An anticipative kinematic limitation avoidance algorithm for collaborative robots: Three-dimensional case

Article · September 2017

DOI: 10.1109/IROS.2017.8206147

CITATION
1

READS
104

3 authors:

Philippe Lebel
Laval University
1 PUBLICATION **1** CITATION
SEE PROFILE

Clément Gosselin
Laval University
573 PUBLICATIONS **17,546** CITATIONS
SEE PROFILE

Alexandre Campeau-Lecours
Laval University
44 PUBLICATIONS **180** CITATIONS
SEE PROFILE

An Anticipative Kinematic Limitation Avoidance Algorithm For Collaborative Robots: Three-Dimensional Case

Philippe LeBel, Clément Gosselin and Alexandre Campeau-Lecours[1]

This paper is a Post-Print version (ie final draft post-refereeing). For access to Publishers version, please access http://ieeexplore.ieee.org/document/8206147/

IEEE/RSJ 2017.

***Abstract*— This paper presents an anticipative robot kinematic limitation avoidance algorithm for collaborative robots. The main objective is to improve the performance and the intuitivity of physical human-robot interaction. Currently, in such interactions, the human user must focus on his task as well as on the robot configuration. Indeed, the user must pay a close attention to the robot in order to avoid limitations such as joint position limitations, singularities and collisions with the environment. The proposed anticipative algorithm aims at relieving the human user from having to deal with such limitations by automatically avoiding them while considering the user's intentions. The framework developed to manage several limitations occurring simultaneously in three-dimensional space is first presented. The algorithm is then presented and detailed for each individual limitation of a spatial RRR serial robot. Finally, experiments are performed in order to assess the performance of the algorithm.**

I. INTRODUCTION

Collaborative robots working alongside humans are now used in many fields such as industrial applications, service robotics and medical applications [1]. By harnessing both the strengths of humans and robots simultaneously, human-robot interaction has the potential to increase human performance and to provide effective assistance in many applications. However, in order to achieve this, the interaction between the human user and the robotic device must be safe and intuitive [1], [2], [3]. To this end, the robot motion can be designed to behave similarly to that of human beings [4], [5]. Additionally to being intuitive and predictable, the robot motion should also be legible, that is to enable the collaborator to quickly and confidently infer the robot's goal. This aspect was introduced and detailed in [3]. While collaborative robots allow a close cooperation with humans, commercial devices still automatically stop when a limitation (joint position limitation, singularities and collision with the environment) is reached, which is clearly not intuitive. In order to avoid the limitations, the user would have to pay a close attention to said limitations, which can be very difficult and cognitively demanding (especially for internal kinematic limitations such as position limitations and singularities).

The management of limitations such as self-collisions and collisions with the environment has extensively be explored in the literature [6], [7], [8], [9]. However, many of the proposed trajectory planning algorithms are designed for industrial robots where the trajectory is planned off-line.

[1]The authors are with the Department of Mechanical Engineering, Université Laval, Québec, Canada. This work is supported by CRSNG.
philippe.lebel.4@ulaval.ca
gosselin@gmc.ulaval.ca
alexandre.campeau-lecours@gmc.ulaval.ca

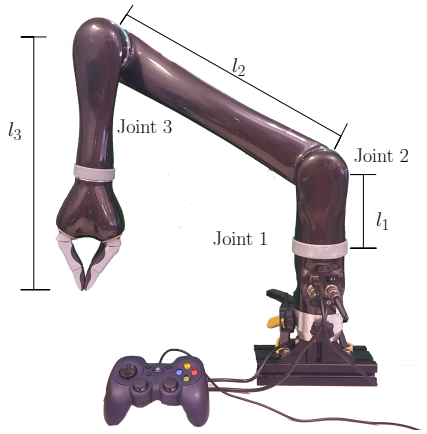

Fig. 1. 3 DOF modifed Kinova robot arm used for the tests

Collaborative robots bring new challenges that may not be effectively handled by the existing techniques. For instance, collaborative robots are usually used in unstructured and dynamic environments and must manage many real-time constraints such as collocation with humans [10]. Additionally, while existing trajectory planning algorithms are effective at finding an optimal path between two points, in a human-robot collaboration application the final destination is often unknown by the planner. Indeed, the user can directly control the robot's direction and velocity (through a joystick or by direct manipulation of the robot) and can thus bring the robot in prohibited configurations. The management of the robot limitations must then be as transparent as possible in order to prevent the user from having to constantly monitor these limitations, which would be detrimental to the principal task.

In order to solve this problem, many solutions have been proposed in the literature such as potential fields [11] ,[12] ,[13], virtual spring-damper systems [10], [14] and virtual-fixtures [15]. The main advantage of these algorithms is that they are able to repel the robot from the limitations regardless of whether the final destination is known or unknown. These reactive algorithms have been implemented in many applications. For instance, a skeleton algorithm able to manage self-collisions of two 7-DOF serial arms in a torso configuration was introduced in [14]. In the proposed scheme, the distance between the robot links is first evaluated and, if necessary, a virtual repulsive force is applied on the joints, by using a virtual spring-damper system, in order to avoid self-collisions.

Even if these methods are successful at avoiding collisions, they present inherent drawbacks. Their reactive nature

produces a change of path only after the trespassing of a limitation. To avoid a collision with an object, the buffer zone around the object must either be much larger than the actual object or present a high stiffness that guaranties that the robot cannot penetrate too far in the limitation. These two methods of managing the reactive nature of the algorithms require an extensive experimental tuning. The former method needs to validate if the space between the limitation boundary and the object is large enough to avoid a collision. The latter needs to verify that the large stiffness of the repulsive force does not produce vibrations or instability [16]. Even after tuning these parameters, the reaction of the robot is still hard to predict for a human user since the virtual force applied on the links varies with the velocity and acceleration of the robot when it collides with an object. These algorithms also have trouble dealing with multiple limitations. Some limitations may contradict each other, pushing the robot in the direction of a less restricting limitation.

The algorithm presented in this paper addresses these considerations by anticipating limitations instead of reacting to them. This anticipative limitation avoidance algorithm aims at making the robot control efficient, intuitive and safe by transforming the user's input in an easily predictable way. The algorithm analyzes the limitations near the robot and computes the components of the user's input that may make the robot collide with a limitation. These components are then removed thus allowing the robot to slide alongside multiple simultaneous limitations with multiple points of contact.

This paper is structured as follows. A brief description of the algorithm's structure and of the virtual limitation detection are presented. Then, the sliding algorithm for a single point defined on the effector is presented. The algorithm is then expanded the multiple and simultaneous contacts case. Experimental results are then reported in order to assess the performance of the algorithm. Finally, the results are discussed and a conclusion is drawn.

II. General structure of the algorithm

The framework required to implement the sliding algorithm includes the following components:

- Limitations defined in the robot's workspace
- Proximity detection algorithm
- Angular position of the robot's joint
- A velocity verification algorithm

The limitations can be defined as external or internal. External limitations include obstacles in the workspace as well as protection zones defined by users. Internal limitations consist of the angular limits of the joints, singularities and self-collision of robot links.

Fig.2. provides an overview of the algorithm. The velocity requested by the user, $\vec{v}_u$, is fed to the Jacobian matrix in order to determine the requested joint velocity $\dot{\boldsymbol{\theta}}_r$. It is also used by the limitation detection algorithm in order to determine the constraints to be used by the sliding algorithm. The algorithm is then applied to modify the requested user velocity $\vec{v}_u$ and produce the effective Cartesian velocity $\vec{v}_m$,

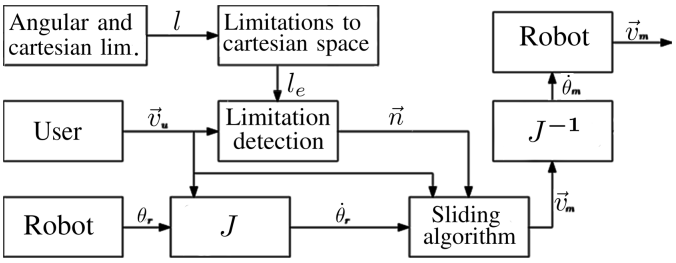

Fig. 2. General structure of the algorithm.

which is converted into the joint velocity array that is finally sent to the robot.

III. Proximity detection algorithm

This section presents the algorithm developed to compute the closest distance between a point defined on the robot's link and a limitation. These limitations can be expressed using techniques such as clouds of points or geometric shapes that include a group of forbidden configurations.

Different approaches can be used to compute the shortest distance with a limitation. Computing the distance between a large number of points defined on the robot links and the limitations is an example of a simple but expensive technique in terms of computing time. The precision of the algorithm directly depends on the number of points used to generate the 3D object and the computation time increases following an $O(n*m)$ complexity (and even $O(n*(m+n))$ if we consider self-collision) [17], where n is the number of points defining the robot's links and m is the number of points defining objects in the robot's workspace. Other techniques such as the GJK algorithm [18] and the octree [19] are more refined techniques used in video games that use geometric relations to reduce the required number of points to process.

An approach similar to the octree technique is used here. Complex limitations are broken down into simple geometries. The shortest distance is computed by defining limitations as prisms and by projecting the points on the edges of these prism. The main difference with the original algorithm is the complex shapes are subdivided only once instead of many times in the regular octree implementation. The need to implement a full octree strategy might come as the number of limitation increases and the geometries become more complex.

After expressing obstacles into simple geometries, the algorithm computes the Euclidean distance between the object and the robot.

IV. Limitation sliding algorithm for a single point

This section presents the proposed algorithm that allows a single point defined along one of the robot's links, such as the end-effector's reference point, to slide on limitations. Since the robot evolves in a three-dimensional space, only three non-redundant limitations are required to fully constrain and immobilize the robot. A higher number of limitations is redundant and can be reduced to a set of

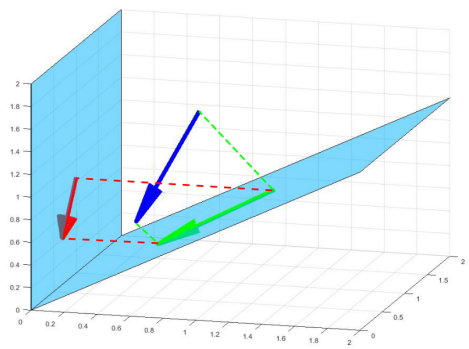

Fig. 3. Example of independent limitation processing. The velocity input is shown in blue and is going toward both limitations. The first projection, in green, is still going toward the second limitation. The second projection, in red, is then going toward the first limitation.

three nonlinearly dependent limitations. Accordingly, it is only possible to slide on a maximum of two limitations simultaneously. When computing the end-effector's modified velocity, all the limitations must be expressed in the same set of coordinates in order to process them simultaneously. If the limitations are not processed simultaneously, for instance if some limitations are processed in the articular space and others in the Cartesian space, contradictory sliding results may occur [4]. Figure 3 shows an example in which the limitations are processed sequentially (independently).

When processing the first limitation (right plane), the resulting velocity (shown in green) is obtained and then processed by limitation two. The resulting vector (shown in red) is then going toward the first limitation.

The method proposed here to process the limitations simultaneously can be described as follows:

1) All the limitations are transformed into Cartesian limitations of the reference point of the end-effector (defined by a plane with a normal vector in the Cartesian space).
2) The active limitations are identified. For a limitation to be considered active, the scalar product of the end-effector velocity vector requested by the user and the limitation's normal vector must be negative.
3) A sliding direction of the requested Cartesian velocity is determined for every combination of two limitations. The only possible direction that allows the sliding on the two limitations simultaneously is the direction defined by the line corresponding to the intersection of the two limitation planes. This direction is obtained using the cross product of the two normal vectors of the limitation planes.
4) The user's input velocity is then projected on each of the sliding direction vectors defined above for each combination of the limitations. The projections are then tested to determine if they satisfy all the limitation. The projections that fail the test are discarded.
5) Then, the same user's velocity is projected on each of the limitation planes. These vectors are also tested and those that fail to satisfy all limitations are discarded in the same manner as the previous step.

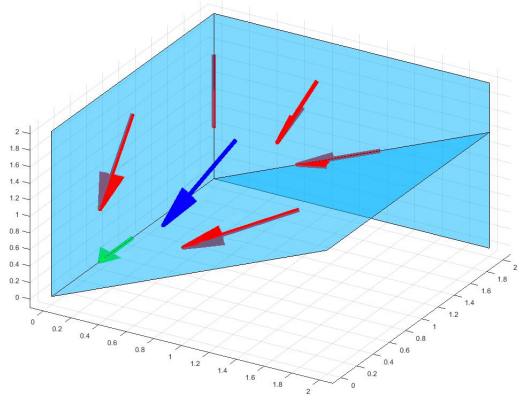

Fig. 4. Example of application of the proposed algorithm, considering limitations simultaneously. The user's input is displayed in blue and is going toward the first two limitation planes. The only possible direction that satisfies the two limitations and is a component of the user's input is the green vector going along the direction of the line defined by the intersection of the two active limitation planes.

6) The final velocity vector is then created by adding all the remaining vectors.

After this process, only viable directions are remaining and the robot can slide along these directions. Figure 4 illustrates this process.

The user's input is displayed as the blue arrow. The green arrow represents the valid directions in which a point can slide on limitations, the red ones represent the components of the user's input that will make the point collide with the planes. In this case, if the limitations were considered independently, the resulting vector would dependent on which limitation is considered first, sometimes not sliding at all, sometimes trespassing limitations. While a sequential processing of the limitations may work in some simple cases, it renders the algorithm's behaviour difficult to predict and unreliable. It is therefore very important to consider all the limitations simultaneously, which is a contribution of the proposed algorithm.

V. Limitation Sliding Algorithm for Multiple Points

In the preceding section, a sliding algorithm was proposed for a single point on the robot. It is also important to consider other points on the robot, such as the elbow as well as other types of limitations, such as angular limitations of the joints and singularities. In order to handle all these limitations simultaneously, the approach proposed here consists in transforming each of the limitation constraints into equivalent constraints at the end-effector. Indeed, it is very important to consider all constraints simultaneously as explained in the preceding section.

A. Physical limitations and protection zones for multiple, simultaneous, potential collision points

Consider first the reference point on the end-effector of the robot and its velocity vector, $\vec{v}$. The mapping between the joint velocity array, $\dot{\theta}$ and vector $\vec{v}$ is given by the Jacobian matrix of the robot, J, defined at the end-effector, as follows:

$$\vec{v} = \boldsymbol{J}\dot{\boldsymbol{\theta}}. \tag{1}$$

Similarly, the relation between the movement of a given point, P, on the robot and the articular velocity array, $\dot{\boldsymbol{\theta}}$, is defined as follows:

$$\vec{\boldsymbol{v}}_{\boldsymbol{p}} = \boldsymbol{J}_{\boldsymbol{p}}\dot{\boldsymbol{\theta}} \tag{2}$$

where $\vec{\boldsymbol{v}}_{\boldsymbol{p}}$ is the Cartesian velocity of a given point, P, on the robot, $\boldsymbol{J}_{\boldsymbol{p}}$ is the corresponding Jacobian matrix (defined at point P) and $\dot{\boldsymbol{\theta}}$ is the joint velocity array of the robot.

When a robot's link approaches a limitation, the collision detection algorithm computes the proscribed direction to avoid the violation of this limit. The only step required before modifying the user's input is to transform this link's forbidden direction into the corresponding end-effector's forbidden direction. This allows the algorithm to process all limitations simultaneously as if they were all limitations of the end-effector motion. Assuming that the robot is not in a singular configuration (matrix $\mathbf{J}$ is invertible), eq.(1) can be inverted and substituted into eq.(1), which yields

$$\vec{\boldsymbol{v}}_{\boldsymbol{p}} = \boldsymbol{J}_{\boldsymbol{p}}\boldsymbol{J}^{-1}\vec{\boldsymbol{v}} \tag{3}$$

where $\vec{\boldsymbol{v}}_{\boldsymbol{p}}$ is the velocity of the point at which the collision potentially occurs and $\boldsymbol{J}_{\boldsymbol{p}}$ is the Jacobian matrix defined at this given point, as described above. This relation allows the computation of the velocity vector of the potential contact point, P, that corresponds to a given velocity, $\vec{\mathbf{v}}$ of the end-effector's velocity.

For example, if point P is defined at the elbow of the manipulator of Fig. 1 [20], eq.3 allows the determination of the velocity of this point for a given end-effector velocity, $\vec{\boldsymbol{v}}$.

It is however required to perform the inverse transformation in order to find relation between the limitation at the elbow and the corresponding limitation at the end-effector. The result will allow to emulate the limitation by giving a normal vector in the end-effector coordinate system, as if the limitation were encountered at the effector thus allowing the algorithm to process all the limitations simultaneously.

However, to achieve this result, it is important to take some considerations into account:

1) Given a potential collision point P defined on link i of a d-dof robot, matrix $\boldsymbol{J}_{\boldsymbol{p}}$ are composed of zero entries since the last $(n-i)$ joint velocities have no impact on the velocity of point P.
2) Given the above observation, matrix J_p is not of full rank and eq.3 cannot be inverted. In the following, the equation that must be used to transfer the limitation at point P into a limitation at the end-effector is presented and the result is an important contribution of this paper.

At point P, the constraint associated with an obstacle of normal vector, $...n$ is written as:

$$\vec{\boldsymbol{n}}^{\boldsymbol{T}}\vec{\boldsymbol{v}}_{\boldsymbol{p}} = 0. \tag{4}$$

Substituting eq.(2) into eq.(3) then yields

$$\vec{\boldsymbol{n}}^{\boldsymbol{T}}\boldsymbol{J}_{\boldsymbol{p}}\boldsymbol{J}^{-1}\vec{\boldsymbol{v}} = 0. \tag{5}$$

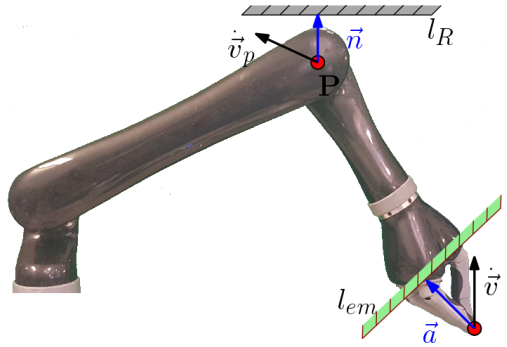

Fig. 5. Example of transformation of a limitation at point P (defined here at the elbow) into a limitation at the end-effector. The vector $\vec{\boldsymbol{v}}_{\boldsymbol{p}}$ represents the velocity of the elbow when the velocity $\vec{\boldsymbol{v}}$ is commanded at the end-effector. Vector $\vec{\boldsymbol{a}}$ represents the limitation at the end-effector that corresponds to the limitation met at the elbow.

This equation represents the constraint at point P and is linear in therms of $\vec{\boldsymbol{v}}$. It can be simply written as

$$\vec{\boldsymbol{a}}^{\boldsymbol{T}}\vec{\boldsymbol{v}} = 0, \tag{6}$$

where

$$\vec{\boldsymbol{a}}^{\boldsymbol{T}} = \vec{\boldsymbol{n}}^{\boldsymbol{T}}\boldsymbol{J}_{\boldsymbol{p}}\boldsymbol{J}^{-1}. \tag{7}$$

It can be noted that, as the scalar product of $\vec{\boldsymbol{a}}^{\boldsymbol{T}}$ and $\vec{\boldsymbol{v}}$ is null, vector $\vec{\boldsymbol{a}}$ represents the direction in which the end-effector's velocity must be constrained to in order to satisfy the limitation at point P defined on the robot.

Finally, the equation providing the limitation $..a$ at the end-effector corresponding to a limitation $..n$ at point P is written as

$$\vec{\boldsymbol{a}} = (\boldsymbol{J}_{\boldsymbol{p}}\boldsymbol{J}^{-1})^{\boldsymbol{T}}\vec{\boldsymbol{n}}. \tag{8}$$

The limitation $\vec{\boldsymbol{a}}$ can be treated as a limitation occurring at the end-effector and processed simulteanously with the other limitations. An example is shown in Fig. 5, where point P is defined at the elbow. The limitation l_R of normal vector $\vec{\boldsymbol{n}}$ at the elbow is transformed into a limitation l_{em} of normal $\vec{\boldsymbol{a}}$ at the end-effector.

B. Angular limitation

In order to manage all the limitations simultaneously, the articular limitations must also be transferred into Cartesian limitations at the end-effector. Such a limitation is active when an angular limitation is reached and when the desired motion requires the joint to move in the limitation's direction. In order to find the equations to transfer the angular limitation constraints into end-effector constraints, the limited joint is considered blocked and the corresponding column of the Jacobian matrix , $\mathbf{J}$, defined in eq. (1) is replaced with a column of zeros.

Referring to eq.(1), it is clear that all the possible Cartesian velocities that satisfy the constraints imposed by the locked joint must be linear combinations of the two remaining columns of this Jacobian matrix $\boldsymbol{J}$, In other words, this the limitation can be expressed as a plane formed by the remaining columns of the 3×3 Jacobian matrix of the robot.

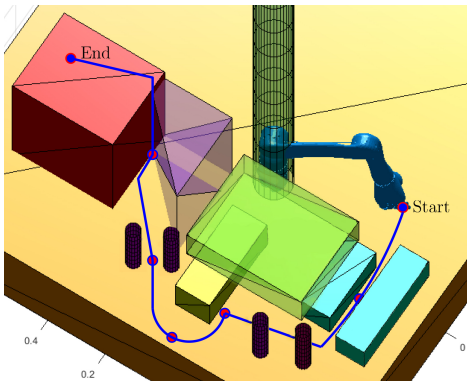

Fig. 6. Geometric description of the experimental set-up. The polyhedra are physical limitations. The four small cylinders are unprotected obstacles that users must avoid during the task. The blue circled red dots are the way points users must cross. The blue line is an example of a path to complete the task.

The forbidden direction is then computed using the cross product of the two column vectors.

$$\vec{n}_f = \vec{c}_1 \times \vec{c}_2 \tag{9}$$

where $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are the non-zero column vectors of matrix $\mathbf{J}$. This forbidden direction $\vec{n}_f$ can now be processed simulteanously with all the other limitations.

C. Singularities

Singularities can be expressed by Cartesian or articular constraints. In both cases, singularity limitations can be expressed in end-effector limitations using the approach presented in Sections V-A and V-B.

VI. Experiments

This section presents the experimental protocol along with the experiment's results. The main objective of the interactive kinematics algorithm is to improve the intuitiveness of the robot control as well as the human performance. In order to assess the effectiveness of the algorithm. A given task was realized by several participants, both with and without the algorithm. In the latter case, the robot velocity was set to zero when a limitation was reached (as with a standard commercial robot). Subjects were not told which algorithm was used and the order was varied between subjects. The experiments were performed on a modified version of the JACO arm from Kinova as shown in Fig. 1. Thirteen (13) subjects aged between 21 and 30 participated in the experiments which were approved by the ethics committee of Université Laval certificate no. 2016-011/12-02-2016. A video shows excerpts of the experiments.

A. The task

The task has been designed to test all cases of possible limitations. Figure 6. shows the workspace and the task trajectory.

B. First experiment: Full attention

The first experiment required the subjects to bring the end-effector to the way points shown in Fig. 6 while avoiding objects in the workspace. and the number of collisions with the objects were recorded. Without the sliding algorithm, the mean completion time is 86.5s with a standard deviation of 32.75s with an average of 1.1 collisions. With the sliding algorithm, the mean completion time is 46.4s with a standard deviation of 15.2s and with an average of 0.4 collisions.

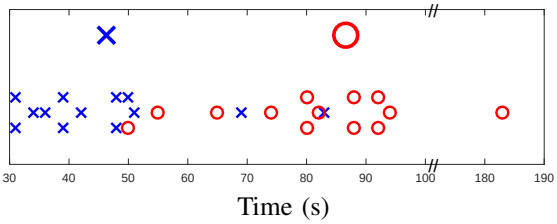

Fig. 7. Experimental results for the first experiment (full attention). Blue 'X's correspond to the test performed with the sliding algorithm while the red 'O's correspond to the test performed without the algorithm. The larger symbols represent the average for each case.

The mean difference between the completion of the task with and without the algorithm is 87% which corresponds to a difference of 40 seconds. The time difference is considered significant according to a Mann-Whitney-Wilcoxon non parametric test one-tailed (p= $9.87 * 10^{-}5$<0:05). It is important to note that the performance improvement is highly dependent on the chosen task. For instance, a task where no limitation is reached would not improve whether the sliding algorithm is used or not. The authors tried to be fair in the selection of the task.

C. Second experiment: Divided attention

In this second experiment, the task required to complete the experiment is similar to the one used in the first experiment. However, a secondary task was added in order to divide the subject's attention (similarly to an industrial task). This secondary task consists in naming a color appearing on a screen at every two seconds. To increase the task's difficulty, the color appearing on the screen is embedded in the font of letters spelling a different color name. For instance, the word blue is spelled on the screen using a red font. The correct answer is then red. The purpose of this secondary task is to assess the additional attentional load of a cognitive process. The time to complete the task and the total number of errors were recorded. Omitting to name a color is counted as two mistakes, mentioning the wrong color is compiled as one mistake and colliding with an object is counted as three mistakes. The objective of this additional task is to assess the level of attention required to accomplish a task while having a secondary task to accomplish. This kind of multitasking experiment may help to represents a normal industrial task where external factors could distract the operator or where the operator must pay attention to several elements. In the context of this experiment, the hypothesis is that the sliding algorithm reduces the level of attention required to complete the task. The user would not have to pay attention to the robot's limitations and self-collisions and could thus give more attention to the main task. The results of this experiment are shown in Fig. 8.

Without the sliding algorithm, the mean completion time is 95.3s with a standard deviation of 34.09s with an average of 9.1 collisions. With the sliding algorithm, the mean comple-

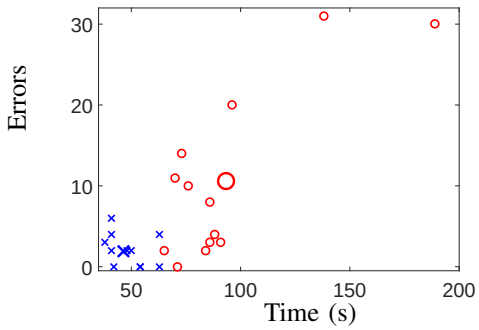

Fig. 8. Experimental results for the second experiment (devided attention). Blue 'X's correspond to the test performed with the sliding algorithm while the red 'O's correspond to the test performed without the algorithm.

tion time is 48.3s with a standard deviation of 9.92s and with an average of 1.9 collisions. The mean difference between the completion of the task with and without the algorithm is 97% which corresponds to a difference of 47 seconds. The time difference is considered significant according to a Mann-Whitney-Wilcoxon non parametric test one-tailed (p= $8.05 * 10^{-6}$<0:05).

D. Comparison between the two tests

For the first experiments, the participants mostly declared that the task was easier to perform it with the algorithm. For the divided attention test, the participants mostly mentioned that the completion of the task was much more stressful and required a lot of attention when the algorithm was not activated. Some participants even said that they would have given up the experiment if it had been longer. When comparing the results between the full attention test and the divided attention test, the efficiency gain translates more in terms of errors avoided than saved time. The time improvement between the two tests (87% for the full attention test, 97% for the divided attention test) is not considered significant according to a Mann-Whitney-Wilcoxon non parametric test one-tailed (p= 0.1582>0:05) while the improvement in terms of error is considered significant according to the same test (p= 0.0032<0:05).

VII. Conclusion

This paper presented an anticipative robot kinematic limitation avoidance algorithm for a 3-DOF serial robot. This algorithm aims at making the robot control easier, more intuitive and safer. Experiments have been performed to observe the efficiency gains while completing tasks realized with the help of the algorithm. One experiment was realized with full attention, another with divided attention. Both tests showed results indicating an increased efficiency (87% and 101% faster completion time) and precision (0.7 and 7.2 fewer errors made) when using the algorithm. These results indicate that the proposed sliding algorithm allows an easier completion of a given task as well as making it safer to accomplish for the robot and the environment. The users are also able to give attention to other secondary tasks while using a serial link manipulator without prior training or knowledge about the basics of the robot's kinematics. Further work will be undertaken to generalize the collision detection algorithm to even more complex shapes as well as being able to treat potential collisions with complete surfaces composing the robot arm. 3D cameras could also be utilized to detect objects that are introduced or moved in the workspace during the completion of the task. The algorithm will also be expanded to the 6 DOF case.

References

[1] M. Vagaš, J. Semjon, M. Hajduk, L. Páchniková, and M. Lipčak, "The view to the current state of robotics," *Proceedings of 2011 International Conference on Optimization of the Robots and Manipulators*, 2011.

[2] A. Lecours, B. Mayer-St-Onge, and C. Gosselin, "Variable admittance control of a four-degree-of-freedom intelligent assist device," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 2, pp. 3903–3908, 2012.

[3] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa, "Effects of Robot Motion on Human-Robot Collaboration," *ACM/IEEE Int. Conference on Human-Robot Interaction*, pp. 51–58, 2015.

[4] A. Campeau-lecours, "An Anticipative Kinematic Limitation Avoidance Algorithm For Collaborative Robots : Two-Dimensional Case," 2016.

[5] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[6] S. R. S. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *University of California, San Diego, Typeset manuscript*, vol. 132, no. 4, pp. 1–19, 2004.

[7] S. R. Buss and J.-S. Kim, "Selectively Damped Least Squares for Inverse Kinematics," *Journal of Graphics, GPU, and Game Tools*, vol. 10, no. 3, pp. 37–49, 2005.

[8] D. E. Schinstock, "Approximate solutions to unreachable commands in teleoperation of a robot," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 3, pp. 219–227, 1998.

[9] J. Kuffner and K. Nishiwaki, "Self-Collision Detection and Prevention for Humanoid Robots.," *the 2002 IEEE International Conference on Robotics & Automation,*, no. Icra, 2002.

[10] F. Seto, K. Kosuge, and Y. Hirata, "Self-collision Avoidance Motion Control for Human Robot Cooperation System using RoBE," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Robotics and Automation. Proceedings. 1985 IEEE International Conference*, vol. 2, pp. 500–505, 1985.

[12] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic Motion Planning for Robots in Partially Unknown Environments," *IFAC World Congress (IFAC2011), Milano, Italy*, vol. 18, 2011.

[13] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *IEEE Int. Conference on Robotics and Automation*, pp. 1398–1404, 1991.

[14] A. De Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1–6, 2007.

[15] L. B. Rosenberg, "Perceptual tools for Teleroboic Manipulation," *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 76–82, 1993.

[16] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A Depth Space Approach to Human-Robot Collision Avoidance," *IEEE International Conference on Robotics and Automation*, pp. 338–345, 2012.

[17] A. Mohr, "Quantum Computing in Complexity Theory and Theory of Computation," *Carbondale, IL*, 2014.

[18] G. V. D. Bergen, "A Fast and Robust GJK Implementation for Collision Detection of Convex Objects," *Journal of Graphics Tools*, vol. 4, no. 2, pp. 7–25, 1999.

[19] H. Samet, "Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods.," 1989.

[20] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, "JACO Assistive Robotic Device: Empowering People With Disabilities Through Innovative Algorithms," *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA) Annual Conference*, 2016.