Safe and Efficient Robot Control

Sandesh Rajendra Jain
Electrical and Computer Engineering Department
Virginia Polytechnic and State University
Blacksburg, USA

***Abstract*—In the modern-day and age of automation, a majority of the workforce is aided in some way or the other by Robotic Systems. The gap between employing humans for a cognitive job and getting the monotonous and sometimes even dangerous tasks done autonomously necessitates the use of cooperative and safe robotic systems. However, the functioning of such systems can be critical, especially when they work in a team consisting of humans and other machinery, where they have an overlapping workspace. Safety properties for such hybrid systems can be proved by the modeling and verification tools proposed in [3]. In this paper, we would be referring to the KeymaeraX theorem prover [10] but the higher-level instructions can be easily implemented on other modeling software as well. We discuss a general kinematic analysis approach for multi-axis robots and the construction of a provably safe controller model. Another issue that is addressed is the use of transcendental functions in the formal proof methods. These functions cannot be easily used in logical proof methods due to their undecidable arithmetic. Finally, an extended study is provided for Task Planning for single or multi-robot systems as well an added sensory processing unit has also been included to allow for flexibility for Human-robot interaction. Motivated by the aforementioned objectives, we list out the methods of constructing a provably safe and efficient time-triggered control design with the flexibility of using non-polynomial functions for popular configurations deployed by the industries.**

***Index Terms*—Robot Control, Hybrid Systems, Safety-centered, KeymaeraX, Cooperative Robots.**

I. Introduction

The properties of employing an industrial robot have become an attractive choice, especially due to features that enable them to work alongside their human counterparts [11]. As much of the human-controlled units get modified to fully/ partially autonomous systems, at times, they are forced to work with mechanical and safety constraints. If not in parallel production, such an autonomous robotic system could cause serious damage to the production rate in sequential production plants if it damages itself or other moving parts. One possible solution is to schedule the movement to be completely synchronized so that no static or dynamic parts ever collide with each other [1]. This seemingly straightforward solution is difficult to implement and shall increase the computational cost to simulate and rule out every possible unsafe position that the robot might find itself in after some large amount of time.

The workspace of mobile robots can be expanded further to avoid any intersections with other units, however, this is not always possible because some of these systems are intrinsically built to collaboratively carry out a task thus having an approachable workspace turns out to be one of the constraints to be accounted for while designing them [6]. This may not always be an option due to several scenarios such as lack of additional space, or the requirement for the robot to act in proximity to the assembly line and co-robots [1]. Additionally, the Human-Robot interaction has become an increasingly popular field, a number of well-established companies have already made significant progress in the field e.g., Kuka Robotics Corporation's LBR iiwa Robot is targeted toward Human-Robot Collaboration. Hence, the conservative approach of securing the workspace perimeter of the robot will not be useful in such scenarios.

The use of a simpler time-triggered control mechanism would be feasible and can incorporate delays in the sensing involved. The robot geometry could be analyzed with the rotation and translation matrices produced by the revolute and prismatic joints respectively. The actuator control can be decided based on the target position to be reached. Now, different tasks have a demand for different degrees of freedom (DOF) to be present. While a DOF of six can have three rotational and 3 translational, a few times redundant links are required to increase the flexibility of the machine. Kinematic analysis of such systems can be done in a modular fashion through the use of Rotation and Translation matrices. However, for the purpose of proof, we shall derive the angular positions with respect to initial states.

With predefined link-lengths and angular constraints, one can perform either the forward or inverse kinematics analysis. The former is required when the machine parameters such as angles for rotational and displacement for translational joints are given and coordinates of the end-effector are to be derived and vice-versa for the latter. For the purpose of safe control, we apply inverse kinematics for the robot resulting in the requisite angular positions. Now, we may generalize the proof rules, with arbitrary acceleration and deceleration values so that specific cases are trivially true. In this paper, we state the safety and efficiency rules for general configurations of multi-axis robots and two control mechanisms depending on the type of job assigned to the robot. We may accommodate initial angular conditions and specify a few rigid constraints to followed at all times so that the geometry of the robot remains intact and no arbitrary control takes place.

To make the control decisions applicable to higher-dimensional robot mechanisms than the one discussed, a modular approach is considered i.e., given the geometric analysis of robot parameters leading to the target location, the control decisions given in the paper can be adopted and modified to encompass additional blocks for more links. Needless to say, the addition of links does affect the geometric consistency of the robot and should be taken care of in the initial conditions, model predictive control, and the domain constraints. Moving along, the forward kinematics should conform to the coordinate estimates of the inverse kinematics-based control. This may not be critical if the sensory units are functional, however, in the case of longer-term robots, such forward analysis will act as an additional safety unit.

As discussed above, the use of forward or inverse kinematic analysis usually results in the use of trigonometric functions which are transcendental in nature. In formal proof methods of such hybrid systems, trigonometric functions cannot be a part of the proof rules because of the undecidability introduced due to the same. This can be handled by the use of Taylor Series Approximation. Trigonometric functions can be expressed as an infinite series of polynomials that can be used for control purposes with a certain tolerance capacity defined by the number of terms used in the approximate series representing the original function [7]. The upper bound on such tolerance can be computed given the number of terms and the range of input values. For trigonometric functions, the approximation is very close to the original function within the zero-centered first period. The generalized error can be stated as per Lagrange's Formula:

$$R_n \leq \frac{M(x-a)^{n+1}}{(n+1)!} \tag{1}$$

Where R_n is the maximum error given the bounds between a and x, M is the maximum value of the original function in the same bounds, and n is the highest polynomial power of x in the approximated function. Using this formula as a well-established axiom concrete bounds can be established on the tolerance with which the function can be expressed. Now, other safety properties for control can be proved based upon the current axiom. For the present scenario, acceleration-based linear control is employed for the purpose of proof, however, PID controllers can be used to achieve the objective as well.

On a higher level, we can classify joint control to be either parallel or sequential in nature. The applications to better understand the two classes are continuous and point-to-point end-effector motion with robotic spray-painting being an example of the former [11]. The point-to-point motion is usually used for pick-and-place operation and is much easier to implement since the controlled actions are independently taken one at a time. While implementing parallel control, one should note that in this case not only the relative angles and distances between the joints should be considered but also the relative motion between the same. The reason for this is; When multiple links are actuated simultaneously then inter-link collision might occur. Hence, the domain constraints should ensure reachable geometric states at all times.

This can be shown by the following example: Let there be a $2R$ robot (i.e., with two revolute joints) with the geometry as $(l_1, \theta_1, l_2, \theta_2)$ where l_1 and l_2 are the link-lengths and θ_1 and θ_2 are the relative angular positions of the links from the base and the joint between the links respectively. Now, under sequential control, given the final target location for the end-effector, one of l_1 or l_2 shall move from their current angular positions θ_1 or θ_2 and then the other one moves. Since at any time during operation, only a single link is mobile, there is no need of considering the relative motion. However, in the case of parallel control if $theta_1 = -10$ and $theta_2 = 170$ then moving both of them anti-clockwise simultaneously might result in link collision as depicted below. In this paper, control guards for such configurations have been discussed.

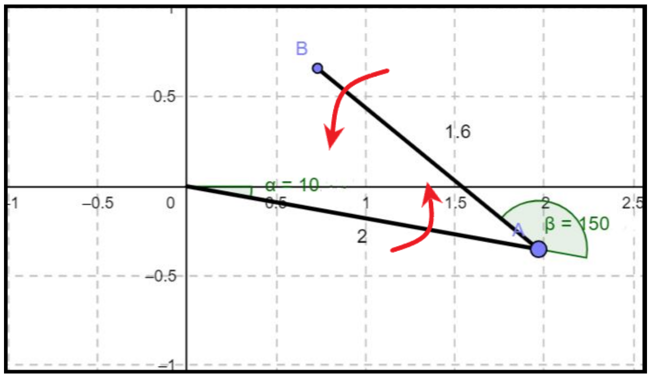

Fig. 1: Parallel Control for a 2R Robot

In addition to the predefined geometric parameters, we can also safeguard human-robot interactions by transforming the human posture to skeleton-based models wherein key-points [5] of the human body are connected together to downsized from the higher-dimensional visual inputs to geometric figures with certain boundary surfaces representing critical 'hitboxes' where the robot should not function at high velocities [8]. Any time-triggered controller relies upon how well the future states can be predicted given a sensing delay of T. However, a flexible controller that can function alongside humans should also be conservative enough to avoid entering unsafe states [11]. Thus modeling human behavior is the bottle-neck problem of such systems.

Safety is one of the aspects that we have been looking at for Robotic control design. Efficiency is another aspect that is significant to the development of a good design. We must ensure, that the robot is not constrained to the extent that its actuators remain static all the time. Such a conservative model is not useful [1], also, the controller's actions must be exhaustive in nature, we do not wish the system to enter dead-ends from where no actions are possible and safety rules may become trivially true. In addition to actuator-level controller efficiency, task-level efficiency can also be used as a performance metric. For this purpose, state-space search algorithms can be applied and according to the nature of growth of the state-space, one needs to consider the optimality-speed trade-off. Planning algorithms can also be used for this purpose such as Forward/Backward State-space search algorithms [2] that have been applied to block-world problems in the past. Similar architecture could be used to improve the planning abilities of robots that are expected to perform cognitive tasks.

The rest of the paper is organized as follows. Section II presents the motivation. The properties and problem formulation are discussed in Section III. The proposed proof construction for safe and efficient robot control is discussed in IV. Finally, Section V concludes the paper.

II. Motivation

Multiple industries ranging from Pharmaceutical to Automobile to Construction-related fields are transitioning from initial human-controlled systems to fully autonomous systems. Such transitions usually have a mid-level zone where humans and robots collaborate together to complete the task. The systems in this section can be termed as $Semi - AutonomousSystems$ [11]. This is where the issue of safety is placed as the highest priority and in the majority of the cases, there is a trade-off between having a highly conservative model and a flexible design allowing cooperative functioning. These systems are required to be rigorously tested for their safety and efficiency simultaneously [1]. Additionally, the field of Multi-agent autonomous systems is turning popular as individual robotic systems become flexible.

Previous work in the field has been done mainly at a higher level wherein the $Planning$ section of the robot is exploited to develop trajectories that ensure safety. However, an actuator-level control should be analyzed as well to ensure the overall safety and efficiency of the system [6]. There is a need to blend both of these requirements so as to incorporate maximum flexibility and safety during the operation [4]. E.g., actuator control can ensure safe actions for its domain of states by time-triggered model-predictive control and flexibility can be achieved for HIR by using pose detection algorithms like [5] whose key-point coordinates can then be fed to the actuator-level control. Other innovations can be employed to improve task planning as well without any hindrance to the aforementioned example system. This is shown in the block diagram below.

III. Problem Formulation

Now, we may formulate the given problem of actuator control by a generalized representation of the robot parameters, and performing kinematic analysis [9] for the same. We consider a Robot Mechanism consisting of N links with lengths $\{l_1, l_2, ..., l_N\}$ and N joints. The coordinate pairs for the joints are $(x_i, y_i, z_i)_{i \in \{0,1,2,...,N\}}$. The joint-angle offsets are α_i, translations $d_i + 1$, and relative angular rotation $\theta_i + 1$. While a number of different joints can be used, for simplicity, we consider two kinds of joints: Revolute (Rotational Effect) and Prismatic (Translational Effect). The table given below for the 2-axis robot given in Fig. 1 representation is also known as the D-H parameter table:

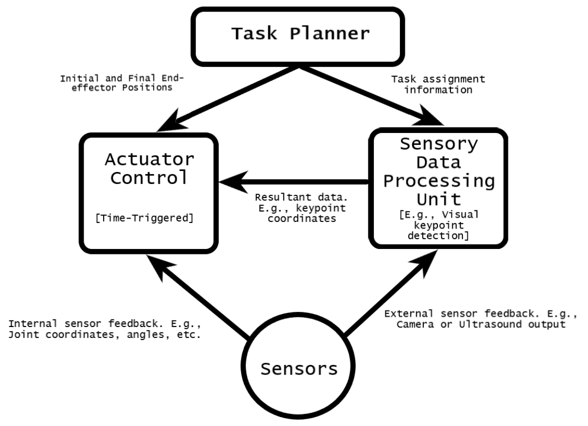
Task Planner
Initial and Final End-effector Positions
Task assignment information
Actuator Control
[Time-Triggered]
Resultant data. E.g., keypoint coordinates
Sensory Data Processing Unit
[E.g., Visual keypoint detection]
Internal sensor feedback. E.g., Joint coordinates, angles, etc.
External sensor feedback. E.g., Camera or Ultrasound output
Sensors

Fig. 2: High-level block-diagram for HRI and cooperative robots

l_i	$d_i + 1$	α_{i+1}	θ_{i+1}
0	0	0	-10°
l_1	0	0	-170°
l_2	0	0	0

The above parameters, should conform with the coordinates defined earlier to preserve the geometry of the system. The link length preservation invariant can be given as:

$$l_{i+1} = \sqrt{(x_{i+1} - x_i)^2) + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \quad (2)$$

We can observe that for a revolute joint between the i_{th} and $i + 1_{th}$ link, the above equation can act as the model invariant for controlling the joint angle and establishing at all times that the path is always circular. Now, given the set of initial geometric constraints and the type of joints being used, we can proceed to express the end-effector coordinates in terms of the angles θ_{i+1}. This process is also known as forward kinematic analysis. One can construct the homogeneous transformation matrix by combining the rotation matrices and the translation matrices to a single transformation matrix and multiplying a series of such matrices with each other in the order of rotations and translations involved. The component-wise representation is given below:

$$[T_i] = \begin{bmatrix} R_i & t_i \\ & s_i \end{bmatrix} \quad (3)$$

$$(R_i) = \begin{pmatrix} R_{xx} & R_{yx} & R_{zx} \\ R_{xy} & R_{yy} & R_{zy} \\ R_{xz} & R_{xy} & R_{zy} \end{pmatrix} \quad (4)$$

Here, R_i represents the rotation of the i_{th} link, t is the displacement matrix which is a column vector of size 3x1 and shows the displacement for the initial (x_i^1, y_i^1, z_i^1) coordinates to final (x_i^2, y_i^2, z_i^2) through prismatic joints. And, s is the scale that is usually set to 1. Inside the rotation matrix, R_{yx} is the projection of y axis after rotation onto the x axis before rotation, the rest of the combination follow the same rule but with a different axial combination.

Now, the final position of the end-effector or any other joint can be derived by post-multiplying or pre-multiplying the transformation matrix with the original coordinate matrix. The Transformation matrix has 12 entries with 3 of them being the displacement and 9 rotation-based entries. Additionally, given a rotation matrix, the internal entries are dependent only on the trigonometric combinations of 3 distinct variables at maximum. In all, we have 6 variables in the matrix that can be used for forward kinematics. However, the inverse of this transformation matrix also gives us the equations to perform inverse kinematics from which the requisite angles and displacement can be derived [9]. For the aforementioned example in Fig. 1., the resultant final angular positions are given below:

$$\beta = \frac{x_2^2 + y_2^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{5}$$

$$\alpha = \arctan\frac{y_2}{x_2} + \arctan\frac{l_2 \sin\beta}{l_1 + l_2\cos\beta} \tag{6}$$

Similarly, through the transformation matrices, we can find inverse relations of the coordinates with the robot parameters. Moreover, we have the choice of using the controller itself to find these parameters which can be included in the proof rule. This can be done by decomposing the inverse trigonometric functions into their respective polynomial series by Taylor's function approximation. Note that when we use only a few terms of the series, we have a monotonically increasing error [7] in the given range of inputs to the series. Moreover, for different ranges of inputs, we might need to change the approximation series accordingly. Once, we form well-defined axioms for these hurdles, we can use them in the proof rules for safe control. Given all the requisite preconditions and the model, we make a choice to either accelerate or decelerate at a predefined rate. The choice of a safe angular acceleration for uni-directional motion of the i_{th} link can be represented as given below:

$$(?Q_A)a_i := A_i \cup a_i := -B_i \tag{7}$$

This brings us to the conditions required to take an action. These choices of angular accelerations are predefined corresponding to a very general case, however, during the application, we may have a range of acceleration to be used which is a subset of the original model and has trivially true safety properties as well. In the above equation, we need to ensure the signs of A and B are the same which will impact the direction of motion of the link.

IV. Methods

The methods section is arranged as follows: Part A describes how the proposed model-predictive control can be formulated for proof. Part B shows methods to improve the flexibility of a robot by introducing external sensory processing. Lastly, Part C throws some light on more higher-level task planning that can be used to build cognitive robots that can find solutions given a descriptive state-space.

A. Hybrid Program for Controller Design

The Hybrid Program for a given system can be seen as a transitioning edge from a set of preconditions to a set of postconditions that needs to be true. This can be shown as $Preconditions \implies [HybridProgram]Postconditions$ [10]. To prove the safety and efficiency of a robotic system, we can describe each of the components from this equation. The time-triggered controller is a method of control relying on model predictions after some time T in the future, this time is the sensor and processing delay that occur during the hardware implementations of the model. We will employ this method for the hybrid program.

1) Preconditions: The preconditions in our case directly correspond to the initial Robot position (home position) and the geometric constraints. The acceptable range of initial states would require us to ensure that no i_{th} link is at an angle of more than 180° with respect to the $i - 1_{th}$ link. All the link lengths must be greater than zero. The end-point of each link must follow a circular trajectory for a revolute joint and linear for a prismatic joint. Additional constraints like positive delay, acceleration, and deceleration values are also added to the constraints. Finally, there must be constraints on the motion and reachability of the destination point. This means that the initial linear and angular velocities must not be so high that the deceleration is not high enough to stop the end-effector from ramming into the object. For an anti-clockwise rotational motion this condition can be represented as below, a similar equation can be constructed for linear velocity for prismatic joints by replacing the required parameters. :

$$\omega_i^2 \geq 2B(\theta_i^{final} - \theta_i^{init}) \tag{8}$$

2) Postconditions: Now, the postconditions are fairly simple and can be broken down into components for each link. To ensure, safe as well as efficient control, we can design the Hybrid program to have the safety-efficiency conditions for a single revolute joint as:

$$\theta_i^{init} = \theta_i^{final} \implies \omega_i = 0 \;\;\&\;\; \theta_i^{init} \neq \theta_i^{final} \implies \omega_i \neq 0 \tag{9}$$

Conjunction of such conditions for all the N components will give us the resultant postcondition for the system. Here, θ_i^{init} is the initial angular position of the i_{th} link and ω_i is the angular velocity of that particular link. The LHS of the Boolean equation represents safe operation i.e., when the end-effector reaches the object then the angular velocity much is 0. The RHS is for efficient operation i.e., until the destination has not been reached there must be some motion otherwise the system can always stay in the state of rest trivially satisfying the safety condition. This can also be modified and applied for a prismatic joint where instead of the angular velocity we use the linear velocity and the linear parameter d_i shown in the previous table is used in place of θ_i.

3) The Hybrid Program: The Hybrid Program in our case will have a Time-triggered model for control. Such a model accommodates the delays involved in sensing and pertinent processing. This is done by safely choosing a control action that is exhaustive in nature to avoid trivially true states for the safety conditions. The criteria for the control action in our case can be delineated as predicting motion-position estimates over the time $t < T$ where T is the maximum time delay that can be accommodated [10]. Now, we can select a direction of motion for the particular link which moves it towards the target position d_i^{final} for a prismatic joint. Further, we can check if applying an acceleration of A_i would leave enough time for the controller to take to the decision of decelerating by B_i and stop just at the final position. In other words, the velocity profile must be such that the area under the velocity-time curve is always less than the displacement from the initial to the final angle.

Through the graphical illustration, we can compile that even for any other kind of control that doesn't employ linear acceleration we can establish the safety condition as an integral of the velocity function with respect to time over the limits of the maximum delay must always be within the safety condition that allows the controller to choose deceleration in the next cycle and lets the respective link come to rest within the available displacement. For our case, wherein linear motion dynamics has been considered, the following equation is derived from the concepts motioned above that represents the condition Q_A mentioned in Equation (7):

$$Q_{A_i} \equiv 2B_i(\theta_i^{final} - \theta_i^t) = \omega_i^2 + (A_i + B_i)(A_i T^2 + 2T\omega_i) \quad (10)$$

Now, the next steps shall be: Constructing the differential equations for the continuous dynamics and defining an invariant that can aid the looping process. The differential equations need to combine the rotational aspects with the translational one to get the overall aspect of the system dynamics correctly.

As mentioned earlier, the serial control should enable the model to take control decisions one at a time, the postconditions for both cases involve the conjunction of all the links. This implies that the control actions should be taken for the i_{th} link only after one for the $i - 1_{th}$ has been taken. Adding or removing links has very little impact on the design. An example of Serial control is the Yamaha's Cartesian Robot which consists of 3 prismatic joints for movement along the X, Y, and the Z axis as shown below:

However, when continuous motion is needed for complex tasks like spray-painting, then parallel control is applied for simultaneous link movement. This requires a stricter evolution domain to restrict the relative motion between the links so that they do not collide with each other in the process of achieving

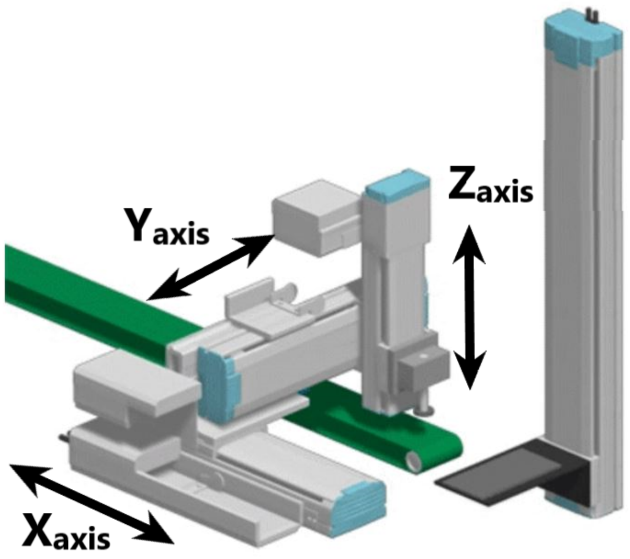

Fig. 3: A Serially controlled Cartesian Robot

their final positions. An example of Parallel control for a 4-DOF Robot from the Matlab Robotics toolbox is given below:

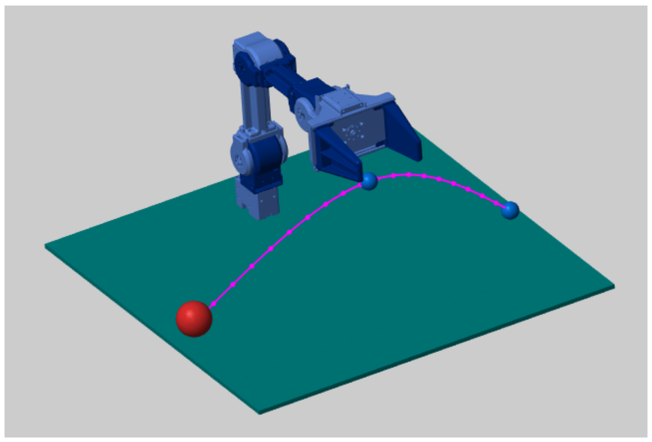

Fig. 4: Parallel Control for a 4-DOF robot to move along the curve

B. *Sensory Perception for Human-Robot Interaction*

Until now, the above sections assumed that the requisite destination points shall be provided by the user. We can have a another system in place for this purpose that collects the higher-dimensional sensory inputs and provides the robot with the requisite coordinates [8]. Such an additional data processing unit can make use of varied sensors. IR-based sensing or ultrasonic sensor-based motion detection can be applied for this information. In any case, the job of a separate data processing unit is to provide a lower-dimensional mapping of the data for the controllers. e.g., in the case of marine robotics, Ultrasound signals for underwater robot navigation can be implemented. This is applicable to industrial robots as well, however, the use of visual sensors i.e., cameras is a popular option as well.

For visual sensors, there exists a stereo camera which can estimate the depth of the target in front of it. The coordinates of human-body key-points can be produced as an output from a pre-trained neural network e.g., CMU's OpenPose library [5] is capable of producing such coordinates in a robust manner. Since these neural networks are not immune to factors like occlusion, we may add a Kernelized Correlation Filter-based tracker to introduce robustness in the process. After these performing these steps to generate the coordinate set, we can generate a safe heat-map in 3D which would define a certain allowable range of the positions of the end-effector and set constraints to the dynamics of the same near sensitive organs like the eyes.

C. *High-level Task Planning*

As automation evolves, the significance of planning increases. To cope with this, one of the earliest planners "STRIPS" [2] was introduced for Shakey the robot from Stanford. Given the set of actions, domain description, initial state, and the goal state, the planner is required to find a set of actions (edges of a state-space graph) that lead to the goal state. Various planning algorithms like Forward and Backward State-Space Planning can be used for this purpose. Moreover, novel algorithms extensively make use of the blocks-world problem which acts as the beginning stage for validation of the same. Having efficient planners can heavily impact the productivity of the machines and improves coordination among them at the same time.

V. Conclusion

In this paper, we saw how robotic systems require safe and efficient control to boost productivity and minimize the risk of damage. Robot kinematics has been explored in-depth to provide a clear picture of the problem. We discussed how the inverse kinematics of a robot contributes to the control decisions for all the links of the particular mechanism [9]. Preconditions, Postconditions, and the Hybrid Program for a general N link robot with prismatic or revolute joints have been discussed. The controller proofs are implemented on the KeymaeraX Modelling and Verification tool [10] for various configurations of multi-axis robots.

The hybrid program has also been generalized to be used for various configurations of multi-axis robots. Additional analysis of the motion profiles has been done to get the bigger-picture of non-linear control as well. In sub-sections of the paper, we have discussed subtler aspects of robotic systems such as Parallel and Serial control, and the inherent properties of both of these. We mentioned briefly about the external sensor processing units for flexible and cooperative robots citeb8. This included Visual and Ultrasound processing units to provide the robot with the final end-effector coordinates. We saw how higher-level Planners [2] can be used to formalize the notion of tasks and employing algorithms to search through the state-space and reach the goal destination.

Finally, we would like to mention that the for more complex non-linear control to handle unmodelled dynamics in the system such as variable payload and the requirement of variable torque for the same. Such systems require Lyapunov-based analysis for guaranteed stability around a set-point. These aspects haven't been covered in the paper. Advanced non-linear control and its optimization can be done on open-source packages like in [3]. There still exists a need for further exploration in area of multi-robot systems, human-robot interaction, and flexible cooperative robots that are safer to use and can perform multiple tasks as opposed to typical task-centric robot design. Only after a thorough investigation of these fields can we employ robots for higher-level cognitive tasks as done the humans.

References

[1] Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay, and Kerstin Dautenhahn. 2014. Formal verification of an autonomous personal robotic assistant. In AAAI FVHMS, 74–79.

[2] Richard E. Fikes, Nils J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving", Artificial Intelligence, Volume 2, Issues 3–4, 1971, Pages 189-208,

[3] Russ Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics", 2019, https://drake.mit.edu

[4] Z.M. Bi, Chaomin Luo, Zhonghua Miao, Bing Zhang, W.J. Zhang, Lihui Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing", 2021 Robotics and Computer-Integrated Manufacturing, Volume 67.

[5] Cao, Zhe et al. "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): 1302-1310.

[6] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero and J. Pérez-Oria, "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments," in IEEE Access, vol. 5, pp. 26754-26773, 2017, doi: 10.1109/ACCESS.2017.2773127.

[7] Singh, Pushpendra Joshi, S.D. Patney, Rakesh Kaushik, Saha. (2015). The Taylor's nonpolynomial series approximation (version 2).

[8] Das, A., Fierro, R., Kumar, V., Ostrowski, J., Spletzer, J., Taylor, C.: A vision-based formation control framework. IEEE Transactions on Robotics and Automation 18(5), 813–825 (2002)

[9] Manseur R, Doty KL. A Fast Algorithm for Inverse Kinematic Analysis of Robot Manipulators. The International Journal of Robotics Research. 1988;7(3):52-63. doi:10.1177/027836498800700304

[10] Quesel, Jan-David Mitsch, Stefan Loos, Sarah Aréchiga, Nikos Platzer, André. (2016). How to model and prove hybrid systems with KeYmaera: a tutorial on safety. International Journal on Software Tools for Technology Transfer. 18. 67-91. 10.1007/s10009-015-0367-0.

[11] Abdelfetah Hentout, Mustapha Aouache, Abderraouf Maoudj Isma Akli (2019) Human-robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017, Advanced Robotics, 33:15-16, 764-799, DOI: 10.1080/01691864.2019.1636714