Dynamics and Control of Modular and Self-Reconfigurable Robotic Systems

Eugen Meister, Alexander Gutenkunst, and Paul Levi
Institute of Parallel and Distributed Systems
University of Stuttgart, Germany
Email: {Eugen.Meister, Alexander.Gutenkunst, Paul.Levi}@ipvs.uni-stuttgart.de

***Abstract*—In this paper, we introduce a framework for automatic generation of dynamic equations for modular self-reconfigurable robots and investigate a few adaptive control strategies. This framework enables to analyse the kinematics, dynamics and control for both, serial and branched multi-body robot topologies with different dyad structures. The equations for kinematics and dynamics are automatically generated using geometrical formulation methods and recursive Newton-Euler method. Different benchmarks examples are used to evaluate serial and branched robot configurations. As control strategies, computed torque method linearisation method together with Extended Kalman Filter estimator are used. A graphical tool has been developed, that enables easy to use interface and functionalities such as graphically selecting of robot topologies, visual feedback of trajectories and parameters editing.**

***Keywords*—multi-body kinematics and dynamics, self-adaptive systems, automatic model generator, adaptive control.**

I. INTRODUCTION

Modular robotics has its origin in the late eighties with the robotic platforms such as CEBOT developed by Fukuda. Most of the existing modular robotic platforms can only be assembled into different configurations manually, however, a few novel platform designes are able to locomote and therefore to assemble into various configurations autonomously. The framework introduced in this paper is primarily developped to be applied on two modular robots Scout and Backbone [1], which have been developped in projects Symbrion [2] and Replicator [3], but can also be used for diverse other modular systems. The state of the art of modular robot system can be reviewed for example in [4] or in [5]. Autonomy is one of the key challanges for such systems and opens a new spectrum of possible application scenarios especially in dangerous and hazardous environments [6], where robots are able to operate without human intervention.

Modular robotic systems are advantageous in comparison to specialised robots because they can help to reduce the developmental and production costs and can also easily be adapted to different situations and applications. However, the complexity for modelling and control also grows rapidly with each additional degree of freedom (DOF). The dynamics of such systems differ from those systems, which are operating in a fixed environment and without the capability to reconfigure and move. Consequently, new methods are required, which can reduce the modelling complexity of kinematics and dynamics as well as for control design.

In classical mechanics, dynamical systems are usually described by setting up the equations of motion. The most common methods in the robotics are Newton-Euler [7], Lagrange [8], and Hamilton [9] formulations, all ending up with equivalent sets of equations. Different formulations may better suit for analysis, teaching purposes or efficient computation on robot.

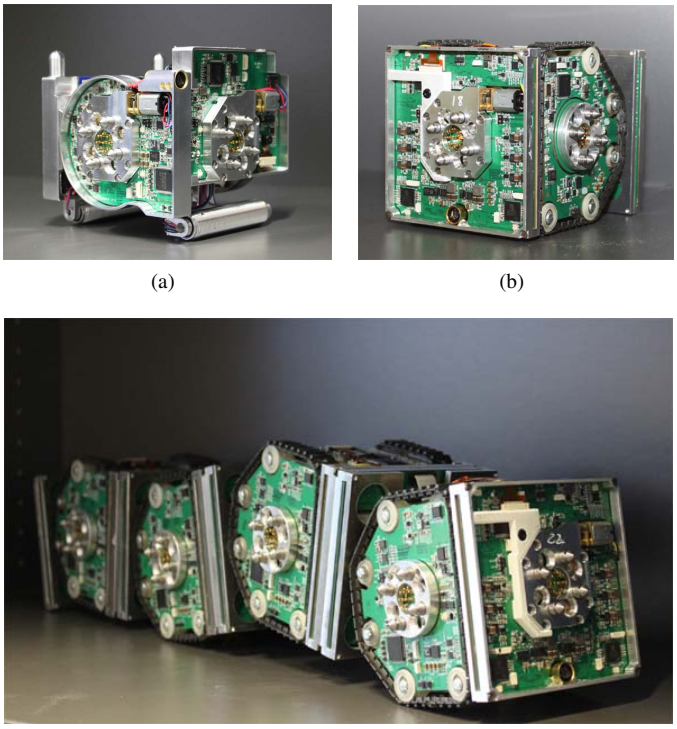
Lagrange's equations, for example, rely on energy properties of mechanical systems considering the multi-body system as a whole. This method is often used for study of dynamics properties and analysis in control design.

More applicable on real robots are the Newton-Euler formulation of dynamics. In this method, the dynamic equations are written separately for each body. This formation consists of two parts describing linear (Newton) and angular (Euler) motion [7].

In case of modular reconfigurable multi-body systems, obtaining of equations of motions can be a challenging and time consuming task. In this paper, we use a method using geometric formulation of motion equation, which was originally introduced by Park and Bobrow [10]. This method is based on recursive formulation of robot dynamics using recursive Newton-Euler combined with mathematical calculus of Lie groups and Lie algebras [11]. The description of motion is based on **twist** and **wrenches** summarizing angular and linear velocities as well as applied forces and moments in six-dimensional vectors [12]. This theory is also known as a Screw Theory [13] and was first published by Sir Robert Stawell Ball in the year 1900.

In the framework presented in this paper, the Newton's second law ($F = ma$) and Euler's equations are applied in two recursions: the forward (outward) and the backward (inward) recursion. Therefore, this approach is also called as a two-step approach. In the forward recursion, the velocities and accelerations of each link are iteratively propagated from a chosen base module to the end-links of multi-body system. During the backward recursion the forces and moments are propagated vice versa from the end-link to the base forming the equations of motions step-by-step. Recursive derivation of the equations makes it applicable to different types of robot geometries and moreover allows automatizing the process. There exist several publications generalising this method for variety of applications [14][15][16]. Most of efficient results use Newton-Euler algorithms, for example Luh, Walker, and Paul [17] expressed the equations of motion in local link

reference frames and by doing this reduced the complexity from $O(n^3)$ to $O(n)$. This approach was lately improved by Walker and Orin [18] providing more efficient recursive algorithm. Featherstone [19] proposed the recursive Newton-Euler equations in terms of spatial notation by combining the linear and angular velocities and wrenches into six dimensional vectors (Plücker notation). His 'Articulated Body Inertia' (ABI) approach becomes widely accepted in current research and is also of complexity $O(n)$.



(a) (b) (c)

Figure 1: Modular robots developed in projects Symbrion and Replicator [20]. (a) Backbone, (b) Scout, (c) Robots docked.

In the projects Symbrion and Replicator two autonomous modular self-reconfigurable robots have been developped, which are capable of building multi-robot organisms (Figure 1(c)) by aggregating or disaggregating into various different configurations [6]. In this paper, we orientate our approach on the method proposed by Chen and Yang [21], which allows generating the motion equations in closed form based on Assembly Incidence Matrix (AIM) representation for serial as well as for tree-structured modular robot assemblies. The approach has been adapted to modular robots Backbone and Scout, because the geometry of modules differs from those proposed by Chen and Yang. Autonomous deriving of the motion equations enables studying model based control strategies for modular and self-reconfigurable multi-robot asseblies. In this paper, the second focus is set to investigate the robustness and scalability of classical control strategies for non-linear systems. There are different possibilities how to deal with non-linearities in complex systems either using standard linearisation approach such as Jacobi linearisation or try to compenstate the non-linear terms through feedback linearisation techniques. In this paper, we use the second method, additionally combining it with Extended Kalman Filter (EKF), which allows to simulate the behaviour of real robots with limited sensor capabilities.

The paper is organized in the following way. In Section II, we give basic theoretical background about geometrical formulations for rigid body transformations. In Section III, we describe how the robot kinematics can be formulated for modular robots. Section IV describes representation techniques for self-reconfigurable robot assemblies. Section V contains the recursive approach for calculation of dynamics equations. In order to evaluate the approach a graphical user interface called MODUROB is built and is explained in Section VI. Section VII describes the evaluation of a branched robot organism. Section VIII contains the control strategies for self-reconfigurable robotic systems and present the results based on computed torque method and Extended Kalman Filter. Finally, Sections IX concludes the work and gives a short outlook.

II. Theoretical Background

For kinematics analysis two Lie groups play an important role, the Special Euclidean Group $SE(3)$ and the Special Orthogonal Group $SO(3)$. $SE(3)$ group of rigid body motions consists of matrices of the form

$$\begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where $R \in SO(3)$ is the group of 3×3 rotation matrices and $p \in \mathbb{R}^{3\times 1}$ is a vector.

Lie algebra is also an important concept associated with the Lie groups. Lie algebra of $SE(3)$, denoted as $se(3)$, is a tangent space at the identity element of G. It can be shown that the Lie algebra of SE(3) consists of matrices of the form

$$\begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{2}$$

where

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3}$$

Lie algebra is defined together with the bilinear map called Lie bracket, which satisfy following conditions:

- Skew-symmetry: $[a, b] = -[b, a]$.
- Jacobi identity: $[a, [b, c]] + [c, [a, b]] + [b, [c, a]] = 0$

If elements are square matrices, the Lie bracket is a matrix commutator $[A, B] = AB - BA$.

The connection between Lie Group $SE(3)$ and Lie algebra $se(3)$ is the exponential mapping [22], which maps $se(3)$ onto $SE(3)$. Exponential mapping allows an elegant way to formulate rigid body motions.

The exponential mapping $e^{\hat{s}q}$ can be interpreted as an operator that transforms a rigid body from their initial pose

to new pose combining rotations and translations at the same time [15]:

$$g_{ab}(q) = e^{\hat{s}q} g_{ab}(0), \tag{4}$$

where $g_{ab}(0) \in SE(3)$ is an initial pose and $g_{ab}(q)$ is the final pose. A twist associated with a screw motion is formulated as

$$s_i = \begin{bmatrix} -\omega_i \times p_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}, \tag{5}$$

where $\omega \in \mathbb{R}^{3\times1}$ is a unit vector denoting the direction of the twist axis, $v_i \in \mathbb{R}^{3\times1}$ is a unit vector facing in the direction of translation and $q_i \in \mathbb{R}^{3\times1}$ is an arbitrary point on the axis. Revolute joints perform only pure rotations about an axis. Therefore the twist has the form:

$$s_i = \begin{bmatrix} 0 \\ \omega_i \end{bmatrix}. \tag{6}$$

Analogous, the pure translation is much simpler,

$$s_i = \begin{bmatrix} v_i \\ 0 \end{bmatrix}. \tag{7}$$

Linear mapping between an element of a Lie group and its Lie algebra can be performed by the adjoint representation. When X is given by $X = (R, p) \in SE(3)$, then the adjoint map $Ad_X : se(3) \mapsto se(3)$ acting on $y \in se(3)$ is defined by $Ad_X(y) = XyX^{-1}$. In [15] is also shown that $Ad_X(y)$ admits the 6×6 matrix representation

$$Ad_X(y) = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{8}$$

where $\hat{p}$ is the skew-symmetric matrix representation of $p \in \mathbb{R}^3$. Linear mapping between an element of Lie algebra and its Lie algebra can be performed via the Lie bracket

$$ad_x(y) = [x, y]. \tag{9}$$

Given $x = (v_1, \omega_1) \in se(3)$, and $y = (v_2, \omega_2) \in se(3)$, the adjoint map admits corresponding 6×6 matrix representation

$$ad_x(y) = \begin{bmatrix} \hat{\omega}_1 & \hat{v}_1 \\ 0_{3\times3} & \hat{\omega}_1 \end{bmatrix} \begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix}. \tag{10}$$

Similar to twists that contain angular and linear velocities in one vector, wrenches or general forces are described in a similar way. Wrenches are vector pairs containing forces and moments acting on a rigid body.

$$F = \begin{pmatrix} f \\ \tau \end{pmatrix}, \tag{11}$$

where $f \in \mathbb{R}^3$ is a linear force component and $\tau \in \mathbb{R}^3$ represents a rotational component. In contrast to general velocities as elements of $se(3)$, wrenches are acting on $se(3)^*$, the dual space and therefore behaves as covectors. For this reason wrenches transform differently under a change of coordinates by using so called adjoint transformation,

$$F_a = Ad^T_{g_{ba}} F_b, \tag{12}$$

where forces acting on the body coordinate frame B are written with respect to coordinate frame A. In spatial representation, this is equivalent as if the coordinate frame A were attached to the object.

III. Robot Kinematics

In modular reconfigurable systems the robot kinematics varies according to modules that are connected to each other. In homogeneous systems with the same physical parameters the kinematics depends only on the orientations of modules relative to each other. Such modular design is advantageous for autonomous systems. Using heterogeneous modules the modelling complexity grows with the number of different modules that are used. Therefore, in most cases we assume identical or similar structure of the modules with similar physical properties. Both robots have been designed with similar geometry, same docking units and differ mostly in several insignificant properties such as number of sensors, different sensors or actuators. Nevertheless, even if the differences are not crucial, we speak about heterogeneous modules because of the additional Degree of Freedom (DOF) in Scout robot that is able to rotate the docking element even if only in limited way. Table I summarizes the mechanical properties of Backbone and Scout modular robots.

TABLE I: Major differences between mechanical properties of Scout/Backbone robots.

	Cubic Modules (I. M. Chen)	Backbone / Scout
Module types	homogeneous	heterogeneous
Joint types	revolute, prismatic	revolute
# ports	6	4
DOFs	rot.: $\pm180^\circ$	Backbone: bend.:$\pm90^\circ$; Scout: bend.:$\pm90^\circ$, rot.: $\pm180^\circ$

Using only revolute joints without any prismatic joints simplify additionally the autonomous and recursive model generator for kinematics and finally for the dynamics model.

A. Dyad Kinematics

Dyad dependencies are common in recursive formulations because the calculation proceeds from one module to the next comprising only two modules. The calculation is done from the base module to all pendant links. In the approach proposed by Chen and Yang [21], a dyad is defined as two adjacent modules (v_i,v_j) connected by a joint e_j (Figure 2(a)). A link assembly is defined by taking one of those modules (link) together with one joint. The relative position and orientation of one frame attached to one module with respect to next frame in the second module can be described under joint displacement by a homogeneous 4×4 matrix $H_{i,j}(q) \in SE(3)$:

$$H_{i,j}(q_j) = H_{i,j}(0) e^{\hat{s}_j q_j}, \tag{13}$$

where $\hat{s}_j \in se(3)$ is the twist of joint e_j and q_j is the angle of rotation. The relative position and orientation between the modules can be recognized by the robot through different kind of on-board sensors such as accelerometers, compass or by vision system. In project Symbrion and Replicator the geometry of the Backbone (Figure 1(a)) and Scout (Figure 1(b)) robots differ from modules proposed by Chen and Yang. Backbone and Scout modules consist of two moving parts and one main hinge motor placed inside of each module and for this reason already implies a complete dyad as defined by Chen and Yang in each robot. In order to adapt the recursive kinematics approach to Backbone and Scout robot we need to extend the system boundaries of a dyad (Figure 2(b)). Since the most weight is concentrated in the middle of the modules where the main motors are placed, the attached coordinate frames for each module coincide with the centre of mass. Because of two robots and hence two revolute joints in a dyad only one joint is involved into calculation in each recursive step.

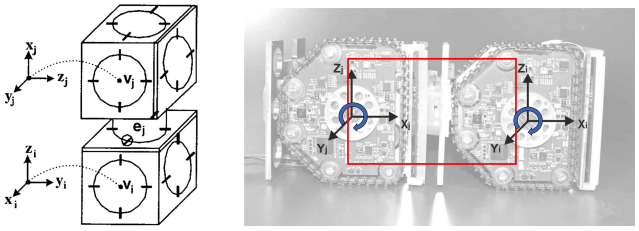


Figure 2: (a) A dyad defined by Chen and Yang [21], (b) Dyad for two Scout robots.

The orientation of axes of rotations depends on how robots are docked to each other. The relative pose can be described by 4×4 homogeneous matrix like in Equation (13).

B. Forward Kinematics

Forward kinematics for modular reconfigurable robotic systems determines the poses of the end-links providing joint angles as an input. In this section, we introduce the modelling technique for forward kinematics based on local frame representation of the Product-of-Exponential (POE) formula originally proposed in [23] or in [15]. This technique can be easily applied to tree-structured robots with many branches (e.g., multi-legged robots). Based on recursive dyad kinematics, the calculation can be done simultaneously for all branches. In this paper, all robots are considered to be cube shaped robots based on Backbone or Scout geometries consisting of one major DOF. In general case, the forward kinematics for serial connected robots can be obtained for an arbitrary number of links by simply multiplying the exponential maps as follows:

$$g_{st}(q) = e^{\hat{s}_1 q_1} e^{\hat{s}_2 q_2} e^{\hat{s}_3 q_3} \dots e^{\hat{s}_n q_n} g_{st}(0), \qquad (14)$$

where $\hat{s}_1$ to $\hat{s}_n$ have to be numbered sequentially starting with the chosen base module (Figure 3).

For a tree or branch structured robot configurations, the forward kinematics can be obtained in parallel way starting

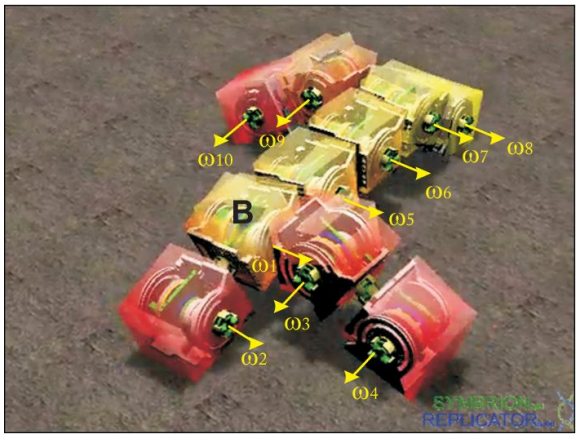


Figure 3: Multi-robot organism [24].

the calculation from a chosen base module to each pendant end-link in all branches. One possibility how the connecting order can be obtained is to use the AIM proposed by Chen and Yang [21]. For branched type of robots, two traversing algorithms are common to find the shortest paths: the Breadth-first search (BFS), and the Depth-first search (DFS) algorithms. The forward kinematic transformations for the branched robot configuration starting from base to each of the pendant links a_n of path k with m branches can be formulated as follows:

$$H(q_1, \dots, q_n) = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_k \\ \vdots \\ H_m \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ \vdots \\ \prod_{i-1}^{n} (H_{a_{i-1}a_i}(0) e^{\hat{s}_{a_i} q_{a_i}}) \\ \vdots \\ \dots \end{bmatrix}, \qquad (15)$$

where $H(q_1, \dots, q_n)$ represent all poses of all pendant end-links by using homogeneous 4×4 matrix representation.

IV. Robot Assembly Representations

Matrix notation is a powerful method to represent modular robots kinematic dependencies. The most common matrices used in robotics are the Adjacency and the Incidence matrix. Both matrices represent the connections between the neighbouring nodes. In [21], Chen proposes a method based on AIM that allows to represent the whole robot assembly consisting from links and joints additionally carrying the information about the type of robot and about used joints. A dynamic model for modular robot assembly is created autonomously from the AIM. This method was developed for a homogeneous kind of robots varying only in size with different joint possibilities including revolute or prismatic joints. Scout and Backbone robots contain only revolute joints however the number is not limited to one DOF. Therefore, the approach proposed in [21] cannot be directly used for this kind of modules and need to be adapted.

A. Adapted Assembly Incidence Matrix

The Backbone and the Scout robots are both cubic shaped robots, however provide only four sides that are equipped with docking units. Therefore, using the notation of gaming dice only ports $2-5$ are able to set a connection. A difference between modular robots proposed in [21] and the Scout/Backbone modules is that joints are not considered as a separate mechanical parts (joint modules), which are required to connect two modules, but rather are placed inside each of the modules. For these reasons each robot builds a full dyad already.

For simplicity, we allow docking only in horizontal plane and we also use the principle of gaming dice for side notations. Robot organisms have to go into initial configuration when additional robots decide to dock. Using this assumption, we distinguish between three major dyad configuration classes: the serial DS, the parallel DP and the orthogonal DO dyad class, where the second letter determines the axes of rotation of module j with respect to module i. A serial coupled dyad (DS) is given when the axes of rotation are in one line. When the rotational axes are parallel than the dyad becomes a member of a parallel class (DP). Finally, when the axes are orthogonal to each other, the robots are classified as the orthogonal to each other connected robot assembly (DO). This information can be easily extracted from the matrix and used for direct computation. Additionally, the symmetry of the platform allows neglecting the sign of the orientation because it does not affect the calculation. Table II summarizes all possible configurations considering that top and bottom side of the robots and hence the sides 1 and 6 of a gaming dice do not contain docking units.

TABLE II: Connections table between Scout and Backbone robots.

Set DS: Dyad: Serial Axes		**Set** DP: Dyad: Parallel Axes		**Set** DO: Dyad: Orthogonal Axes	
1^{st} Mod.	2^{nd} Mod.	1^{st} Mod.	2^{nd} Mod.	1^{st} Mod.	2^{nd} Mod.
2	2	3	3	2	3
2	5	3	4	2	4
5	2	4	3	3	2
5	5	4	4	3	5
				4	2
				4	5
				5	3
				5	4

The autonomous docking procedure is based either on IR sensor communication [25] or also can be fulfilled by using vision system [26][27]. Backbone and the Scout robots have one revolute joint as a major actuator, therefore, the information about the kind of actuators in the last row of the AIM is unnecessary. Instead, we use the last row for the three types of docking orientations for serial, parallel or orthogonal case. The last column in the AIM contains the information about the kind of robot, which is used. We denote the modified AIM as AIM_{SB}, where index 'SB' denotes the first letters of both robots: the Scout and the Backbone robot. In Figure 4, a small example of an organism and the corresponding graph is shown. The AIM_{SB} for this organism is shown in Figure 5.

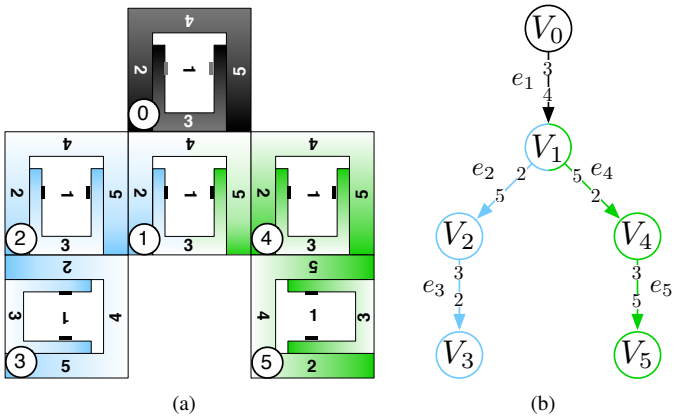


Figure 4: (a) Multi-robot organism example, (b) Directed graph representation.

$$AIM = \begin{array}{c|ccccc|c} & e_1 & e_2 & e_3 & e_4 & e_5 & \text{Links} \\ v_0 & 3 & 0 & 0 & 0 & 0 & S \\ v_1 & 4 & 2 & 0 & 5 & 0 & S \\ v_2 & 0 & 5 & 3 & 0 & 0 & S \\ v_3 & 0 & 0 & 2 & 0 & 0 & S \\ v_4 & 0 & 0 & 0 & 2 & 3 & S \\ v_5 & 0 & 0 & 0 & 0 & 5 & S \\ \hline \text{Joints} & DP & DS & DO & DS & DO & \end{array}$$

Figure 5: AIM of robot assembly from Figure 4(a).

B. Direct/Indirect Recursive Transformations

Structuring the kinematics dependencies into an AIM_{SB}, we are able to apply the transformations T_{ij} between the modules directly once the AIM is determined. By reusing the already calculated dependencies that are stored into lists it is fast and efficient to calculate the kinematics for big robot organisms. We use two lists: one list containing transformation results between consecutive joints, we call it a Direct-Transformation-List (DTL) and another list called Indirect-Transformation-List (IDTL) for non-consecutive transformations between joints however still in the same kinematics path.

In DTL as shown in Table III, each line represents one direct transformation. The first two columns indicate the connected modules and the last two columns hold the information, which sides are connected. IDTL contains the indirect transformations, which are calculated by two successive transformations ($T_{ij} = T_{ix} \cdot T_{xj}$). The first two columns denote the desired transformation. Next four columns hold two multiplied transformations that are stored in DTL or in IDTL. Both tables refer to the example shown in Figure 4.

A short example demonstrates the first traversing calcula-

TABLE III: Direct and indirect transformation list referred of example in Figure 4(a).

DTL			
T_{ij}		Sides	
i	j	from	to
0	1	3	4
1	2	2	5
2	3	3	2
1	4	5	2
4	5	3	5

IDTL					
T_{ij}		$=$	T_{ix}	$\cdot$	T_{xj}
i	j	i	x	x	j
0	2	0	1	1	2
0	3	0	2	2	3
0	4	0	1	1	4
0	5	0	4	4	5

tions using both lists:

$$\begin{aligned} T_{01} &= T_{01}(0)e^{\hat{s}_1 q_1} && \text{direct} \\ T_{12} &= T_{12}(0)e^{\hat{s}_2 q_2} && \text{direct} \\ T_{02} &= T_{01} \cdot T_{12} && \text{indirect} \\ T_{23} &= T_{23}(0)e^{\hat{s}_3 q_3} && \text{direct} \\ T_{03} &= T_{02} \cdot T_{23} && \text{indirect} \\ T_{14} &= T_{14}(0)e^{\hat{s}_4 q_4} && \text{direct} \\ T_{04} &= T_{01} \cdot T_{14} && \text{indirect} \\ T_{45} &= T_{45}(0)e^{\hat{s}_5 q_5} && \text{direct} \\ T_{05} &= T_{04} \cdot T_{45} && \text{indirect} \end{aligned} \tag{16}$$

This algorithm can be compared with DFS algorithm, providing a flexible way to calculate the order all possible transformations can be calculated during run-time.

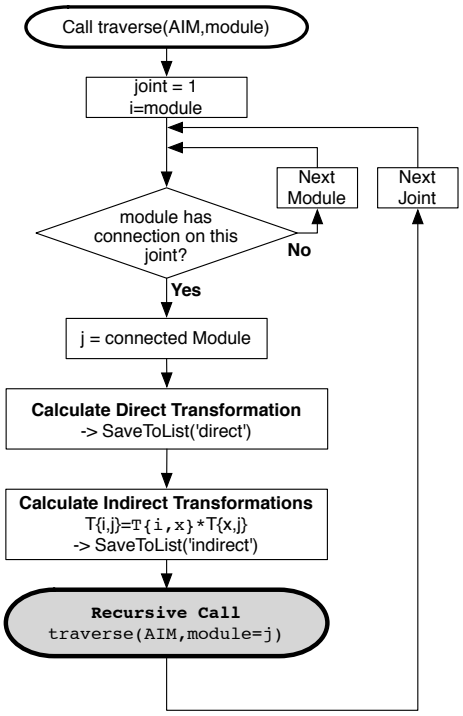


Figure 6: Traversing algorithm.

The flowchart of the algorithm is illustrated in Figure 6.

V. Modular Robot Dynamics

In general, two main branches of robot dynamics problems are mostly considered, namely the forward and the inverse dynamics problems. Forward dynamics play an important role in simulation of multi-body systems, also called as direct dynamics. Forward dynamics problem determines accelerations and external reaction forces of the system giving initial values for positions, velocities and applied internal/external forces, whereas the inverse dynamics problem determines the applied forces required to produce a desired motion. The first problem that appears in modular self-reconfigurable robotics is that the model of the robot assembly cannot be known a priori. Therefore, the robot should be able to generate its own model autonomously without human intervention.

A. Recursive Two-Step Approach

The original idea for recursive formulation and computation of the closed form equation of motion was introduced by Park and Bobrow [10]. The idea was extended by Chen and Yang by introducing the AIM. Starting with the AIM, that contains the information about how robots are assembled, the formulation of equations of motion is done in two steps: first applying forward transformation from base to the end-link, followed by the second recursion backwards from the end-link to the base module. Finally, we get the equation of motion in a closed-form. Before starting the recursion, some assumption and initializations should be done. In the first step, the system has to choose the starting module denoted as the *base* module. Starting from this module, the AIM is filled based on path search algorithms such as BFS or DFS. After the AIM is built and all paths are determined the recursive approach can be started.

- **Initialization:** Given V_0, $\dot{V}_0$, F^e_{n+1}

$$V_b = V_0 = (0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0)^T \tag{17}$$

$$\dot{V}_b = \dot{V}_0 = (0 \;\; 0 \;\; g \;\; 0 \;\; 0 \;\; 0)^T, \tag{18}$$

where V_b and V_0 denote generalized velocities expressed in the starting frame 0 and all other quantities are expressed in link frame i.

- **Forward recursion: for** $i = 1$ **to** n **do**

$$V_i = Ad_{H^{-1}_{i-1,i}}(V_{i-1}) + S_i\dot{q}_i, \tag{19}$$

$$\dot{V}_i = Ad_{H^{-1}_{i-1,i}}(\dot{V}_{i-1}) - ad_{Ad_{H^{-1}_{i-1,i}}(V_{i-1})}(S_i\dot{q}_i) + S_i\ddot{q}_i. \tag{20}$$

- **Backward recursion:** for $i = n$ to 1 do

$$F_i = Ad^*_{H^{-1}_{i-1,i}}(F_{i-1}) - F^e_i + \mathcal{M}_i\dot{V}_i - ad^*_{V_i}(\mathcal{M}_i V_i), \tag{21}$$

$$\tau_i = s_i^T F_i. \tag{22}$$

Here, $\mathcal{M}_i$ is the generalized mass matrix of the form

$$\mathcal{M}_i = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & mI_3 \end{bmatrix}, \tag{23}$$

where $\mathbf{I}$ is 3×3 inertia matrix and I is the identity matrix. The non-diagonal terms are zero because in our case the center

of mass coincides with the origin. F^e_{i+1} are the forces acting on the end-links of chained robots. This values can either be estimated or read from force or tactile sensors such as [28][29], which can be attached to the robots. F_i is the total generalized force traversed from link $i-1$ to i consisting of internal and external wrenches and τ_i is the applied torque by the corresponding actuator.

B. Equations of Motion

By expanding the recursive equations (19)-(22) in body coordinates, it can be shown that the equations for generalized velocities, generalized accelerations and forces can be obtained in matrix form:

$$V = TS\dot{q} \tag{24}$$
$$\dot{V} = T_{H_0}\dot{V}_0 + TS\ddot{q} + Tad_{S\dot{q}}V \tag{25}$$
$$F = T^T F^e + T^T M\dot{V} + T^T ad^*_V MV \tag{26}$$
$$\tau = S^T F \tag{27}$$

where

$$\begin{array}{lcl}
\dot{q} & = & column[\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n] \in R^{n\times 1} \\
\ddot{q} & = & column[\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_n] \in R^{n\times 1} \\
V & = & column[V_1, V_2, \ldots, V_n] \in R^{6n\times 1} \\
\dot{V} & = & column[\dot{V}_1, \dot{V}_2, \ldots, \dot{V}_n] \in R^{6n\times 1} \\
F & = & column[F_1, F_2, \ldots, F_n] \in R^{6n\times 1} \\
F^e & = & column[F^e_1, F^e_2, \ldots, F^e_n] \in R^{6n\times 1} \\
\tau & = & column[\tau_1, \tau_2, \ldots, \tau_n] \in R^{n\times 1} \\
S & = & diag[S_1, S_2, \ldots, S_n] \in R^{6n\times n} \\
M & = & diag[\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n] \in R^{6n\times 6n} \\
ad_{S\dot{q}} & = & diag[-ad_{S_1\dot{q}_1}, -ad_{S_2\dot{q}_2}, \ldots, -ad_{S_n\dot{q}_n}] \in R^{6n\times 6n} \\
ad^*_V & = & diag[-ad^*_{V_1}, -ad^*_{V_2}, \ldots, -ad^*_{V_n}] \in R^{6n\times 6n}
\end{array}$$

The n index represents the number of elements containing also virtual joints that are required to move the robot in space [30].

$$T_{H_0} = \begin{bmatrix} Ad_{H^{-1}_{0,1}} \\ Ad_{H^{-1}_{0,2}} \\ \vdots \\ Ad_{H^{-1}_{0,n}} \end{bmatrix} \in R^{6n\times 6} \tag{28}$$

$$T = \begin{bmatrix} I_{6\times 6} & 0_{6\times 6} & 0_{6\times 6} & \cdots & 0_{6\times 6} \\ Ad_{H^{-1}_{1,2}} & I_{6\times 6} & 0_{6\times 6} & \cdots & 0_{6\times 6} \\ Ad_{H^{-1}_{1,3}} & Ad_{H^{-1}_{2,3}} & I_{6\times 6} & \cdots & 0_{6\times 6} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Ad_{H^{-1}_{1,n}} & Ad_{H^{-1}_{2,n}} & Ad_{H^{-1}_{3,n}} & \cdots & I_{6\times 6} \end{bmatrix} \in R^{6n\times 6n}, \tag{29}$$

where T is the transmission matrix for the whole robot assembly. The elements $H_{i,j}$ in T_{H_0} and in T can be read out directly from the DTL and IDTL lists.

The closed-form equation of motion of the classical form

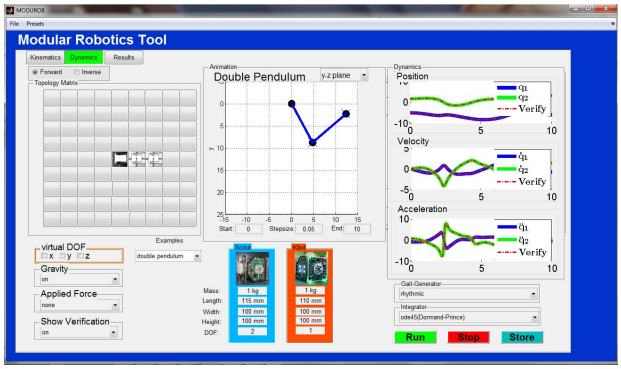
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau \tag{30}$$

is obtained by substituting the equations 24-27, where $M(q)$ is the mass matrix; $C(q,\dot{q})$ describes the Coriolis and centrifugal accelerations and $N(q)$ represents the gravitational forces as well as the external forces.

$$M(q) = S^T T^T MTS \tag{31}$$
$$C(q,\dot{q}) = S^T T^T (MTad_{S\dot{q}} + ad^*_V M)TS \tag{32}$$
$$N(q) = S^T T^T MT_{H_0}\dot{V} + S^T T^T F^e \tag{33}$$

VI. MODUROB - Modular Robotics Software Tool

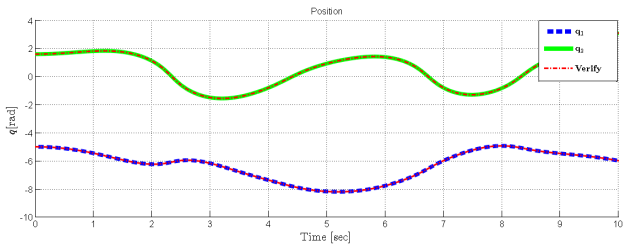
MODUROB is a tool built in MATLAB® that contains a possibility to build robot topologies by simply clicking on topology matrix grid (Figure 7). Currently, two types of robots are provided: the Backbone (Figure 1(a)) and the Scout robot (Figure 1(b)). For simplification, robots are only allowed to assemble or disassemble in planar configurations on the ground. The automatic model can be built in two ways: analytically or numerically. The symbolic formulation in MATLAB is done by using the symbolic toolbox. For solving of differential equations the user can choose between the numerical integrators that are provided by MATLAB. In order to move the robot in a joint space, different gait generators are provided either using rhythmic generators based on rhythmic functions [30] or gait generators that use chaotic map. We use an approach proposed in [31], that allows to generate periodic gaits that result from synchronization effects of coupled maps. Such approach can help to control complex multi-body structures by mapping the active joints to an individual chaotic driver [32].



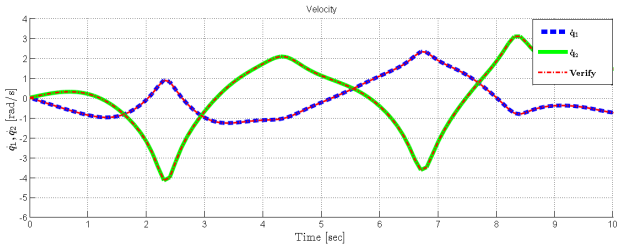
(a)

Figure 7: Benchmark example of Double pendulum.

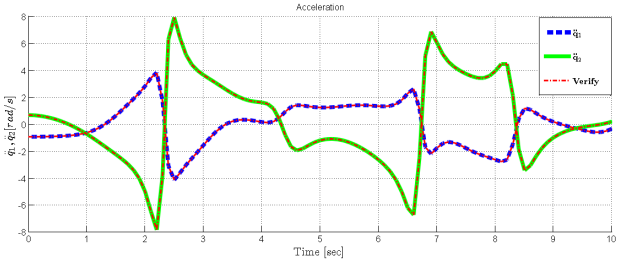
For evaluation or benchmarking of the framework two examples are implemented based on Lagrangian equations and can be compared with the geometrical approach. One example is a double pendulum example (Figure 7(a)) for example derived in [33] and the second is an extended pendulum that is movable on a shaft like a crane presented in [1].



(a)



(b)



(c)

Figure 8: Verification (dashed line) of geometrical POE approach with Lagrangian method using double pendulum model. (a) Position; (b) Velocity; (c) Accelerations.

The example (Figure 8) shows absolutely identical behaviour between implementation based on Lagrangian equations and with the geometrical approach based on twist and wrenches.

VII. Evaluation of Branched Multi-body Systems

In the previous section, two basic examples, a pendulum and a crane example are used to evaluate the correct functionality of the geometrical approach proposed in this thesis. However, pundulum-like structures are the simplest form of robot assemblies, therefore, another benchmark with branched robot configuration (Figure 9) has been generated to make deeper evaluation of the framework.

While single or double pendulum example code can be easily found and downloaded from many sources, e.g. from [33], the code for more complex structures of multi-body systems are hardly available. For this reason, in order to evaluate other robot configurations, a new branched modular robot has been

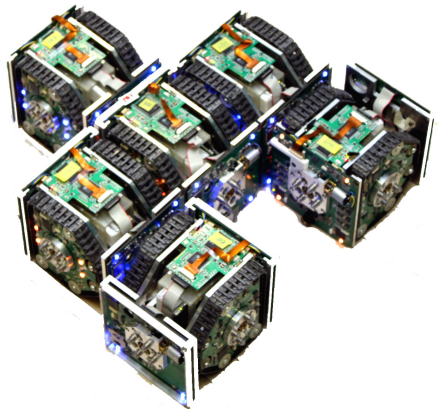


Figure 9: Crab-like multi-robot structure assembled with Scout robots.

modelled within the MATLAB SimMechanics toolbox [34]. This tool allows to simulate physical properties and dynamics of multi-body systems by use of Euler-Lagrangian modelling technique [35].

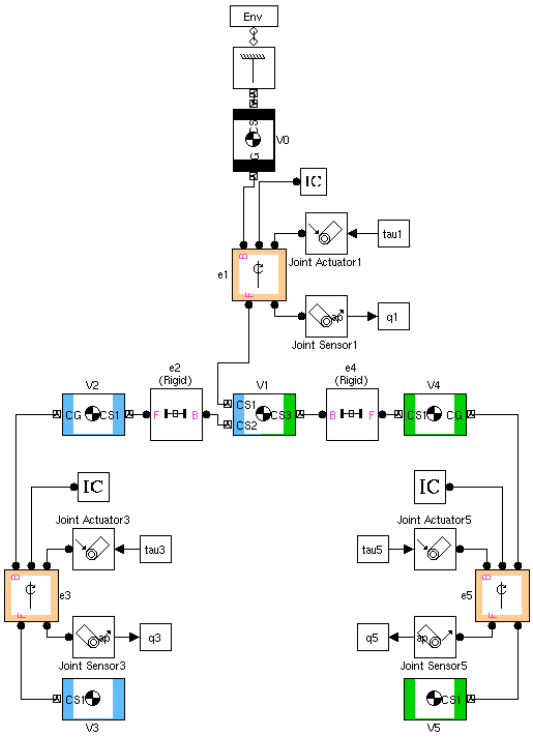


Figure 10: Simulink SimMechanics model of crab-like robot.

Figure 10 present a SimMechanics model of a crab-like robot organism, which is shown in Figure 9. The corresponding AIM and the graph representation of this organism structure can be reviewed in Figures 4-5. This structure has been chosen because it contains all three types of dyad structures (serial,

parallel and orthogonal), which are also listened in Table II. This structure implies also the case, where due to mechanical limitations, robots are not able to move with respect to each other and in this case the state space of dynamic model needs to be reduced. For this reason the framework should detect such cases and adapt the framework accordingly.

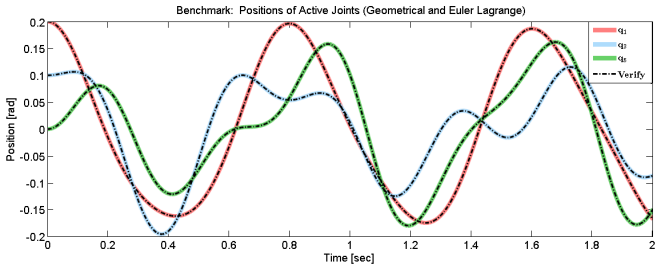


Figure 11: Position, velocity and acceleration of links based on crab-like robot structure.

In order to evaluate the proposed geometrical framework, identical robot structure, which is used in SimMechanics simulation (Figure 10) is also modelled with our model generator. The behavaiour of dynamics of both models can be examined in Figure 11. In this plot, it can be observed that both models produce identical set of results that similar like in the previous section, also proves the correctness of the framework.

VIII. Control of Multi-body Systems

Classical control theory provides a huge spectrum of controller design strategies for linear and non-linear systems including for example methods such as sliding mode control [36], optimal control [37], robust control [38], backstepping control [39] and different adaptive control mechanisms [40][41][42]. In the previous sections, an automatic model generator is presented, that enables to generate the dynamics model of the multi-robot organism based on geometrical formulation of motion equations. The generated model is a non-linear model, and to design a controller for a robot that is able to reconfigure and hence requires a new model representation is a huge challenge. One of the most used methods to deal with non-linear models is to linearise the model at a certain operation point by use of Jacobian linearisation techniques [43]. These methods are most common used in control theory, because it distinctly reduces modelling and computational complexity. During the last decades, the technological progress in microprocessor technologies opens new opportunities to apply methods for nonlinear control design techniques direct on embedded systems. Feedback linearisation also known as exact linearisation is an approach of nonlinear control design, that has attracted lots of research in recent years and is probably the first choice for mechanical systems containing only active joints.

In this section, a non-linear control design for a multi-body reconfigurable robots is presented based on computed torque method and EKF.

A. Inverse Dynamics

Most of robotic systems has a non-linear character. One of the concepts that currently becomes popular is the **feedback linearisation** [44] control strategy. This technique is a generalized concept and a special branch of it in robotics is also known as **inverse dynamics** or **computed torque** [45] control.

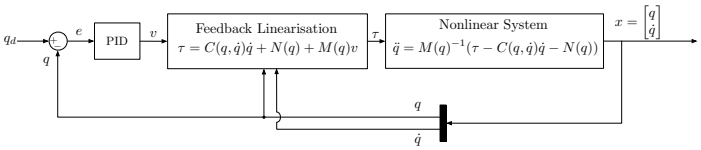


Figure 12: Structure of Computed Torque Method.

The block diagram of this approach is depicted in Figure 12. The inverse dynamics problem can be formulated in a joint space as follows:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q), \tag{34}$$

which applied to Equation (30) yields to $\ddot{q} = v$. The new control input v needs to be designed and is typically chosen as:

$$v = \ddot{q}_d - \underbrace{K_0(q - q_d)}_{e_q} - \underbrace{K_1(\dot{q} - \dot{q}_d)}_{\dot{e}_q}, \tag{35}$$

where K_0 and K_1 are positive definite matrices. The error dynamics for the closed loop system can be formulated as:

$$\ddot{e}_q + K_1\dot{e} + K_0 e_q = 0, \tag{36}$$

where matrices K_0 and K_1 are the gain matrices. The error dynamics can be achieved to be exponentially stable by the proper choice of K_0 and K_1. In the framework presented in this paper, the torque can be computed for every selected robot configuration. The limitations are not given by the framework itself, but rather by the computation time for very big robot structures. Figure 13 shows step responses of the active joints in a crab-like robot structure from the Figure 9. Due to selected

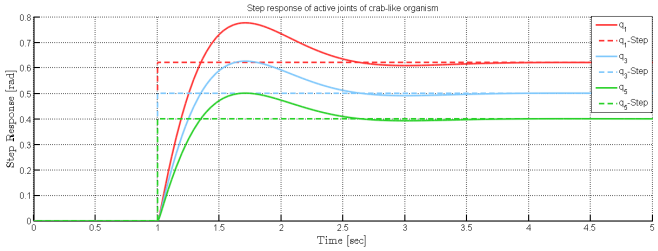


Figure 13: Step response of active joints of simulated crab-like robot configuration with applied computed torque linearisation.

robot configuration and its mechanical restrictions the number of active joints is reduced to three DOFs. As a conclusion we can summarise that with computed torque method and PID control we are able to stabilise the non-linear system in an acceptable short time.

B. States Estimation with EKF

Mutli-robot organisms, which are introduced in this paper are built of modular robot with one rotational degree of freedom. The electronics of the modular robots [46] Backbone or Scout allows to detect the absolute hinge angle of rotation by the use of hall sensors, which are placed inside the hinge motor. However, measuring only the hinge position is not enough to get the full set of states for a nonlinear system model. One of the promising approaches to estimate the missing or noisy states of a dynamical system is the Kalman filter. The EKF [47] is the nonlinear version of the classical Kalman filter. This recursive predictive filter runs in two steps: the prediction and the correction step. In this section, we give a short introduction of theoretical background of EKF and present the achieved results.

The structure of the model containing computed torque linearisation from previous section and the EKF are illustrated in the Figure 14.

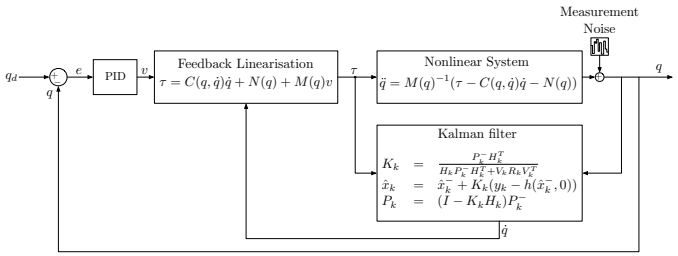


Figure 14: Schematic of system structure with computed torque, EKF and PID control.

Consider the nonlinear discrete-time model written in the standard state-space representation:

$$\begin{aligned} x_k &= f(x_{k-1}, w_{k-1}) \in \mathbb{R}^n, \\ y_k &= h(x_k, v_k), \end{aligned} \tag{37}$$

where k describes the time step; x_k is a vector of actual states; y_k is the actual output vector; w_k and v_k the system and the output noise.

The actual state and measurement vector can be approximated as

$$\begin{aligned} \hat{x}_k^- &= f(\hat{x}_{k-1}, 0), \\ \hat{y}_k &= h(\hat{k}_k^-, 0). \end{aligned} \tag{38}$$

The linear approximation of the Equation (37) can be formulated as:

$$\begin{aligned} x_k &\approx \hat{x}_k^- + A(x_{k-1} - \hat{x}_{k-1}^-) + W w_{k-1}, \\ y_k &\approx y\hat{x}_k^- + H(x_k - \hat{x}_k^-) + V v_k, \end{aligned} \tag{39}$$

where A_k is the Jacobian matrix defined as

$$A_{ij} = \frac{\partial f_i}{\partial x_j}(\hat{x}_{k-1}, 0). \tag{40}$$

W_{ij} is the Jacobian matrix of partial derivatives of f with respect to w:

$$W_{ij} = \frac{\partial f_i}{\partial w_j}(\hat{x}_{k-1}, 0). \tag{41}$$

H_{ij} is the Jacobian matrix of partial derivatives of h with respect to x:

$$H_{ij} = \frac{\partial h_i}{\partial x_j}(\hat{x}_k, 0). \tag{42}$$

V_{ij} is the Jacobian matrix of partial derivatives of h with respect to v:

$$V_{ij} = \frac{\partial h_i}{\partial x_j}(\hat{x}_k, 0). \tag{43}$$

The a priory prediction error can be defined as:

$$\hat{e}_{x_k}^- = x_k - \hat{x}_k^-. \tag{44}$$

The a priory measurement residual is:

$$\hat{e}_{y_k}^- = y_k - \hat{y}_k^-. \tag{45}$$

Using both equations (44) and (45), the prediction and measurement error can be approximated:

$$\begin{aligned} \hat{e}_{x_k}^- &\approx A(x_{k-1} - \hat{x}_{k-1}) + \epsilon_k, \\ \hat{e}_{y_k}^- &\approx H\epsilon_{x_k} + \eta_k, \end{aligned} \tag{46}$$

where ϵ_k and η_k are independent random variable with zero mean and covariance matrices WQW^T and VRV^T.

The second Kalman process that models the error estimates over time can be obtained by

$$\hat{x}_k = \hat{x}_k^- + \hat{e}_{x_k}. \tag{47}$$

The Kalman filter equation used to estimate $\hat{e}_k$ becomes:

$$\hat{e}_{x_k} = K_k \hat{e}_{y_k}^-. \tag{48}$$

Substituting the Equation (48) back into (47) leads to:

$$\hat{x}_k = \hat{x}_k^- + K_k \hat{e}_{y_k}^-. \tag{49}$$

The prediction step can be computed as:

$$\begin{aligned} \hat{x}_k^- &= f(x_{k-1}, 0), \\ P_k^- &= A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T. \end{aligned} \tag{50}$$

Finally, the correction step becomes:

$$\begin{aligned} K_k &= \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R_k V_k^T}, \\ \hat{x}_k &= \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-, 0)), \\ P_k &= (I - K_k H_k) P_k^-. \end{aligned} \tag{51}$$

The correction step corrects the state and covariance estimates with the measurement y_k.

C. Example

In the previous section, computed torque method (Figure 12) is applied to linearise the non-linear system 34. However, on a real robot not all states can be measured and therefore the EKF as an estimator has been included into the system. The structure of the system together with EKF is illustrated in Figure 14. In this example, the crab-like robot configuration is used again to investigate the system behaviour. Figures 15(a)-15(d) show the smulated output, states, estimate states and the error, which is the substraction between the real and estimated states.

For this example the matrices Q, W, R, V have been selected as:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}, \quad (52)$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (53)$$

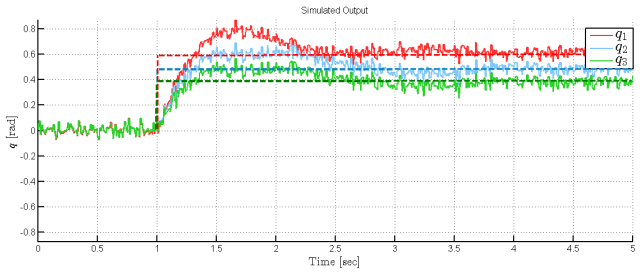
$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (54)$$

$$V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (55)$$

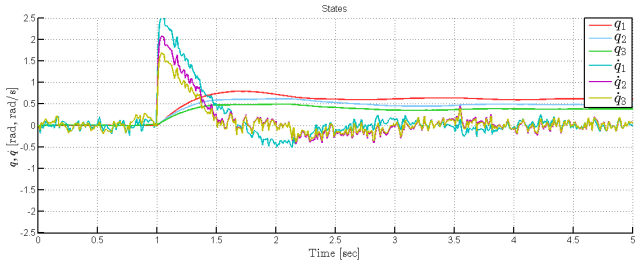
Looking the results produced by the system that uses the EKF, we can assume that the framework enables correct estimations of missing or noisy states and therefore coclude that this control strategy can be used in modular multi-body robot configurations.

IX. Conclusion and Future Work

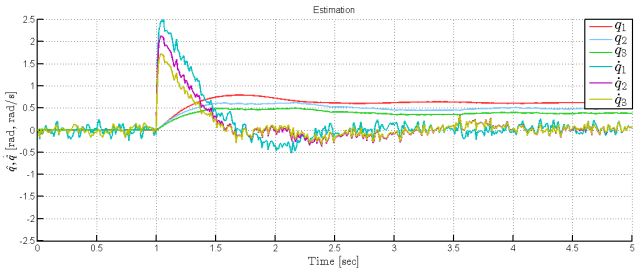
In this paper, we demonstrate a MATLAB framework that enables to analyse the kinematics, dynamics and control of modular self-reconfigurable robots. The autonomous calculation of self-adaptive model is based on a recursive geometrical approach. The proposed two-step approach for autonomous generation of motion equations was inspired by the work from Chen and Yang and has been modified and adapted to the needs of robot modules developed in projects Symbrion and Replicator. This paper is an extended version of the work presented in [1] and has been extended with non-linear control methods and with a state estimator for values that cannot be measured on a real mashine. The analysis and evaluation of the model and control strategies are inalienable before porting the software to the robots. A corresponding ongoing C++ framework, which will run on robots is currently in development and will be presented in a future publications.



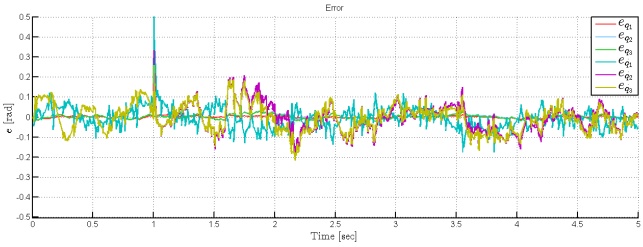
(a)



(b)



(c)



(d)

Figure 15: Simulation results for crab-like robot configuration. (a) Step response of active joints with simulated sensor noise; (b) system states; (c) estimated states; (d) error.

Acknowledgment

The "SYMBRION" project is funded by the European Commission within the work programme "Future and Emergent Technologies Proactive" under the grant agreement no. 216342. The "REPLICATOR" project is funded within the work programme "Cognitive Systems, Interaction, Robotics" under the grant agreement no. 216240.

REFERENCES

[1] E. Meister and A. Gutenkunst. Self-adaptive framework for modular and self-reconfigurable robotic systems. In *Proc. of the Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2012)*, Nice, France, 2012.

[2] SYMBRION. *SYMBRION: Symbiotic Evolutionary Robot Organisms, 7th Framework Programme Project No FP7-ICT-2007.8.2*. European Communities, 2008-2012.

[3] REPLICATOR. *REPLICATOR: Robotic Evolutionary Self-Programming and Self-Assembling Organisms, 7th Framework Programme Project No FP7-ICT-2007.2.1*. European Communities, 2008-2012.

[4] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G.S. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine, Magazine*, pages 43–52, 2007.

[5] S. Kernbach, O. Scholz, K. Harada, S. Popesku, J. Liedke, H. Raja, W. Liu, F. Caparrelli, J. Jemai, J. Havlik, E. Meister, and P. Levi. workshop on "modular robots: State of the art". In J. Guerrero, editor, *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA-2010)*, pages 1–10, Anchorage, Alaska, 2010.

[6] P. Levi and S. Kernbach, editors. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer-Verlag, 2010.

[7] M. D. Ardema. *Newton-Euler dynamics*. Springer, 2005.

[8] R. A. Howland. *Intermediate dynamics: a linear algebraic approach*. Mechanical engineering series. Springer, 2006.

[9] W. Greiner. *Classical Mechanics: Systems of Particles and Hamiltonian Dynamics*. Springer, 2010.

[10] F. C. Park and J. E. Bobrow. A recursive algorithm for robot dynamics using lie groups. In *Proc. of IEEE Conference on Robotics and Automation*, pages 1535–1540, 1994.

[11] W. Pfeifer. *Lie Algebra Su(N):*. Birkhuser, 2003.

[12] J. M. Selig. *Geometric Fundamentals of Robotics (Monographs in Computer Science)*. Springer-Verlag, 2004.

[13] R. Ball. *A Treatise on the Theory of Screws*. Cambridge University Press, 1900.

[14] I.-M. Chen. *Theory and Applications of Modular Reconfigurable Robotics Systems*. PhD thesis, Calfornia Institute of Technology, CA, 1994.

[15] R. M. Murray, Z. Li., and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[16] S. R. Ploen. *Geometric Algorithms for the Dynamics and Control of Multibody Systems*. PhD thesis, 1997.

[17] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980.

[18] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 1982.

[19] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, 2008.

[20] S. Kernbach, F. Schlachter, R. Humza, J. Liedke, S. Popesku, S. Russo, R. Matthias, C. Schwarzer, B. Girault, and ... P. Alschbach. Heterogeneity for increasing performance and reliability of self-reconfigurable multi-robot organisms. In *In Proc. IROS-11*, 2011.

[21] I.-M. Chen and G. Yang. Automatic model generation for modular reconfigurable robot dynamics. *ASME Journal of Dynamic Systems, Measurement, and Control*, 120:346–352, 1999.

[22] J. M. Selig. *Geometric Fundamentals of Robotics*. Monographs in Computer Science. Springer, 2005.

[23] R. Brockett. *Mathematical theory of net-works and systems*, chapter Robotic manipulators and the product of exponential formula, pages 120–129. Springer, New York, 1984.

[24] L. Winkler and H Wörn. Symbricator3d a distributed simulation environment for modular robots. In Ming Xie, editor, *Intelligent Robotics and Applications, Lecture Notes in Computer Science*, pages 1266–1277, 2009.

[25] W. Liu and A. F. T. Winfield. Autonomous morphogenesis in self-assembling robots using ir-based sensing and local communications. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 107–118, Berlin, Heidelberg, 2010. Springer-Verlag.

[26] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil. Simple, yet Stable Bearing-only Navigation. *Journal of Field Robotics*, October 2010.

[27] R. Saatchi M. S. Ahmed and F. Caparrelli. Support for robot docking and energy foraging - a computer vision approach. In *Proc. of 2nd International Conference on Pervasive and Embedded Computing and Communication Systems*, 2012.

[28] E. Meister and D. Kryvoukhov. An artificial tactile sensor skin for modular robots. In P. Fisette J.C. Samin, editor, *Proc. of Multibody Dynamics 2011, ECCOMAS Thematic Conference*, Brussels, Belgium, 2011.

[29] E. Meister and I. Zilberman. Fuzzy logic based sensor skin for robotic applications. In *Proc. of the The 14th International Conference on Artificial Intelligence (ICAI 2012)*, Las Vegas, USA, 2012. IEEE Computer Society Press.

[30] E. Meister, S. Stepanenko, and S. Kernbach. Adaptive locomotion of multibody snake-like robot. In P. Fisette J.C. Samin, editor, *Proc. of Multibody Dynamics 2011, ECCOMAS Thematic Conference*, Brussels, Belgium, 2011.

[31] S. Kernbach, E. Meister, F. Schlachter, and O. Kernbach. Adaptation and self-adaptation of developmental multi-robot systems. *International Journal On Advances in Intelligent Systems*, 3:121–140, 2010.

[32] S. Kernbach, P. Levi, E. Meister, F. Schlachter, and O. Kernbach. Towards self-adaptation of robot organisms with a high developmental plasticity. In J. Guerrero, editor, *Proc. of the First IEEE International Conference on Adaptive and Self-adaptive Systems and Applications (IEEE ADAPTIVE 2009)*, pages 180–187, Athens/Glyfada, Greece, 2009. IEEE Computer Society Press.

[33] J. E. Hasbun. *Classical Mechanics With MATLAB Applications*. Jones & Bartlett Publishers, 1 edition, March 2008.

[34] W. D. Pietruszka. *MATLAB Und Simulink in Der Ingenieurpraxis: Modellbildung, Berechnung Und Simulation*. Lehrbuch : Maschinenbau. Vieweg+teubner Verlag, 2006.

[35] G. D. Wood and D. C. Kennedy. *Simulating Mechanical Systems in Simulink with SimMechanics.* Mathworks, The MathWorks, Inc.,, 2003. http://www.mathworks.com.

[36] S. V. Emel'janov. *Automatische Regelsysteme mit veränderlicher Struktur.* Akademie-Verl, 1971.

[37] D. E. Kirk. *Optimal Control Theory: An Introduction*. Dover books on engineering. Dover Publications, 2004.

[38] J. Ackermann and P. Blue. *Robust Control: The Parameter Space Approach*. Communications and Control Engineering. Springer, 2002.

[39] J. Zhou and C. Wen. *Adaptive Backstepping Control of Uncertain Systems: Nonsmooth Nonlinearities, Interactions Or Time-Variations*. Lecture Notes in Control and Information Sciences. Springer, 2008.

[40] A. S. I. Zinober and D. H. Owens. *Nonlinear and Adaptive Control: Ncn4 2001*. Lecture Notes in Control and Information Sciences. Springer, 2002.

[41] M. Krstić, I. Kanellakopoulos, and P.V. Kokotović. *Nonlinear and adaptive control design*. Adaptive and learning systems for signal processing, communications, and control. Wiley, 1995.

[42] R. Freeman. Nonlinear and adaptive control with applications. *Control Systems, IEEE*, 31(2):90–92, 2008.

[43] H. P. Geering. *Regelungstechnik: Mathematische Grundlagen, Entwurfsmethoden, Beispiele*. Springer-Lehrbuch. Springer-Verlag GmbH, 2003.

[44] I. Alberto. *Nonlinear Control Systems: An Introduction (Communications and Control Engineering)*. Springer, 1994.

[45] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Gale virtual reference library. Springer, 2008.

[46] E. Meister, O. Scholz, J. Jemai, J. Havlik, W. Liu, S. Karout, G. Fu, and S. Kernbach. *Computation, Distributed Sensing and Communication.* Springer-Verlag, 2010.

[47] G. Welch and G. Bishop. An introduction to the Kalman filter. 1995.