2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Saturation in the Null-Space (SNS) for Tele-operated Surgery: Prioritized Motion Control for RCM and Joint Limit Constraints

Sreekanth Kana[1], Antonia Perez Arias, Robert Kahlau, Pavan Kanajar[1], and Shashank Sharma[1]

***Abstract*— This paper showcases the application of the Saturation in the Null Space (SNS) algorithm to establish task prioritization and coordination within a tele-operated minimally invasive robotic surgical setting. In our work, SNS prioritizes achieving Remote Center of Motion (RCM) constraint, ensuring safe instrument manipulation, while respecting joint constraints for uninterrupted robot operation. This prioritization allows for accommodating the tracking of the surgeon's motion, within the capabilities defined by RCM and joint constraints. We investigate both the velocity and acceleration control variants of the SNS algorithm, incorporating bespoke adjustments to tailor the original algorithm to the intricate requirements of surgical applications. Through simulations and experiments, this work aims to demonstrate the effectiveness of SNS in enhancing the safety and controllability of tele-operated surgery, paving the way for its integration in various surgical procedures.**

I. INTRODUCTION

Robot-assisted minimally invasive surgery (RAMIS) has revolutionized modern healthcare, offering significant advantages over traditional open surgery. Utilizing robotic arms and high-definition visualization systems, RAMIS allows surgeons to perform intricate procedures through small incisions, leading to faster patient recovery, reduced pain and scarring, and minimized risk of infection. However, ensuring safe and accurate manipulation within the confined space of the surgical field necessitates strict adherence to a set of constraints, the most prominent of which is the Remote Center of Motion (RCM) constraint [1].

Fig. 1 illustrates a tele-operated robot-assisted minimally invasive surgery setting. The RCM constraint dictates that surgical instruments must pivot around a fixed point located at the entry point (typically a trocar) into the patient's body. This pivotal motion plays a critical role in safe operation by limiting the lateral movement at the incision site, which significantly decreases tissue damage.

Researchers have investigated diverse methods for implementing RCM in the setting of tele-operated robotic-assisted surgery. These approaches fall into two main categories: *programmable* and *mechanical RCM*. While mechanical RCM [2], [3] utilizes the physical design of the robotic system itself to achieve the necessary constraints, programmable RCM (which we focus in this work) relies on control algorithms and programming [1], [4], [5] to enforce the desired RCM behavior.

While the majority of literature on RAMIS primarily concentrates on the RCM constraint by itself, it is essential

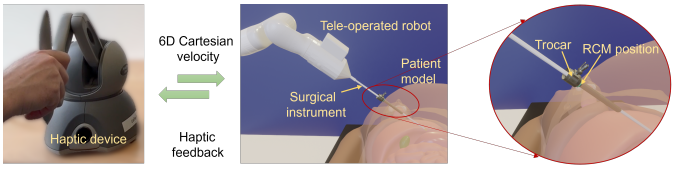


Fig. 1: Image depicting a tele-operated robotic surgical setup. A trocar is positioned at the incision site, serving as the reference point for the robot's Remote Center of Motion (RCM). Through this trocar, the surgical tool is inserted and maneuvered. The haptic device generates a velocity which is tracked by the robot, with optional haptic feedback available to the operator.

to ensure that the desired RCM behavior can be achieved within the robot's physical capabilities. Neglecting to observe joint constraints (position, velocity, and acceleration) during robotic-assisted surgery poses substantial risks. Exceeding these limits can result in physical damage to the robot's components, stemming from excessive stress and potential component failure. Moreover, reaching joint limits may trigger emergency stops, disrupting the surgical workflow, which also compromise the integrity of the RCM and generate undesirable motion at the tool-tip. Operating the joint within the limits of velocity and acceleration safeguards the robot against the rapid overshooting of velocity in the vicinity of a singularity. This is important in surgical settings, especially when articulated instruments introduce kinematic singularity. Oftentimes, in addition to the robot's inherent limitations, the inclusion of user-defined joint constraints is pivotal for specific surgical tasks and collision avoidance. Hence, ensuring strict adherence to joint constraints is imperative for upholding the safety, performance, and successful execution of robotic-assisted surgery.

Joint limit avoidance is a fundamental challenge in robotics and has been extensively researched. We focus more on the redundancy resolution based approaches, where, the additional degrees of freedom (redundancy) present in a robotic system is effectively used for satisfying joint constraints. A few of the popular approaches from this category include artificial-potential based [6], [7], SNS (Saturation in the Null-Space) [8], [9], [10], [11] and SJS (Saturation in the Joint Space) [12], [13]. While the joint constraints can be categorized as *hard* (constraints are enforced without flexibility) and *soft* (more flexible limits that allow some degree of deviation), for surgical applications, we prefer hard joint constraints.

We prefer not to employ a potential-field based approach

[1]Sreekanth Kana, Pavan Kanajar, and Shashank Sharma are affiliated with KARL STORZ VentureONE Germany GmbH
sreekanth.kana@ks-ventureone.com, pavan.kanajar@ks-ventureone.com, shashank.sharma@ks-ventureone.com

because of its susceptibility to problems such as local minima and oscillations. Although SJS guarantees the fulfillment of joint constraints, there's a risk that the lower priority tasks may not always be upheld, particularly when multiple joints hit the limits, leading to reduced null-space dimension. Our research on joint limit avoidance suggests that the existing SNS framework has promising potential for application in surgical robotics because it offers two key features aligned with the requirements of RAMIS:

a) Prioritization of tasks: SNS inherently supports prioritizing tasks, which is crucial for meeting the RAMIS objective of ensuring safety and performance. b) Strict enforcement of joint limits: SNS enforces strict constraints on joint positions, velocities, and accelerations, further contributing to the safety and control needed in surgical robotics. In addition, while joint saturation is done to comply to the joint constraints, the lower priority task is still maintained by using a scaling feature.

In this paper, our contribution lies in harnessing the SNS framework to achieve the following objectives:

- To the authors' knowledge, there has been no prior implementation of the SNS framework in minimally-invasive robot-assisted surgery. In this paper, we illustrate how it can be adapted to surgical contexts through suitable modifications.
- We devise a unified framework wherein the Remote Center of Motion constraint is incorporated by adhering to the joint constraints of the robotic system.
- We introduce modifications to the original algorithm [9], focusing specifically on the acceleration stage to address some of the practical limitations.

With this framework, we also aim to mitigate high-velocity motions at kinematic singularities by enforcing joint velocity and acceleration limits through SNS. However, singularity analysis is beyond the scope of this work.

The paper is structured as follows: In section II, the SNS velocity-level control with RCM constraints is presented. This is followed by III, where we discusses acceleration-level control implementation of RCM, also with our own modifications for better performance. Section IV outlines the experimental validation of our approach on a tele-operated Kinova robot. This is followed by conclusion and future works in Section V.

II. SNS VELOCITY CONTROL FOR MINIMALLY INVASIVE SURGERY

The Saturation in the Null-Space (SNS) algorithm is a redundancy resolution approach that enforces strict adherence to joint constraints. The underlying idea is as follows. At any instant, joint velocity limits are defined by a set of box-constraints [8] based on the instantaneous joint kinematics, effectively representing the proximity of each joint to its respective position and velocity limits. When the joints are expected to exceed those limits, the joints are saturated to their maximum values while utilizing any remaining degrees of freedom to maintain the desired task.

For a stack of k tasks where the priority increases from 1 to k, the SNS framework for multi-task for task priority can be expressed [8] as

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + (\boldsymbol{J}_k \bar{\boldsymbol{P}}_k)^{\#}(s_k \dot{\boldsymbol{x}}_k - \boldsymbol{J}_k \dot{\boldsymbol{q}}_{k-1}) + \tilde{\boldsymbol{P}}_k(\dot{\boldsymbol{q}}_{N,k}) \quad (1)$$

Here $\bar{\boldsymbol{P}}_k$ is called the auxiliary projection matrix and is given as

$$\bar{\boldsymbol{P}}_k = \left(\boldsymbol{I} - ((\boldsymbol{I} - \boldsymbol{W}_k)\boldsymbol{P}_{k-1})^{\#}\right)\boldsymbol{P}_{k-1}$$

where,

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - (\boldsymbol{J}_k \boldsymbol{P}_{k-1})^{\#}(\boldsymbol{J}_k \boldsymbol{P}_{k-1})$$

In this paper, we use the superscript $\#$ to denote the pseudo-inverse of a matrix. That is, $(\boldsymbol{J}_k \bar{\boldsymbol{P}}_k)^{\#}$ represents the pseudo-inverse of $(\boldsymbol{J}_k \bar{\boldsymbol{P}}_k)$.

The matrix $\boldsymbol{P}_k$ is a specialized projector that incorporates joint velocity saturation while maintaining the task adhering to the hierarchical priorities (refer [8] for more details) which can be obtained as

$$\tilde{\boldsymbol{P}}_k = (\boldsymbol{I} - (\boldsymbol{J}_k \bar{\boldsymbol{P}}_k)^{\#}\boldsymbol{J}_k)((\boldsymbol{I} - \boldsymbol{W}_k)\boldsymbol{P}_{k-1})^{\#}$$

with $\boldsymbol{P}_0 = \boldsymbol{I}$, $\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$.

In the above equations, for the k^{th} task, $\boldsymbol{W}_k$ is the saturation matrix for which $w_{j,j} = 0$ when joint j is saturated. When further saturation of joints not possible, the task is scaled down by factor s_k to stay within the joint limits. The task scaling factor is computed individually for all the joints such that the resultant scaled velocity is in agreement with the box constraints, and the minimum of these used as the common scaling factor for all the joints. Thus, these two parameters ensure that the task is performed within the joint limits.

A. SNS velocity-level control for Remote Center of Motion

In this section, we leverage SNS velocity-level control in the context of tele-operated robotic surgery by incorporating RCM constraints into the framework. With the velocity-level control we expect the system to operate within the joint position and velocity limits while satisfying a set of task velocities. For each of the task, we need to define the task velocity ($\dot{\boldsymbol{x}}_k$) with an associated Jacobian ($\boldsymbol{J}_k$), forming a task pair ($\dot{\boldsymbol{x}}_k, \boldsymbol{J}_k$). Consider the following tasks for a tele-operated surgery:

Task 1: Remote Center of Motion

The RCM implementation is based on the work outlined in [15]. For a surgical instrument inserted at the access port via a trocar positioned at $\boldsymbol{p}_{rcm}$, the desired velocity at the RCM can be defined as the output of a PD controller as

$$\dot{\boldsymbol{x}}_{rcm} = \boldsymbol{K}_p \begin{pmatrix} \widehat{\boldsymbol{i}}^T \\ \widehat{\boldsymbol{m}}^T \end{pmatrix} \boldsymbol{e}_{rcm} + \boldsymbol{K}_v \begin{pmatrix} \widehat{\boldsymbol{i}}^T \\ \widehat{\boldsymbol{m}}^T \end{pmatrix} \dot{\boldsymbol{e}}_{rcm}, \quad (2)$$

where, $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are diagonal matrices of proportional and derivative of gains. Here, $\boldsymbol{e}_{rcm}$ is the RCM error and $\widehat{\boldsymbol{i}}$ and $\widehat{\boldsymbol{m}}$ are the orthonormal vectors on the plane tangent to the access port.

The Jacobian at the RCM is computed in terms of $\boldsymbol{J}_{start}$ and $\boldsymbol{J}_{end}$, the Jacobians at the starting and ending points of the linear part of the articulated instrument as follows.

$$\boldsymbol{J}_{rcm} = \begin{pmatrix} \widehat{\boldsymbol{i}}^T \\ \widehat{\boldsymbol{m}}^T \end{pmatrix} (\lambda_{rcm}\boldsymbol{J_{end}} + (1 - \lambda_{rcm})\boldsymbol{J_{start}}) \quad (3)$$

Here, λ_{rcm} is the ratio of the instrument penetration (see [15] for more details).

Task 2: Cartesian velocity tracking

In a standard tele-operated robotic surgery setup, the surgeon guides the instrument using an input console, resulting in the task velocity being directly determined by sensors on the input console. Hence, the instrument tip-velocity is expressed as

$$\dot{\boldsymbol{x}}_{inst} = K_p' \, \dot{\boldsymbol{x}}_{inp} \quad (4)$$

where, $\dot{\boldsymbol{x}}_{inp}$ is the Cartesian velocity from the input console and $K_p^{'}$ is a proportional constant which can be adjusted to achieve the desired sensitivity.

Here, the task Jacobian is the Jacobian ($\boldsymbol{J}_{inst}$) at the instrument tip itself.

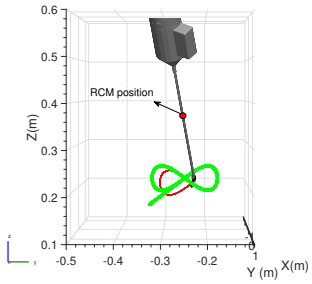
Therefore, from eq. 2 and eq. 3, the task pair for task 1 can be expressed as $(\dot{\boldsymbol{x}}_1, \boldsymbol{J}_1) = (\dot{\boldsymbol{x}}_{rcm}, \boldsymbol{J}_{rcm})$ and the same can be written for task 2 as $(\dot{\boldsymbol{x}}_2, \boldsymbol{J}_2) = (\dot{\boldsymbol{x}}_{inst}, \boldsymbol{J}_{inst})$ which are then applied in eq. (1) to obtain the commanded joint velocities. It is to be noted that for each of the above tasks, an associated saturation matrix and scaling factor will be assigned to ensure adherence to the joint constraints.

In our implementation[1], additional care was taken for detecting algorithmic singularities [10] and handled using the damped least-squares pseudoinverse approach is used to provide a stable behaviour [14]. However, we do not address this in detail as it is out of scope of this work.

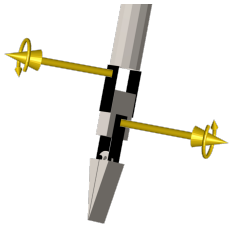
B. Simulation

In this section, we validate our approach via simulation on a 7DoF Kinova robot arm mounted with an articulated surgical instrument with 2DoF (see Fig. 2b). The surgical instrument features a pair of open and close grasper jaws at the end, which is not considered to be the part of the kinematic chain.

For the Cartesian task, we mathematically generate a 6D Cartesian velocity to emulate the haptic device. The velocity is derived from a pre-defined trajectory within Cartesian space (Fig. 2a). It is crucial to emphasize that the objective is not to replicate the trajectory itself. Instead, the focus lies on tracking the instantaneous Cartesian velocity. This decision is made to align with practical surgical scenarios where the trajectory is not predetermined, and the primary goal is to mimic the hand movements of the surgeon. Furthermore, the depicted boundaries in this simulation represent user-defined constraints on specific joints, serving to ensure that saturation occurs during the task execution. Note that that the timestep used here is T = 1ms.



(a) The curve in green shows the pre-defined path used to generate the 6D velocity. The actual path traced by the instrument is shown in red.



(b) Articulated instrument with 2 DoF.

Fig. 2: Illustration of the articulated instrument motion

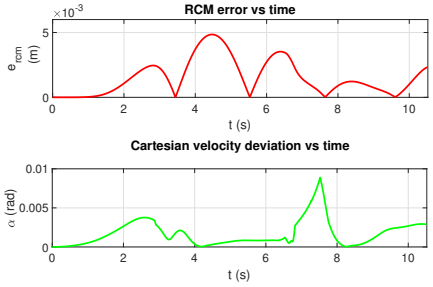


Fig. 3: Plots depicting the RCM error (i.e., the resultant displacement from the desired RCM position) and the deviation (in terms of the angle) of the computed Cartesian velocity vector from the commanded Cartesian velocity vector

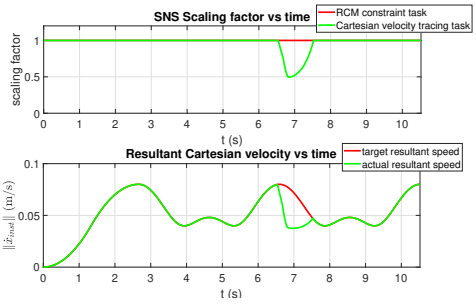


Fig. 4: The upper plot illustrates the fluctuation of the scaling factor for both the RCM and Cartesian tasks, while the lower plot visualizes the impact of the scaling factor on the resulting Cartesian velocity.

As the motion begins, the algorithm successfully achieves the desired Cartesian velocity (Fig. 4) while keeping low RCM error (Fig. 3). With reference to Fig. 5, it can be seen that between time 2.8 s and 4 s, joint 1 attains its position limit. However, as there are still enough degrees of freedom available to accomplish the task, the velocity does not get scaled. From 6 s to 8 s, joint 1 reaches a velocity limit, followed by a position limit, while joint 6 also encounters a velocity limit. At this stage, task scaling becomes necessary

[1]Our implementation of SNS framework is enriched by the insights provided by the implementation in *Park, A., & Smith, C. (2017, October 30). sns ik. [GitHub repository]. GitHub. Retrieved from https://github.com/RethinkRobotics-opensource/sns_ik.*

to maintain the overall task while ensuring all joints stay within their limits. This scaling adjustment is visible in Fig. 4, where the task is scaled down from time 6.5 s to 7.5 s, after which it recovers. This is also reflected in Fig. 2a, where the path traced by the instrument is deviating from the predefined path used to compute the velocity. This behavior is expected since we are tracking the instantaneous velocity but not the path. As to the task errors, the RCM error, depicted in Fig. 3, is influenced by the choice of the PD gains eq. (2). The PD gain values used here are heuristic, and further fine-tuning is expected to improve the RCM error. Additionally, we observe a small Cartesian task deviation (the angle between commanded and computed Cartesian velocity), which can be attributed to, a) the additional velocity introduced by the higher-priority task (i.e., RCM correction) and b) numerical errors during the mapping of joint velocities to Cartesian velocity.

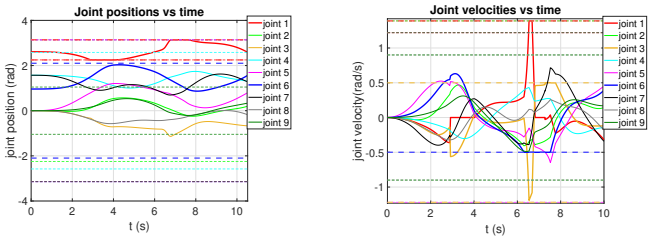


Fig. 5: Joint positions (on the left) and velocity (on the right) plots. The dotted lines represent the minimum and the maximum limits for each joint.

III. SNS ACCELERATION-LEVEL CONTROL

As discussed in [8], velocity-level implementation of the SNS algorithm, while effective in enforcing joint constraints, can introduce sub-optimal robot behavior. As the robot approaches joint limits, rapid changes in joint velocities may be required to maintain task execution. These abrupt velocity shifts create a jarring experience for the surgeon and can induce undesirable stress on the robot's actuators. Furthermore, the focus on velocities alone neglects the true dynamic capabilities of the robot. To address these limitations, we extend our approach leveraging acceleration-level SNS, allowing for intrinsically smoother control signals, complying to robot dynamics, and an overall robust surgical experience.

A. Box-constraints: practical limitations

The success of SNS-based approaches hinges critically on the quality of the box-constraints, which defines the maximum and minimum allowed acceleration at any point of time. The box-constraints for acceleration-level control, in their original form [8], [9] is as follows.

$$\ddot{Q}_{min,i} = \max\left\{\frac{2(Q_{min,i} - q_i - \dot{q}_i T)}{T^2}, -\frac{V_{max,i} + \dot{q}_i}{T}, -A_{max,i}\right\} \quad (5)$$

$$\ddot{Q}_{max,i} = \min\left\{\frac{2(Q_{max,i} - q_i - \dot{q}_i T)}{T^2}, -\frac{V_{max,i} - \dot{q}_i}{T}, A_{max,i}\right\} \quad (6)$$

Here, i ranges from 1 to n, where n represents the total number of joints. The variables $\ddot{Q}_{min,i}, \ddot{Q}_{max,i}$ denote the minimum and maximum allowed joint acceleration limits. Similarly, $Q_{min,i}, Q_{max,i}$ represent the minimum and maximum joint position limits. The current position and velocity for joint i is given by $q_i, \dot{q}_i$ respectively. Finally, $V_{max,i}, A_{max,i}$ correspond to the maximum allowed velocity and acceleration for joint i (here, the minimum and maximum limits for both velocity and acceleration limits are symmetrically defined). The term T here, is the sampling time.

However, these constraints entail certain limitation that could hinder the smooth tele-operation of the robot, particularly when the robots operate in the vicinity of the joint limits. We address these scenarios below.

Smooth tele-operation in surgical robotics necessitates fast computation cycles, typically within the order of $T <= 1$ms. As the robot's joint approaches its upper limit $Q_{\text{max},i}$, with a very small sampling time T, the term $\frac{2(Q_{\text{max},i} - q_i - \dot{q}iT)}{T^2}$ in eq. (6) becomes highly negative. Consequently, in eq. (6), $\ddot{Q}_{\text{max},i}$ is assigned the minimum value, yielding $\ddot{Q}_{\text{max},i} = \frac{2(Q_{\text{max},i} - q_i - \dot{q}iT)}{T^2}$.

Consider a scenario where $\ddot{Q}_{\text{max},i} < A_{\text{min},i}$. Should $\ddot{q}_i > \ddot{Q}_{\text{max},i}$, $\ddot{q}_i$ becomes saturated, meaning $\ddot{q}_i = \ddot{Q}_{\text{max},i}$. This implies that the commanded acceleration saturates to a value exceeding the maximum deceleration capability of the joint ($-A_{\text{max},i}$). A representative numerical example could entail $\ddot{Q}_{\text{max},i} = -15$ rad/s^2, $A_{\text{max},i} = -10$ rad/s^2, and $\ddot{q}_i = -5$ rad/s^2.

Now, we examine another scenario where $\ddot{Q}_{\text{max},i} < \ddot{Q}_{\text{min},i}$. If $\ddot{q}_i < \ddot{Q}_{\text{max},i}$, $\ddot{q}_i$ saturates to the minimum limit, i.e., $\ddot{q}_i = \ddot{Q}_{\text{min},i}$. However, this adjustment causes the joint to decelerate at a slower rate than required, potentially leading to the joint's motion exceeding the position constraint. An example scenario would be where $\ddot{Q}_{\text{max},i} = -15$ rad/s^2, $\ddot{Q}_{\text{min},i} = -10$ rad/s^2, and $\ddot{q}_i = -15$ rad/s^2.

Furthermore, it's noteworthy that when the robot operates in the vicinity of joint constraints and a software emergency stop is triggered, the system initiates rapid deceleration to swiftly halt the robot's movement, prioritizing safety. For surgical tasks, its crucial to ensure that the residual tool motion as well as the RCM motion is minimal during an emergency stopping. A common practice to achieve this is rapid deceleration of all joints with maximum (mechanically) possible values [16]. However, this bypasses the box-constraints, potentially resulting in joint limit violation. This necessitates, the need for tailored solutions to suit surgical applications.

B. Extended box-constraints

In order to address the aforementioned challenges, we propose an extension to the original box-constraints. We adopt a reverse approach, to determine a parameter, that we would like to call as *Deceleration Trigger Point* (DTP). The term DTP denotes a threshold joint position at which a joint, nearing its limit with maximum velocity, must employ maximum deceleration to come to a halt within the defined

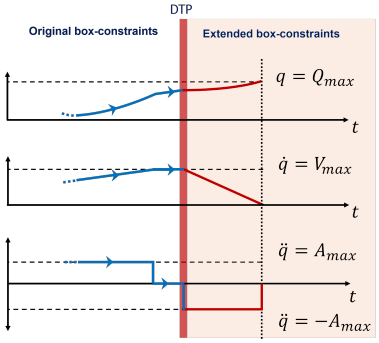


Fig. 6: Illustration of how the extended constraints modifies the kinematic profile of the joints to comply with the desired boundary conditions. Until the DTP is reached, the original box constraints dictate the acceleration limits. Once the joint surpasses the DTP, the extended constraints come into effect.

position boundaries. Here, we start at the desired boundary condition as work our way back to the the DTP. That is, for a single joint, with the initial position, $q = Q_{max}$, velocity, $u = 0$, maximum acceleration is iterative applied and the resulting trajectory profile is computed with the following update law.

$$u = u + A_{\text{max}} \cdot T$$
$$q = q - \Delta q,$$
$$\text{where, } \Delta q = u \cdot T + \frac{1}{2} \cdot A_{\text{max}} \cdot T^2$$

The motion is allowed to continue until the velocity (v) attains or surpasses V_{max}, the maximum allowed velocity (this ensures that the constraints work for lower velocities as well). The joint angle (q) corresponding to this instance is identified as the DTP (in practise, to ensure timely action triggering, it's pragmatic to set this point slightly ahead of the actual joint position ($q - V_{max}T$)). As an example, for a joint steadily approaching its position limit with maximum velocity, the kinematics follow the profile as shown in Fig. 6.

The extended constraints guarantee that commanded deceleration never surpasses the maximum joint acceleration. Additionally, upon crossing the DTP, the joints are forced to adhere to a kinematic profile, enabling them to come to a stop with maximum deceleration while remaining within the joint position limits.

With reference to Algorithm 1 (lines 17-23), when a joint approaches its maximum limit, if it continues to move towards the limit ($\dot{q} > 0$) and the user commands an acceleration ($\ddot{q} > 0$) that pushes it further towards the limit, the maximum acceleration is constrained by the expression $max(-\frac{\dot{q}}{T}, -A_{\text{max}})$. This applies maximum deceleration until the velocity $\dot{q}$ reaches zero. However, if the previous velocity is guiding the joint away from the limit (i.e., $\dot{q} < 0$), the user can apply a maximum positive acceleration of $min(-\frac{\dot{q}}{T}, A_{\text{max}})$ until the direction of motion changes.

Conversely, if the commanded acceleration moves the joint away from the maximum limit (i.e., $\ddot{q} < 0$), and the previous velocity guides the joint towards the limit (i.e., $\dot{q} > 0$), the maximum acceleration is limited to $-A_{\text{max}}$ until the term $-\frac{\dot{q}}{T}$ dominates as $\dot{q}$ becomes sufficiently small. If both the previous velocity and the commanded acceleration guide the robot away from the joint limit, no modifications are necessary for the computation of maximum acceleration limit.

C. Task infeasibility in SNS-acceleration control

When the SNS algorithm fails to find a solution, the task is considered infeasible and the scaling factor is set to zero. While a zero scaling factor in velocity-level SNS effectively stops the instrument tip motion (i.e., zero task velocity), the same action in SNS-acceleration-level control does not achieve a complete halt. Instead, the instrument maintains its previous velocity with zero acceleration. However, this presents an undesired scenario as it may lead to unintended motion of the surgical instrument. To address this issue, we trigger a software-based emergency stop, where the robot is stopped as quickly as possible with all the joints driven with maximum deceleration.

D. Scaling factor oscillation

Task scaling in SNS helps in keeping the task as more joints get saturated. However, at times, the scaled task acceleration exhibits oscillatory behavior due to the dynamic interplay between the constraints and the scaling mechanism within the SNS planner. For better understanding, consider the following scenario. Assume the robot is at a state where the scaling factor is 1, indicating no limitations on achieving the commanded acceleration. If the task is found to be infeasible (i.e., zero scaling factor), for instance, due to multiple joints meeting their velocity limits, the software-based emergency stopping gets triggered. Consequently, the robot's velocity drops to zero. But at this stage, with the robot now stationary, the possibility to attain the commanded acceleration is reinstated as the joints are now starting from rest. However, there is a strong likelihood that the commanded acceleration is once again infeasible to carry out after a few iterations. This cyclic process results in alternating phases of unrestricted and restricted velocity, manifesting as the observed oscillations in the resulting robot velocity.

To overcome this limitation, we implement a *look-ahead* approach, whereby, for a commanded task velocity, SNS is performed for a duration mT (where m is an integer chosen heuristically). During this window, we monitor the value of the scaling factor, to check if any oscillatory behavior is expected. If we foresee oscillations, the robot stays stationary, and the process is repeated until the planner encounter a task velocity that does not yield an oscillatory scaling behavior.

E. SNS acceleration control with RCM constraints

In SNS acceleration-level multi-task [9], the task priority control can be expressed as

$$\ddot{\boldsymbol{q}}_k = \ddot{\boldsymbol{q}}_{k-1} + (\boldsymbol{J}_k \bar{\boldsymbol{P}}_k)^{\#}(s_k \ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k - \boldsymbol{J}_k \ddot{\boldsymbol{q}}_{k-1}) + \tilde{\boldsymbol{P}}_k(\ddot{\boldsymbol{q}}_{N,k} - \ddot{\boldsymbol{q}}_{k-1}) \quad (7)$$

where, the $\bar{\boldsymbol{P}}_k, \boldsymbol{P}_k, \tilde{\boldsymbol{P}}_k$ follow the same definitions as given in section II and $\dot{\boldsymbol{J}}_k$ is the time derivative of the k^{th} task Jacobian.

Similar to the velocity-level control, the task pairs for the acceleration level control can be defined as $(\ddot{\boldsymbol{x}}_1, \boldsymbol{J}_1, \dot{\boldsymbol{J}}_1) = (\ddot{\boldsymbol{x}}_{rcm}, \boldsymbol{J}_{rcm}, \dot{\boldsymbol{J}}_{rcm})$ and $(\ddot{\boldsymbol{x}}_2, \boldsymbol{J}_2, \dot{\boldsymbol{J}}_2) = (\ddot{\boldsymbol{x}}_{inst}, \boldsymbol{J}_{inst}, \dot{\boldsymbol{J}}_{inst})$.

Here, the RCM task acceleration $\ddot{\boldsymbol{x}}_{rcm}$ is the time derivative of the velocity $\dot{\boldsymbol{x}}_{rcm}$ given by eq. (2) which also takes the form of an error-based PD controller. Similarly, $\ddot{\boldsymbol{x}}_{inst}$ is the time derivative of $\dot{\boldsymbol{x}}_{inst}$ from eq. (4).

Algorithm 1 Modified constraints for acceleration-level control

$c_{1,min} = \frac{2(Q_{min} - q - \dot{q}T)}{T^2}$
$c_{2,min} = -\frac{V_{max} + \dot{q}}{T}$
$c_{3,min} = -A_{max}$
$\dot{Q}_{min} = max\{c_{1,min}, c_{2,min}, c_{3,min}\}$
if $q \leq Q^{min}_{DTP}$ **then**
  if $\ddot{q} < 0$ **then**
    $\ddot{Q}_{min} = \begin{cases} min(-\frac{\dot{q}}{T}, A_{max}) & \text{if } \dot{q} \leq 0 \\ max(\frac{-\dot{q}}{T}, -A_{max})) & \text{if } \dot{q} > 0 \end{cases}$
  end if
  if $\ddot{q} > 0$ **then**
    $\ddot{Q}_{min} = \{min(-\frac{\dot{q}}{T}, A_{max}) \quad \text{if } \dot{q} \leq 0$
  end if
end if
$c_{1,max} = \frac{2(Q_{max} - q - \dot{q}T)}{T^2}$
$c_{2,max} = \frac{V_{max} - \dot{q}}{T}$
$c_{3,max} = A_{max}$
$\dot{Q}_{max} = min\{c_{1,max}, c_{2,max}, c_{3,max}\}$
if $q \geq Q^{max}_{DTP}$ **then**
  if $\ddot{q} > 0$ **then**
    $\ddot{Q}_{max} = \begin{cases} max(-\frac{\dot{q}}{T}, -A_{max}) & \text{if } \dot{q} \geq 0 \\ min(-\frac{\dot{q}}{T}, A_{max}) & \text{if } \dot{q} < 0 \end{cases}$
  end if
  if $\ddot{q} < 0$ **then**
    $\ddot{Q}_{max} = \{max(-\frac{\dot{q}}{T}, -A_{max}) \quad \text{if } \dot{q} \geq 0$
  end if
end if

F. Simulation

For acceleration-level simulation, we generate the commanded acceleration as the time differential of the difference in the target and the actual Cartesian velocity. The timestep used for simulation is T = 1ms.

With reference to Fig. 7, sharp spikes in the acceleration, (but confined to the limits), can be observed, at 2.2 s and 7.8 s. It can be inferred from the joint velocity plot that this abrupt acceleration change occurs as velocity of one of the joints (in this case, joint 1) drastically drops due to saturation (enforced by the position limit). At this instant, the velocity of the remaining joints undergo sudden change to compensate for the saturated joint.

Upon further analysis, it is observed that at 2.2 s and 7.8 s, the joint accelerations again hit their limits. This behavior is triggered by the software-emergency braking caused by an infeasible target acceleration. That is, at these instants, the SNS is not in control of the motion, instead all joints are commanded with maximum deceleration. The impact of the emergency stop is also evident in the task errors, as depicted in Fig. 8b, where both the RCM error and velocity deviation experience overshoots. Even though the peaks in the actual velocity results in huge computed target acceleration (Fig. 8a), the target acceleration is irrelevant during these moments as actual acceleration is solely dictated by the maximum deceleration of joint values.

It is important to highlight that despite the activation of emergency braking, the joints still operate within their designated position, velocity, and acceleration limitations. This controlled response is ensured by the kinematic profile set by the extended box-constraints before emergency stop was initiated. Furthermore, this emergency braking differs from a physical emergency button press. Since the braking is triggered by the software, there is no need for a physical hardware reset. This allows the robot to resume operation much quicker. As soon as a valid (achievable) motion command is issued, the robot can resume its motion.

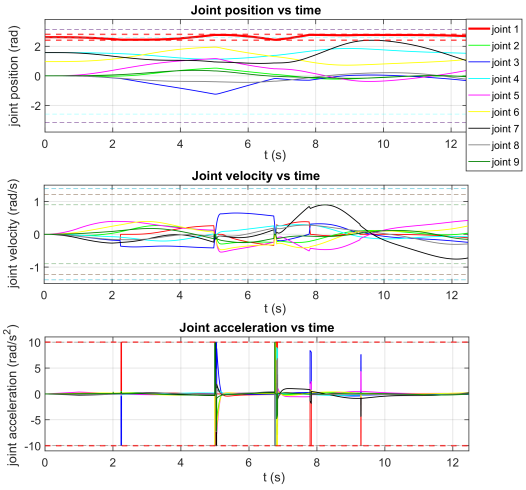
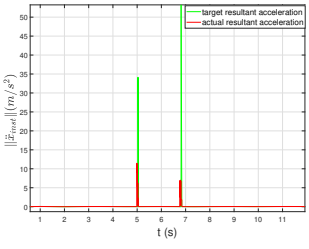
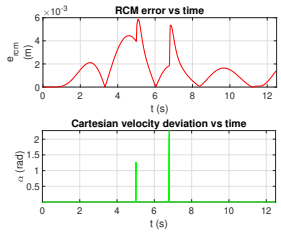


Fig. 7: Joint position, velocity, and acceleration profile with dotted lines representing the maximum and the minimum limits



(a) Target and actual resultant Cartesian acceleration error



(b) RCM error and the Cartesian velocity deviation

Fig. 8: Cartesian acceleration profile together with the primary and secondary task errors

IV. EXPERIMENT AND ANALYSIS

In this section, we perform experimental validation of the SNS-velocity control based RCM on a real hardware under a tele-operated setting.

A. Experimental setup

For our experiment, a 12th Gen Intel Core i7-12700, 4.5 GHz processor Preempt RT Linux based system is used to execute our C++ implementation of the framework. The hardware setup comprises of the Kinova Gen3 6DoF robot arm with the end effector interfaced with a 3D printed housing containing a representative instrument rod. Additionally, the robot is equipped with an EtherCAT communication interface, enabling real-time updates of joint position commands at 1 ms. For tele-operation, we utilize a Phantom Omni haptic device as an input device (see Fig. 9), providing cartesian linear velocities every 1 ms. The setup ensures tele-operation input is supplied to the SNS velocity planner in real-time, allowing for the computation and control of desired joint positions on the robot to accurately adhere to RCM point constraint during RAMIS procedures. The SNS velocity algorithm with RCM task achieves an average runtime of 0.1ms on the target hardware, satisfying the real-time requirement for motion control.

Even though the SNS-velocity algorithm outputs joint velocity values, the motion commands to the robot is joint position input (see Fig. 10) as the hardware supports only position inputs.

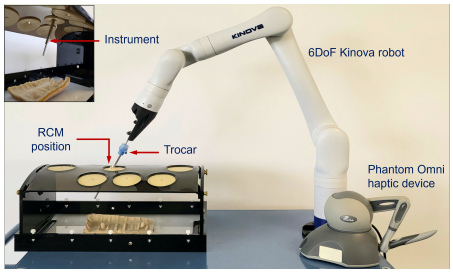


Fig. 9: Experimental setup for the RCM-constrained tele-operation of a Kinova robot with a Phantom Omni haptic device. The inset image on the top-left corner represents the view from the inside of the human body.

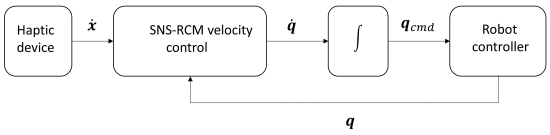
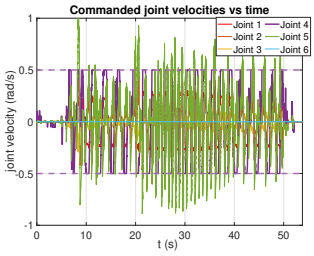


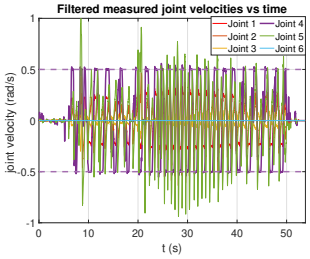
Fig. 10: 3D Cartesian velocity $\dot{x}$ from the haptic device serve as input to the SNS planner. The planner computes the desired joint velocity $\dot{q}$, which is then integrated into the position command sent to the robot.

Note that for the tele-operation on the real hardware, we only consider tracking the translational motion of the haptic device due to the absence of an articulated instrument. To validate that our algorithm respects the joint position limits, we implemented preemptive hard limits on just one joint, encompassing both velocity and position thresholds. For example, we set a joint position limit of +/- 0.1 rad for joint 1 and joint velocity limits of +/-0.5 rad/s for joint 4. This strategy was adopted to safeguard sufficient degrees of freedom for accomplishing all tasks. As a joint nears its limit, the algorithm explores alternate solutions by saturating the affected joints. Conversely, if the desired solution remains within the hard limits, the algorithm reactivates any previously disabled joint.

As the user guides the haptic device, the robot attempts to track the desired Cartesian velocity while simultaneously maintaining adherence to the RCM constraints as dictated by the SNS velocity planner. Referring to Fig. 11a, we observe that starting at 6.4 seconds, the commanded velocity for joint 4 frequently approaches the user-defined limit without ever exceeding it. This observation is further confirmed by the measured velocity (filtered) data from the robot in Fig. 11b, which remains consistently within the anticipated range. Slight deviations (within an acceptable tolerance) above the defined velocity limit can be observed in a position-controlled robot due to factors like integration errors, and control loop tuning. While position control prioritizes reaching target positions, the internal control system calculations might cause short-lived instances where the velocity transiently exceeds the intended limit.



(a) Commanded joint velocity for the robot (i.e. output of the SNS velocity algorithm) Here, robot joint 4 frequently hits the user-defined velocity limit.



(b) Measured joint velocities (filtered) from the robot sensors. Here, joint 4 hits the limits as expected. The slight overshoot is inferred to be caused by the position controller.

Fig. 11: Commanded and the measured joint velocity plots. The maximum velocity for the joints are [0.75, 0.75, 0.75, 0.5, 1.0, 1.0] rad/s, with the corresponding minimum velocities being the negative of these values. Note that the joint limits are displayed only for saturated joints to enhance visual legibility.

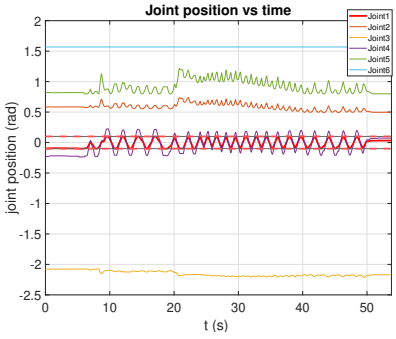


Fig. 12: Measured joint position from the robot encoders. The upper limits for the position are [0.1, 2.24, 2.57, 3.14, 2.09, 3.14] rad, with the corresponding minimum positions being the negative of these values. Here, it can be seen that the joint 1 frequently encounters its limit during the task. The Joint limits are displayed only for saturated joints to enhance visual legibility.

Fig. 12 depicts the measured positions of the robot's joints. While the data reveals instances where joint 1 reaches its user-defined position limit, it is to be noted that these occurrences did not impede the tele-operation task during the

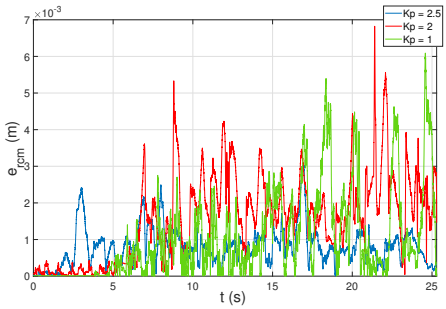


Fig. 13: RCM error comparison for different gain values

experiment. Similarly, by analysing the RCM error in Fig. 13, the low error (corresponding to gain K_p = 2.5) demonstrates the ability of our framework to maintain the RCM while confining the joints to their constraints. That is, the robot is able to operate even at the joint limits as long as the task is feasible. This feature carries significant practical implications, as it eliminates the interruptions, caused by joint limit violations commonly encountered in most hardware setups.

Furthermore, in Fig. 13, we present the impact of different gain (K_p) values on the measured RCM error. Three different K_p values were chosen heuristically for analysis and it is shown that with increasing gains, there is a reduction in the RCM error. Therefore, fine-tuning of the gain is required to achieve an optimal RCM error.

V. CONCLUSIONS AND FUTURE WORK

This work explores the application of Saturation in the Null Space (SNS) for tele-operated minimally invasive surgery. We propose a novel framework that leverages SNS to seamlessly integrates the Remote Center of Motion (RCM) task within the robot's joint position, velocity, and acceleration constraints. Additionally, we proposed the idea of the *extended box-constraints* and the *look-ahead* approach for SNS to address some of the practical implementation challenges.

We performed simulation and experimental validation, which confirmed the effectiveness of the proposed SNS framework. Simulations with both SNS-velocity and acceleration-level control demonstrated successful adherence to the RCM constraint without violating joint limitations during various Cartesian space tasks. Hardware experiments involving a Kinova robot controlled by a Phantom Omni haptic device further corroborated these findings. Experiments have been performed on a hard real-time system with 1 ms cycle time. Consequently, a smooth tele-operation was achieved while maintaining all constraints throughout the tasks with the SNS velocity-level control.

Our work presents a unified framework for tele-operated minimally invasive surgery, guaranteeing adherence to surgical task constraints while respecting hardware limitations. This approach ensures seamless operation even during scenarios where the robot operates at the edge of its joint capabilities.

In our future work, we intend to prioritize hardware experimentation, focusing on incorporating acceleration-level control alongside an active tool (offering independent degrees of freedom). In addition, we intend to fine-tune the PD gains associated with the RCM constraints to ensure optimum trade-off between the RCM error and the interference of RCM task onto the lower-priority task.

ACKNOWLEDGMENT

We would like to express our gratitude to Ms. Jule Bender for her support in documenting and simulating the SNS acceleration-level algorithm.

REFERENCES

[1] Aghakhani, N., Geravand, M., Shahriari, N., Vendittelli, M., Oriolo, G. (2013, May). Task control with remote center of motion constraint for minimally invasive robotic surgery. In 2013 IEEE international conference on robotics and automation (pp. 5807-5812). IEEE.

[2] Zhou, X., Zhang, H., Feng, M., Zhao, J., Fu, Y. (2018). New remote centre of motion mechanism for robot-assisted minimally invasive surgery. Biomedical engineering online, 17(1), 1-16.

[3] Li, J., Xing, Y., Liang, K., Wang, S. (2015). Kinematic design of a novel spatial remote center-of-motion mechanism for minimally invasive surgical robot. Journal of Medical Devices, 9(1), 011003.

[4] Pham, C. D., Coutinho, F., Leite, A. C., Lizarralde, F., From, P. J., Johansson, R. (2015, September). Analysis of a moving remote center of motion for robotics-assisted minimally invasive surgery. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1440-1446). IEEE.

[5] Azimian, H., Patel, R. V., Naish, M. D. (2010, September). On constrained manipulation in robotics-assisted minimally invasive surgery. In 2010 3rd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (pp. 650-655). IEEE.

[6] Mansard, N., Khatib, O., Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks. IEEE Transactions on Robotics, 25(3), 670-685.

[7] Foresi, G., Freddi, A., Kyrki, V., Monteriu, A., Muthusamy, R., Ortenzi, D., Pagnotta, D. P. (2017). An avoidance control strategy for joint-position limits of dual-arm robots. IFAC-PapersOnLine, 50(1), 1056-1061.

[8] Flacco, F., De Luca, A., Khatib, O. (2012, May). Motion control of redundant robots under joint constraints: Saturation in the null space. In 2012 IEEE International Conference on Robotics and Automation (pp. 285-292). IEEE.

[9] Flacco, F., De Luca, A., Khatib, O. (2012, October). Prioritized multitask motion control of redundant robots under hard joint constraints. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3970-3977). IEEE.

[10] Flacco, F., De Luca, A., Khatib, O. (2015). Control of redundant robots under hard joint constraints: Saturation in the null space. IEEE Transactions on Robotics, 31(3), 637-654.

[11] Fiore, M. D., Meli, G., Ziese, A., Siciliano, B., Natale, C. (2023). A General Framework for Hierarchical Redundancy Resolution Under Arbitrary Constraints. IEEE Transactions on Robotics.

[12] Muñoz Osorio, J. D., Fiore, M. D., Allmendinger, F. (2018, August). Operational space formulation under joint constraints. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 51814, p. V05BT07A022). American Society of Mechanical Engineers.

[13] Osorio, J. D. M., Allmendinger, F., Fiore, M. D., Zimmermann, U. E., Ortmaier, T. (2019, December). Physical human-robot interaction under joint and cartesian constraints. In 2019 19th International Conference on Advanced Robotics (ICAR) (pp. 185-191). IEEE.

[14] Siciliano, B., Khatib, O. & Kröger, T. (2008). Springer Handbook of Robotics. *Springer*. pp. 252-25.

[15] Azimian, H., Patel, R. V., Naish, M. D. (2010, September). On constrained manipulation in robotics-assisted minimally invasive surgery. In 2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (pp. 650-655). IEEE.

[16] Kröger, T. (2011, May). Opening the door to new sensor-based robot applications—The Reflexxes Motion Libraries. In 2011 IEEE International Conference on Robotics and Automation (pp. 1-4). IEEE.