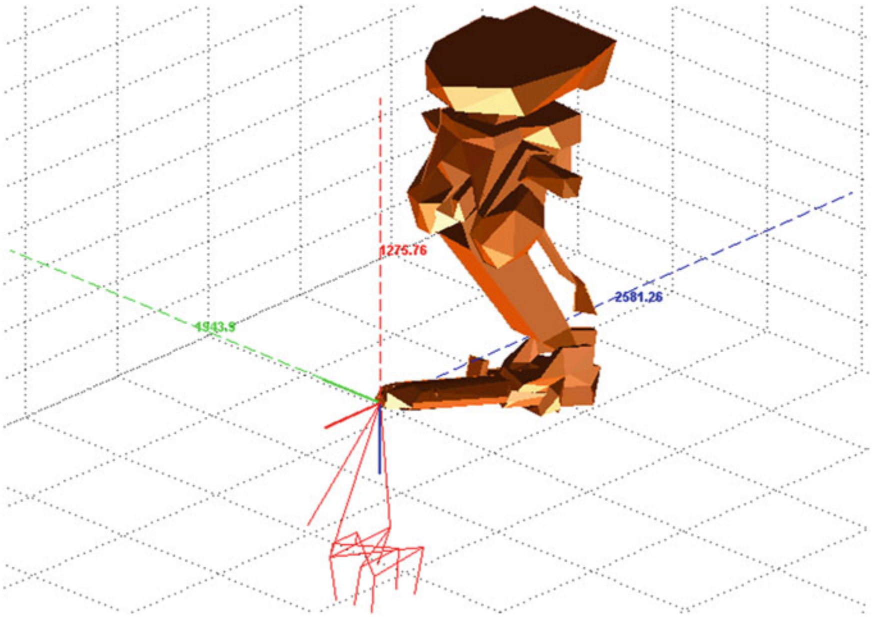

Fig. 5 The robot and the generated trajectory

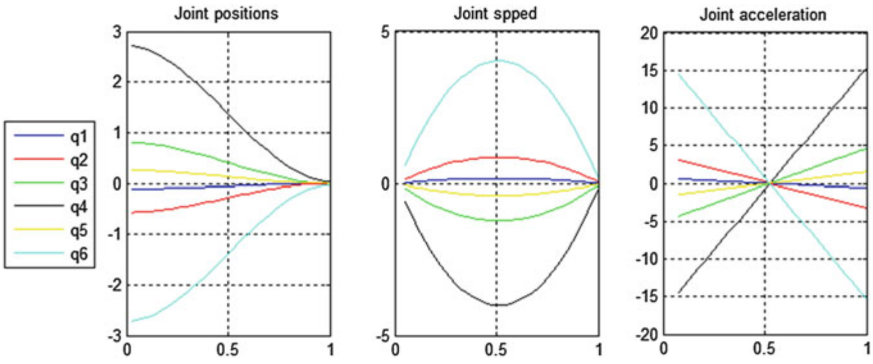

Fig. 6 The joint positions, speeds and accelerations

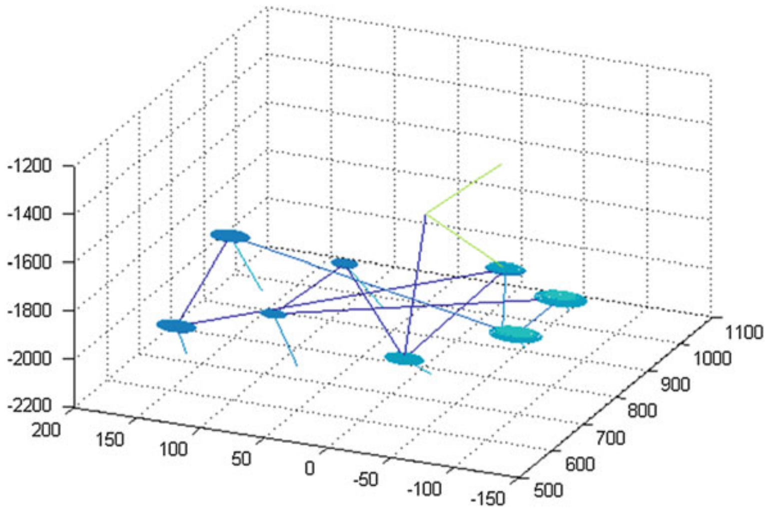

Fig. 7 The resulted zones

5 Conclusion

The resulted speed is v300: this is the maximum speed which the robot can obtain because of the important load which it handles (screwdriver and pipe wrench) and also due to the relative short distances between points.

The speed can be increased by using a set of PAx points defined at a bigger distance from the Px points but this will also increase the execution time. Another way in which the speed can be increased is by modifying the robot's accelerations, but this was not possible from the RAPID robot program, and also this could lead to premature wear of the robot.

The obtained zones were (z50, z50, z30, z25, z80, z80, z50, z40) for PA1 to PA8.

Using these settings in the simulation we obtained a cycle time of 16.53 s only for the robot motion. After the implementation on the robot we obtained a cycle time for the robot and screwdriver operation between 28 and 30 s, this is because the screwdriver operation depends on how well the screws are pre fixed and also because on the simulation the communications between the PLC and the robot were not considered.

Comparing the results with other approaches [13–16] the solution we propose can be easily integrated in manufacturing lines and adds the advantage that the trajectory can be tested offline.

Future developments will consider the possibility to define different zones for the same PA point depending on the trajectory, and also the possibility for the PA points to be tuned on different directions not only on the initial approach direction; this will allow to tune the PA position without increasing the distance that the robot will have to travel.

References

1. Carlson, J.S., Spensieri, D., Wärmefjord, K., Segeborn, J., R. Söderberg: Minimizing dimensional variation and robot traveling time in welding stations, Procedia CIRP. **23**, 77–82 (2014)
2. Huang, Y., Chiba, R., Arai, T., Ueyama, T., Ota, J.: Robust multi-robot coordination in pick-and-place tasks based on part-dispatching rules. Robot. Auton. Syst. **64**, 70–83 (2015)
3. Abdel-Malek, L.L., Li, Z.: Robot location for minimum cycle time. Eng Costs Prod Econ **17** (1–4), 29–34 (1989)
4. Nilakantan, J.M., Huang, G.Q., Ponnambalam, S.G.: An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. J. Clean. Prod. **90**(1), 311–325 (2015)
5. Björkenstam, S., Spensieri, D., Carlson, J.S., Bohlin, R., Gleeson, D.: Efficient sequencing of industrial robots through optimal control. Procedia CIRP, **23**, 194–199 (2014)
6. Fung, R.-F., Cheng, Y.-H.: Trajectory planning based on minimum absolute input energy for an LCD glass-handling. Appl Math Model **38**(11–12), 2837–2847 (2014)
7. Paes, K., Dewulf, W., Van der Elst, K., Kellens, K., Slaets, P.: Energy efficient trajectories for an industrial ABB robot. Procedia CIRP. **15**, 105–110 (2014)
8. Todtermuschke, M., Findeisen, Bauer, M.A.: Methodology for creation a reference trajectory for energetic comparability of industrial robots in body shop. Procedia CIRP. **23**, 122–126 (2014)
9. http://cadfiles.desouttertools.com/files/0003-Documentation/0100-CVI_Range_Tools_-_Controllers_-_Accessories/0140-Fixtured_Electric_Spindles_EM-EME/0141-In-Line/0001-EME_%28One_cable%29/EME38-20J_6159933813-02_pdf.pdf
10. Corke, P.I.: Robotics, vision and control, Springer, Berlin. ISBN 978-3-642-20143-1 (2011)
11. http://www.mathworks.com/matlabcentral/fileexchange/22409-stl-file-reader
12. https://www.youtube.com/watch?v=fa7GwwA3498
13. Jin, J., Gans, N.: Parameter identification for industrial robots with a fast and robust trajectory design approach. Robotics and Computer-Integrated Manufacturing **31**, 21–29 (2015)
14. Menasri, R., Nakib, A., Daachi, B., Oulhadj, H., Siarry, P.: A trajectory planning of redundant manipulators based on bilevel. Appl. Math. Comput. **250**(1), 934–947 (2015)
15. Abu-Dakka, F.J., Rubio, F., Valero, F., Mata, V.: Evolutionary indirect approach to solving trajectory planning problem for industrial robots operating in workspaces with obstacles. Eur. J. Mech. A. Solids. **42**, 210–218 (2013)
16. Kohrt, C., Stamp, R., Pipe, A.G., Kiely, J., Schiedermeier, G.: An online robot trajectory planning and programming support system for industrial use. Robotics and Computer-Integrated Manufacturing **29**(1), 71–79 (2013)