Offline Programming for a Complex Welding System Using DELMIA Automation

Joseph Polden, Zengxi Pan, Nathan Larkin,
Stephen Van Duin, and John Norrish

Faculty of Engineering, University of Wollongong,
Wollongong, NSW, 2530, Australia
e-mail: zengxi@uow.edu.au

Abstract. This paper presents an offline programming (OLP) system for a complex robotic welding cell using DELMIA Automation. The goals of this research are aimed at investigating the feasibility of taking a commercially available robotic simulation package, DELMIA, and to use a Visual Basic Automation interface to reduce the programming time by creating automated 'modules' to carry out some of the tasks in the OLP process. The paper first investigates and presents the structure of OLP as a discreet method of individual steps. These steps are then evaluated for their potential as an automation candidate. The methods in which these steps are automated are then presented. A general analysis of the developed OLP system was carried out, providing a scope for future research and development.

1 Introduction

A manufacturing facility in Australia has, over the last number of years, been tasked with handling the manufacture of an automobile hull. The vehicle hull is composed of steel plates, and is constructed in a monocoque type assembly with all of the various plates that make up the hull being welded together by the same way as a ship vessel. At the time of writing, the welding of the hull was being carried by manual processes alone; however an increase in future production demand has pushed the manufacturing facility to automate the welding processes on the hull via implementation of a complex robotic welding system.

Due to the high number of seams to be welded and the complex geometry, which is inherent in the hull's design, a specialized robotic cell was needed to maximize the number of external and internal seams that could be completed by the cell. This cell consists of two 6-DOF articulated welding robots. Each of these robots is then in-turn mounted on another, larger, 6-DOF 'auxiliary' robot and linear rail to create a form of homogenized 13-DOF robot, as shown in Figure 1. The hull itself is also mounted on a rotating trunnion to allow the welding robot access to areas such as the roof of the hull, or allow the weld robot better internal access through the opening such as the windscreen orifice. The robotic cell features appendages such as laser scanner and heat sensors for calibration purpose.

T.-J. Tarn et al. (Eds.): Robotic Welding, Intelligence and Automation, LNEE 88, pp. 341–349.
springerlink.com

The robot-on-robot set up required a state-of-the-art communication system to ensure each robot was interfaced and could communicate correctly.

The robotic cell was originally programmed via online jog-and-teach method. However the highly complex nature of the cell is a great hindrance to an efficient programming solution. A lot of time was invested in teaching the welding robot all the seam locations, as the extra degrees of freedom added by the auxiliary robot removed a lot of intuitiveness in manually jogging the robot to a specific target location without clashing with the vehicle hull; particularly when negotiating through complex internal geometry.

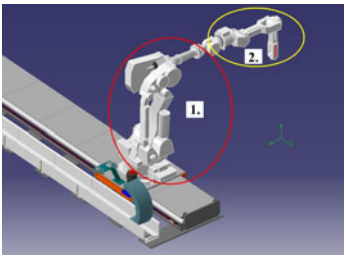

Fig. 1. A model of the homogenized 13-DOF robotic welding system, featuring two separate 6-DOF robot and a linear rail

The manufacturing company is now anticipating orders of other configurations of the vehicle, meaning that the robot cell will need to be reprogrammed to accommodate these various models of the vehicle. After the initial difficulties experienced when utilizing the online programming method; it was deemed necessary to explore options in which the robotic cell can be re-programmed for these new designs in a much more efficient manner. Researchers at the University of Wollongong proposed an offline programming approach as an alternative, hoping to create an automated programming system utilizing a simulation package widely used in industry today.

At the outset of the project, a literature review of current OLP software was conducted [1]. The review indicated that OLP software mainly came from 3 sources; from generic robotics software producers, from robot manufacturers [2-5] and finally from research institutions that produce their own programming and simulation software, usually developed around existing CAD software such as AutoCAD or SolidWorks [6-9] or from scratch using OpenGL, VRML and Java technology [10-12].

To create an OLP software package for this complex welding cell, it was decided to utilize a commercially available generic robotic software package. This was chosen as a generic system can be much more flexible in its compatibility with various brands of system hardware and also features virtual reality, allowing the user to be fully immersed into the simulation environment. The DELMIA software package was chosen. Its current and widespread use in robot programming for industry and manufacturing processes, along with its Visual Basic

programming interface, were major factors behind its selection. Details of the DELMIA robotics simulation software package are covered below, in section II.

2 DELMIA for Offline Programming and Automation

DELMIA is a robotics and manufacturing based OLP package that is utilized widely within various respective manufacturing industries today. DELMIA utilizes a 3D simulation environment to test and optimize robot programs before implementation into real world applications. DELMIA features assorted 'toolboxes' that are available to provide a programmer with numerous functions which are useful to the various specific areas of robotic OLP, such as; robot target definition, reachability analysis, clash testing, path planning/augmentation etc.

A user carrying out the programming of a robotic cell with DELMIA would follow the general steps for OLP highlighted in Figure 2.

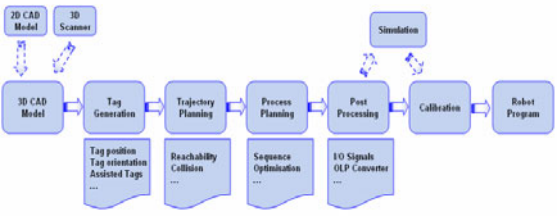

Fig. 2. Block diagram of overall offline programming structure [1]

To aid in the programming of the complex welding cell in question, it is proposed that some of the steps in the above OLP process be automated. These areas related to 3 specific sections of the OLP process: The automatic extraction of seam data from CAD models, the automated generation of reachability and collision assessments and the automatic creation of the robot process with simulation.

The proposed automation of these DELMIA functions was carried out with the Visual Basic interface that comes with DELMIA. DELMIA's robotics related commands which were not accessible through the VBA functionality were controlled via the windows GUI automation program; AutoIT.

3 Automatic Seam Extraction

Defining the weld seams to be carried out by the robotic cell is the first step in the OLP process. In DELMIA, these seams are defined by first loading a 3D CAD model of the work object to be welded. The programmer identifies a seam, and then defines it by individually allocating two separate tags at each end of the specific seam. The tag's individual XYZ orientation angles are then augmented by the

user to specify the correct approach angles for the manipulator/weld-torch. This process is not a difficult one, however when a high number of seams are to be defined by the user, it has proven to be a monotonous and time-consuming task. The VBA/DELMIA interface was identified as a tool, which can analyze the drawing features that make up the CAD model of the work object. A programming module was created to aid the programmer in efficiently defining these weld seams. The module assists by providing a 'semi-automated' method of defining the weld seam and then automatically snapping the tags to the seam.

The semi automated approach reduces the time taken to define a seam by first prompting the user to select the edge they wish to be defined as a seam. The tags are then attached automatically to each end of the selected seam. The user is then prompted to click the two adjacent faces that make up the seam, this is done to define approach/orientation angles for the weld torch as it carries out the weld process, as seen in Figure 3 below. The semi-automated approach developed provides the programmer with a more effective interface to tag their work objects than standard methods available in DELMIA. This is due to the fact that standard methods for tagging in DELMIA require that the user first add two separate tags in the correct location to define the seam, and then they are able to orientate it for the correct approach angles. The semi-automated approach, however, already assumes you are searching for an edge to define as a seam; once the edge is selected by the user the tags are added and orientated automatically. This significantly reduces the amount of individual operations the user has to carry out in order to define a seam, hence cutting down the overall time required. Initial tests on tagging a vehicle hull indicate that the time taken to define each seam in the entire hull was more than halved when using the semi automated approach over standard methods currently available in DELMIA.

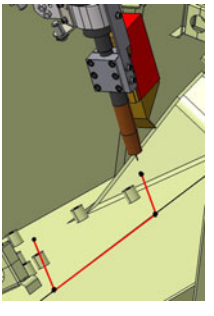

Fig. 3. The semi-auto hull tagging module defines the seam edges and approach/orientation angles for the weld torch

Once the seams have been defined by the user, the data relating to their position and orientation is automatically added to an excel spreadsheet for further use at a later stage of the OLP process.

4 Path Planning

Once the programmer has defined all of the seams within the vehicle that are to be welded they then move onto the next programming task, which relates to planning the motions required for the robotic cell to correctly carry out the welds. To undertake this task with the complex welding cell being used in this research, the programmer first has to program the auxiliary robot to carry the weld robot to a specific point in space which fulfils two criteria. Firstly, this point in space has to be close enough to the seam to ensure that the weld robot can reach its target. Secondly, specific orientations for the weld robot have to be selected so that it can carry out the welding of the seam without colliding with the work object. As each seam to weld has different locations and orientations in space; the programmer essentially has to find a new point in space that for fills the above criteria for each individual seam that is to be welded. Due to the high number of individual seams that feature on the vehicle in the complex welding cell, the repetitive nature of finding these points becomes monotonous and time consuming. However the repetitiveness of the task also highlighted that it is an ideal candidate for automation, meaning that a lot of time to program the cell will be saved if it were possible to control DELMIA in a way so that it can define automatically for the programmer these points in space, meaning that they don't have to spend the time to find it 'manually'.

The approach for automating this task was to address the two criteria mentioned above as separate modules. The first, which related to automatically finding points in space which fulfilled reachability criteria, was addressed before moving onto the second criteria of defining the correct motions and orientations for clash free motion when welding the seams. To automate the reachability assessments, a 3D array of potential positions for the auxiliary robot is defined by the programmer in an excel spreadsheet; the user can define the position of the array, the volume it occupies and the number of nodes or targets within the array. The inverse kinematics of the weld robot were mapped in Matlab, which utilized the previously defined excel data to calculate the reachability of the weld robot from each test node in the array to the previously defined weld seams within that area. These reachability results are fed back to the spreadsheet so that the programmer can see how many seams can be welded from each potential target in the 3D array of nodes. DELMIA is capable of carrying out reachability tests; however the VBA interface in DELMIA is restricted in its access to this class of commands, making automation of these commands with the VBA interface a non-functional pursuit. Matlab was used and the M-file created was able to automatically carry out these tests in a very fast and efficient manner. The results displayed to the programmer were offered in an intuitive yet detailed presentation. The programmer is able to see which nodes can provide the weld robot with reachability to a single seam, multiple seams (which are listed) or no seams at all. Data relating to where the node is in 3D Cartesian space is also displayed, along with the orientation (roll/pitch/yaw angles) of the node.

After the reachability testing is concluded, the next 'module' is to automatically test whether the weld robot can move from the specific pre-defined nodes to the

weld seam without clash. To automate these commands, the DELMIA program was interfaced with AutoIT, which is a program designed to control and operate with the windows GUI; meaning that instead of a human operator manually clicking through the DELMIA commands and testing clash, AutoIT was programmed to carry out the same operations. AutoIT was programmed to read the excel spreadsheet utilized in earlier sections of this research. It reads the calculated reachability data and then prompts DELMIA to move the position of the weld robot to each node previously deemed to have a positive reachability result. The robot is then commanded to move to a specified seam that has been deemed reachable at that particular node. As this motion is carried out, DELMIA monitors for clash. AutoIT cycles through each node and returns the clash results back to the excel spreadsheet. Each available robot configuration is also tested for clash, and the data relating to which specific configurations provide a clash free motion is also displayed in the excel spreadsheet of results.

The result of these automated tests is essentially an array of targets for the auxiliary robot to carry the weld robot to. The robot programmer can be safe in the knowledge that these targets will firstly have at least one seam that is within reach of the weld robot from the particular array node. These nodes will also provide a clash free motion for the weld robot as it carries out its weld process on the seams. For optimisation, the programmer can easily check the created excel spreadsheet to select which nodes will be utilized in the final robot program. This is done easily and intuitively as they are provided with the data relating to the seams that are reachable from specific nodes; so they can easily select the fewest individual nodes required to carry out all seams within one particular area of the hull. This will effectively cut down the number of times the weld robot will have to be repositioned within the hull as it carries out its weld processes.

Once the correct positioning for the weld robot and suitable configurations for clash free motion have been defined the robot tasks for each robot is created using this data. At the users request, the complete process is simulated from the beginning to verify that the procedure is carried out correctly and without clash. Once the process is verified, the Visual Basic interface exports the native robot programs to a folder on the computer desktop. If, for some reason, during the verification process an error such as clash or unreachability occurs then an error file is also exported with the program. This text file contains the nature of the error and the simulation time at which it occurred, making it easy for the programmer to trace through the simulation and fix any problems before exporting the program again.

5 Experimental Results

The effectiveness of the created automation system was tested on a number of seams on the vehicle hull. Visual Basic was used to create a user interface, in which the programmer has access to the various buttons and controls that manage the operation of the OLP automation system created. The first automation control the user utilizes in the OLP process is the automated generation of the seam targets for the welding robot. The proposed seams to be created were located in an

internal section at the rear of the hull in order to fully test the capabilities of the automation system.

The user first enters a desired seam name into an input box on the Visual Basic user control panel, and then clicks the 'Create Tags' button. A blank model of the vehicle hull in a new DELMIA window with all of its drawing features exposed to the user is then opened. A pop up window prompts the user to select the edge between two plates that will form the first seam that is to be defined. The user is then prompted to click on the incident faces to this selected edge. Once the faces have been selected the blank model of the vehicle hull is closed and two tags are placed at each end of the previously selected edge on the hull in the complex welding cell. The previously selected faces automatically orientate the seam tags with the appropriate approach angles for the weld torch, which includes any push/pull angle if required in a corner. These tags are then automatically renamed to the previously defined seam name and are saved into a specific 'semi-auto' tag group in DELMIA. The seam is then fully defined and the user can now move on to repeat the process as many times as they want. Three seams were defined inside the rear section of the hull using the above method. The 'export seam data to excel' button on the user control panel was used to export all data relating to the seam tags locations/orientations to a specific Excel spreadsheet. A list of seams in the excel sheet was created, and the new tags imported are added to the end of this list. This data is used during the later stages of the OLP automation process. Another button on this tab has the capability of importing this list of tags back into the DELMIA environment if required, allowing the user to make modifications to the tag's location/orientation in the excel environment and then import these modified tags back into the simulation model.

Once the Seams have been defined the user then clicks on the 'Path Planning' tab on the user interface to expose the next set of automation controls. Before beginning the path planning automation functions, a matrix of targets for the Auxiliary Robot to place the Weld Robot has to be defined about the rear of the Hull. This is done in the same spreadsheet as the stored seam tag data. The user defines this matrix by specifying in a specific section of the excel sheet the upper left and lower right hand corners of the desired positioning matrix and also entering the desired number of nodes in the matrix. These coordinates are found using the DELMIA simulation model and the DELMIA compass tool.

The user selects which robot programs they would like to generate as a result of these automated tests, in this instance the button to generate both the weld robot and auxiliary robot programs was selected. The programmer can also check the 'Validate with Simulation' box, which runs a final validation check on reach and clash at the end of the OLP process. All that remains for the user to do is enter the name of the weld seam they wish to create the programs for and utilize the 'Generate Code' button on the user interface to initialize the Matlab, Visual Basic and AutoIT automation components listed in section IV.

After validating the process with simulation from start to finish the final output is two separate robot programs. The first program commands the auxiliary robot to move the weld robot to a suitable position close to the weld seams. The second robot program commands the weld robot to move from this base position to carry

out the seam weld without clashing with the vehicle hull. If desired the user can at this stage make 'manual' adjustments to the robot task within the simulation environment by using traditional DELMIA commands and then export the modified robot programs again.

Once the first seam has been completely programmed, the operator then moves on to obtaining the robot programs for the remaining two seams at the rear section of the hull. This is done by inputting the next remaining seam name into the user control panel and then clicking the 'Generate Program' button again. Once the program has finished obtaining the code for these seams, the final seam at the rear of the hull was addressed in the same fashion. The time to obtain the programs to correctly weld these three seams took the automation module approximately 5 minutes from start to finish. This result provides a significant improvement over using traditional 'online' jogging methods currently employed by the manufacturing facility.

6 Conclusion and Future Work

This research has shown that it is possible to modify currently available simulation software to automate some of the steps in the OLP process. The developed modules provide automation functionality to both the tag generation and trajectory planning stages of the OLP process, giving an overall improvement to the time taken for a programmer to produce code for a robotized welding cell.

Whilst the created system is able to provide a level of success in delivering a working package, there are a number of noted issues, which have a negative impact on its operation. The main issue negatively affecting the process relates to the level of functionality that is exposed in the DELMIA/VBA interface. DELMIA, in its current development state, provides access to the majority of its functions via the VBA programming interface. However, access to DELMIA's robotics related functions was minimal. This resulted in having to use indirect methods, such as accessing the commands through DELMIA's GUI rather than getting direct access to use certain functions or commands. Examples of this inaccessibility include restricted access to the reachability and clash checking commands. Whilst the overall speed is a significant improvement over 'manual' methods of offline programming, there exists room for improvement in the developed OLP automation package. To overcome this, work has begun on creating new OLP modules in Matlab to replace some of the tasks in the OLP process, removing the reliance on the slower VBA/DELMIA interface in much the same way Matlab was implemented in section IV to carry out the reachability assessments of the weld robot. The main goal behind this is to improve the efficiency of these tasks, whilst also improving the reliability of the program by moving the role of DELMIA/VBA towards being just a tool for simulation.

Acknowledgement. This work is funded by the Australian Defence Materials Technology Centre (DMTC).

References

[1] Pan, Z., Polden, J., Larkin, N., Van Duin; Norrish, J.: Recent Progress on Programming Methods for Industrial Robots. In: ISR/ROBOTIK, Munich, Germany, June 7-9 (2010)

[2] Brown, R.G.: Driving digital manufacturing to reality. In: Proceedings of 2000 Winter Simulatin Conference, December 10-13, vol. 1, pp. 224–228 (2000)

[3] Bruccoleri, M., D'Onofrio, C., La Commare, U.: Off-line Programming and simulation for automatic robot control software generation. In: 5th International Conference on Industrial Informatics, June 23-27, vol. 1, pp. 491–496 (2007)

[4] Dong, W., Li, H., Teng, X.: Off-line programming of Spot-weld Robot for Car-body in White Based on Robcad. In: International Conference on Mechatronics and Automation, ICMA 2007, August 5-8, pp. 763–768 (2007)

[5] Lee, D.M.A., Elmaraghy, W.H.: OBOSIM: a CAD-based off-line programming and analysis system for robotic manipulators. Computer-Aided Engineering Journal (October 1990)

[6] Pries, J.N., Godinho, T., Ferreira, P.: CAD interface for automatic robotic welding programming. Industrial Robot: An International Journal 31(1), 71–76 (2004)

[7] Mitsi, S., et al.: Off-line programming of an industrial robot for manufacturing. International Journal of Advanced Manufacturing Technology 26, 262–267 (2005)

[8] Soron, M., Kalaykov, I.: Generation of continuous tool paths based on CAD models for Friction Stir Welding in 3D. In: Mediterranean Conference on Control & Automation, MED 2007, June 27-29, pp. 1–5 (2007)

[9] Yang, Y., Chen, X., Ling, C., Kang, B.: A Robot Simulation System Basing on AutoLisp. In: 2nd International Conference on Industrial Electronics and Applications, ICIEA 2007, May 23-25, pp. 2154–2156 (2007)

[10] Dai, W., Kampker, M.: PIN-a PC-based robot simulation and offline programming system using macro programming techniques. In: The 25th Annual Conference of the Industrial Electronics Society, November 29 –December 3, vol. 1, pp. 442–446 (1999)

[11] Jaramillo-Botero, A., Matta-Gomez, A., Correa-Caicedo, J.F., Perea-Castro, W.: ROBOMOSP. IEEE Robotics & Automation Magazine 13(4), 62–73 (2006)

[12] Kim, C.-S., Hong, K.-S., Han, H.Y.-S., Kim, S.-H., Kwon, S.-C.: PC-based off-line programming using VRML for welding robots in shipbuilding. In: IEEE Conference on Robotics, Automation and Mechatronics, December 1-3, vol. 2, pp. 949–954 (2004)