Self-Adaptive Framework for Modular and Self-Reconfigurable Robotic Systems

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

***Abstract*—In this paper, we introduce a framework for automatic generation of dynamic equations for modular self-reconfigurable robots. The equations for kinematics and dynamics are generated recursively in two steps by using geometrical formulations and recursive Newton-Euler method. This framework has the purpose to analyse the kinematics and dynamics for serial as well as for branched multibody robot topologies with different dyad structures. A multi-functional and easy to use graphical interface provides functionalities such assembling of topologies, visual feedback of trajectories and parameters editing. Two benchmark examples show, that the proposed framework results coincide with the results produced by classical Lagrangian method.**

***Keywords*-multi-body kinematics and dynamics; self-adaptive systems; automatic model generator.**

I. INTRODUCTION

Self-reconfigurable modular robots [1] open a spectrum of applications especially in dangerous and hazardous environments [2]. Self-reconfiguration is necessary when robots are operating completely autonomously without human intervention. Modular systems are on the one hand advantageous in comparison to specialized systems because they are adaptable to different situations and applications; however on the other hand, the complexity for modelling and control can grow rapidly.

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, Lagrange, and Hamilton [3] 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 multibody system as a whole [4]. This method is often used for study of dynamics properties and analysis in control design.

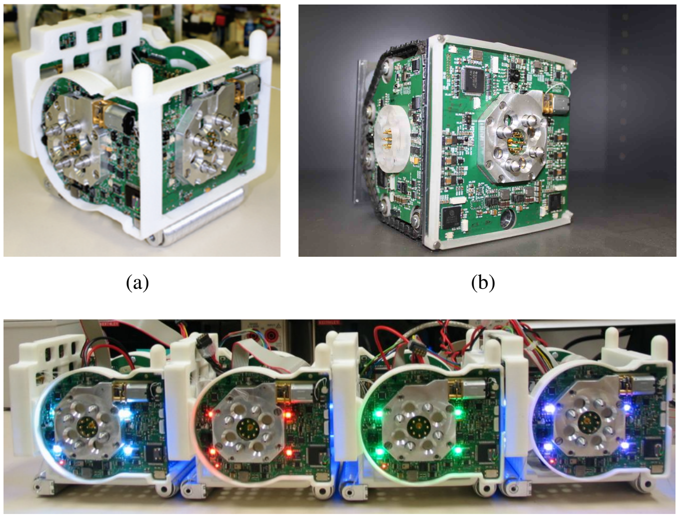
More applicable on real robots are the Newton-Euler formulation of dynamics. In this method, the dynamic equations are written separately for each body. This formation consists of two parts describing linear (Newton) and angular (Euler) motion [5].

In case of modular reconfigurable multibody 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 the equation of motion originally introduced by Park and Bobrow [6]. This method is based on recursive formulation of robot dynamics using recursive Newton-Euler combined with mathematical calculus of Lie groups and Lie algebras. 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.

In this approach, 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, we speak about 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 multibody 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 generalizing this method for variety of applications [7], [8], [9]. Most of efficient results use Newton-Euler algorithms, for example Luh, Walker, and Paul [10] expressing the equations of motion in local link reference frames and by doing this reduce the complexity from $O(n^3)$ to $O(n)$. This approach was lately improved by Walker and Orin [11] providing more efficient recursive algorithm. Featherstone [12] 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 $O(n)$.

In the projects Symbrion [14] and Replicator [15], we develop autonomous modular reconfigurable robots that are capable to build multi-robot organisms by aggregating/disaggregating into different topologies [2]. In this paper we orientate our approach on the method proposed by Chen and Yang [16], 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.

The paper is organized in the following way. In Section II, we give basic theoretical background about geometrical formulation for rigid body transformations. In Section III, we describe how the robot kinematics can be formulated for modular robots. In Section IV, robot assembly representation technique is introduced. Section V contains the recursive



(a) (b)

(c)

Fig. 1. Modular robots developed in projects Symbrion and Repicator [13]. (a) Backbone, (b) Scout, (c) Robots docked.

approach for calculation of dynamics equations. In order to evaluate the approach a graphical user interface (GUI) called MODUROB is built and is explained in Section VI. Finally, Sections VII 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 consist 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, which maps $se(3)$ onto $SE(3)$. Exponential mapping allows an elegant way to formulate rigid body motions. The formula originates from the solution of the time-invariant linear differential equation for velocity $\dot{p}$ of a point that rotates about an axis ω

$$\dot{p}(t) = \omega \times p(t) = \hat{\omega} p(t). \tag{4}$$

By integrating the equation we receive

$$p(t) = e^{\hat{\omega} t} p(0), \tag{5}$$

where $p(0)$ is the initial position at $t = 0$ of the point. $\hat{\omega} \in so(3)$ is a skew symmetric matrix and the $e^{\hat{\omega} t}$ is the so-called the matrix exponential

$$e^{\hat{\omega} t} = I + \hat{\omega} t + \frac{(\hat{\omega} t)^2}{2!} + \frac{(\hat{\omega} t)^3}{3!} + \ldots \tag{6}$$

Considering rotations with unit velocity ($\|\omega\| = 1$), the net rotations can be formulated as follows:

$$e^{\hat{\omega} q} = I + q\hat{\omega} + \frac{q^2}{2!}\hat{\omega}^2 + \frac{q^3}{3!}\hat{\omega}^3 + \ldots \tag{7}$$

Using the Rodrigues' formula, a closed-form expression of this formula can be obtained without computing the full matrix exponent and therefore more efficient from the computational point of view.

$$e^{\hat{\omega} q} = I + \hat{\omega} + sinq + \hat{\omega}^2 (1 - cosq). \tag{8}$$

The robot kinematics can be obtained by using the fact that rigid body motion can be achieved by a rotation about an axis combined with a translation parallel to it (Chasles's Theorem) [17].

In this case, 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

$$g_{ab}(q) = e^{\hat{s} q} g_{ab}(0), \tag{9}$$

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

$$s_i = \begin{bmatrix} -\omega_i \times p_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}, \tag{10}$$

where $\omega \in \mathbb{R}^{3\times 1}$ is a unit vector showing in the direction of the twist axis and $q_i \in \mathbb{R}^{3\times 1}$ is an arbitrary point on the axis. Revolute joints perform only pure rotations about an axis. Therefore the twist has the form:

$$s_i = \begin{bmatrix} 0 \\ \omega_i \end{bmatrix}. \tag{11}$$

Analogous, the pure translation is much simpler,

$$s_i = \begin{bmatrix} v_i \\ 0 \end{bmatrix}, \tag{12}$$

where $v_i \in \mathbb{R}^{3\times 1}$ is a unit vector facing in the direction of translation.

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 [8] is also shown that $Ad_X(y)$ admits the 6×6 matrix representation

$$Ad_X(y) = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{13}$$

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

$$ad_x(y) = [x, y] \tag{14}$$

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

$$ad_x(y) = \begin{bmatrix} \hat{\omega}_1 & \hat{v}_1 \\ 0_{3\times 3} & \hat{\omega}_1 \end{bmatrix} \begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix}. \tag{15}$$

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 (angular components) and moments (rotational component) acting on a rigid body.

$$F = \begin{pmatrix} f \\ \tau \end{pmatrix}, \tag{16}$$

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,

$$F_a = Ad^T_{g_{ba}} F_b, \tag{17}$$

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 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.

	Cubic Link Modules (Chen)	Backbone / Scout (Symbrion / Replicator)
Module types	homogeneous (large/small)	heterogeneous
Joint types	revolute, prismatic	revolute
# ports	6	4
DOFs	rot.: ±180°	Backbone: bend.:±90°; Scout: bend.:±90°, rot.: ±180°

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

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 [16], 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 $H_{i,j}(q) \in SE(3)$:

$$H_{i,j}(q_j) = H_{i,j}(0)e^{\hat{s}_j q_j}, \tag{18}$$

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.

 ISBN: 978-1-61208-219-6

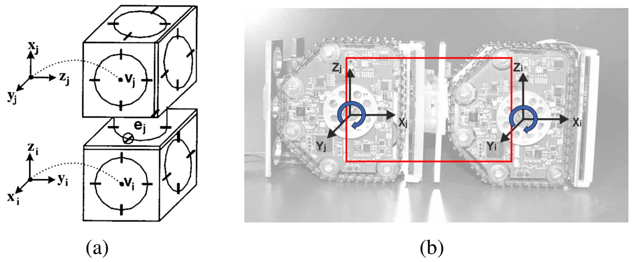


Fig. 2. (a) A dyad defined by Chen and Yang [16], (b) Dyad for two Scout robots.

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 Eq. 18.

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 [18] or in [8]. 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:

$$g_{st}(q) = e^{\hat{s}_1 q_1} e^{\hat{s}_2 q_2} e^{\hat{s}_3 q_3} \dots e^{\hat{s}_n q_n} g_{st}(0), \tag{19}$$

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

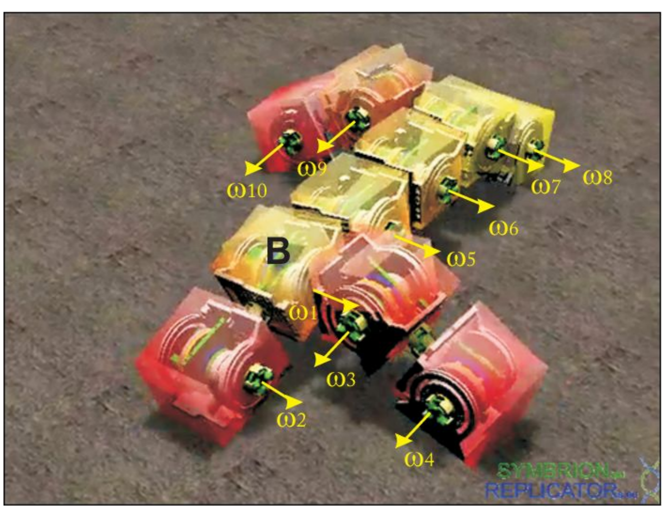


Fig. 3. Multi robot organism [19].

For a tree or branch structured robot configurations, the forward kinematics can be obtained in parallel way starting 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 [16]. 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:

$$H(q_1, q_2, \dots, q_n) = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_k \\ \vdots \\ H_m \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ \vdots \\ \prod_{i=1}^{n} (H_{a_{i-1}a_i}(0) e^{\hat{s}_{a_i} q_{a_i}}) \\ \vdots \\ \dots \end{bmatrix}, \tag{20}$$

where $H(q_1, q_2, \dots, q_n)$ 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 [16], 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 [16] cannot be directly used for this kind of modules and need to be adapted.

A. Adapted Assembly Incidence Matrix

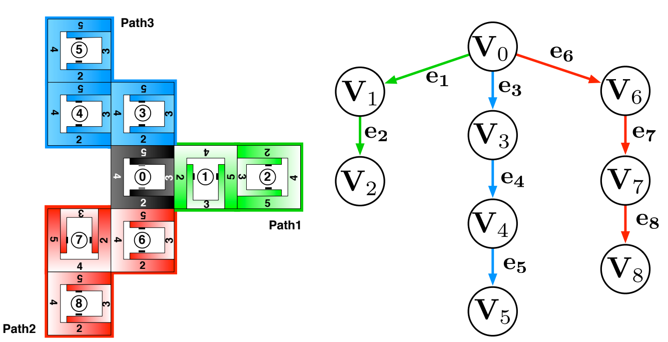
The Backbone and the Scout robots are both cubic shaped robots, however provide only four sides that are equipped with docking units. Therefore, using the notation of gaming dice only ports $2-5$ are able to set a connection. A difference between modular robots proposed in [16] and the Scout/Backbone 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.

Set *DS*: Dyad: Serial Axes		Set *DP*: Dyad: Parallel Axes		Set *DO*: Dyad: Orthogonal Axes	
1st Mod.	2nd Mod.	1st Mod.	2nd Mod.	1st Mod.	2nd 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

TABLE II
DYADS LOOK-UP TABLE FOR SCOUT AND BACKBONE.

The autonomous docking procedure is based either on IR sensor communication or also can be fulfilled by using vision system [20], [21]. 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.



(a) (b)

Fig. 4. (a) Multi robot organism example, (b) Directed graph representation.

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.

$$
AIM_{SB} = \begin{array}{c} \\ \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ \text{Joints} \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} \text{Path1} & & \text{Path2} & & \text{Path3} & & & & \\ e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & \text{Links} \end{array} \\
\left[\begin{array}{ccccccccc}
3 & 0 & 5 & 0 & 0 & 2 & 0 & 0 & B \\
4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & B \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & B \\
0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & B \\
0 & 0 & 0 & 3 & 5 & 0 & 0 & 0 & B \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & B \\
0 & 0 & 0 & 0 & 0 & 5 & 4 & 0 & B \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & B \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & B \\
DO & DO & DS & DP & DS & DS & DO & DO &
\end{array}\right]
\end{array}
$$

Fig. 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.

DTL			
T_{ij}		Sides	
i	j	from	to
0	1	3	2
1	2	5	3
0	3	5	2
3	4	4	3
4	5	5	2
0	6	2	5
6	7	4	2
7	8	4	5

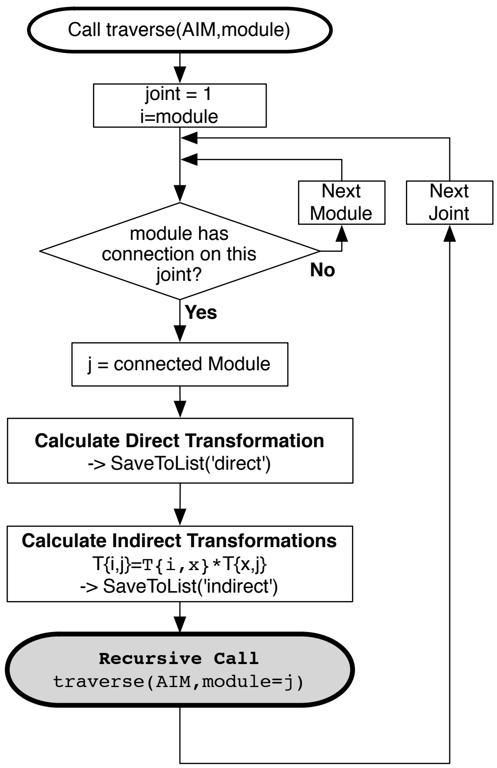
IDTL					
T_{ij}		= T_{ix}		· T_{xj}	
i	j	i	x	x	j
0	2	0	1	1	2
0	4	0	3	3	4
3	5	3	4	4	5
0	5	0	4	4	5
0	7	0	6	6	7
6	8	6	7	7	8
0	8	0	7	7	8

TABLE III
DIRECT AND INDIRECT TRANSFORMATION LISTS.

A short example demonstrates the first traversing calculations using both lists:

$$
\begin{array}{lll}
T_{01} & = T_{01}(0)e^{\hat{s}_1 q_1} & \text{direct} \\
T_{12} & = T_{12}(0)e^{\hat{s}_2 q_2} & \text{direct} \\
T_{02} & = T_{01} \cdot T_{12} & \text{indirect} \\
\vdots & \vdots &
\end{array}
\tag{21}
$$

This algorithm can be compared with DFS algorithm, providing a flexible way to calculate the order in which all possible transformations can be calculated during runtime.



(a)

Fig. 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 multibody 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 [6]. 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}

$$V_b = V_0 = (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)^T \tag{22}$$
$$\dot{V}_b = \dot{V}_0 = (0\ \ 0\ \ g\ \ 0\ \ 0\ \ 0)^T \tag{23}$$

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

$$H_{i-1,i} = H_i e^{\hat{s}_i q_i} \tag{24}$$
$$V_i = Ad_{H^{-1}_{i-1,i}}(V_{i-1}) + S_i \dot{q}_i \tag{25}$$
$$\dot{V}_i = Ad_{H^{-1}_{i-1,i}}(\dot{V}_{i-1}) - ad_{Ad_{H^{-1}_{i-1,i}}}(V_i) + S_i \ddot{q}_i \tag{26}$$

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. F_{n+1} is the force acting on the end-link of chained robots. This values can either be estimated or read from force sensors attached to the robots.

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

$$F_i = Ad^*_{H^{-1}_{i,i+1}}(F_{i+1}) - F^e_i + M_i \dot{V}_i - ad^*_{V_i}(M_i V_i) \tag{27}$$
$$\tau_i = s_i^T F_i \tag{28}$$

Here, M_i is the generalized mass matrix of the form

$$M_i = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & mI_3 \end{bmatrix}, \tag{29}$$

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_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 (25) to (28) in body coordinates, it can be shown that the equations for generalized velocities, generalized accelerations and forces can be obtained in matrix form:

$$V = TS\dot{q} \tag{30}$$
$$\dot{V} = T_{H_0}\dot{V}_0 + TS\ddot{q} + Tad_{S\dot{q}}V \tag{31}$$
$$F = T^T F^e + T^T M\dot{V} + T^T ad^*_V MV \tag{32}$$
$$\tau = S^T F \tag{33}$$

 ISBN: 978-1-61208-219-6

where

$$
\begin{aligned}
\dot{q} &= column[\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n] \in R^{n\times 1}\\
\ddot{q} &= column[\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_n] \in R^{n\times 1}\\
V &= column[V_1, V_2, \ldots, V_n] \in R^{6n\times 1}\\
\dot{V} &= column[\dot{V}_1, \dot{V}_2, \ldots, \dot{V}_n] \in R^{6n\times 1}\\
F &= column[F_1, F_2, \ldots, F_n] \in R^{6n\times 1}\\
F^e &= column[F_1^e, F_2^e, \ldots, F_n^e] \in R^{6n\times 1}\\
\tau &= column[\tau_1, \tau_2, \ldots, \tau_n] \in R^{n\times 1}\\
S &= diag[S_1, S_2, \ldots, S_n] \in R^{6n\times n}\\
M &= diag[M_1, M_2, \ldots, M_n] \in R^{6n\times 6n}\\
ad_{S\dot{q}} &= diag[-ad_{S_1\dot{q}_1}, -ad_{S_2\dot{q}_2}, \ldots, -ad_{S_n\dot{q}_n}] \in R^{6n\times 6n}\\
ad_V^* &= diag[-ad_{V_1}^*, -ad_{V_2}^*, \ldots, -ad_{V_n}^*] \in R^{6n\times 6n}
\end{aligned}
$$

The index n represents the number of elements containing also virtual joints that are required to move the robot in a space [22].

$$
T_{H_0} = \begin{bmatrix} Ad_{H_{0,1}^{-1}} \\ Ad_{H_{0,2}^{-1}} \\ \vdots \\ Ad_{H_{0,n}^{-1}} \end{bmatrix} \in R^{6n\times 6} \tag{34}
$$

$$
T = \begin{bmatrix} I_{6\times 6} & 0_{6\times 6} & 0_{6\times 6} & \cdots & 0_{6\times 6} \\ Ad_{H_{1,2}^{-1}} & I_{6\times 6} & 0_{6\times 6} & \cdots & 0_{6\times 6} \\ Ad_{H_{1,3}^{-1}} & Ad_{H_{2,3}^{-1}} & I_{6\times 6} & \cdots & 0_{6\times 6} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Ad_{H_{1,n}^{-1}} & Ad_{H_{2,n}^{-1}} & Ad_{H_{3,n}^{-1}} & \cdots & I_{6\times 6} \end{bmatrix} \in R^{6n\times 6n}, \tag{35}
$$

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

$$
M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau \tag{36}
$$

is obtained by substituting the equations 30 to 33, 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.

$$
M(q) = S^T T^T M T S \tag{37}
$$

$$
C(q,\dot{q}) = S^T T^T (M T ad_{S\dot{q}} + ad_V^*) T S \tag{38}
$$

$$
N(q) = S^T T^T M T_{H_0} \dot{V} + S^T T^T F^e \tag{39}
$$

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 8). 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 [22] or gait generators that use chaotic map. We use an approach proposed in, [23], that allows to generate periodic gaits that result from synchronization effects of coupled maps. Such approach can help to control complex multibody structures by mapping the active joints to an individual chaotic driver [24].

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 8(a)) for example derived in [25] and the second is an extended pendulum that is movable on a shaft like a crane. In the second example, we use a virtual joint that allows moving the crane along one axes (Figure 8(b)).

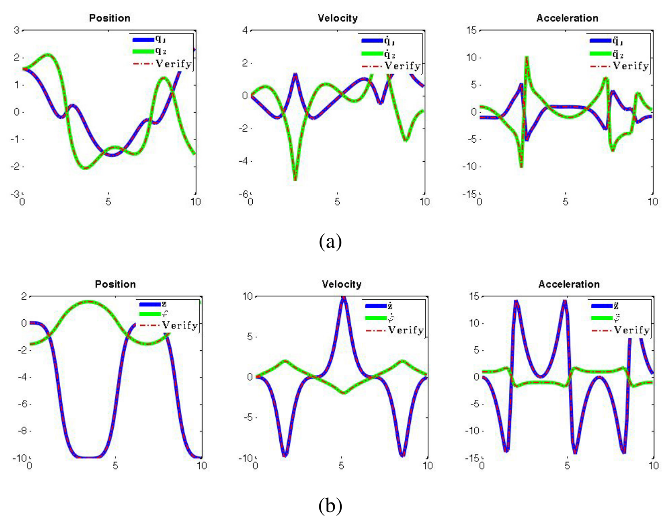


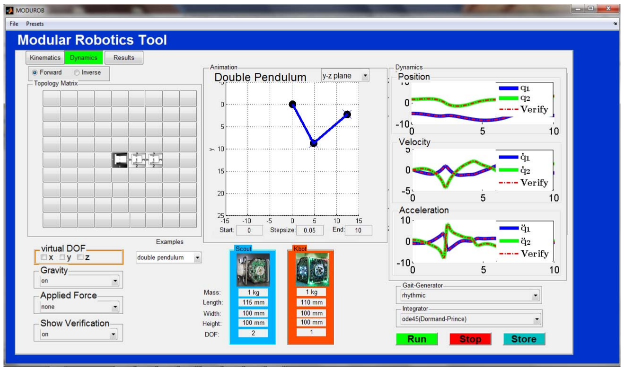
Fig. 7. Verification (dashed line) of geometrical POE approach with Lagrangian method using two examples: (a) Double pendulum model, (b) Crane model.

Both examples (Figure 7) show absolutely identical behaviour with examples implemented based on Lagrangian equations as well as with the geometrical approach based on twist and wrenches and therefore evaluates the approach.

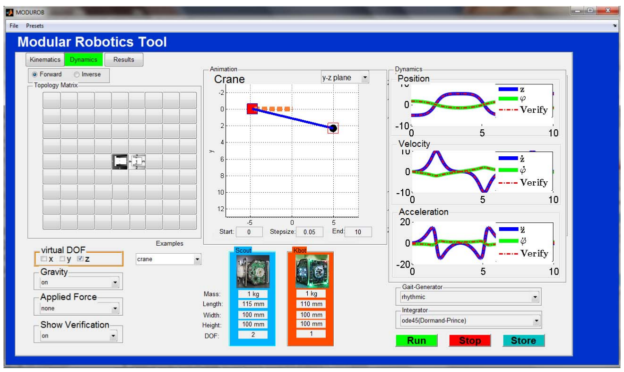
VII. Conclusion and Future Work

In this paper, we demonstrate a MATLAB framework that allows analysing the kinematics and dynamics of modular robots. The calculation of self-adaptive models is based on recursive geometrical approach built on Screw Theory [26]. The proposed algorithm is inspired by the work from Chen and Yang and has been modified and adapted to the needs of robot

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-219-6



(a)



(b)

Fig. 8. (a) Implemented benchmark example of a double pendulum [25], (b) Crane example.

modules developed in projects Symbrion and Replicator. Such tool can not only be used for studying of topology behaviours of modular robots but also open a easy way to understand the theory behind the geometrical recursive approach. After we are able to build the models for kinematics and dynamics autonomously the next step will be to investigate different control design strategies such as feedback linearisation, self-organized and learning control mechanisms.

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] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S. Chirikjian. Modular self-reconfigurable robot systems – challenges and opportunities for the future. *IEEE Robotics and Autonomation Magazine*, March:43–53, 2007.

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

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

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

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

[6] 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.

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

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

[9] Scott Robert Ploen. *Geometric Algorithms for the Dynamics and Control of Multibody Systems*. PhD thesis, 1997.

[10] 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.

[11] 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.

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

[13] 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.

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

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

[16] 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.

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

[18] 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.

[19] Lutz Winkler and Heinz 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.

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

[21] Reza Saatchi M. Shuja Ahmed and Fabio 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.

[22] E. Meister, S. Stepanenko, and S. Kernbach. Adaptive locomotion of multibody snake-like robot. In *Proc. of Multibody Dynamics 2011, ECCOMAS Thematic Conference*, 2011.

[23] 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.

[24] 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.

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

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