Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 17 (2018) 270–277

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USA

A Collaborative Framework for Robotic Task Specification

Steven Brown, Harry A. Pierson

University of Arkansas: Department of Industrial Engineering, 800 W Dickson St., Fayetteville, AR 72701, USA

Abstract

Implementation of automated robotic solutions for complex tasks currently faces a few major hurdles. For instance, lack of effective sensing and task variability – especially in high-mix, low-volume processes – creates too much uncertainty to reliably hard-code a robotic work cell. Current collaborative frameworks generally focus on integrating the sensing required for a physically collaborative implementation. While this paradigm has proven effective for mitigating uncertainty by mixing human cognitive function and fine motor skills with robotic strength and repeatability, there are many instances where physical interaction is impractical, as human reasoning and task knowledge are still needed. The proposed framework consists of key modules such as a path planner, path simulator, and result simulator. An integrated user interface facilitates the operator to interact with these modules and edit the path plan before ultimately approving the task for automatic execution by a manipulator that need not be collaborative. The software modules are integrated using the Robotic Operating System. Application of the collaborative framework is illustrated for a pressure washing task in a remanufacturing environment that requires one-off path planning for each part. The framework can also be applied to various other tasks, such as spray-painting, sandblasting, deburring, grinding, and shot peening.

Keywords: collaborative robotics, path planning, flexible automation

1. Introduction

Since the first industrial implementations of robotic solutions in manufacturing environments, task specification has been one of the toughest and most time-consuming parts of the implementation process. As robotics has advanced, so has the technology surrounding task specification; however, there is still a need for the operator to physically program the robot. While this is fine for low-mix, high-volume production processes, it is a very

10.1016/j.promfg.2018.10.046

restrictive requirement for the automation of lower-volume processes. Automated task specification would go a long way toward alleviating some of the hurdles faced by high-mix, low-volume processes. However, the implementation of automated robotic solutions for complex tasks currently faces a few major hurdles. Lack of effective sensing and task variability create too much uncertainty to reliably hard-code a robotic work cell. Collaborative robotics have proven effective for mitigating uncertainty by mixing human cognitive function and fine motor skills with robotic strength and repeatability. Yet, there are many instances where physical interaction is impractical, as human reasoning and task knowledge are still needed. The solution is a framework that blends the latest developments in automated task specification with the experience and cognition of a human operator to provide a more accurate task specification. While this paper does focus on surface finishing tasks such as pressure washing, sandblasting shot peening, deburring, grinding, sanding, and wire brushing, the framework can also be applied to any robotic task that does not have a predefined path, such as assembly, inspection, packaging, and pick-and-place operations.

The inspiration for this paper is a pressure washing work cell. The current work cell is an entirely manual operation with the operators being subjected to high ergonomic risk factors [1]. As such, this process is a strong candidate for automation, but high degrees of variability and uncertainty, combined with extremely difficult perception problems (e.g., differentiating black paint from grease) make traditional robotic automation impractical. By designing an automated system to suggest a toolpath for cleaning and then using the operator's intelligence and understanding to inform the automated side of potential changes, the system can consistently handle the variability in the process.

2. Literature Review

While there are many well-documented methods, safety measures, and best practices for general robotic implementations, there are few frameworks designed for the challenges and needs of automating a specific task. The most significant research has been in a software-based approach to connect various sensors and actuators together to create complex systems, such as robots, and has resulted in the formation of the open-source Robotic Operating System [2]. While the ROS consortium and others focus on the integration of tools, sensors, and some external software, other research has focused on how robots communicate within themselves [3]. Depending on how intra-robot communication is viewed, this can be interpreted in one of two ways: either by considering each piece of a robot as its own robotic module or by considering a group of similar robots focused on the same task. When considering a modular robot, there are steps that can be taken to design the optimal robot based on the available modules and the needs of the task [4]. While this method does help when deciding what style or configuration of robots is needed, it does not address anything other than the physical requirements of the task. When considering communication across multiple robots, there are a variety of methods being used to manage the interactions of multiple robots, from linked pathed planners to swarm intelligence [5, 6]. This has predominately been a focus of mobile robotics, especially with the rise of cleaning and delivery robots [7, 8]. With the ability to control multiple robots or parts of a robot independently to accomplish a task, other research has looked at how a distributed system might manage multiple simultaneous requests either by prioritizing certain tasks over others or by attempting to complete multiple tasks at the same time [9]. On the collaborative side, there are some general frameworks for how a robot could communicate with a human, but they are focused around mobile robotics and collision avoidance [10].

Over the past few years there has been a major push in the robotics community toward a new style of robot that can better interact with human operators. Called collaborative robots, or cobots, they are designed to work together with humans to accomplish a task in the most productive way possible by leveraging the strength and endurance of robots with the flexibility and decision making of humans [11]. They are able to do this by integrating new safety standards and methods into this new generation of robots and by refitting systems with older industrial robots to meet the new safety standards as discussed below. These new safety standards have allowed for numerous new automation opportunities, both in how robots are used and where they can be used [12, 13].

Traditionally, whenever an operator needs to physically interact with a robot in any way, they need to use a lock-out procedure to ensure that either the robot's servos are turned off or the robot is locked in place by some other mechanism. With safety-rated monitored stops, this is no longer necessary. As long as the robot does not move from its current position, the operator is free to enter the workspace without shutting down the robot or going through a lock-out procedure. A hand-guided cobot allows for the operator to directly affect the position of the robot with their hands without deactivating the servo motors. This is especially useful for robotic arms with axes that can be easily

affected by gravity and would usually require power to maintain their position. This style of collaboration allows for faster and easier teaching and programming, and allows humans to use the robots to lift the majority of a heavy load while the operator guides it into place. A cobot utilizing speed and separation monitoring allows the operator to move freely throughout the workspace while the robot is in motion, as long as a dynamically defined minimum separation distance is maintained between the robot and the operator; otherwise, the robot will immediately initiate a protective stop. Power- and force-limiting robots are specifically built for physical contact with the operator that can occur both intentionally and unintentionally by limiting the maximum capable applied forces to comply with defined threshold limits [14].

Aside from the physical interpretations of collaborative operation, there are also many human interface changes that can make a robotic implementation collaborative. As discussed above, some of the collaborative operating methods can be used to enhance the programming experience by allowing the human to interact with the robot [15]. While this does not lead to a collaborative operation, it does minimize the time spent on setting up the operation, which can be just as valuable.

3. GENERAL FRAMEWORK DESIGN

This section discusses the modules required within the framework as well as reveals the key attributes for the success of each module. Figure 1 illustrates how the individual modules interact with each other, the external components of the system, and the human operator. From a high level, the system takes the provided 3D data and initial user input as parameters into the path planner to generate a path. The path is then sent to the path analyzer before the simulation displays the original 3D input, the path, and the analysis. From here, the operator can decide to accept the proposed task as is or make adjustments. If necessary, the adjustments are made by the path modifier module and then sent back through analysis before the operator has the opportunity to make another decision. Upon approval, the path is passed to the robot and the task is completed. However, if the operator notices that there are still unsatisfactory spots, the process can be started again with either the full part or a smaller section being passed to the path planning module.

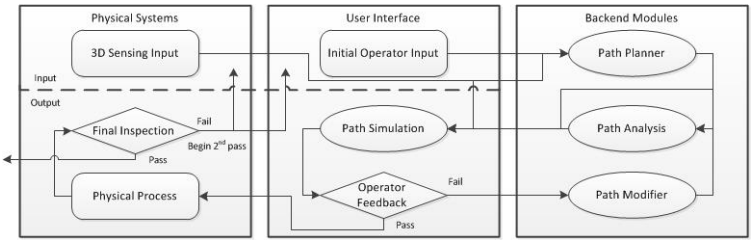

Fig. 1. System framework design.

There are two types of data required for any robotic task specification system: the specific geometry of the object and the parameters of the process being specified. Each type of data can be acquired through various means and faces its own unique issues. For instance, there are a variety of representations of 3-dimensional data, and the data can be easily affected by many environmental variables. The same is true of process parameters. There can be quite different types of parameters that may need to be derived or subjectively chosen by a human operator, which creates another layer of uncertainty. The most important piece of any data collection module or process is that there is a standardized method for doing so that eliminates as much uncertainty as possible.

One might assume that it is feasible to fully automate the process and eliminate a vast majority of the uncertainty, but there are still some large issues surrounding 3D data collection that need to be solved before that is possible for a one-off task specification system [16, 17]. One of the biggest is the need to ensure that the data is completely accurate. 3D sensing technologies can still be easily fooled due to environmental or surface conditions. Lighting plays a huge part in achieving an accurate scan of the part. If a surface is not reflecting the light as the system expects an error may occur in the final rendering. This could ultimately lead to collisions during the task. There is also the possibility that the true part geometry is being obscured by dirt or debris. While this is not problematic in processes such as pressure washing, it can cause challenges in processes such as deburring, where debris could be interpreted as integral to the piece and thus not be removed. Another concern is that the environment or the process

itself could cause problems for the sensing mechanism. Any spray, smoke, particles, or general debris being scattered about during the process, along with any additional environmental variable, could obscure or alter the view of the sensors. Covering the sensors to protect them and scanning ex situ both present issues. Not only is it necessary to touch the part twice, but there is the concern about orientation and registration issues once the part is placed in the workspace. Achieving the appropriate orientation and registration of an unfixtured part with no on-site 3D sensing equipment puts a significant burden on the operator to get things exactly right every time.

Other big concerns for the initial data collection are how to ensure that the information provided by the operator is accurate and how to determine the right amount of human interaction required to maintain a flexible yet accurate system. These questions are interdependent. For the right amount of human interaction to be determined, one must understand how much the input can vary based on human judgement and error, and how much that particular input affects the task specification process. For example, orienting and registering an unfixtured part, as described above, requires the user to match points on the part in the robot's workspace with the same points on the 3D model. This brings in not only error from the operator's judgment about it being "close enough" but also the error in the measuring device used. Finding the right balance between accuracy and usability while maintaining operator support can be difficult. It is important to note that while robotic automation has seen a huge surge in popularity across many industries, only about 10 percent of manufacturing jobs have been automated [18]. Part of that is due to the fact that many workers are less likely to accept robots that completely replace their job without any mistakes. In fact, recent studies show that "clumsy robots" that sometimes need help or make mistakes are better received by humans [19]. This is where collaborative systems can be most beneficial, by allowing limited user control and feedback to inform the robot of what should be done while maintaining a higher level of accuracy and precision.

3.1. Path Planner

The path planner has the largest influence on system performance, which is not surprising considering the large amount of work that has already been done in the area. From seed and slicing based models to advanced genetic algorithms, there are multitudes of ways to generate an initial tool path [20, 21]. There are several key factors to consider when designing a path planning module. First, knowing and understanding the process and its requirements is key to building an accurate path planner. The path needed to deburr a surface must be much more precise than the path needed to sandblast that same surface. The controls are also different. A deburring operation needs to control which grinding tool is used and its cutting speed, whereas a sandblasting operation needs to control the type and quantity of sand being used. The sandblasting operation would also be more affected by the excess coverage issues, which places more significance on finding non-overlapping paths. Another key consideration would be how to link parts of the path with the corresponding area on the surface. This is essential when considering location-specific user input and feedback. Without any way to link areas in need with specific segments, the entire path would need to be rebuilt every time a change is needed to be made. Another staple of a good path planner is robustness to the noise surrounding the part. Depending on the process, this can be achieved by generalizing the original geometry or sometimes ignoring specific pieces during the planning phase. Finally, a good path planner should take into consideration the robot's capabilities when building the final path. A path that may be technically feasible may not be the best path overall due to limits on the robot's reach, joint limits, and singularities.

3.2. Path Analysis

The path analysis module provides a measure of how effective the path planner was in achieving the desired results for the process. When designing this module, there are a few key issues. Most importantly, an accurate mathematical representation of the end effector and the process is required. Without this, any generated feedback will be inaccurate. While there are plenty of existing models of various processes, each model relies on input parameters unique to the particular setup. After an accurate process model has been achieved, there should be a methodology defined for quantifying the effect of the process on the surface. For instance, a pressure washing model could be quantified by the cumulative energy impingement on the surface as a function of distance and incidence angle, while a sanding operation could be quantified by the grit of the sander and the pressure applied. These quantifications can then be used to track the overall work done on the surface over time. This process must then be applied to the path. The most reliable method for modeling the entire path is to discretize the path into individual points derived from a consistent time period. After iterating through the entire path, there will be a corresponding

impingement value for all of the affected facets on the surface at each time value. These values can then be summed by facet to create the total impingement value for each facet for the entire path. These final values may then be scaled and trimmed in a way that accurately represents the process. For instance, a pressure washing operation is typically not concerned with overtreating a surface, and path evaluation can consider any value over a given threshold acceptable, whereas sandblasting would require limiting the impingement values to within a certain range. Once the impingement values have been scaled, they can be used to inform a variety of internal and external decisions moving forward based on the requirements of the process.

3.3. Path Simulation and User Interface

Assuming all of the above modules are working correctly, none of them output data in a way that is easily comprehensible by the operator, making simulation and the human-machine interface critical. The interface and simulation should be robust enough to allow the human operator to quickly and easily understand what is happening, and allow them to make the appropriate judgements and adjustments. This section will discuss some guidelines for what can be included in this module as well as some novel ways of using the data from the previous modules.

In order to provide the necessary depth of information in simulation, there are three particular pieces of data that should be displayed. First, there should always be a properly oriented and accurate representation of the part. This ensures that the operator knows exactly which pieces correspond to the physical part. However, assuming the operator is using a stationary workstation, there is a point-of-view problem, where the operator can see only part of the object. One solution is to mount a camera on the robot and allow for visual inspection as the robot moves through the path, which could be time consuming. Another solution is to reskin the simulated part with actual images of the part, similar to photogrammetry, which would allow for the most accurate initial representation. Secondly, the simulation should show the proposed toolpath, as shown in Figure 2a. This is necessary not only to ensure there are no collisions or errant movements, but also to make sure the user understands the possibilities when choosing a method for improving the path. To further improve the simulation, a model of the end effector and the corresponding process should be added. Also, depending on the feedback methods, the displayed path could be selectable and manually editable for more advanced users. Finally, the simulation should also have an efficient way to represent the impingement values that does not interfere with other aspects of the simulation. The easiest way to achieve this is by using the values corresponding to each facet and applying a color scale effect to create a heat map of the process' effectiveness as shown in Figure 2b, where red represents maximum coverage and blue represents no coverage. While this would interfere with the visualization method proposed above, the two could be toggled or have a slightly transparent version overlaid over the other.

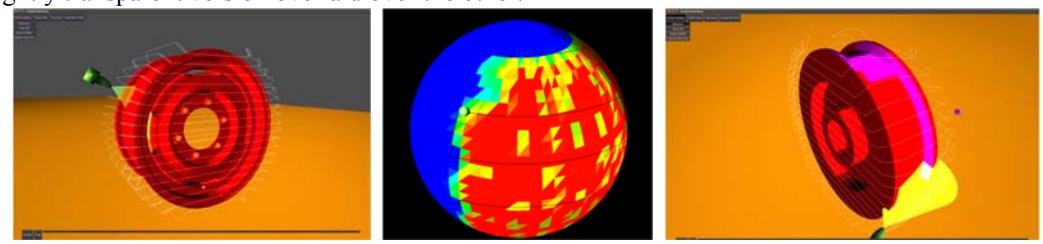

Fig. 2. (a) Path with process simulation (left); (b) Process analysis visualization (center); (c) User interface with process simulation (right)

Given those three pieces of data, there are certainly plenty of other things that can be done to make the overall simulation environment more intuitive and pleasing for the user. Having intuitive controls for rotation and navigation is a must. Not only do the controls impact the user's opinion of the software and thus their willingness to work with it, but inefficient control schemes can also take up valuable space on the screen. Enhanced visuals and options also go a long way toward improving the interface. Whether it is as simple as better shading and color choices or advanced options, such as the ability to turn features on and off and customize the interface, anything that makes the overall product feel more polished helps to integrate the user and automation.

Once the user has been given all of the necessary information from simulation, they should be given the opportunity to provide feedback and request further iteration if necessary. This step also has the same pitfalls as the initial user input where the interface needs to provide enough feedback options for effective information gathering, while minimizing the effect the user's subjective opinions have on the end product. While the initial planning

algorithm takes into account only the condition of the part as a whole, localized options are much more efficient when creating directions for second-pass modifications. This means that the capability for selecting individual parts of the surface with a high-enough resolution is essential. Figure 2c provides an example of surface selection, the pink colored facets have been selected for some rework or modification. As discussed above, most of the operator's decisions are based on their understanding of part condition, the proposed path, and process simulation results. There is no single, simple answer for displaying all of this information in an integrated fashion. For example, one possibility is to display the original image underneath a faded representation of the impingement values and then completely cover both with a fully opaque color once the facet has been selected. This allows for the operator to see and interpret both datasets before deciding whether to select them. Another opportunity for surface selection is to use statistical grouping techniques to find and classify larger sets of facets that may require additional processing. While this still relies on having a selectable surface, it helps eliminate the resolution issues created by selecting individual facets. Ideally this method would be able to take all of the facets not meeting the requirements of the process and group them into contiguous surfaces of similar impingement values, and then present the operator with the option to make modifications to each such surface. The specific modification options will be process dependent, but regardless of what changes are made, the operator should always get feedback as to how the changes impact the quality of the specified path before giving the final approval to execute the task.

3.4. Path Modifier

As with almost all of the modules above, the path modification module will be very process specific, but there are a few key capabilities needed to be effective for any process. First and foremost, it should not make any changes to the formatting of the data, meaning that any changes made should not affect the other modules' ability to understand them. This is especially important for maintaining version control since there is the possibility that the operator makes a poor choice and makes the planned path worse. One method for maintaining version control is by keeping the original path plan in the planning module, building an entirely new path from it using whatever pieces have been deemed acceptable, and then finally making the necessary changes as the new version is built. However, this can be difficult because it is not always clear what needs to be changed to make the path better: Maybe pieces of the path need to be removed and replaced, maybe the pieces need only to be removed, maybe there need to be additional passes added, or maybe there needs to be a change in the process parameters for that particular piece. This also requires that this module be able to link specific pieces of the path to the facets they effect on the surface and have been selected for modification. One way to do this would be to mark the correlation earlier in the planning process, but that still can leave multiple pieces of path affecting the same facets—thus doing little to make a decision on which one should be modified. Because of this indecision, another possibility for accounting for low impingement values is simply to tack on additional pieces of path for the robot to execute after completing the original path. While this can certainly work, constantly jumping from place to place means that there is a greater chance for collisions. To combat this issue, there needs to be some collision avoidance logic built into the path modifiers to ensure that everything will flow smoothly. Some potential modifications to the path could include changing the attack angle, slowing down the tool head's movement, replanning for only the problem areas, and adjusting the offset distance for non-contact operations. At the very least, this module should be prepared to be iterated multiple times, making the finer changes to eventually produce a good path plan. In this case, some internal iteration might be preferred so that the operator does not need to keep checking and rechecking all of the time.

4. IMPLEMENTATION

In order to realize the distributed nature of the system, the Robotic Operating System (ROS) was used to coordinate communication amongst the nodes in the network. ROS was chosen because it supports multiple languages and since there are no modifications to the data between the languages. This allows a variety of sensors, hardware, software, and various other accessories to communicate easily. It also has convenient methods for managing node executions and version control. For example, a ROS service node holds program execution on the client side until it has completed. This helps to ensure that the current task plan is not being modified by something else when it is accessed by another node.

For a system like this to work, maintaining data integrity is key, especially when all of the newly created data points need to be linked back to the originals. As discussed above, version control and communication between

languages are what ROS does well. While most languages use different mixes of lists, arrays, tuples, and various other data structures, almost all languages hold true to the basics such as text strings, integers, and floating-point numbers. ROS is no different, as it supports the basic data types but uses its own structures, called messages. For this framework system, custom ROS messages were written to accurately represent the data and transfer it between the nodes, where it was translated to and from the native data structures for use.

Currently, the prototype implementation is being built on Ubuntu 16.04 with the majority of the code written in Python 2.7. The GODOT gaming engine is used for simulation and the main user interface. Various open-source python packages are used to handle the rest of the process. As shown in Figure 2, the planned path is visualized along with a representation of the spraying process. The user can then consult the impingement data as displayed on the part and select facets needing rework. Since this is non-contact, the user can choose to modify speed or offset distance to better clean the selected facets, and this process can then be iterated until an acceptable path is found.

Although this paper focuses on full-part coverage for surface-finishing task specification, the framework can also be applied to tasks with less obvious goals, such as assembly, inspection, packaging, and pick-and-place operations. For these tasks, many of the modules discussed are still useful, but will have different goals and may be utilized in a different order. For the initial data input, the system still needs to understand the pose of all objects, but the input parameters can look different. In some cases, there may not be any input from the operator until a simulation has been rendered. The same can be said of the path planning, analysis, and modification modules. In these tasks, unlike full-coverage path planning, the goal may not be completely clear, since user input could be required before anything other than a simulation of the part and environment could be done. In these cases, an additional module could be added for identifying the task at hand and determining what needs to go where. For instance, a task identifier module could leverage visual and 3D data to identify where each piece fits with the other.

Consider an assembly robot that has been designed to help assemble a wide variety of products. Once the system has been shown all of the parts for the assembly, a task identifier module could make the initial decisions about what goes where before passing on those decisions to the operator to be confirmed or edited. Once the operator gives the go-ahead, the path planner module can take over and make a first pass at determining the appropriate trajectories for the assembly process. The trajectories would then be reviewed by the operator who could then approve them or make edits and suggestions for an improved trajectory. Measuring the goodness of these trajectories could be difficult, especially when considering how to convey the results clearly to the operator, but assembly is a bit more intuitive than finding an optimal surface covering path.

Other additional hurdles for implementing a system for non-surface finishing operations are inevitable because there is no easy and reliable way to fully complete the entire task in simulation before executing. For example, the above assembly trajectories rely on the gripper being able to replicate the grip that was achieved in simulation for the rest of the trajectory to be accurate. An alternative to this problem would be to rescan and replan each step after the robot picks up a piece with its gripper. While this would help increase accuracy, it would add significant time to the process. This also means that the operator needs to keep checking on the simulation throughout the assembly process, which makes the human more of a tele-operator than a supervisor. While that is not necessarily a bad thing and could even be ideal in some industries, it does not do much to improve the operator's efficiency.

5. CONCLUSIONS

As robotic technologies continue to advance, so should the way humans can interact with them. Collaborative robotics both physically and virtually are the next step in furthering the integration of robotics into industry and our everyday lives. While it is impossible to know what new technologies might dramatically shift how we think about robotic implementations in the future, having a framework for collaboration between robots and humans to complete complex and diverse tasks will be essential. The collaborative robotic framework for task specification proposed here is one step toward fully automated systems capable of very high-mix, low-volume manufacturing.

6. REFERENCES

[1] A. Woods and H. Pierson, "Developing an Ergonomic Model and Automation Justification for Spraying Operations," in *Proceedings of the 2018 Institute of Industrial and Systems Engineers Annual Conference*, Orlando, FL, 2018.

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System".

[3] R. Johansson, A. Robertsson, K. Nilsson, T. Brogard, P. Cederberg, M. Olsson, T. Olsson and G. Bolmsjo, "Sensor integration in task-level programming and industrial robotic task execution control," *Industrial Robot: An International Journal ,* vol. 31, pp. 284-296, 2004.

[4] Z. M. Bi and W. J. Zhang, "Concurrent Optimal Design of Modular Robotic Configuration," *Journal of Robotic Systems ,* vol. 18, pp. 77-87, 2001.

[5] M. Montemerlo and S. Thrun, FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics, Springer, 2007.

[6] G. Beni and J. Wang, "Theoretical Problems for the Realization of Distributed Robotic Systems," in *IEEE International Conference on Robotics and Autanatioo*, Sacremento, CA, 1991.

[7] E. Prassler, A. Ritter, C. Schaeffer and P. Fiorini, "A Short History of Cleaning Robots," *Autonomous Robots ,* vol. 9, pp. 211-226, 2000.

[8] A. Martinoli, K. Easton and W. Agassounon, "Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation," *The International Journal of Robotics Research ,* vol. 23, pp. 415-436, 2004.

[9] B. Siciliano and J.-J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics*, 1991.

[10] T. Fong, C. Thorpe and C. Baur, "Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation," AAAI, 1999.

[11] A. Djuric, R. J. Urbanic and J. L. Rickli, "A Framework for Collaborative Robot (CoBot) Integration in Advanced Manufacturing Systems".

[12] T. Anandan, "Robotic Industry Insights: Collaborative Robots and Safety," Robotic Inustries Association, 26 Jan 2016. [Online]. Available: https://www.robotics.org/content-detail.cfm?content_id=5908.

[13] P. Waurzyniak, "Putting Safety First in Robotic Automation," *Manufacturing Engineering,* pp. 61-66, September 2016.

[14] S. Brown, A. Woods, H. Pierson and G. Parnell, "An Operations Management Perspective on Collaborative Robotics," in *American Society for Engineering Management International Annual Conference*, Huntsville, AL, 2017.

[15] Robotiq, "Teaching Robots Welding," [Online]. Available: http://robotiq.com/solutions/robot-teaching/. [Accessed 6 May 2017].

[16] F. Remondino and S. El-Hakim, "IMAGE-BASED 3D MODELLING: A REVIEW," *The Photogrammetric Record,* vol. 21, pp. 269-291, 2006.

[17] J. D. Schutter, T. D. Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes and H. Bruyninckx, "Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty," *The International Journal of Robotics Research,* vol. 26, pp. 433-455, 2007.

[18] H. Sirkin, M. Zinser and J. Rose, "The Robotics Revolution: The Next Great Leap in Manufacturing," BCG, 2015. [Online]. Available: https://www.bcg.com/publications/2015/lean-manufacturing-innovation-robotics-revolution-next-great-leap-manufacturing.aspx.

[19] N. Mirnig, G. Stollnberger, M. Miksch, S. Stadler, M. Giuliani and M. Tscheligi, "To Err Is Robot: How Humans Assess and Act toward an Erroneous Social Robot," *Frontiers in Robotics and AI,* vol. 4, p. 21, 2017.

[20] W. Chen and D. Zhao, "Path Planning for Spray Painting Robot of Workpiece Surfaces," *Mathematical Problems in Engineering,* 2013.

[21] A. UGUR, "Path planning on a cuboid using genetic algorithms," *Information Sciences,* vol. 178, pp. 3275-3287, 2007.