CASPR-ROS: A Generalised Cable Robot Software in ROS for Hardware

Jonathan Eden, Chen Song, Ying Tan, Denny Oetomo and Darwin Lau

Abstract In this paper, the software platform CASPR-ROS is introduced to extend the author's recently developed simulation platform CASPR. To the authors' knowledge, no single software framework exists to implement different types of analyses onto different hardware platforms. This new platform therefore takes the advantages of CASPR, including its generalised CDPR model and library of different analysis tools, and combines them with the modular and flexible hardware interfacing of ROS. Using CASPR-ROS, hardware based experiments can be performed on arbitrarily CDPR types and structures, for a wide range of analyses, including kinematics, dynamics and control. The case studies demonstrate the potential to perform experiments on CDPRs, directly compare algorithms and conveniently add new models and analyses. Two robots are considered, a spatial cable robot actuated by PoCaBot units and an anthropomorphic arm actuated by MYO-muscle units.

1 Introduction

Cable-driven parallel robots (*CDPR*s) are a class of mechanisms in which actuation is transmitted through cables in place of rigid links. CDPRs possess a range applications including payload manipulation [1, 2, 3], motion simulation [4], exoskeletons [5] and musculoskeletal robots [6, 7, 8, 9]. An important feature of cable actuation is that cables can only transmit forces in tension. This creates many unique challenges in CDPR modelling [6], design [10], inverse dynamics [11, 12, 13], forward kinematics [14, 15, 16] and motion control [17, 18].

Jonathan Eden* · Ying Tan · Denny Oetomo
The University of Melbourne, Melbourne, Australia
e-mail: jpeden@student.unimelb.edu.au, yingt@unimelb.edu.au, doetomo@unimelb.edu.au

Jonathan Eden · Chen Song · Darwin Lau
The Chinese University of Hong Kong, Hong Kong
e-mail: chensong@link.cuhk.edu.hk, darwinlau@mae.cuhk.edu.hk

Most CDPR algorithms can be applied onto different classes of CDPR. However, CDPR research typically either validates algorithms in simulation or through implementation only on the research group's robots. This inhibits CDPR development and research in a number of ways: 1) The evaluation of new techniques often neglects the effect of CDPR structures such that the impact of varying attachments/degrees of freedom/cables may be unknown. 2) There are no benchmarking algorithms for performance comparison. 3) There is a significant cost in implementing new results on hardware, where researchers often re-implement existing models and algorithms.

To address these concerns, different software platforms have been developed for the study of CDPRs. In [19], a MATLAB/Simulink control simulation software interfaced to the dynamics simulator XDE was presented. This platform was developed for single link CDPRs, where the addition of other CDPRs is not simple. For planar and spatial CDPRs, the ARACHNIS [20] and WireCenter [3] software platforms were developed. These platforms were not designed for algorithm benchmarking and additional algorithms cannot be added. Recently, CASPR [21], a MATLAB based simulation platform for the study of CDPRs, was developed. This platform addresses the previous issues by allowing the analysis of arbitrary CDPRs with the possibility to accommodate different algorithms.

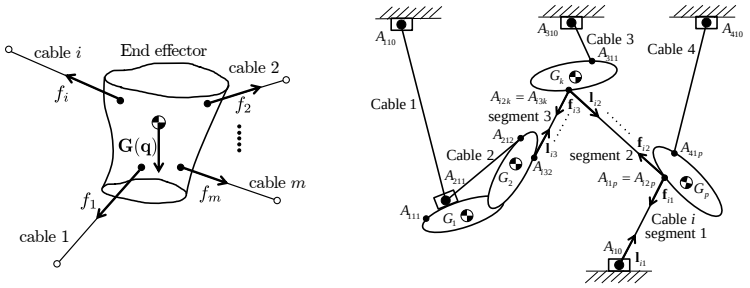
CASPR is primarily a simulation platform and does not favour online hardware control and analysis due to its object oriented MATLAB implementation. The extension of CASPR for hardware implementation would allow the operation of arbitrary CDPR hardware to benefit from the flexibility, robustness and extendibility of CASPR. ROS represents one existing means of interfacing robotics hardware which provides a modular and well supported interface for extension and integration [22].

In this paper, CASPR-ROS is introduced as a software platform for CDPR hardware implementation. This platform implements the generalised and object-oriented principles of CASPR into ROS to take advantage of ROS's flexible and modular hardware interfacing capabilities. Through the addition of a new extendible hardware interfacing layer, it is shown that hardware implementations can be performed onto arbitrary CDPRs using arbitrary hardware units. The convenience and efficiency of benchmarking and online implementation in CASPR-ROS is then demonstrated through experimental results obtained on different hardware platforms.

2 Background

2.1 System Model

Consider the general single (SCDR) and multi-link (MCDR) CDPRs depicted in Figure 1. The n degree of freedom robot configuration (*joint space*) is represented by the pose vector $\mathbf{q} \in \mathbb{R}^n$. The m cable actuation can be described by the cable length and force (*cable space*) vectors $\mathbf{l} = [l_1 \quad \dots \quad l_m] \in \mathbb{R}^m$ and $\mathbf{f} = [f_1 \quad \dots \quad f_m] \in \mathbb{R}^m$, respectively, where $l_i, f_i \geq 0$ denote the length and force of cable i.



(a) Single-link Cable-Driven Robot (b) Multi-link Cable-Driven Robot

Fig. 1: General Single-link and Multi-link CDPRS

The kinodynamic equations for the CDPRs depicted in Figure 1 are given by

$$\dot{\mathbf{l}} = L(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

$$M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) + \mathbf{G}(\mathbf{q}) + \mathbf{w}_e = -L^T(\mathbf{q})\mathbf{f}$$

$$\mathbf{0} \leq \mathbf{f}_{min}(\mathbf{q}) \leq \mathbf{f} \leq \mathbf{f}_{max}(\mathbf{q}) \ , \tag{2}$$

where $L \in \mathbb{R}^{n\times m}$ is the cable Jacobian matrix, $M \in \mathbb{R}^{n\times n}$ is inertia matrix and $\mathbf{C}, \mathbf{G}, \mathbf{w}_e \in \mathbb{R}^n$ are the Coriolis/centrifugal vector, the gravitational vector and the external wrench, respectively. The vectors $\mathbf{f}_{min}, \mathbf{f}_{max} \in \mathbb{R}^m$ are the minimum and maximum cable force bounds. They are constant for ideal cables and pose dependent for spring, variable stiffness [10, 23] and muscle inspired cables [24, 25].

2.2 Models in CASPR

CASPR is a software platform for the simulation and analysis of CDPRs [21]. CASPR models CDPRs with fundamental equations (1) and (2) using the cable routing matrix based model [6]. This model can represent SCDRs and MCDRs provided that it is given the inertia and joint properties of each link as well as the actuation and attachment specifications of each cable. These specifications are provided in an easily reconfigurable manner through the use of XML scripts[1].

The model representation used by CASPR provides a generic form from which any new CDPR can be added. As such, CASPR-ROS makes use of this same representation allowing for new models to be easily added in addition to the existing CASPR supported models including: NIST RoboCrane [1], CoGiRo [2], IPAnema family [3], the MyoArm [9], and CAREX [5]. To use these models on hardware it is necessary that the algorithms are made suitably computationally efficient and that

[1] A detailed explanation of CASPR models and the use of XML scripts is provided in [21].

interfaces are provided to connect the computational component of CASPR with the hardware.

2.3 Analysis in CASPR

CASPR supports CDPR analysis using an inheritance based object oriented approach. For each analysis problem, an abstract based class is created to represent the problem. New algorithms are then generated by inheriting the base class and implementing abstract methods which map the input to the appropriate outputs. Using this approach, different algorithms for a range of problems including inverse dynamics, forward kinematics, control and workspace analysis are supported in CASPR[21].

In hardware implementation, the resolution of joint space wrench into cable forces, conversion of cable lengths into joint space pose and the tracking of reference trajectories must be considered. CASPR-ROS therefore uses the generalised inheritance based paradigm of CASPR [21] for the inverse dynamics, forward kinematics and control problems. CASPR-ROS currently contains the following algorithms:

- Inverse Dynamics - The computationally efficient closed form method [11] and the quadratic programming method using the qpOASES solver [26].
- Forward Kinematics - The Jacobian pseudo-inverse method [14] and the non-linear least squares method [15].
- Control - The computed torque [17] and Lyapunov based static [18] controllers.

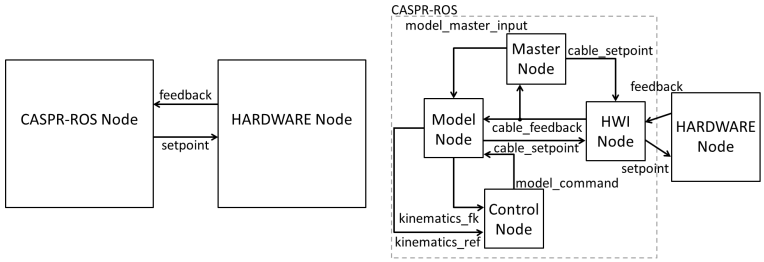
3 Interfacing Hardware in CASPR-ROS

3.1 Integrating CASPR with ROS

Using the reconfigurable model representation and inheritance based object oriented paradigm discussed in Section 2, CASPR-ROS users can apply a range of CDPR analysis techniques onto different CDPR models. To connect this generic CASPR-ROS computational module to hardware it is necessary for the module to be interfaced with hardware specific sensors and actuators. ROS is chosen for this connection due to its widespread usage, existing hardware support capabilities and open source nature. ROS messaging is then used to translate between the cable space variables and the actuator and sensor information.

Figure 2 shows two different ROS-based hardware communication schemes supported in CASPR-ROS: the centralised method and the distributed method. It can be seen that the centralised method (Figure 2(a)) connects hardware to CASPR-ROS through a single experiment node. This node facilitates all possible CASPR-ROS operations including forward kinematics, trajectory generation, control and inverse dynamics. In addition, the node is also responsible for translating the generic cable

space information into hardware specific feedback and setpoint ROS messages. As a result, the centralised method allows for a more direct process of porting CASPR code into ROS. However, the method prevents common operations, such as forward kinematics, from being continuously operated while the hardware is active and limits its algorithm changes to only occur in between experiments. This approach is therefore best used for single experiments that are not repeated such as calibration.



(a) Centralised Communication Scheme (b) Distributed Communication Scheme

Fig. 2: ROS based Hardware Communication Methods used in CASPR-ROS

In contrast, the distributed method (Figure 2(b)) distributes the common CASPR-ROS operations across a number of ROS nodes thereby allowing for the nodes to be run independently if desired. As a result, this method can allow for nodes and analysis algorithms to be changed within single experiments and is best used for repetitive operation of experiments. In addition to having ROS publishers and subscribers associated with the feedback and setpoint messages, the distributed method also requires CASPR-ROS messages to be defined for communication between CASPR-ROS nodes. The following messages are provided:

- `joint_kinematics` - Contains the kinematic vectors $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$.
- `master_command` - Contains the current operation and timing information.
- `model_update_command` - Provides the command variables $\mathbf{q}_{cmd}$, $\dot{\mathbf{q}}_{cmd}$, $\ddot{\mathbf{q}}_{cmd}$ and $\mathbf{w}_{cmd}$ for updating the equation of motion (2).

3.2 Hardware Interfaces

Common to all CASPR-ROS communication is the need to translate between generic cable space variables and hardware specific ROS messages. CASPR-ROS does this by providing an abstract hardware interfacing class. This class therefore contains all of the basic ROS messaging publisher and subscriber objects in addition to providing the rules for cable space-hardware translation. In this manner, the CASPR-ROS computational core remains generic and need not consider hardware specific requirements such as filtering and relative/absolute data conversion.

The abstract class `HardwareInterfaceBase` therefore sits between the hardware and the CASPR-ROS computational core. Implementations of this class are responsible for translating the hardware specific contents of the feedback message into the associated cable space variables and in writing the actuator specific commands given knowledge of the command cable space variables. This is achieved by implementing the abstract methods `updateFeedback(..)`, `publishForceSetpoint(..)`, `publishLengthSetpoint(..)` and `publishVelocitySetpoint(..)`. To show CASPR-ROS's use of hardware interfaces, the `FlexrayInterface` and `PoCaBotInterface` classes have been constructed for MYO-muscle modules [27] and PoCaBot units[2].

4 Experiments in CASPR-ROS

4.1 Adding new Experiments

Experiments in CASPR-ROS represent executables which define the operation of a CDPR using the analysis techniques and hardware interfaces discussed in Sections 2 and 3. New experiments can be added into CASPR-ROS through the addition of new master ROS nodes. A single master node is therefore responsible for generating a reference in addition to possibly specifying the CDPR model, hardware interface and analysis algorithm when the centralised scheme is used.

Like the modelling and hardware interface classes described in [21] and Section 3, respectively, CASPR-ROS master nodes use an inheritance based object oriented design principle. The abstract classes `ScriptBase` and `MasterNodeBase` are classes which comprise of a single `mainLoop` function which is to be implemented by all new centralised and distributed experiments, respectively. Code Sample 1 illustrates the `mainLoop` function used in CASPR-ROS.

Code Sample 1: mainLoop function used in `ScriptBase` class.

```
// Variable Initialisation
bool is_initialised = 0, first_time = 1;
// General Operation
while((ros::ok()) && (!terminating_condition(t))){
  // Only proceed once the hardware is ready
  if(hardware_interface.hardwareReady()){
    // Check the initialisation status of the experiment
    if(!is_initialised){
      // Run the initialisation procedure
      is_initialised = initialising_function(first_time);
      if(first_time){ first_time = false; }
    } else {
      // Run the main procedure
      main_function(t); t += SYSTEM_PERIOD;
```

[2] PoCaBot unit specifications can be found at `https://github.com/darwinlau/CASPR/wiki`

```
    }
  }
  // ROS management
  ros::spinOnce(); loopRate.sleep();
}
// Terminate the script
terminating_function();
```

It can be seen that the `mainLoop` function represents a single function that defines the behaviour of the robot throughout operation. This behaviour is defined for each particular experiment through the implementations of four abstract methods: `terminating_condition(..)` which defines the terminating condition for the main function, `initialising_function(..)` which initialises the hardware, `main_function(..)` which defines the desired general CDPR behaviour and `terminating_function(..)` which safely terminates the experiment.

4.2 Operating Procedure

To run experiments in CASPR-ROS the following procedure is required: 1) Configure the model parameters and trajectories using CASPR XML scripting. 2) Configure the experiment settings using the roslaunch files associated with the desired experiments. 3) Run the relevant ROS nodes using the appropriate ROS launch files.

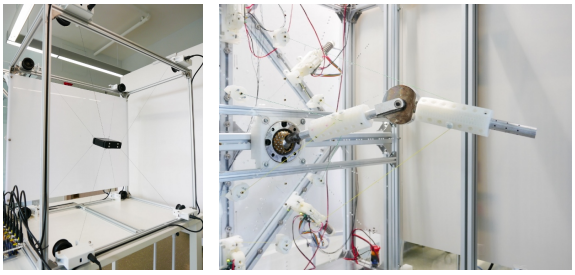
5 Experimental Results

Two case studies are presented using the CDPRs depicted in Figure 3. These studies show the application of CASPR-ROS on arbitrary CDPRs and CASPR-ROS's potential application for benchmarking different analysis algorithms on hardware[3].

5.1 Case Study: Spatial Cable Robots using PoCaBot Units

This case study shows the use of CASPR-ROS in the online length control of the spatial CDPR driven by PoCaBot units (depicted in Figure 3(a)) with the units attached on the corners of a $84 \times 54 \times 80$cm frame. Figure 4 depicts the performance of the system for each of the trajectories depicted in Figure 5, where $\begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^T$ and the orientation $\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T$ is represented by the XYZ Euler angle convention. It can be seen from Figure 4 that the robot

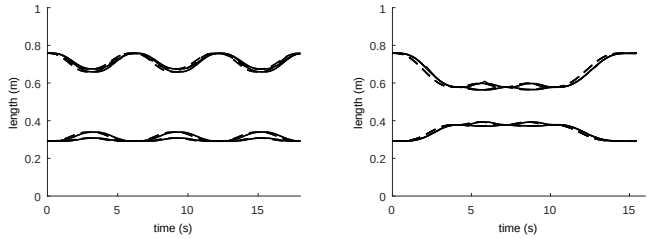
[3] The case study specifications can be found in the folder `data/model_config/models` at the repository `https://github.com/darwinlau/CASPR`. Case Studies 1 and 2 are contained in the folders `PoCaBot_spatial` and `BM_arm`, respectively.



(a) Spatial PoCaBot CDPR (b) 2 Link BioMuscular-Arm

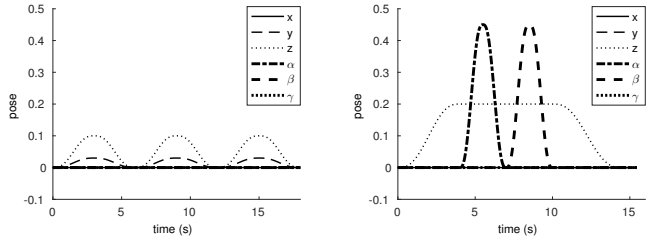
Fig. 3: Case Study CDPRs

tracks the desired lengths with only a small lag and tracking error. In addition it can be seen from Figure 4 that the obtained length feedback is provided within the PoCaBot operating frequency of 20Hz (for 8 motors) indicating the capability of CASPR-ROS to be configured for this online constraint.



(a) Trajectory 1 (b) Trajectory 2

Fig. 4: Cable Length Command (Dashed) and Feedback (Solid)



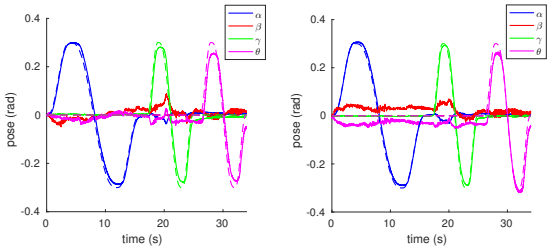
(a) Trajectory 1 (b) Trajectory 2

Fig. 5: Reference Joint Space Trajectory

From this case study it can be seen that online kinematic length control can be achieved using CASPR-ROS. Furthermore, by using XML scripts and modular ROS nodes, the resulting code is flexible to changes in the experimental set-up, such as different experimental trajectories, without the need for separate experiment scripts.

5.2 Case Study: BioMuscular Arm using MYO-muscles

This case study illustrates the use of CASPR-ROS in benchmarking two different inverse dynamics algorithms over a range of cable sets. The experiment is performed using the 2 link BioMuscular Arm (*BM-Arm*) with Myomuscle units, depicted in Figure 3(b). The performance of the closed form [11] and minimum force norm quadratic program (QP) based inverse dynamics algorithms are compared by tracking the reference joint space trajectories (shown with dashed lines) in Figure 6 with the cable sets `CS1` and `CS2`, where $\begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T = \begin{bmatrix} \alpha & \beta & \gamma & \theta \end{bmatrix}^T$. To ensure accurate tracking, a computed torque controller is also implemented.

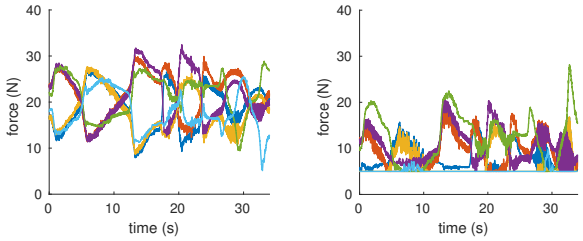


(a) Closed Form Method (b) QP Method

Fig. 6: Joint Space Command (dashed) and Forward Kinematics (solid) - CS1

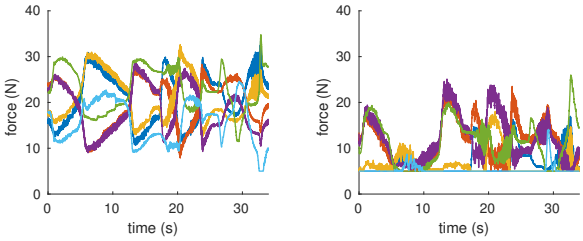
Figure 7 shows the cable force solutions for each algorithm using cable set `CS1`. It can be seen that the QP solver (due to its solving objectives) requires typically lower forces and that in both cases the closed loop control has resulted in oscillating cable forces. The resulting tracking performance (obtained using the pseudo-inverse forward kinematics method) of each inverse dynamics solver is shown in Figure 6. In this case, the closed form solver results in a slightly larger lag and steady state error particularly in the twist axis β and revolute θ axes. This is likely the result of the discretised MYO-muscle sensor resolution, where the resolution is larger over smaller force values.

Figures 8 and 9 depict the cable forces and tracking performance for cable set `CS2`, respectively. It can be seen that the resulting solutions for both methods are different to that observed using cable set `CS1`, however the tracking performance is quite similar. The relative performance of the two solvers is however similar in which the use of lower cable forces leads to more reliable tracking performance.



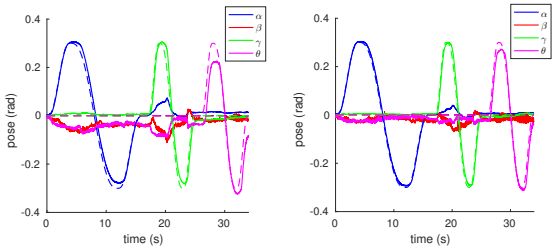
(a) Closed Form Method (b) QP Method

Fig. 7: Closed Loop Inverse Dynamics Solutions for CS1



(a) Closed Form Method (b) QP Method

Fig. 8: Closed Loop Inverse Dynamics Solutions for CS2



(a) Closed Form Method (b) QP Method

Fig. 9: Joint Space Command (dashed) and Forward Kinematics (solid) - CS2

Table 1 shows the computational time used in solving the inverse dynamics, forward kinematics and control for each experiment. It can be seen that the closed form method is on average slightly faster and possesses a lower maximum time. It is also noted that the period of operation was less than that required by the 150Hz frequency of the BM-Arm in all cases for both algorithms.

From this case study the use of CASPR-ROS in comparing different analysis techniques can be observed. This case study also displays the flexibility of CASPR-ROS to consider arbitrary cables sets without the need for system model derivation.

	Closed Form - CS1	QP - CS1	Closed Form - CS2	QP - CS2
Maximum time (ms)	5.20	4.79	2.12	5.14
Average Time (ms)	1.33	1.41	1.27	1.38

Table 1: Computational Time Specifications

6 Acknowledgements

The work was supported by a grant from the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service of Germany (Reference No. G-CUHK410/16). Acknowledgements to the CUHK T-Stone Robotics Institute for supporting this work.

7 Conclusion

CASPR-ROS was presented as a tool for the hardware implementation of algorithms onto arbitrary CDPRs. The platform aims to address the lack of a comprehensive CDPR hardware implementation software by integrating the hardware connectivity of ROS with the generic and flexible qualities of CASPR. The modular design of CASPR-ROS makes it convenient to develop new models, analysis algorithms, hardware interfaces and executable scripts. The presented case studies illustrate the flexibility of using CASPR-ROS on different hardware platforms. Future work for CASPR-ROS will look to increase the types of analyses provided and to broaden the cable models considered to include sagging cables and other actuator dynamics.

References

1. Albus, Bostelman, and Dagalakis, "The NIST robocrane," *J. Robot. Syst.*, vol. 10, no. 5, pp. 709–724, 1993.
2. Lamaury and Gouttefarde, "Control of a large redundantly actuated cable-suspended parallel robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4659–4664.
3. Pott, Mütherich, Kraus, Schmidt *et al.*, "Ipanema: A family of cable-driven parallel robots for industrial applications," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., vol. 12, pp. 119–134.
4. Miermeister, Lächele, Boss, Masone *et al.*, "The cablerobot simulator large scale motion platform based on cable robot technology," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 3024–3029.
5. Mao and Agrawal, "Design of a cable-driven arm exoskeleton (CAREX) for neural rehabilitation," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 922–931, 2012.
6. Lau, Oetomo, and Halgamuge, "Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1102–1113, 2013.

7. Kozuki, Mizoguchi, Asano, Osada *et al.*, "Design methodology for thorax and shoulder of human mimetic musculoskeletal humanoid kenshiro - a thorax with rib like surface -," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 3687–3692.
8. Lau, Eden, Halgamuge, and Oetomo, "Cable function analysis for the musculoskeletal static workspace of a human shoulder," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., vol. 32, pp. 263–274.
9. Richter, Sören Jentzsch, Garrido, Ros *et al.*, "Scalability in neural control of musculoskeletal robots," *IEEE Robot. Autom. Mag.*, 2016.
10. Yeo, Yank, and Lim, "Design and analysis of cable-driven manipulators with variable stiffness," *Mech. Mach. Theory*, vol. 69, pp. 230–244, 2013.
11. Pott, "An improved force distribution algorithm for over-constrained cable-driven parallel robots," in *Computational Kinematics*, 2014, pp. 139–146.
12. Müller, Reichert, and Bruckmann, "Analysis of a real-time capable cable force computation method," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., 2015, vol. 32, pp. 227–238.
13. Lau, Oetomo, and Halgamuge, "Inverse dynamics of multilink cable-driven manipulators with the consideration of joint interaction forces and moments," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 479–488, 2015.
14. Bruckmann, Mikelsons, Brandt, Hiller *et al.*, "Wire robots part I: Kinematics, analysis & design," in *Parallel Manipulators New Developments*, ser. ARS Robotic Books, 2008.
15. Pott and Schmidt, "On the forward kinematics of cable-driven parallel robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 3182–3187.
16. Merlet, "A generic numerical continuation scheme for solving the direct kinematics of cable-driven parallel robot with deformable cables," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 4337–4343.
17. Williams II, Xin, and Bosscher, "Contour-crafting-cartesian-cable robot system: Dynamics and controller design," in *DETC 2008: 32nd Annual Mechanisms and Robotics Conference, Vol. 2, Parts A & B*, 2009, pp. 39–45.
18. Alp and Agrawal, "Cable suspended robots: Design, planning and control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 4275–4280.
19. Michelin, Baradat, Nguyen, and Gouttefarde, "Simulation and control with XDE and matlab/simulink of a cable-driven parallel robot (CoGiRo)," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., vol. 32, pp. 71–83.
20. Ruiz, Caro, Cardou, and Guay, "ARACHNIS: Analysis of robots actuated by cables with handy and neat interface software," in *Cable-Driven Parallel Robots*, ser. Mech. and Mach. Sci., vol. 32, pp. 293–305.
21. Lau, Eden, Tan, and Oetomo, "CASPR: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 3004–3011.
22. Quigley, Faust, Foote, and Leibs, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, 2009, p. 5.
23. Yang, Yang, Wang, Zheng *et al.*, "Design analysis of a 3-dof cable-driven variable-stiffness joint module," in *IEEE Int. Conf. Robot. and Biomimetics*. IEEE, 2015, pp. 529–534.
24. Zajac, "Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control," *Crit. Rev. in Biomed. Eng.*, vol. 17, no. 4, pp. 359–411, 1989.
25. Lau, Eden, Oetomo, and Halgamuge, "Musculoskeletal static workspace of the human shoulder as a cable-driven robot," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 978–984, 2015.
26. Ferreau, Kirches, Potschka, Bock *et al.*, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.
27. Marques, Maufroy, Lenz, Dalamagkidis *et al.*, "MYOROBOTICS: a modular toolkit for legged locomotion research using musculoskeletal designs," in *Proc. 6th International Symposium on Adaptive Motion of Animals and Machines (AMAM'13)*, 2013.