IEEE 15th International Workshop on Advanced Motion Control
March 9-11, 2018, Tokyo, Japan

On Prioritized Inverse Kinematics Tasks: Time-Space Decoupling

Wael Suleiman [*], Ko Ayusawa [†], Fumio Kanehiro [‡] and Eiichi Yoshida [†]
[*] Electrical and Computer Engineering Department, University of Sherbrooke, Canada
Email: Wael.Suleiman@USherbrooke.ca
[†] CNRS-AIST, JRL (Joint Robotics Laboratory), UMI 3218/RL, Tsukuba, Japan
Email: {k.ayusawa, e.yoshida}@aist.go.jp
[‡] Intelligent Systems Research Institute, AIST, Tsukuba, Japan
Email: f-kanehiro@aist.go.jp

***Abstract*— In this paper we propose a new robust and computationally efficient algorithm to solve the problem of prioritized inverse kinematics (IK) tasks. The main idea of the new algorithm is assuming that the sampling time is a free variable and solving the IK problem with respect to an extended variable that includes the joint variation and the sampling time. The main advantage of the new formulation is that it can always provide a physically consistent solution, moreover the user only needs to provide a geometric description of the desired operational space paths as the algorithm provides the required time on the fly to convert those paths into trajectories for the robot joints.**

The proposed algorithm has been extensively validated in simulation, and the results revealed its efficiency to deal with the priority of tasks as well as the robot physical limits, furthermore, the algorithm always succeeded in providing an admissible solution.

I. Introduction

The Inverse Kinematics (IK) problem is one of the most important problems in robotics, and it has been extensively studied in the literature [1], [2], [3], [4], [5]. Recently, the concept of prioritized IK tasks has received a lot of attention [4], [5], [6] as many robotic systems are nowadays redundant, i.e. they have more Degrees of Freedom (DoF) than required for a specific kinematic task, for instance 6 DoF for a standard three-dimensional task. As the tasks cannot be simultaneously achieved, the user role is to define the task priority in accordance with their importance. However, if the first priority task becomes infeasible, the other tasks with lower priority will not be achieved.

The prioritized IK tasks problem is usually defined for n kinematic tasks, however, without loss of generality we only consider the case of two tasks as follows:

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}(t)} \dot{\boldsymbol{q}}(t)^T Q \dot{\boldsymbol{q}}(t) \\ &\text{subject to} \\ &\text{First priority:} \quad J_1 \dot{\boldsymbol{q}}(t) = \dot{r}_1(t) + K_1\, e_1(t) \\ &\text{Second priority:} \quad J_2 \dot{\boldsymbol{q}}(t) = \dot{r}_2(t) + K_2\, e_2(t) \end{aligned} \tag{1}$$

where Q is a diagonal and positive semi-definite matrix, $J_\times$ is the Jacobian matrix, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity and $\dot{r}_\times(t)$ is the linear and angular velocity of the desired trajectory, K is a positive definite (usually diagonal) matrix and e is the operational space error defined as follows:

$$e = x_d - x_e \tag{2}$$

(a) HRP-2 (b) HRP-3 (c) HRP-4

Fig. 1. Robots having similar kinematic structures but different velocity and joint limits

where x_d is the desired position and orientation and x_e is the actual one.

The optimization problem (1) can be efficiently solved using the pseudo-inverse technique [4], [5] as follows:

$$\begin{aligned} \dot{\boldsymbol{q}} =& J_1^\dagger(\dot{r_1} + K_1\, e_1(t)) + \cdots \\ & (I - J_1^\dagger J_1)\hat{J}_2^\dagger((\dot{r_2} + K_2\, e_2(t)) - J_2 J_1^\dagger(\dot{r_1} + K_1\, e_1(t))) \end{aligned} \tag{3}$$

where:

$$\begin{aligned} J_\times^\dagger &= Q^{-1} J_\times^T \left(J_\times\, Q^{-1}\, J_\times^T\right)^{-1} \\ \hat{J}_2 &= J_2(I - J_1^\dagger J_1) \end{aligned} \tag{4}$$

If Q is the identity matrix, $J_\times^\dagger$ becomes the well-known Moore-Penrose pseudo-inverse.

Even though the analytical solution in (3) is very attractive from a computational point of view, it cannot allow the integration of constraints on the velocity and joint limits or avoiding obstacles. Therefore, most of recent research in the field are instead considering the following optimization problem:

$$\min_{\dot{\boldsymbol{q}}(t)} \dot{\boldsymbol{q}}(t)^T Q \dot{\boldsymbol{q}}(t)$$

subject to

$$\begin{array}{ll} \text{First priority:} & \left\{ \begin{array}{l} J_1\dot{\boldsymbol{q}}(t) = \dot{r}_1(t) + K_1\, e_1(t) \\ A_1\dot{\boldsymbol{q}}(t) \le b_1 \end{array} \right. \\ \text{Second priority:} & \left\{ \begin{array}{l} J_2\dot{\boldsymbol{q}}(t) = \dot{r}_2(t) + K_2\, e_2(t) \\ A_2\dot{\boldsymbol{q}}(t) \le b_2 \end{array} \right. \\ & \hat{\dot{\boldsymbol{q}}}^- \le \dot{\boldsymbol{q}}_t \le \hat{\dot{\boldsymbol{q}}}^+ \end{array} \tag{5}$$

As in the previous case, Q is a diagonal and positive semi-definite matrix. $\hat{\dot{\boldsymbol{q}}}^-$ and $\hat{\dot{\boldsymbol{q}}}^+$ are respectively the generalized lower and upper limits of the velocity defined as in [7], [8]. Those limits provide a compact and efficient way to deal with both velocity and joint limits. $A_\times$ and $b_\times$ represent the additional constraints that result from considering geometric constraints, such as collision avoidance, or avoiding the obstruction of the robot visual field, for more details the reader is referred to [7], [9], [10], [11], [6].

The optimization problem (5) can be efficiently solved by a Quadratic Programming (QP) solver. However, by adding the inequality constraints the problem can easily become infeasible, this is mainly true if the desired Cartesian trajectories are not well defined. Defining a suitable Cartesian trajectory is tricky as robots having similar kinematic structures can have very different velocity and joint limits, an example is given in Fig. 1.

A possible solution to deal with the numerical infeasibility issue would be relaxing the constraints, thus the optimization problem (5) becomes:

$$\min_{\dot{\boldsymbol{q}}, w_1, w_2} \dot{\boldsymbol{q}}(t)^T Q \dot{\boldsymbol{q}}(t) + \alpha_1\, w_1^T\, w_1 + \alpha_2\, w_2^T\, w_2$$

subject to

$$\begin{array}{l} J_1\dot{\boldsymbol{q}}(t) = \dot{r}_1(t) + K_1\, e_1(t) + w_1^0 \\ A_1\dot{\boldsymbol{q}}(t) \le b_1 + w_1^1 \\ J_2\dot{\boldsymbol{q}}(t) = \dot{r}_2(t) + K_2\, e_2(t) + w_2^0 \\ A_2\dot{\boldsymbol{q}}(t) \le b_2 + w_2^1 \\ \hat{\dot{\boldsymbol{q}}}^- \le \dot{\boldsymbol{q}}_t \le \hat{\dot{\boldsymbol{q}}}^+ \end{array} \tag{6}$$

Where $w_1^T = \left[{w_1^0}^T\, {w_1^1}^T \right]$ and $w_2^T = \left[{w_2^0}^T\, {w_2^1}^T \right]$ are slack variables, α_1 and α_2 are user-defined constants ($\alpha_1 >> \alpha_2$, for instance $\alpha_1 = 10^3 \times \alpha_2$). Even though the infeasibility issue is solved and the obtained motion is physically consistent, the generated Cartesian (position and orientation) trajectory might significantly diverge from the desired one or have a collision with the environment as the constraints are not fully satisfied [12].

In [12], we have proposed an approach that is based on considering the sampling time as an additional parameter, thus solving the infeasibility issue. The main contribution in this paper is adapting that method to the case of prioritized tasks and proposing an efficient and robust algorithm to solve it in real time as well as solving the discontinuity issue of discrete time control. We have also proposed an extension to handle the case of floating-base robots, e.g. humanoid robots.

II. Time-Space Decoupling

As above-mentioned the main idea is transforming the sampling time (T), which is usually supposed fixed, into a free variable, and solving the inverse kinematics problem as a function of $\Delta\boldsymbol{q}$ and T. Thus, the IK problem in (5) is transformed into the following equivalent QP problem:

$$\min_{\Delta\boldsymbol{q}, T, w_0, w_1} \Delta\boldsymbol{q}^T Q \Delta\boldsymbol{q} + \beta T^2 + \alpha \left[w_0^T\, w_1^T \right] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

subject to

$$\begin{array}{l} J_1\Delta\boldsymbol{q} = \Delta r_1 + K_1\, e_1\, T \\ A_1\Delta\boldsymbol{q} \le b_1\, T \\ J_2\Delta\boldsymbol{q} = \Delta r_2 + K_2\, e_2\, T + w_0 \\ A_2\Delta\boldsymbol{q} \le b_2\, T + w_1 \\ \hat{\dot{\boldsymbol{q}}}^-\, T \le \Delta\boldsymbol{q} \le \hat{\dot{\boldsymbol{q}}}^+\, T \\ \epsilon \le T \end{array} \tag{7}$$

where Δr is defined in the same way as in [12]. $0 < \epsilon \ll 1$, $\beta > 0$, and $\alpha > 0$ are user-defined constants. In practice, a possible choice for α and β is $\alpha = \beta = 10^3 \times \|Q\|$, where $\|Q\|$ is the norm of the matrix Q. The role of the slack variable $w = \left[w_0^T\, w_1^T \right]^T \in \mathbb{R}^p$ is defining the priority between the tasks.

The QP problem (7) can be efficiently solved, in real time, using an appropriate QP solver such as qpOASES solver [13].

III. Discrete Time Control

As modern robotic systems are usually controlled by a control unit with a fixed time step, the sampling time T should satisfies the following additional constraint:

$$T = n\, T_s \tag{8}$$

where $n \in \mathbb{N}$ is a strictly positive integer and T_s is the robot control loop fixed time step.

Thus, the optimization problem (7) becomes:

$$\min_{\Delta\boldsymbol{q}, n, w_0, w_1} \Delta\boldsymbol{q}^T Q \Delta\boldsymbol{q} + \beta n^2 + \alpha \left[w_0^T\, w_1^T \right] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

subject to

$$\begin{array}{l} J_1\Delta\boldsymbol{q} = \Delta r_1 + K_1\, e_1\, n\, T_s \\ A_1\Delta\boldsymbol{q} \le b_1\, n\, T_s \\ J_2\Delta\boldsymbol{q} = \Delta r_2 + K_2\, e_2\, n\, T_s + w_0 \\ A_2\Delta\boldsymbol{q} \le b_2\, n\, T_s + w_1 \\ \hat{\dot{\boldsymbol{q}}}^-\, n\, T_s \le \Delta\boldsymbol{q} \le \hat{\dot{\boldsymbol{q}}}^+\, n\, T_s \\ 1 \le n \end{array} \tag{9}$$

The above optimization problem is a Mixed-Integer Quadratic Programming (MIQP) problem, which is generally more complex and computationally more expensive than a standard QP.

However, as mentioned in [12], one does not need to solve the above MIQP problem, but instead one can still solve (7) by replacing ϵ by T_s, and once $\Delta\boldsymbol{q}$ and T are obtained, the parameter n can be obtained such as:

$$(n-1)\,T_s + \frac{T_s}{2} < T \leq n\,T_s + \frac{T_s}{2} \tag{10}$$

However, in practice, the operation in (10) could lead to discontinuity in the joint velocity $\dot{\boldsymbol{q}}_t \cong \frac{\Delta\boldsymbol{q}}{n\,T_s}$.

To solve the discontinuity issue, a solution would be to modulate the vector $\Delta\boldsymbol{q}$ by the factor $\frac{n\,T_s}{T}$, thus:

$$\begin{aligned}\dot{\boldsymbol{q}}_t &\cong \frac{\frac{n\,T_s}{T}\Delta\boldsymbol{q}}{n\,T_s} \\ &\cong \frac{\Delta\boldsymbol{q}}{T}\end{aligned}$$

As a result, the solution to (9) would be:

$$\mathcal{X}^* = \begin{bmatrix}\frac{n\,T_s}{T}\Delta\boldsymbol{q} \\ n\,T_s\end{bmatrix} \tag{11}$$

where $\Delta\boldsymbol{q}$ and T are the solutions to (7), and n satisfies the constraint in (10).

IV. Floating-base Robots

Since a humanoid robot is not fixed to the environment, the robot has the same mechanical property of a free-floating system. The base link of a humanoid robot is often modeled as a floating base which has 6 DoF. In the case of classical manipulators, the numerical integration of joint angles is simply computed as follows.

$$\boldsymbol{q}_t = \boldsymbol{q}_{t-1} + \Delta\boldsymbol{q} \tag{12}$$

On the other hand, the generalized coordinates of a floating base includes not only its position but also its orientation. The integration of the generalized velocities cannot be computed according to (12); the integration of rotational motion needs to be considered.

Let us represent the generalized coordinates of a humanoid robot by using the following variables:

$\boldsymbol{p}$: the vector of the position of the base link
$\boldsymbol{R}$: the orientation matrix of the base link
$\boldsymbol{\theta}$: the vector of the joint angles

Then, let us define the generalized velocities $\dot{\boldsymbol{q}}$ as follows:

$$\dot{\boldsymbol{q}} = \begin{bmatrix}\boldsymbol{\nu} \\ \boldsymbol{\omega} \\ \dot{\boldsymbol{\theta}}\end{bmatrix}$$

where, $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ are the linear and angular velocity (or the spatial velocity) of the base link, and $\dot{\boldsymbol{\theta}}$ indicates the joint angle velocities. Note that, $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ are expressed in the local coordinates, this is because the local expression is suitable when computing the Jacobian matrices.

Let us consider the following vector $\Delta\boldsymbol{q}$ by linearly integrating $\dot{\boldsymbol{q}}$ with the small-time step ΔT.

$$\Delta\boldsymbol{q} = \begin{bmatrix}\Delta\widehat{\boldsymbol{p}} \\ \Delta\widehat{\boldsymbol{r}} \\ \Delta\boldsymbol{\theta}\end{bmatrix} = \dot{\boldsymbol{q}}\Delta T \tag{13}$$

where, $\Delta\widehat{\boldsymbol{p}}(=\boldsymbol{\nu}\Delta T)$, $\Delta\widehat{\boldsymbol{r}}(=\boldsymbol{\omega}\Delta T)$ and $\Delta\boldsymbol{\theta}(=\dot{\boldsymbol{\theta}}\Delta T)$ are the sub-vectors of $\Delta\boldsymbol{q}$. Note that $\Delta\boldsymbol{q}$ does not have physical meaning due to the linear integration of the spatial velocity. Then, the numerical integration of the generalized velocities is formulated by using $\Delta\boldsymbol{q}$ as follows:

$$\begin{aligned}\boldsymbol{p}_t &= \boldsymbol{p}_{t-1} + \boldsymbol{R}_{t-1}\Delta\widehat{\boldsymbol{p}} \\ \boldsymbol{R}_t &= \boldsymbol{R}_{t-1}e^{[(\Delta\widehat{\boldsymbol{r}})]^\wedge} \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}\end{aligned} \tag{14}$$

where, $e^{\boldsymbol{M}}$ means the matrix exponential of $\boldsymbol{M}$, and $[.\,]^\wedge$ designs the skew operator defined as follows:

$$[.\,]^\wedge : \boldsymbol{\omega} \in \mathbb{R}^3 \rightarrow so(3)$$
$$[\boldsymbol{\omega}]^\wedge = \begin{bmatrix}0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0\end{bmatrix}$$

When implementing the discrete time control for a humanoid robot, the solution $\Delta\boldsymbol{q}$ of the optimization problem (7) needs to be integrated according to the update rule of (14) and (13) in order to compute $\boldsymbol{p}_t$, $\boldsymbol{R}_t$, and $\boldsymbol{\theta}_t$ of the robot at each time instance.

V. Complexity Analysis

Let us compare the complexity of the new algorithm with the conventional inverse kinematics algorithm with relaxed constraints, i.e. Eq. (6).

The dimension of the optimization variable in the proposed algorithm ($\mathcal{X}$) is $n+p+1$ where n is the dimension of the joint vector ($\boldsymbol{q}_t$), and p is the dimension of equality and inequality constraints of the second priority task, while the optimization variable in (6) is $\mathrm{X}^T = [\dot{\boldsymbol{q}}_t^T\; w_1^{0\,T}\; w_1^{1\,T}\; w_2^{0\,T}\; w_2^{1\,T}]$ has a dimension of $n+2p$ if we suppose that both tasks have the same number of equality and inequality constraints. Regarding the equality and inequality constraints, the new algorithm has one additional constraint in comparison with the conventional method, this constraint is on the variable T.

As a result, the complexity of the new algorithm is comparable to the conventional inverse kinematics one. However, it is important to recall that the proposed algorithm has the advantage of overcoming the problem of infeasibility and avoiding a significant deviation from the desired trajectory.

The numerical simulations in Section VI also confirmed the above conclusions.

VI. Simulation Results

We have conducted five simulation scenarios:

A. Scenario 1: Comparing with Conventional Method

To compare the performance of the proposed algorithm with the conventional method in (6), we consider a planar redundant manipulator, which has 4 degrees of freedom (Fig. 2). We defined two prioritized kinematic tasks as follows:

- First priority task: The end effector has to follow a path that is designed as a Bezier curve as shown in Fig. 2.
- Second priority task: The path of third joint (Joint 3 in Fig. 2) has to stay parallel to the vertical axis (y axis).

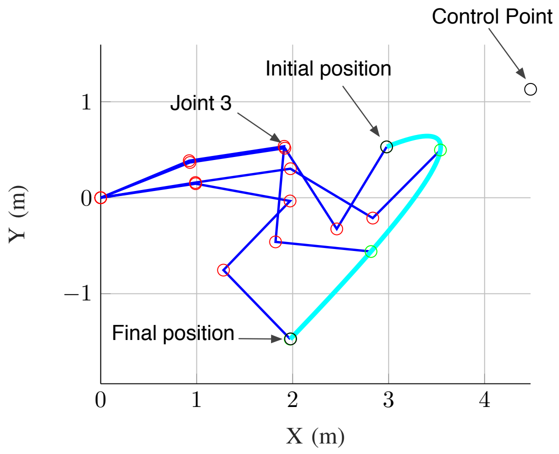
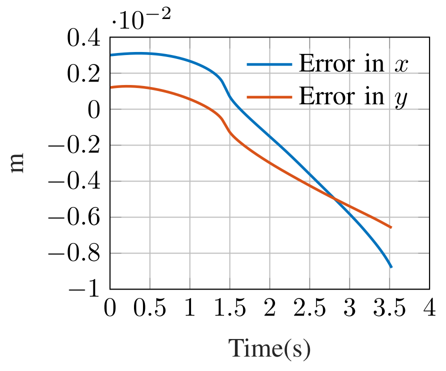


Fig. 2. Planar robot configurations

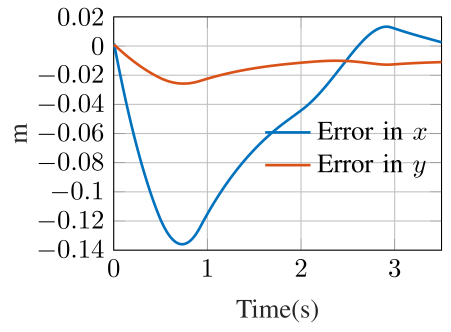
The joint velocity limits have been chosen as follows:

$$\dot{\boldsymbol{q}}^{+} = -\dot{\boldsymbol{q}}^{-} = 0.5 \times [1 \quad 1 \quad 1 \quad 1]^T \tag{15}$$

For comparison reasons, we also solved the IK problem in (6) by transforming the end-effector path into a trajectory using a uniform time distribution with a sampling time $T = 5 \times 10^{-3} s$ and the final time $T_f = 3.5\, s$ is set equal to the one obtained by the proposed algorithm. Both algorithms generate a motion within the robot physical limits, however, Fig. 3 shows that the tracking error of the end-effector trajectory in the case of conventional method is significant in comparison with the proposed algorithm.



(a) Proposed algorithm



(b) Conventional method Eq.(6)

Fig. 3. **Scenario 1**: End-effector tracking error

B. *Scenario 2: Different Velocity Limits*

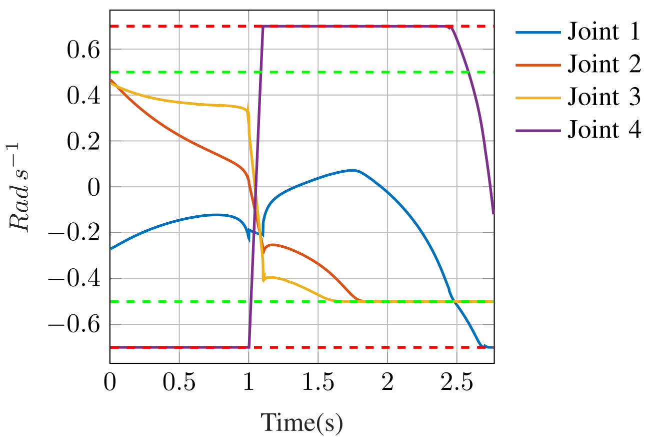
In this scenario, we consider the same prioritized tasks as in Scenario 1 but with two different sets of velocity limits:

1) Set 1:

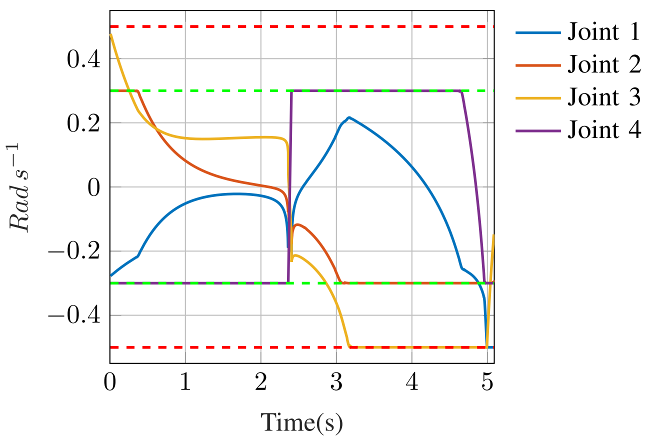
$$\dot{\boldsymbol{q}}^{+} = -\dot{\boldsymbol{q}}^{-} = [0.7 \quad 0.5 \quad 0.5 \quad 0.7]^T$$

2) Set 2:

$$\dot{\boldsymbol{q}}^{+} = -\dot{\boldsymbol{q}}^{-} = [0.5 \quad 0.3 \quad 0.5 \quad 0.3]^T$$



(a) Set 1 of joint velocity limits



(b) Set 2 of joint velocity limits

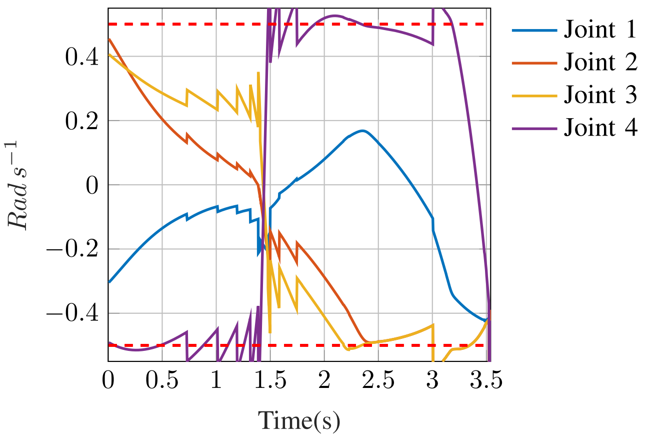
Fig. 4. **Scenario 2**: Impact of different joint velocity limits

Fig. 4 shows how the proposed algorithm automatically modified the joint trajectories according to the joint velocity limits and the trajectory time is also accordingly modified.

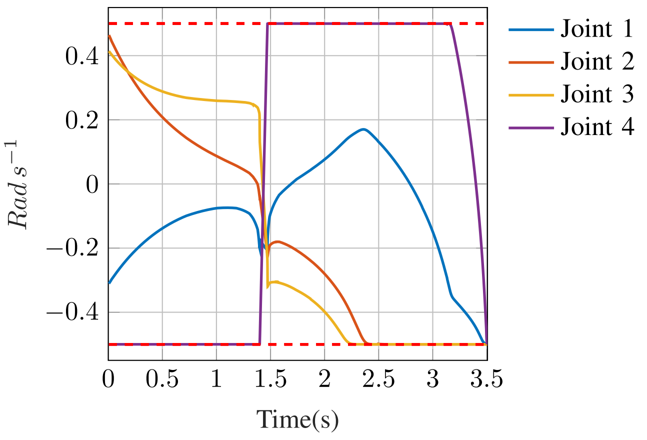
C. *Scenario 3: Discrete Time Control*

We consider again Scenario 1 with a control unit of fixed sampling time $T_s = 5 \times 10^{-3}\, s$, we compare the results of applying the proposed algorithm in Section III without and with modulation Eq. (11). Fig. 5 shows the results for this scenario, one can observe that the modulation proposed in Section III successfully solved the discontinuity problem of joint velocity. Moreover, Fig. 5(a) shows that the joint velocity limits are not satisfied because of the discontinuity issue.

The profile of discretized sampling time is shown in Fig. 6, recall that the sampling time is $T = nT_s$.



(a) Joint velocity without modulation



(b) Joint velocity with modulation Eq.(11)

Fig. 5. **Scenario 3**: Discrete time control

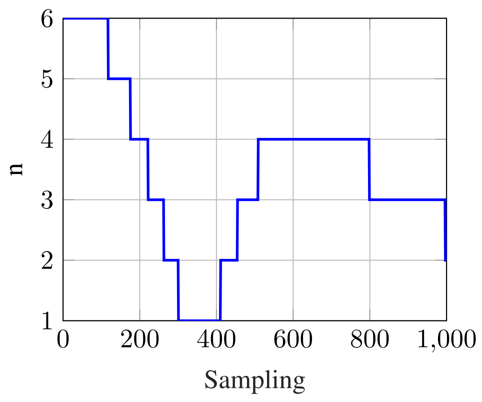


Fig. 6. **Scenario 3**: Discretized sampling time

D. Scenario 4: Avoiding Obstacles

We added an obstacle to Scenario 1, the obstacle is represented by the red circle in Fig. 7. The new prioritized tasks are:

- First priority task: The end effector has to follow a Bezier curve as in Scenario 1 and the robot should avoid the collision with the circle. The collision avoidance is expressed as an inequality constraint, more details are in [7].
- Second priority task: The path of the third joint has to stay parallel to the vertical axis.

The joint velocity limit is the same as in Scenario 1.

Fig. 8 shows that the discretized sampling time rapidly increased around the sampling 620, at that instance the distance between the robot and the obstacle became smaller than the interference distance. This increase of sampling time, slowing down the motion, is the direct result of the velocity damping in the collision avoidance formulation.

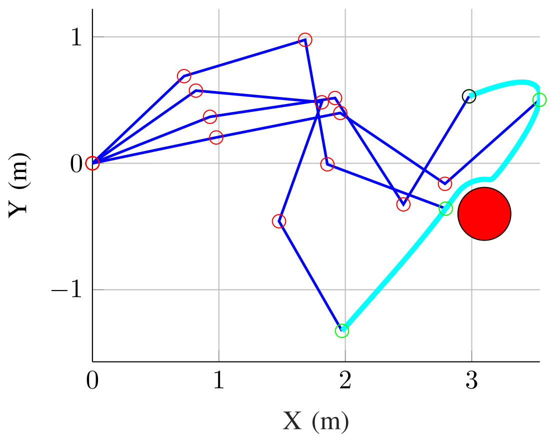


Fig. 7. **Scenario 4**: Planar robot trajectory and avoiding the obstacle (red cercle)

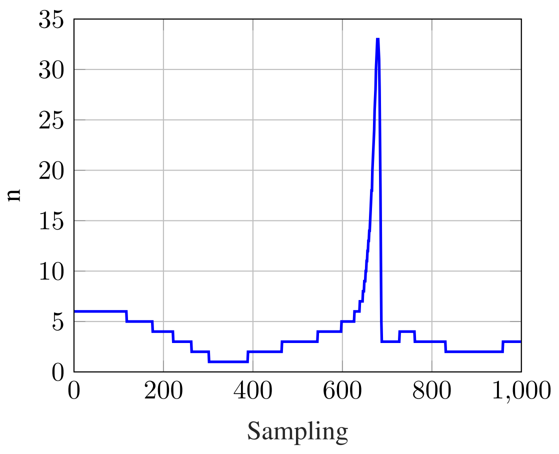


Fig. 8. **Scenario 4**: Discretized sampling time

E. Scenario 5: Real Time Motion Retargeting

The objective in this scenario is to convert human capture data to the humanoid robot HRP-4 while considering the velocity and joint limits of the robot. We have chosen a large valve opening task which is similar to one of the tasks in DARPA Robotics Challenge (DRC).

The first priority tasks are the positions and orientations of the feet, they are fixed during the motion, the second priority tasks are the positions and orientations of the hands and the floating-base, which is the waist joint.

Snapshots of the robot obtained motion is shown in Fig. 9, and the discretized sampling time is given in Fig. 10. We consider that the control loop sampling time of the robot is $T_s = 10^{-3}\,s$.

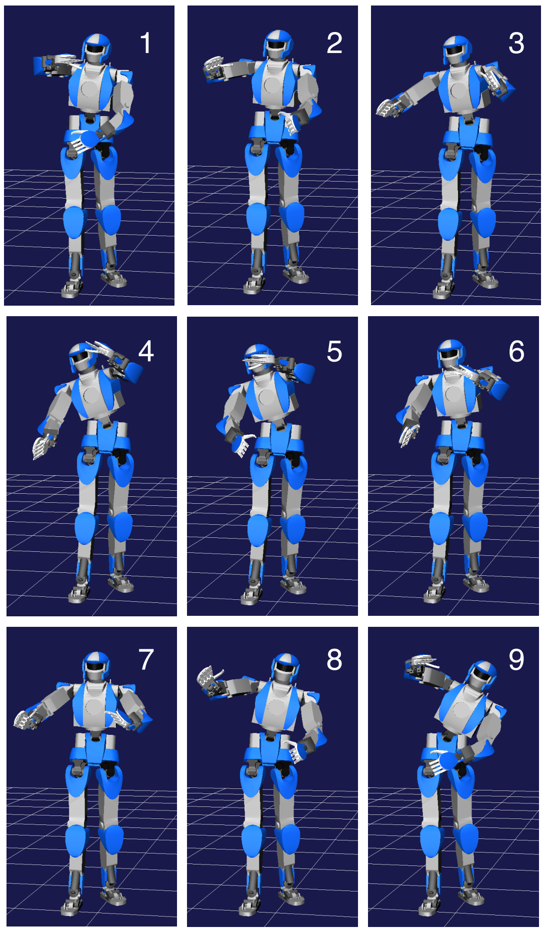


Fig. 9. **Scenario 5**: Snapshots of the retargeted motion with the humanoid robot HRP-4

VII. Conclusion and Future Work

In this paper, we proposed a robust algorithm to handle the case of prioritized kinematics tasks, it has as input the desired path instead of the conventional desired trajectory. As a result, it is easier for the user to define the tasks without worrying about the robot joint velocity limits, moreover, the desired tasks can be applied to different robotic systems having a similar kinematic structure, but different velocity and joint limits. The method is simple to implement, as it is formulated as Quadratic Programming (QP) problem that can be efficiently solved in real time.

As a future work, we are planning to implement and validate the proposed algorithm on real robotic systems such as Baxter research robot or the humanoid robot HRP-4.

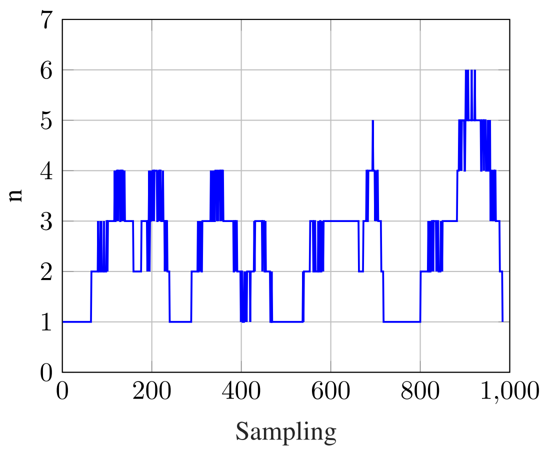


Fig. 10. **Scenario 5**: Discretized sampling time

Acknowledgment

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the grant RGPIN-2012-419406.

References

[1] D. E. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 94, no. 4, pp. 303–309, 12 1972.

[2] Y. Nakamura and K. Yamane, "Dynamics computation of structure-varying kinematic chains and its application to human figures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 2, pp. 124–134, Apr 2000.

[3] S. Chan and P. Lawrence, "General inverse kinematics with the error damped pseudoinverse," in *IEEE International Conference on Robotics and Automation*, Apr 1988, pp. 834–839 vol.2.

[4] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[5] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[6] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, pp. 1006–1028, 2014.

[7] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A Local Collision Avoidance Method for Non-strictly Convex Objects," in *2008 Robotics: Science and Systems Conference*, Zurich, Switzerland, June 2008.

[8] W. Suleiman, "On inverse kinematics with inequality constraints: new insights into minimum jerk trajectory generation," *Advanced Robotics*, vol. 30, no. 17-18, pp. 1164–1172, 2016.

[9] F. Kanehiro, M. Morisawa, W. Suleiman, K. Kaneko, and E. Yoshida, "Integrating geometric constraints into reactive leg motion generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 4069–4076.

[10] O. Kanoun, "Real-time prioritized kinematic control under inequality constraints for redundant manipulators," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[11] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 3733–3738.

[12] W. Suleiman, F. Kanehiro, and E. Yoshida, "Infeasibility-free inverse kinematics method," *IEEE/SICE International Symposium on System Integration (SII)*, pp. 307–312, 2015.

[13] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.