Qualification of a Collaborative Human-robot Welding Cell

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 41 (2016) 352 – 357

www.elsevier.com/locate/procedia

48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015

Qualification of a collaborative human-robot welding cell

Dario Antonelli[a,]*, Sergey Astanin[a]

[a]*DIGEP — Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy*

* Corresponding author. Tel.: +39-011-090-7288; fax: +39-011-090-7299. *E-mail address:* dario.antonelli@polito.it

Abstract

This work is focused on evaluating performance of a collaborative robot welding cell developed in our previous research. Such a cell is based on an interactive cooperation between a human supervisor and a welding robot. This approach to organizing a workstation allows to employ robots even in the case of prototypes or small productions. Research on collaborative robots usually focuses on safety issues and on the programming techniques. Present work deals with a complementary problem crucial to industrial applications: the qualification of the welding cell performance in terms of accuracy, repeatability and dependability.

In this application, the human worker is responsible for handling of the parts to be assembled and for teaching the robot. The robot is in charge of actual welding. Teaching is executed by demonstration: the teacher shows the welding trajectories with a pointer observed by a motion capture system. The program is generated automatically and executed by the robot. Robots and humans share the same workspace in different times therefore human risk exposure is minimal.

Industrial applications of this or similar technology require that the process reliability and capability be assessed. We describe and analyze error accumulation along the entire data flow from the measurement tool, through the reference system transformations, to the actual representation and execution of the robot program.

Peer-review under responsibility of the scientific committee of 48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015

Keywords: Welding Robot; Accuracy and Precision; Programming by Demonstration;

1. Introduction

Interactions between human and robot is a well-established field in robotics since early 1990s. The research started mostly from the design of human-compatible robotic hardware [1] then expanding to human-friendly control modalities [2], social aspects of the interaction [3], natural user interfaces [4], and representation of the complex tasks [5,6]. The Goodrich's survey on Human-Robot Interaction (HRI) [7] summarizes the progress made up to mid-2000s and presents a thorough description of different interaction modes, application domains, and the principal open problems in the field. The industrial robots were not considered the principal application field.

In the 2000s the search for a better human-robot interface continued, tapping into direct brain-computer [8], augmented reality [9,10], and advanced verbal [11] interfaces. The progress in the field of human-compatible robotic hardware is also undeniable [12,13]. An effort to standardize the robotic devices designed for HRI was undertaken [14]. Machine learning methods remained the staples of robot knowledge representation [15]. Notwithstanding numerous applications to mobile, bio-inspired, medical and service robots, HRI research has seen only sporadic industrial applications [16,17].

Notably, different application fields tend to use different performance metrics [7]. Paying attention to industrial domain, [18] identify the main issues for HRI in safety and dependability. Dependability is an integrated concept, combining availability, reliability, safety, integrity and maintainability. However, accuracy and precision (in ISO 5725 sense of terms) are mostly ignored in HRI research [19].

Peer-review under responsibility of the scientific committee of 48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015
doi:10.1016/j.procir.2015.12.036

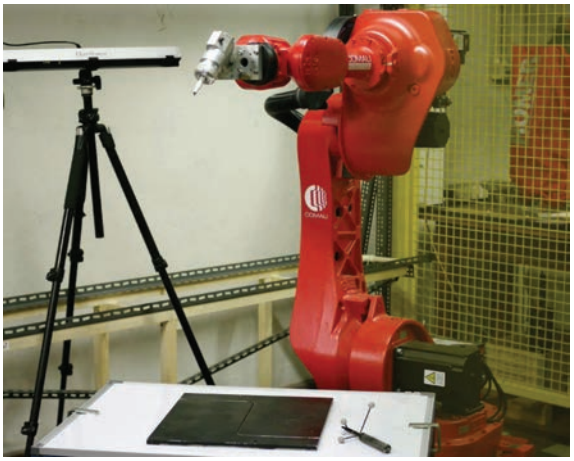

Fig. 1. Components of the programming by demonstration setup: a) a motion capture system Optitrack V120:Trio; b) a Comau Smart NS-16 industrial robot arm, equipped with a custom felt-pen holder as an end effector; c) a hand-held pointer tool equipped with IR reflective markers, d) benchmark trajectory template for reproducible test execution.

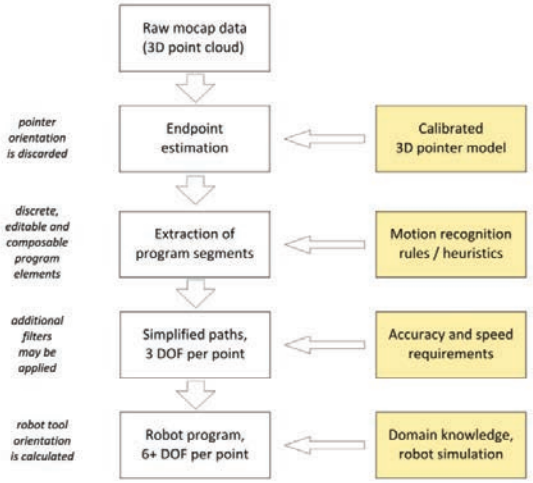

Fig. 2. The system matches the calibrated model of the pointer to every data frame, and calculates the position of the end-point. Ad hoc rules allow for separating program paths from other pointer motions.

In the framework of our previous projects [20,21] a collaborative robot welding cell was developed. Our experiment was focused on collaboration between a standard industrial robot arm and a human worker.

This implies some degree of conservatism in our choice of technologies, and different metrics to assess the work done. We chose a motion capture system as a principal input device. This technology is safe, reliable and very mature. We see it as a good balance between precision and user friendliness. Being an industrial application, the proposed system is required to repeat the tasks reliably and accurately. Hence we focus on precision and accuracy of the system as the most important metrics.

In this paper we have tried to qualify the performance of the collaborative robot cell and analyze the precision of the system throughout the entire data flow, from the measurement tool, through the reference system transformations, to the actual representation and execution of the robot program.

2. Description of the collaborative cell

2.1. Experiment design

Our Programming by Demonstration (PbD) system allows for path execution on a standard industrial robot. We used a motion capture system and a custom pointer tool both to record precise spatial location, and to convey symbolic instructions through gestures. The programs are executed on a Comau Smart NS-16 robot arm, but can be easily ported to other robot controllers. A felt-pen holder is mounted on the robot arm. It allows to draw trajectories on paper that are used as a tangible trace of the program execution.

In order to indicate tool paths to the robot, we produced a special pointer tool equipped with three spherical IR reflective markers and a spherical end-point. The markers are arranged in a scalene triangle, thus the three markers are sufficient to calculate tool's location and orientation. The spherical endpoint has two advantages: it does not stuck when sliding over the work surface in any direction, and it is easy to produce and measure precisely. The entire tool is measured on a coordinate measurement machine, so all dimensions are known. Three dimensional coordinates of the markers are measured and its end-point location is calculated.

The experiment area (Fig. 1) is equipped with a motion capture system (NaturalPoint Optitrack V120:Trio). It is managed by its own software (NaturalPoint Motive) which can multicast point cloud data over network (or loopback interface) to other programs. Optitrack Streaming Engine uses a binary UDP-based protocol, NatNet. As a side effect of the project, we have developed a Python library to read NatNet binary packets, and the library was released as an open source Python package [22].

2.2. System implementation

Conceptually, the system transforms the raw motion capture data into a set of disjoint one-dimensional manifolds, by splitting it into distinct “paths” to execute. Missing program parameters (tool state and orientation) are calculated based on domain knowledge, path geometry, and the specifics of the robot motion planning implementation (Fig. 2).

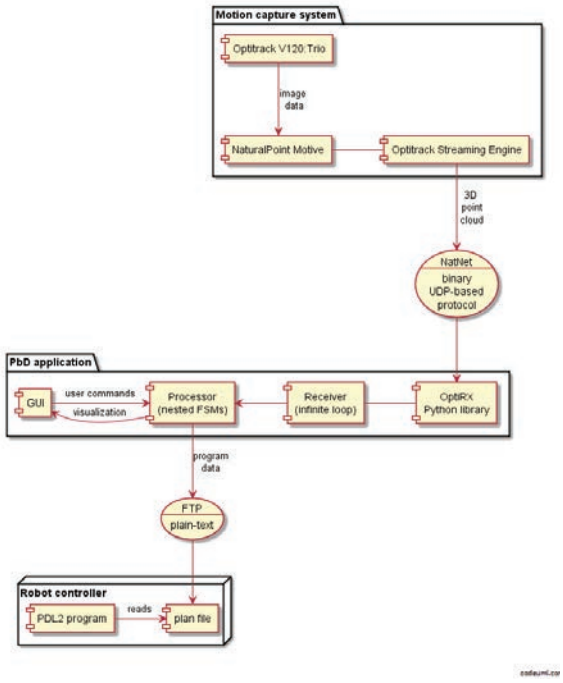

Fig. 3. UML Component diagram of the programming by demonstration system. The motion capture software and the PbD application may run on the same or different machines, and communicate via network protocol. PbD application and the robot controller communicate via an Ethernet connection.

We underline, that though technically the orientation of the pointer tool is known to our system, the actual orientation of the robot tool does not necessarily coincide with the orientation of the pointer. This implies that the amount of information, that the operator is supposed to *demonstrate*, is reduced. The operator has to demonstrate the trajectory, but does not have to care about the welding tool's orientation. This feature may be particularly important when creating programs longer or more complex than a human can conveniently demonstrate in a single gesture, such as when working with very large work pieces.

The PbD application is implemented in Python and is designed as three loosely coupled components (Fig. 3):

- *Receiver*, which reads raw binary data from the motion capture system and transforms them to Python data structures;
- *Processor*, which does the bulk of the work: estimates pointer's end point location and orientation, recognizes gestures, builds a valid program plan; its design will be explained in detail further in the paper.
- *User interface (GUI)*, which represents the state of the system to the user and receives out-of-band user commands.

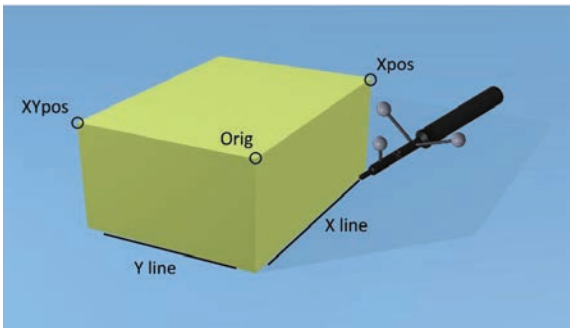

Fig. 4. A reference cuboid is used to calculate a rigid transform between the motion capture system reference frame and the robot reference frame. The robot reference frame is defined by the three vertices *Orig*, *Xpos*, *XYpos*. The user indicates two edges, *X line* and *Y line*, of the dihedral angles between the cuboid and the table using the usual pointer tool equipped with IR markers.

The components run in separate processes and communicate only via message queues. The execution plan may be transferred to the robot either via File Transfer Protocol (FTP) or via the direct socket connection.

The robot program is written in PDL2, the Comau programming language. It interprets the program plan and translates it into vendor-specific motion commands. The program plan acts as a simple, vendor-agnostic format to specify robot motion and may be considered an ad hoc domain specific language, and like most such languages is concerned with coordinates' presentation and motion control [23]. Proper design of the DSL is our future priority.

3. Reference frame definition

3.1. Common reference frame

Robot programs often run in a Cartesian reference frame defined by the user. For the hardware we used in our experiments, this reference frame is defined by touching three reference points (*Orig*, *Xpos*, *XYpos*) with a tool of known dimensions mounted on the robot flange. The reference points are often chosen to be the vertices of a rectangular cuboid mounted in the work area. The user reference frame is calculated with respect to a robot base reference frame (fixed unless the robot is misplaced).

The reference frame of the motion capture system is linked to one of the cameras or a specially crafted calibration tool with three markers in the shape of an 'L' [24]. Either way, the rigid transformation between the reference frame of the robot, and the reference frame of the motion capture system remains unknown. Neither location of the robot base origin, nor the camera's reference origin are accessible to be measured.

We have developed a method to calculate a rigid transformation between these two reference frames (camera reference frame and user-defined robot reference frame). The method is supposed to be fast and easy to execute by the end-user, and reasonably accurate in the same time.

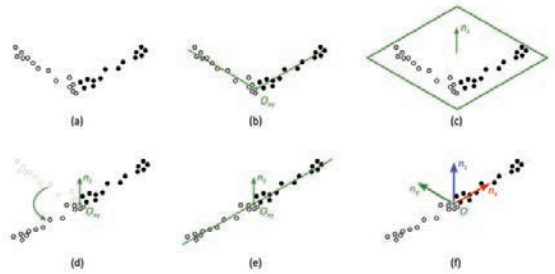

Fig. 5. The method to calculate a reference frame: (a) acquire some points along two perpendicular edges; (b) fit two lines to each of the edges independently, use their intersection as a new origin O_{xy}; (c) fit a plane to all points; (d) rotate points of the second line around the plane's normal $\boldsymbol{n}_z$; (e) fit a single line to all points simultaneously; (f) choose a positive direction $\boldsymbol{n}_x$ of the *X* axis along the line, build an orthonormal basis.

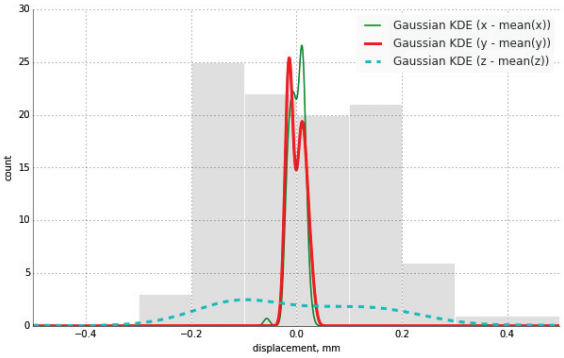

Fig. 6. Kernel density estimates of the calculated end-point location with respect to its average location around the center of the frame. The histogram and the dashed line KDE show the distribution of the depth variable measurements.

The idea is to make the reference rectangular cuboid of the robot somehow "visible" to the motion capture system. The motion capture system may observe only spherical IR markers. One solution would be to mount some markers on the reference cuboid and measure their precise locations. This solution is easy to implement, but it is not very practical because it would require to leave a fragile calibration object in the work area of the robot. An alternative is to use the same pointer tool with IR markers, which we use to indicate paths, and touch at least three reference points on the cuboid (Fig. 4).

It's not possible to uniquely position a spherical end point of the handheld tool on a vertex, but it's easy to follow a straight line in the dihedral angle between the cuboid and the surface it is mounted on. Two such lines coincide with the directions of the robot's X and Y axis, and the translation between their intersection and robot's origin is known if the dimensions of the reference cuboid are known.

The method is practical, because: 1) it doesn't require to leave fragile calibration objects in the robot work area, 2) neither robot nor motion capture reference frame acquisition have to be modified, so the method is brand-agnostic 3) the same tool which is used to indicate program paths is used to calibrate the system, 4) acquisition of only two straight trajectories with motion capture software is fast and easy for the user.

3.2. Fitting algorithm

To calculate the reference frame based on points sampled along *X line* and *Y line*, we do the following:

- Discard initial and final points which are too close to each other. The pointer tool is likely to be still at the beginning and the end of the line, while the motion capture system works at a fixed frame rate. As the still time can be arbitrary long, we may capture arbitrary many points at the extremities of the line. We use only the points when the pointer is in motion.
- Optionally resample points along the curvilinear coordinate. This filter serves the same purpose to give less weight to the initial and the final parts of the trajectory when the pointer is moving slower than in the middle part.
- Fit two lines to two edges independently; use their intersection (or the point of the closest approach) as a new origin O_{xy} (Fig. 5, b)
- Fit a plane to all points (Fig. 5, c)
- Rotate the points of the second line around O_{xy} and $\boldsymbol{n}_z$ for $\pi/2$ (Fig. 5, d)
- Fit a single line to all points simultaneously (Fig. 5, e)
- Choose the positive direction of the *X* axis along the new line so that the X edge has positive coordinates; build an orthonormal basis (Fig. 5, f)
- Take into account the radius of the pointer's end-point and the height of the reference cuboid; adjust the origin to match the origin of the robot's user frame.

This method strives to fit an orthonormal basis to all points simultaneously. Just fitting two lines, like in Fig. 5 (b), and building an orthonormal basis upon them have demonstrated to introduce asymmetry in error distribution. This method does not address a situation when two reference edges are skewed or are not perpendicular. This method does not address a situation when the length of two edges is very different (the edge with more points will have more weight). Almost any arbitrary cuboid of suitable size can be used as a reference object.

4. Qualification of the system

4.1. Repeatability of the end-point calculation

Repeatability of the end-point calculation for the pointer in the center of the frame was measured experimentally. The motion capture system was placed at ≈1400 mm away, inclined 36° with respect to the table surface. The pointer was repeatedly put on an isostatic support, and the end-point location was measured. Its standard deviation components are $\sigma_x = 0.014$ mm, $\sigma_y = 0.015$ mm, $\sigma_z = 0.15$ mm (Fig. 6).

The result is fairly good and indicates, that there are no serious problems with the quality of the pointer nor with the repeatability of the motion capture system. In theory, this is the lower bound for the accuracy of the setup. The repeatability in the *Z* direction (depth variable) is an order of magnitude worse. This is to be expected in similar stereo vision setups and fortunately can be improved by using more cameras or increasing the stereo base.

4.2. Stable fit of the reference frame

To estimate stability of the reference frame fitting method with respect to measurement errors, we run synthetic tests. The testing assumption were:

- i.i.d. measurement errors, $\sigma = 1$ mm
- non-uniform sampling along the line; points are sampled at x values of the points uniformly distributed along the curve $y = 1 - (2x - 1)^4$ for $0 \le x \le 1$, 100 points per each edge
- equal reference edges, 400 mm long

After 1000 computational experiments (generating new points and calculating new reference frame every time), we have seen that

- the median displacement of the calculated origin is 0.1 mm (0.18 mm for 95.45% percentile)
- the median angular error of the calculated X axis is 0.02° (0.034° for 95.45% percentile)

Overall, there are no major fitting issues if the measurement errors are normally independently and identically distributed.

4.3. Real-life repeatability

Real-life repeatability of the reference frame acquisition was measured experimentally. The motion capture system was placed at 1370 mm away from the origin, inclined 36° with respect to the table surface, *N* = 50 measurements of the reference frame were executed with the same relative position of the reference cube and the motion capture system.

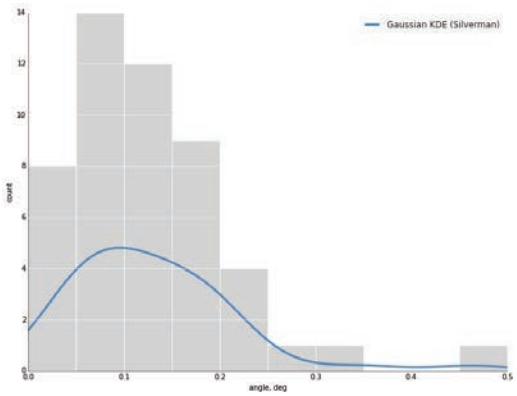

Fig. 7. Histograms and kernel density estimations of the angular misalignment of the calculated reference frame with respect to average results.

The actual standard deviations of the origin location are $\sigma_x = 0.8$ mm, $\sigma_y = 0.3$ mm, $\sigma_z = 0.9$ mm in the reference frame of the motion capture system, where *z* is the depth variable, *y* is the vertical axis.

We may note that the repeatability of the *x* coordinate is much worse than that of the *y* coordinate, in spite of both being the image plane coordinates. We attribute it to the fact that the camera's axis was only slightly inclined with respect to the table surface, thus *y* coordinates of points of both line segments vary much less than *x* and *z* coordinates.

Angular repeatability of the acquired reference frame is a separate issue to consider. We calculated the average direction of the *X* axis, and calculated angular misalignment for every sample. It appears that on average the angle between the calculated and the average direction of the *X* axis is 0.13° with the standard deviation of 0.08°. The histogram and the kernel density estimation are displayed in Fig. 7. This order of magnitude of the angular repeatability implies that transverse errors at 1 m away from the origin may be as high as 2.3 mm.

4.4. Error accumulation

There are several steps in the process, and multiple factors influence the repeatability and the accuracy of the final program execution. If we refer to Fig. 2 and our measurements, we may identify and describe these factors (see Table 1).

Overall, most system components have an acceptable repeatability level. The most likely reason for loss of accuracy are extrapolation of angular errors far away from the reference cube (origin). Depending on user requirements this issue may have to be addressed.

4.5. Form error

To estimate variability of the process with the same input trajectory, we built some benchmark trajectory guides. These items are thick aluminium plates in which we carved channels with a triangular cross-section (Fig. 8). The width and the depth of the channels was chosen to accommodate the end-point sphere of the pointer tool. True dimensions of the

Table 1. Repeatability at various stages of the method.

Step	Repeat-ability, mm
CMM measurement of the pointer tool	0.01
Optitrack V120:Trio @ 1 m away	unknown
Pointer model fitting (measured)	0.2
Line and reference frame fitting (simulated)	0.1
Extrapolation of the angular error @ 1 m away	2.3
Program path approximation	arbitrary
Robot user frame definition	unknown
Robot repeatability ISO 9283 (specification)	0.05

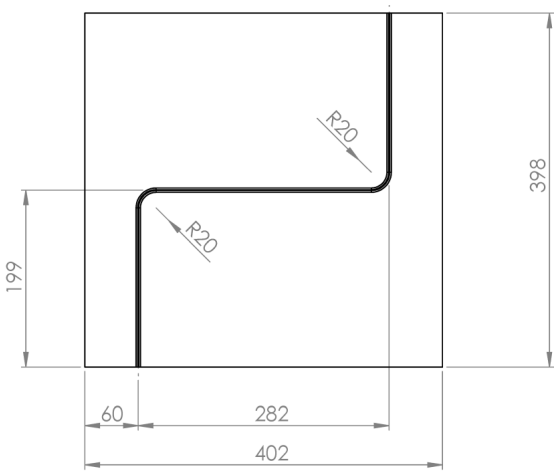

Fig. 8. One of the benchmark trajectory guides. A channel with a V-shaped cross-section allows to repeatedly indicate the same and *known* path.

channel were measured on a coordinate measuring machine. The channels served as guides to repeatedly indicate the same and *known* trajectories with the pointer tool.

The same trajectory was acquired multiple times, a robot program was generated and executed with a felt-pen, and the trace of the pen was scanned and compared to the true channel geometry in several points along the curve.

In the experiment with an S-curve presented in Fig. 8, the median displacement of the traces was found to be 0.4 mm, with the interquartile range of 0.6 mm, though occasionally we did see maximum errors as big as 2.6 mm. These results appear to be in line with our estimations in Sec. 4.3.

5. Conclusions

This work presented the further development of our programming by demonstration system for standard industrial robots. Industrial applications require to assess if human demonstrations are interpreted and executed by the robot reliably and precisely. These aspects are rarely addressed in general Human-Robot Interaction research.

We qualified the entire system and its individual components, identified the weak spots, and benchmarked the generated robot programs against the known human intent. There is obvious room for improvement, but even in its current state the system is already compatible with a number of industrial usage scenarios.

We believe that this kind of approach to HRI performance evaluation is crucial in the industrial domain.

References

[1] Kazerooni H. Human-robot interaction via the transfer of power and information signals. Syst Man Cybern IEEE Trans On 1990;20:450–63.
[2] Luh JYS, Hu S. Comparison of various models of robot and human in human-robot interaction. Syst Man Cybern IEEE Int Conf 1998; 2:1139–44.
[3] Breazeal C, others. A motivational system for regulating human-robot interaction. AAAI/IAAI; 1998. p. 54–61.
[4] Kang SB, Ikeuchi K. Toward automatic robot instruction from perception-temporal segmentation of tasks from human hand motion. Robot Autom IEEE Trans On 1995;11:670–81.
[5] Yang J, Xu Y, Chen CS. Human action learning via hidden Markov model. Syst Man Cybern Part Syst Hum IEEE Trans On 1997;27:34–44.
[6] Klingspor V, Demiris J, Kaiser M. Human-robot communication and machine learning. Appl Artif Intell 1997;11:719–46.
[7] Goodrich MA, Schultz AC. Human-robot interaction: a survey. Found Trends Hum-Comput Interact 2007;1:203–75.
[8] Bell CJ, Shenoy P, Chalodhorn R, Rao RP. Control of a humanoid robot by a noninvasive brain–computer interface in humans. J Neural Eng 2008;5:214.
[9] Reinhart G, Vogl W, Kresse I. A projection-based user interface for industrial robots. Virtual Environ. Hum.-Comput. Interfaces Meas. Syst. 2007 VECIMS 2007 IEEE Symp. On, IEEE; 2007, p. 67–71.
[10] Nee AYC, Ong SK, Chryssolouris G, Mourtzis D. Augmented reality applications in design and manufacturing. CIRP Ann-Manuf Technol 2012;61:657–79.
[11] Mavridis N. A review of verbal and non-verbal human–robot interactive communication. Robot Auton Syst 2015;63:22–35.
[12] Kim S, Laschi C, Trimmer B. Soft robotics: a bioinspired evolution in robotics. Trends Biotechnol 2013;31:287–94.
[13] Knaepen K, Beyl P, Duerinck S, Hagman F, Lefeber D, Meeusen R. Human-robot interaction: kinematics and muscle activity inside a powered compliant knee exoskeleton 2014.
[14] Harper C, Virk G. Towards the development of international safety standards for human robot interaction. Int J Soc Robot 2010;2:229–34.
[15] Nguyen-Tuong D, Peters J. Model learning for robot control: a survey. Cogn Process 2011;12:319–40.
[16] Lenz C, Nair S, Rickert M, Knoll A, Rosel W, Gast J, et al. Joint-action for humans and industrial robots for assembly tasks. Robot Hum. Interact. Commun. 2008 RO-MAN 2008 17th IEEE Int. Symp. On, IEEE; 2008, p. 130–5.
[17] Haddadin S, Suppa M, Fuchs S, Bodenmüller T, Albu-Schäffer A, Hirzinger G. Towards the robotic co-worker. Robot. Res., Springer; 2011, p. 261–82.
[18] De Santis A, Siciliano B, De Luca A, Bicchi A. An atlas of physical human–robot interaction. Mech Mach Theory 2008;43:253–70.
[19] Steinfeld A, Fong T, Kaber D, Lewis M, Scholtz J, Schultz A, et al. Common metrics for human-robot interaction. Proc. 1st ACM SIGCHISIGART Conf. Hum.-Robot Interact., ACM; 2006. p. 33–40.
[20] Antonelli D, Astanin S, Galetto M, Mastrogiacomo L. Training by demonstration for welding robots by optical trajectory tracking. Procedia CIRP 2013;12:145–50.
[21] Antonelli D, Astanin S, Caporaletti G, Donati F. FREE: Flexible and Safe Interactive Human-Robot Environment for Small Batch Exacting Applications. Gearing Accel. Cross-Fertil. Acad. Ind. Robot. Res. Eur., Springer; 2014. p. 47–62.
[22] Astanin S. OptiRX 0.3 [Computer program]. 2015. https://pypi.python.org/pypi/optirx
[23] Nordmann A, Hochgeschwender N, Wrede S. A survey on domain-specific languages in robotics. Simul. Model. Program. Auton. Robots, Springer; 2014. p. 195–206.
[24] NaturalPoint Inc. Calibration. NaturalPoint Product Documentation. 2014. http://wiki.optitrack.com/index.php?title=Calibration (accessed March 30, 2015).