CASPR 101: The Basics of Using CASPR

This documentation serves as the basic getting started document for using CASPR. Please refer to the README document for how to setup CASPR on your computer. For more detailed in-depth documentation on the architectural and technical details of the components of CASPR please refer to the additional documentation located within the folder ∼/`docs/` (Note: ∼/ is the root folder of where CASPR is located).

1 What is CASPR?

The *Cable-robot Analysis and Simulation Platform for Research (CASPR)* is an open-source software platform designed for researchers to perform study on Cable-Driven Parallel Robots (*CDPRs*) with the following benefits:

- **Comprehensive** range of different types of analyses on cable-driven parallel robots (*CDPRs*): kinematics, dynamics, control, workspace analysis, and design optimisation.
- **Extensive library** of existing state-of-the-art CDPR models already included, ranging from single link CDPRs to multilink CDPRs to hybrid acutated CDPRs. XML format of the CDPR model means it is easy to add and simulate your own CDPR.
- **Perform research faster** for both existing and new researchers in CDPRs by removing the need to re-implement other algorithms and re-derive the system model.
- **Modular software architecture** makes it easy to add, test and compare your own algorithms, implementations, cable models, types of joints and much more.
- **Easily compare** different algorithms and implementations on the same machine for the same type of analysis and CDPR using the modular architecture and extensive library.
- **Open-source** for the research community to contribute and share their research with everyone.

The hope of CASPR is to prevent the "reinvent-the-wheel" problem from inhibiting the research progress and advancement of CDPRs. So your support in both using and contributing to CASPR would be greatly appreciated.

1.1 CDPRs in CASPR

Currently, CASPR contains the following CDPR models to be used "out-of-the-box":

Name	Type	No. Links	No. DoFs	Country	Cite
KNTU planar robot	Planar SCDM	1	3	Iran	[1]
2 DoF VSD robot	Planar SCDM	1	2	Singapore	[2]
Passive Spring Planar CDPR	Planar SCDM	1	3	U.S.	[3]
MyoArm Shoulder	Spherical SCDM	1	3	Germany	[4]
ACROBOT	Spatial SCDM	1	6	France	[5]
CoGiRo	Spatial SCDM	1	6	France	[6]
IPAnema1	Spatial SCDM	1	6	Germany	[7]
IPAnema2	Spatial SCDM	1	6	Germany	[7]
NIST ROBOCRANE	Spatial SCDM	1	6	U.S.	[8]
SEGESTA	Spatial SCDM	1	6	Germany	[9]
FAST	Spatial SCDM	1	6	China	[10]
CAREX	MCDM	2	5	U.S.	[11]
Human inspired neck model (8S_neck)	MCDM	8	24	-	[12]

2 Adding a CDPR to CASPR

2.1 XML Model Description

A new CDPR model can be added to CASPR by creating the following XML files:

1. **bodies.xml**: The rigid body structure of the CDPM, such as the inertia properties and the type of joints.
2. **cables.xml**: The set of cable arrangements for the cable attachment locations and cable properties.
3. **trajectories.xml**: The set of possible trajectories that the robot may execute.
4. **operational_space.xml** [*optional*]: Defines the operational space for the CDPR.

Refer to the documentation on the XML model specifications for more details on the XML structure and tags.

2.2 Creating the Model

A new CDPR model can be added to CASPR by performing the following steps:

1. If you are starting a new session navigate to the ∼ folder and execute the script `initialise_CASPR.m`.
2. Create a new folder for the robot at a suitable location within the directory ∼`/data/config/models/SCDM` (for single link CDPMs), or ∼`/data/config/models/MCDM` (for MCDMs).
3. Create the necessary XML files (as described in Section 2.1) in the folder created in step 1, the names of the actual XML files can be named to the users' preferences.
4. Add to the enum class `ModelConfigType` (located at ∼`/data/config/models/ModelConfigType.m`) an enum to represent the added CDPR following the convention `M_ROBOTNAME`.
5. Update the master list file ∼`/data/config/models/master_list.csv` with the information for the robot in the following CSV format, where each row is used for an individual robot:
   `CDPR_enum, folder_path, bodies_file, cables_file, trajectories_file, operational_space_file`
6. Run the unit test function `runtests('ModelConfigTest')` to ensure that your model has been correctly added (it should appear in the set of CDPR models that the test runs through).

2.3 Points to Note

2.3.1 Cable Sets

A *cable set* refers to the definition of an entire set of cables, such as the type of cables and its attachment locations, for the CDPR. The **cables.xml** file for a CDPR can store multiple cable sets for different arrangements and types of cables. Cable sets allow users to easily switch between different cable arrangements for the same robot during analysis and simulation. Each cable set must be created with an `id` attribute in order to identify the cable set. The `default_cable_set` attribute to the `<cables>` specify which cable set should be used by default if it is not specified by the user.

2.3.2 Cable Models

CASPR supports multiple types of cable models, such as ideal cables, cables modelled as a linear spring, variable stiffness cables and the Hill-type physiological muscle model. Furthermore, users can add new cable models to share with the community. Within a `cable_set`, users can specify for each cable which cable model is to be used. This allows for the potential to "mix-and-match" different cable models if desired. The cable model used is determined by the tag of the cable, for example `<cable_ideal>` and `<cable_linear_spring>` for ideal cable model and linear spring model, respectively. Note that different cable models may have different attributes and properties. Please refer to the XML model specifications documentation for more details on the supported cable models and their attributes, and also refer to Section 4.3 on how to create a new cable model in CASPR.

2.3.3 Trajectory Sets

Similar to Section 2.3.1, a set of trajectories can be specified in the **trajectories.xml** file. This allows for different trajectories to be selected, through the `id` attribute, for performing simulation on.

3 Running Simulations

Simulations of different types of analyses can be performed/executed through CASPR in one of two ways: 1) MATLAB script programming and 2) through the CASPR GUI. The GUI is intended to be a more user-friendly approach to use CASPR and only the more stable features of CASPR will be progressively implemented. On the other hand, scripts allow users to access **all features** within CASPR (including those not in the GUI).

Note: before executing a script or GUI in CASPR, the following procedure **should always** be followed

1. Navigate MATLAB to the folder ∼ /.
2. Run the initialisation script `initialise_CASPR.m`.

3.1 Scripts

A set of example scripts to run various simulators of CASPR have been included at the folder location ∼`/scripts/examples/`. These simulators demonstrate how to operate and run simulations for varying analysis techniques.

Note: Scripts should always be run in CASPR from the ∼ directory. The name of the script can be prompted using the tab command.

3.1.1 Loading a Model (`script_loadRobot_example`)

In CASPR, the system model represents an object that contains all of the physical properties, dynamics and kinematics (of both the rigid bodies and the cables) for a particular cable robot. In this example, the method to load a system model is shown (Code Sample 1). This script, which is located in the folder which corresponds to the script ∼`/scripts/examples`, comprises of 3 steps:

1. Creation of the `ModelConfig` object by specifying the type of robot (`ModelConfigType`).
2. Creation of the model (`SystemModel` in ∼`/src/Model/SystemModel.m`) by the `getModel` method.
3. Plotting the created model in order to visualise the CDPR.

Code Sample 1: Example Script for Loading a new Model.

```
% Set up the type of model:
model_config = ModelConfig(ModelConfigType.M_SEGESTA);
% The SystemModel is created
cdpr = model_config.getModel(model_config.defaultCableSetId);
% Plot to see the robot
MotionSimulator.PlotFrame(cdpr, model_config.displayRange);
```

Steps 1 and 2 together are required in all CASPR scripts to load a new cable robot model. In **step 1**, the type of robot to create is specified through the `ModelConfigType` enumeration, where the SEGESTA robot will be created in Code Sample 1. In **Step 2**, the model is now created by the `getModel(cableSetId)` method, where `cableSetId` is the string ID that references to the cable sets from the **cables.xml** XML file. In this example, the `defaultCableSetId` as specified in the XML file is used. **Step 3** shows how the resulting model could potentially be used within simulations, and in this example the robot is simply drawn onto the screen for visualisation.

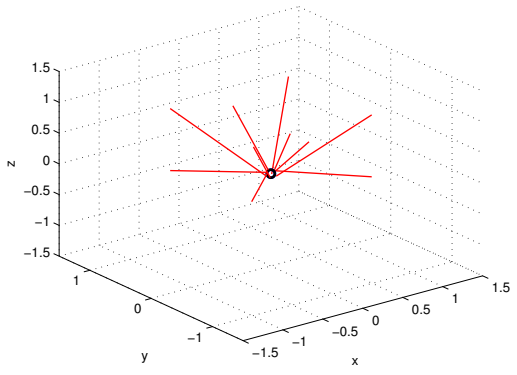


Figure 1: Output Figure for `script_loadRobot_example`

3.1.2 Creating and Running a Simulator

In CASPR, *simulators* are the primary way to execute analysis in simulation, for example, inverse dynamics analysis over a trajectory or workspace analysis over set of poses. Additionally, different simulators allow for analysis specific functionalities such as plotting of results/graphs and generating result statistics. In this tutorial, an inverse dynamics simulator will be created and then used in order to determine a possible set of solution cable forces for a desired trajectory. Examples of the usage of simulators for different types of analyses in CASPR can be found in the `~/scripts/examples` directory.

The script `~/scripts/examples/dynamics/script_ID_closedForm_example.m` is shown in Code Sample 2. This script comprises of 5 steps:

1. Loading a CDPR model (Section 3.1.1).
2. Loading an inverse dynamics solver (`IDSolverClosedForm` in this example).
3. Creation of the inverse dynamics simulator (`InverseDynamicsSimulator`).
4. Evaluation/execution of the simulator on a desired trajectory through the `run` method.
5. Plotting the simulation results.

Code Sample 2: Example Script for Running a Simulator.

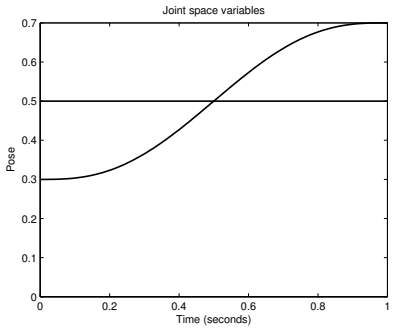
```
% Set up the type of model, trajectory and the set of cables to be used
model_type = ModelConfigType.M_SIMPLE_PLANAR_XY;
model_config = ModelConfig(model_type);
modelObj = model_config.getModel('basic');
% Create an inverse dynamics solver (ID solver)
id_type = ID_CF_SolverType.IMPROVED_CLOSED_FORM;
id_solver = IDSolverClosedForm(modelObj, id_type);
% Setup the ID simulator with the SystemModel object and the ID solver
idsim = InverseDynamicsSimulator(modelObj, id_solver);
trajectory = model_config.getTrajectory('x_simple');
% Run the solver on the desired trajectory
idsim.run(trajectory);
% Plotting simulation graphs
idsim.plotJointSpace();
idsim.plotCableForces();
```

Step 1 creates the CDPR model (*planar SCDM*) with the *basic* cable set configuration. **Steps 2 and 3** create the `InverseDynamicsSimulator` (`idsim`) by providing a CDPR model (`model`) and a desired inverse dynamics solver algorithm (`id_solver`) used to resolve the inverse dynamics. In this code sample, the desired solver is the improved closed form algorithm [13] (`IDSolverClosedForm`). This approach to creating simulators allows different types of inverse dynamics solvers to be selected from within the script. Please refer to the documentation of the analyses and inverse dynamics solvers for more detail. **Step 4** then runs the simulator on the trajectory from the XML configuration file with ID '`x_simple`' (a simple straight line motion trajectory in the x-axis). For every simulator in CASPR, the `run` method is used to execute the simulator. **Step 5** shows how the simulator can also be used to plot the results of the simulation, in this example the joint trajectory and resulting cable force solution (Figure 2).

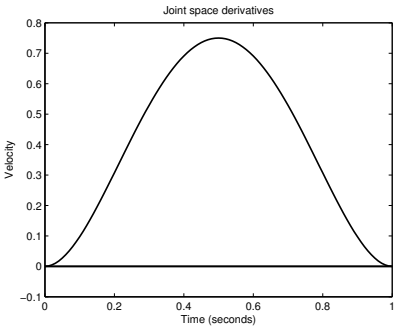
3.2 GUI (*Alpha Version*)

CASPR also supports the loading of models and running of simulations through the in-built CASPR GUI. This feature (currently in *alpha version*) allows users to do almost everything that can be done through scripting with greater ease. The CASPR GUI can be opened through the command `CASPR_GUI`, and will open the **home window** by default (Figure 3). The home window of the GUI contains 4 elements:

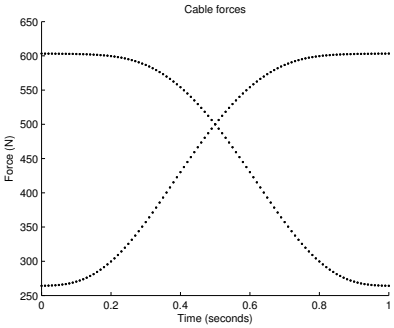
1. *Model setup* options to allow users to select different CDPR models and cable sets.
2. *Simulators* selection to allow users to choose which simulator to open.
3. *Model view* to visually display the CDPR in different poses or cable sets.
4. *Model pose* to allow users to change the pose to display in the model view.



(a) Joint Pose Plot



(b) Joint Pose Derivative Plot



(c) Cable Force Plot

Figure 2: Output plots for ∼/scripts/examples/dynamics/script_ID_closedForm_example.m

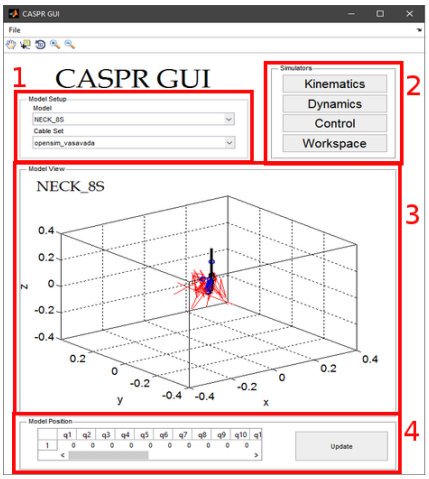


Figure 3: CASPR GUI main window

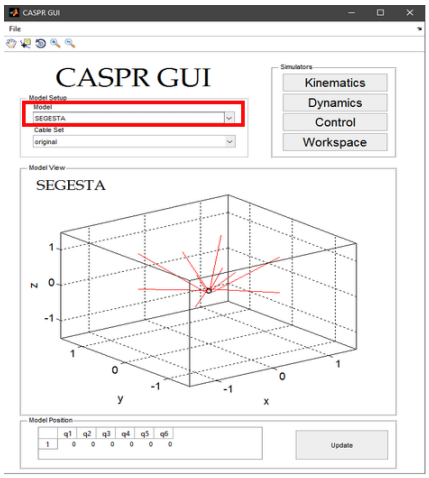


Figure 4: GUI Main Window with SEGESTA Selected

3.2.1 Loading a Model

Using the home window of the GUI, a new model can be loaded by:

- Opening up the *models* dropdown and select the cable robot of your choice, for e.g. `SEGESTA` (Figure 4).
- After the model is loaded, the desired cable set may be selected from the *cable set* dropdown.

The new model is automatically loaded once a model and its cable set are chosen.

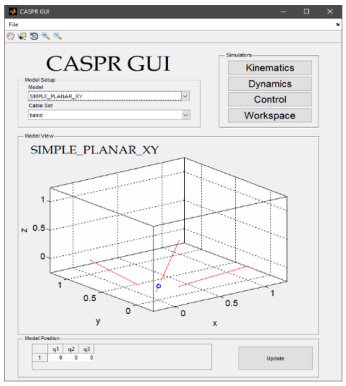
3.2.2 Creating and Running a Simulator

Running a simulator in the CASPR GUI is as simple as selecting one of the simulators on the home window and then choosing your desired options for the simulator. Currently, The CASPR GUI has four simulator windows:

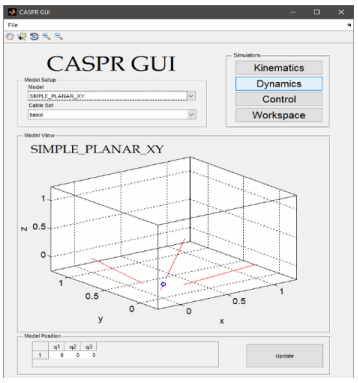
- The `control_GUI` window for running control simulators.
- The `kinematics_GUI` window for running inverse and forward kinematics.
- The `dynamics_GUI` window for running inverse and forward dynamics.
- The `workspace_GUI` window for running workspace analysis (note: currently not set up for wrench feasibility analysis).

As an example of how to use CASPR GUI, Figure 5 shows the necessary steps required in producing the same simulation as that of Section 3.1.2. It can be seen that this consists of the following steps:

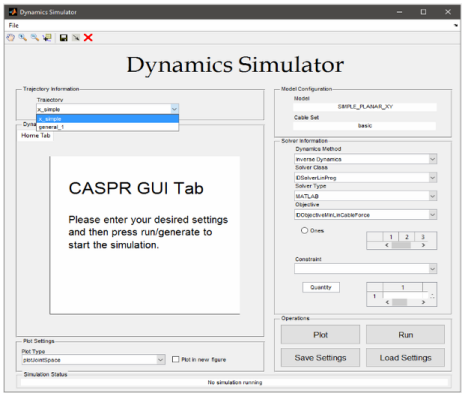
1. Select the model `SEGESTA` and use the `original` cable set.
2. Press the `Dynamics` button to open up the `dynamics_GUI`.
3. Open the `Trajectory` drop down menu and select the trajectory `x_simple`.
4. Open the `Dynamics Method` drop down menu and select inverse dynamics.
5. Set the inverse dynamics options tabs such that the options match Section 3.1.2.
6. Press the run button to start the simulation. The output for plot `plotCableForces` is shown in the figure.
7. If any additional plots are desired, select the plot using the plot tab and press the plot button (note that a simulation must have been conducted for this button to produce a plot).



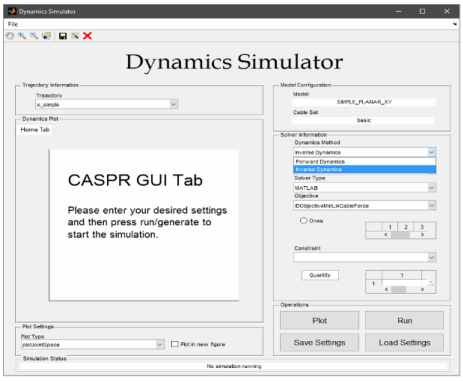
(a) Model Selection



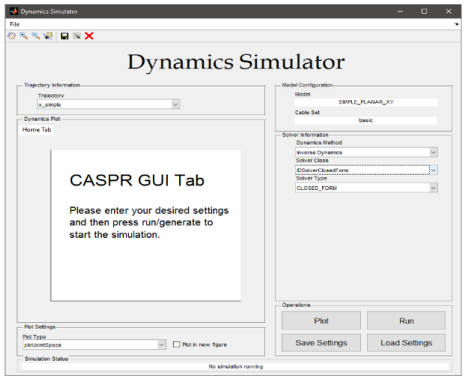
(b) Opening the Simulator



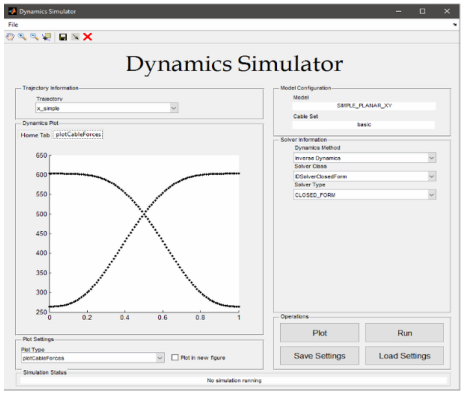
(c) Select the Trajectory



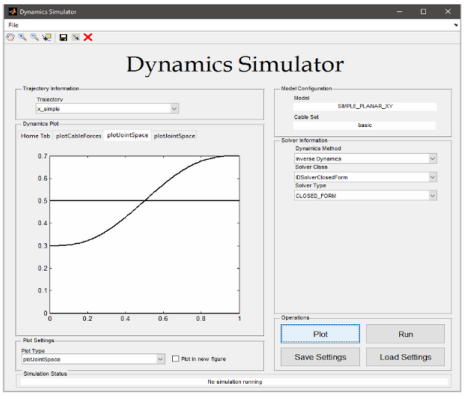
(d) Select Inverse Dynamics



(e) Set the Solver Options



(f) Run the simulator



(g) Add additional Plots

Figure 5: Steps to Run the Inverse Dynamics Simulator

4 Advanced Uses

4.1 Creating a New Analysis Algorithm

To be added.

4.2 Creating a New Joint Type

To be added.

4.3 Creating a New Cable Model

To be added.

4.4 Adding a New Algorithm to the GUI

To be added.

4.5 Unit Testing

In CASPR, newly added code can be tested to ensure that it doesn't break any existing code using the provided unit testing functions (contained in ∼/`unit_testing`). Currently, unit testing includes testing of the following modules

- System Model(including all joints and cable models)
- Model Configuration
- Inverse Dynamics

To run all unit tests, enter the command `CASPRTestScript` into the MATLAB console.

4.5.1 Creating a New Unit Test

To be added.

5 Important/Commonly Used Files and Folders

Below are the locations of a number of important files and folder for the operation of CASPR.

5.1 Additional Example Scripts

Additional example scripts for CASPR can be found in the following locations:

- Control example: ∼/scripts/examples/control/script_control_example.m
- Design optimisation example: ∼/scripts/examples/designOptimisation/script_design_example.m
- Forward dynamics example: ∼/scripts/examples/dynamics/script_FD_example.m
- Inverse dynamics example using a QP solver: ∼/scripts/examples/dynamics/script_ID_QP_example.m
- Forward kinematics example: ∼/scripts/examples/kinematics/script_FK_example.m
- Inverse kinematics example: ∼/scripts/examples/kinematics/script_IK_example.m
- Workspace example: ∼/scripts/examples/kinematics/script_WS_example.m
- Advanced workspace example: ∼/scripts/examples/kinematics/script_WS_example.m

5.2 Important Folder Locations

Here are the important folder locations to know:

- Collection of models: ∼/data/config/models/
- Collection of different analysis algorithms: ∼/src/Analysis/
- Collection of different joints: ∼/src/Model/Bodies/Joints/
- Collection of different cable models: ∼/src/Model/Cables/
- Collection of simulators: ∼/src/Simulation/

5.3 Important Files

5.3.1 Model

The model configuration and model object files for CASPR are located at:

- Model configuration: ∼/data/config/models/ModelConfig.m
- Model object: ∼/src/Model/SystemModel.m
- Model object for bodies: ∼/src/Model/Bodies/SystemModelBodies.m
- Model Object for cables: ∼/src/Model/Cables/SystemModelCables.m
- Base class for different types of joints: ∼/src/Model/Bodies/Joints/JointBase.m
- Base class for different types of cable models: ∼/src/Model/Cables/CableModelBase.m

5.3.2 Analysis

The base classes for different types of analyses types for CASPR are located at:

- Forward kinematics: ∼/src/Analysis/ForwardKinematics/FKAnalysisBase.m
- Inverse dynamics: ∼/src/Analysis/InverseDynamics/IDSolverBase.m
- Control: ∼/src/Analysis/Control/ControllerBase.m
- Workspace:
  - Condition: ∼/src/Analysis/Workspace/Conditions/WorkspaceConditionBase.m
  - Metric: ∼/src/Analysis/Workspace/Metrics/WorkspaceMetricBase.m

- Design optimisation (∼/src/Analysis/DesignOptimisation/):
  - Cable attachment optimisation (./CableAttachmentOptimisation/)
    - Objective function: ./ObjectiveFunctions/CableAttachmentOptimisationFnBase.m
    - Attachment point parameter: ./AttachmentPointParam/AttachmentPointParamBase.m

5.3.3 Simulation

The different simulators in CASPR are located within the ∼/src/Simulation/ folder.

References

[1] R. Babaghasabha, M. A. Khosravi, and H. D. Taghirad, "Adaptive control of KNTU planar cable-driven parallel robot with uncertainties in dynamic and kinematic parameters," in *Cable-Driven Parallel Robots*, ser. Mechanisms and Machine Science, A. Pott and T. Bruckmann, Eds. Springer International Publishing, 2015, vol. 32, pp. 145–159.

[2] W. Lim, S. H. Yeo, G.-M. Yang, and I.-M. Chen, "Design and analysis of a cable-driven manipulator with variable stiffness," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4519–4524.

[3] Q. Duan, V. Vashita, and S. K. Agrawal, "Effect on wrench-feasible workspace of cable-driven parallel robots by adding springs," vol. 86, pp. 201–210, 2015.

[4] C. Richter, R. H. Sören Jentzsch, J. A. Garrido, E. Ros, A. Knoll, F. Röhrbein, P. van der Smagt, and J. Conradt, "Scalability in neural control of musculoskeletal robots," *IEEE Robot. Autom. Mag.*, 2016.

[5] L. Gagliardini, S. Caro, M. Gouttefarde, and A. Girin, "A reconfiguration strategy for reconfigurable cable-driven parallel robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1613–1620.

[6] J. Lamaury and M. Gouttefarde, "Control of a large redundantly actuated cable-suspended parallel robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4659–4664.

[7] A. Pott, H. Mütherich, W. Kraus, V. Schmidt, P. Miermeister, and A. Verl, "Ipanema: A family of cable-driven parallel robots for industrial applications," in *Cable-Driven Parallel Robots*, ser. Mechanisms and Machine Science, A. Pott and T. Bruckmann, Eds. Springer International Publishing, vol. 12, pp. 119–134.

[8] J. S. Albus, R. V. Bostelman, and N. Dagalakis, "The NIST robocrane," vol. 10, no. 5, pp. 709–724, 1993.

[9] C. Reichert and T. Bruckman, "Unified contact force control approach for cable-driven parallel robots using an impedance/admittance control strategy," in *IFToMM World Congress*, 2015, pp. 645–654.

[10] R. Nan, D. Li, C. Jin, Q. Wang, L. Zhu, W. Zhu, H. Zhang, Y. Yue, and L. Qian, "The five-hundred-meter aperture spherical radio telescope (fast) project," *International Journal of Modern Physics D*, vol. 20, no. 06, pp. 989–1024, 2011.

[11] Y. Mao and S. K. Agrawal, "Design of a cable-driven arm exoskeleton (CAREX) for neural rehabilitation," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 922–931, 2012.

[12] D. Lau, D. Oetomo, and S. K. 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.

[13] A. Pott and V. Schmidt, "On the forward kinematics of cable-driven parallel robots," 2015, pp. 3182–3187.