Abstract—This paper introduces a highly flexible and adjustable force controller for modular wheeled robots. Given a proper configuration, the controller can work with wildly different robot configurations and on varying levels of control distribution. The functionality of the controller was tested with simulations and practical tests with real robots are soon to follow.

I. INTRODUCTION

A mobile robot needs a specialized structural configuration to achieve optimal performance with a given environment and task. These specialized configurations can vary wildly and each different configuration has traditionally demanded a specialized controller whose design and implementation takes time and effort. The proposed force controller is highly adjustable and expandable with the minimum of effort. The goal is also to make the structure of the controller as modular as possible to help with the physical distribution of the force controller from a single relatively powerful node to multiple nodes across the robot. The modularity also makes it easier to experiment and integrate different methods of solving specific sub-problems.

Spatial vector algebra [1] simplifies the design of force control algorithms that are used to control potentially highly complex robot configurations and Section 2 provides a quick introduction to the parts of spatial vector algebra used in this paper. Section 3 introduces the force controller whose basic functionality was tested in the simulations described in Section 4. The paper is concluded with a brief description of future work in Section 5.

II. SPATIAL VECTOR ALGEBRA

Like screws and wrenches in screw theory [3], spatial motion- and force vectors combine the linear and angular aspects of a rigid-body motion or force into a single 6D vector. Spatial vectors do not use reference points either, all the necessary information about the locations of velocities, forces and masses resides inside the vectors and inertia matrices themselves [1]. These characteristics make equations written with 6D spatial vectors simpler and more concise than the ones written with the vastly more common 3D vectors [1][2]. This makes the design and implementation of robot controllers easier once the designer has understood the basic concepts behind spatial vectors.

The following is a short introduction to the parts of spatial vector algebra used in this paper, for more extensive and mathematically rigorous material please refer to [1], [4] and [5]. Vectors with hats (e.g. $\hat{v}, \hat{f}$) are 6D spatial vectors. The information on this Section is from [1] unless otherwise noted. The examples are kept planar for the purpose of making the principles of spatial vectors as clear as possible.

A. Spatial Motion

The rigid body in Fig. 1 rotates with a known angular velocity of $\omega = [0, 0, \omega_3]^T$ about point C which is simultaneously translating at unknown velocity $v_C$. At the same time, point A has a known instantaneous velocity of $v_A = [v_{Ax}, v_{Ay}, 0]^T$. The 6D spatial vector combining the rotational and translational velocity from the two known 3D velocity vectors is $v_o = [\omega, v_o]^T$, where

$$v_o = v_A + \vec{OA} \times \omega.$$  

Fig. 1. Rigid body in motion.

The rigid body in Fig. 1 rotates with a known angular velocity of $\omega = [0, 0, \omega_3]^T$ about point C which is simultaneously translating at unknown velocity $v_C$. At the same time, point A has a known instantaneous velocity of $v_A = [v_{Ax}, v_{Ay}, 0]^T$. The 6D spatial vector combining the rotational and translational velocity from the two known 3D velocity vectors is $v_o = [\omega, v_o]^T$, where $v_o = v_A + \vec{OA} \times \omega$. The 6D spatial vector $v_o$ describes the complete motion of the rigid body. The object can now be viewed as rotating around point A with angular velocity $\omega$ while in addition simultaneously translating with velocity $v_o$. The object can also be viewed as having screwing motion along vector $\omega$ that passes through point K that uniquely fulfils,

$$\hat{v} = \begin{bmatrix} \omega \\ v_o \end{bmatrix} = \begin{bmatrix} \omega \\ k \times \omega + p\omega \end{bmatrix},$$  

when $k = \vec{OA}$ and $p = \frac{\omega \cdot v_o}{\omega \cdot \omega}$. Both views are equivalently correct. For the purposes of this paper the former is preferred while the latter is used in screw theory [3].
The velocity of point P located anywhere in the rigid body is,

\[ V(P) = v_0 + \omega \times \overrightarrow{OP}, \]

and is the same regardless of where O is located. With different positioning and orientation of O, the value of \( \omega \times \overrightarrow{OP} \) changes, but so does \( v_0 \). All spatial vectors share this invariance for frame selection and it is a key element of spatial vector algebra. Like conventional 3D velocities, spatial velocities are additive,

\[ \ddot{v}_{total} = \ddot{v}_1 + \ddot{v}_2. \]

Spatial acceleration vector \( \ddot{a}_o = [\omega, \ddot{v}_o]^T = [a, \dot{a}]^T \) is the rate of change of a stream of body-fixed points going through O and not the change of velocity of any single point. Unlike the conventional 3D acceleration vector, spatial accelerations vectors behave like true vectors [5], and have the same combination,

\[ \ddot{a}_{total} = \ddot{a}_1 + \ddot{a}_2, \]

and coordinate transformation rules as spatial velocity vectors.

**B. Spatial Force**

Spatial force vectors are formed similarly to \( \ddot{v}_o \). \( \ddot{f}_o = [n_0, \ f]^T \) where,

\[ n_0 = n_1 + \overrightarrow{OB} \times f, \]

and \( n_1 \) is a force couple affecting the object. In the example of Fig. 1 force \( \ddot{f}_o \) can be viewed as the combination of linear force \( f \) going through O (instead of original point B) and a moment \( n_0 \) at O. The moment in any point P can be calculated by,

\[ N(P) = n_0 + f \times \overrightarrow{OP}. \]

Like spatial motions, spatial forces are also additive,

\[ \ddot{f}_{total} = \ddot{f}_1 + \ddot{f}_2. \]

**C. Spatial Inertia**

The spatial inertia is a 6x6 matrix,

\[ I = \begin{bmatrix} I_c + mc \times c \times c & mc \times c \\ mc \times c & m \mathbf{1} \end{bmatrix}, \]

where \( I_c \) the object’s 3x3 inertia tensor, \( c \) is the vector from O to point C located at the center of mass (COM) and \( m \) is the object’s mass. Spatial inertia is also additive,

\[ I_{total} = I_1 + I_2. \]

**D. Basic Equations**

There are two spatial cross product operators for spatial vectors. \( \times^* \) is a cross product operator between a spatial motion (velocity or acceleration) and a force vector,

\[ \ddot{v} \times^* \ddot{f} = \begin{bmatrix} \omega \\ \ddot{v}_o \end{bmatrix} \times^* \begin{bmatrix} n_o \\ f \end{bmatrix} = \begin{bmatrix} \omega \times n_o + v_o \times f \\ \omega \times f \end{bmatrix}. \]

Spatial cross product operator \( \times \) between two spatial motion vectors is similarly,

\[ \ddot{v} \times \ddot{m} = \begin{bmatrix} \omega \\ v_o \end{bmatrix} \times \begin{bmatrix} m_o \\ m \end{bmatrix} = \begin{bmatrix} \omega \times m_o + v_o \times m \\ \omega \times m \end{bmatrix}. \]

The equation of motion expressed with spatial vectors is

\[ \ddot{f} = \frac{d}{dt}(\ddot{v}) = \dddot{v} + \dot{\omega} \times \dddot{v}. \]

In spatial vector algebra there are three separate frame transformations for motion, force and inertia. Frame transformation operation from frame A to frame B for motion vectors is

\[ \ddot{v}_B = B X_A \ddot{v}_A \]

where \( E \) is a 3x3 rotation matrix , \( r = \overrightarrow{AB} \) in frame A coordinates and \( \times \) is a cross product operator,

\[ \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \]

\( B X_A \) performs the inverse transform of (15),

\[ A X_B = \begin{bmatrix} 1 & 0 & 0 \\ r & 1 & 0 \end{bmatrix} \begin{bmatrix} E^T & 0 \\ 0 & E \end{bmatrix} = \begin{bmatrix} E^T & 0 \\ r \times E^T & E \end{bmatrix}. \]

Similarly for spatial force vectors,

\[ \ddot{f}_B = B X_A \ddot{f}_A \]

where \( \ddot{f}_A \) is a force in frame A coordinates.

\[ B X_A \ddot{f}_A = \begin{bmatrix} E & 0 \\ 0 & E \end{bmatrix} \begin{bmatrix} 1 & -r \times & -r \times & -r \times \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} E^T & 0 \\ 0 & E \end{bmatrix} \begin{bmatrix} r \times E^T & E \end{bmatrix} \]

\[ A X_B \ddot{f}_B = \begin{bmatrix} 1 & r \times & r \times & r \times \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} E^T & 0 \\ 0 & E \end{bmatrix} \begin{bmatrix} r \times E^T & E \end{bmatrix}. \]

For spatial inertia transformations,

\[ B I = B X_A A I A X_B \]

**III. FORCE CONTROLLER**

The intended platform for the controller is a robot which is in essence a co-operating sensor-actuator network of heterogeneous nodes (wheel modules, force controller, sensors, etc.) where the division of tasks between nodes can vary according to the robot’s configuration, complexity and intended mission. The nodes are connected to a shared communication network to facilitate the task distribution. The force controller itself can be physically distributed; for simple robots a single processing unit inside a single node could directly compute the required torques for all the wheels; for more complex robots, the computational burden can be distributed to multiple processing units inside multiple nodes.

Fig.2 displays the force controller architecture and Table 1 the relevant parameters. The environmental force estimation and adaptive fine-tuning are subject to future work and as such they are intentionally left vague.
**TABLE I. PARAMETERS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Origin of auxiliary frame A (Fig. 3)</td>
</tr>
<tr>
<td>C</td>
<td>Origin of frame B centred at the robot’s COM (Fig. 3)</td>
</tr>
<tr>
<td>M</td>
<td>Location of sensors (Fig. 3)</td>
</tr>
<tr>
<td>S</td>
<td>Location of steering point (Fig. 3)</td>
</tr>
<tr>
<td>SC</td>
<td>Vector from point S to C</td>
</tr>
<tr>
<td>S ̈a</td>
<td>Desired spatial acceleration expressed at point S</td>
</tr>
<tr>
<td>CA</td>
<td>Desired spatial acceleration expressed at point C by using (14) with S ̈a and SC</td>
</tr>
<tr>
<td>MC</td>
<td>Vector from point M to C</td>
</tr>
<tr>
<td>MP</td>
<td>Measured velocity at point M</td>
</tr>
<tr>
<td>CP</td>
<td>Measured velocity at point C, (14) used with MP and MC</td>
</tr>
<tr>
<td>CF</td>
<td>The robot’s total inertia expressed at point C</td>
</tr>
<tr>
<td>FC</td>
<td>The force needed to accelerate S ̈a to ̇a</td>
</tr>
<tr>
<td>FE</td>
<td>Estimation of internal forces resisting the motion and external forces affecting the robot</td>
</tr>
<tr>
<td>FA</td>
<td>Adaptive force to compensate for unmodeled forces</td>
</tr>
<tr>
<td>f i</td>
<td>(n x, n y, n z, f x, f y, f z) T , total force demanded from wheel modules</td>
</tr>
<tr>
<td>Wi</td>
<td>Wheel node i</td>
</tr>
<tr>
<td>xi</td>
<td>x coordinate of wheel i in frame A coordinates</td>
</tr>
<tr>
<td>yi</td>
<td>y coordinate of wheel i in frame A coordinates</td>
</tr>
<tr>
<td>di</td>
<td>Moment distance of wheel i</td>
</tr>
<tr>
<td>I wi</td>
<td>Rolling inertia of wheel i</td>
</tr>
<tr>
<td>r wi</td>
<td>Radius of wheel i</td>
</tr>
<tr>
<td>k wi</td>
<td>Force distribution coefficient of wheel i</td>
</tr>
<tr>
<td>K F,K B,K L,K R</td>
<td>≡K = Σk wi, Sum of weight coefficients of wheel groups F (front), B (back), R (right) and L (left)</td>
</tr>
<tr>
<td>LF,L B,L L,L R</td>
<td>L = Σd wi,</td>
</tr>
<tr>
<td>f wi</td>
<td>Force demanded from wheel i, either k wi/f f, k wi/f b, k wi/f l, or k wi/f d depending on which group the wheel belongs to</td>
</tr>
<tr>
<td>o wi</td>
<td>Orientation of wheel i. Value 1 or -1 depending on which way the wheel was installed.</td>
</tr>
<tr>
<td>τ dem wi</td>
<td>Torque due to force demanded wheel wheel i</td>
</tr>
<tr>
<td>τ rot wi</td>
<td>Additional torque needed to accelerate or decelerate wheel wheel i</td>
</tr>
<tr>
<td>τ wi</td>
<td>Final torque command to wheel i, τ i = τ w i + τ dem wi + τ rot wi</td>
</tr>
<tr>
<td>SW i</td>
<td>Vector from point S to the location of wheel i</td>
</tr>
<tr>
<td>W i ̈a</td>
<td>Desired spatial acceleration expressed at wheel i location by using (14) with ̇a and SW i</td>
</tr>
</tbody>
</table>

Fig. 2. Force controller.

Fig. 3 illustrates one very asymmetrical omni-wheel robot configuration that can be controlled with the proposed force controller. Point M is the location of velocity measurement sensors. Point S is the steering point about which the desired spatial acceleration S ̈a is expressed. S ̈a and the location of S provide flexibility on how to express the desired acceleration. For example, when S ̈a = [0, 0, 0] T , point S can be viewed as the instantaneous center of rotation (ICR) about which the robot is ordered to rotate around with the angular acceleration of S ̇ω 3 rad/s.

Basically, the force controller operates by first expressing the robot’s inertia and velocity at point C in spatial vectors and from them calculating the necessary force needed to produce the desired acceleration. The estimated environmental forces and the output of some adaptive sub-controller are then added to this single spatial force vector. The resulting total force is demanded collectively from all the wheels and each individual wheel contributes to the fulfillment of this demand according to its position and capability.

The operation of the force controller is divided between initialization that is done only once and run-time.

**A. Initialization**

The goal is to use the same controller with various robot configurations with as little effort as possible, so the configuration of the controller should be as automated as possible. Communication is done via a shared channel, thus, all nodes have access to all the messages sent during initialization (and runtime). The initialization of the force controller can be divided into eight steps. Note that only steps 1, 2 and 3 need any input from the designer.

**Step 1.** It is assumed that for each node with a significant mass, there is a pre-calculated spatial inertia matrix expressed about the point of the node where it will be attached to the robot’s rigid chassis. The chassis inertia is expressed about its COM. The nodes send the controller their pre-calculated inertia matrices. The designer sends the inertias for nodes without communication capabilities such as the chassis and any fixed payloads.

**Step 2.** The designer selects the origin O and orientation of an auxiliary frame A about which the mass distribution and
nodes locations and orientation will be initially expressed. The exact position of O does not matter but should be in a central position to minimize rounding errors in subsequent initialization steps. The orientation of A’s z-axis is upwards and x-axis points to the main motion direction of the robot.

**Step 3.** The designer sends in frame A coordinates the locations of where the nodes are attached to the chassis and orientations of all the nodes. For this paper, only the locations of powered wheels, motion sensors (point M in Fig. 3), payloads and COM of the chassis are relevant.

**Step 4.** Using the inertias from step 1 and locations and orientations from step 3 the controller calculates the robot’s total inertia $I_T$ in frame A with (21) and (10). In the example of Fig. 3,

$$0I = 0I_{chassis} + 0I_{payload} \sum_{i=1}^{2} 0I_{wheeli}.$$  

**Step 5.** The controller extracts the calculated location of COM from $I_T$ with (9) and then with (21) forms $I_T$ which is the whole robot’s inertia expressed in frame B. Frame B has the same orientation as the auxiliary frame A and its origin point C coincides with the robot’s calculated COM.

**Step 6.** The controller calculates the locations of nodes, in frame B coordinates, then stores and broadcasts them to the other nodes.

**Step 7.** Wheels are divided into four groups depending on their location and orientation in regards to frame B. Currently, wheels can only have a fixed ±x or ±y axis orientation and the sign is stored in $\omega_{wi}$. The division into wheel groups is illustrated in Fig. 4 with C being the robot’s COM, as in Fig 3. For example, wheels in the negative y half-plane that have orientation parallel with x-axis, belong to R (right) group and wheels in positive x plane with orientation parallel with y-axis belong to F (front) group.

![Division into wheel groups.](image)

The wheel groups in Fig. 3 are $F = \{\emptyset\}, B = \{W_3, W_6\}, L = \{W_1, W_4, W_7\}, R = \{W_2, W_5\}$. The moment distance $d_i$ of a wheel is determined according to Table 2 by the group the wheel belongs to and its location coordinates.

<table>
<thead>
<tr>
<th>Wheel group</th>
<th>Wheel i moment distance $d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F (front)</td>
<td>$X_{wi}$</td>
</tr>
<tr>
<td>B (back)</td>
<td>$X_{wi}$</td>
</tr>
<tr>
<td>L (left)</td>
<td>$Y_{wi}$</td>
</tr>
<tr>
<td>R (right)</td>
<td>$Y_{wi}$</td>
</tr>
</tbody>
</table>

**Step 8.** Wheel nodes send the controller the rolling inertias and diameters of the wheels.

**B. Runtime**

As shown in Fig. 2, the path planner (or some other higher level control unit) gives the location of steering point S and the desired acceleration $\ddot{q}$ expressed about that point. The location of the steering point S can be changed during runtime. The location of measurement point M and measurements $\dot{\theta}$ can be either actual physical values or processed combinations based on the locations and measurement data of multiple sensors.

$^c f_i$ is calculated by (13) with $^c a$, $^c \dot{\theta}$ and $^c T$. Force $^c f_i$ is estimated using any applicable method with the addition of expressing the forces as a single spatial vector at point C. The fine-tuning force $^c f_A$ is achieved by some adaptive method such as a PID or neural network. Because $^c f_i$ has only up to three relevant parameters ($f_x$, $f_y$, and $n_z$), the adaptive system behind $^c f_A$ should remain relatively simple. The details on how $^c f_E$ and $^c f_A$ are formed are out of scope of this paper.

Force distribution coefficient $k_{wi}$ can be used to adjust the relative amount of force demanded from the wheel based on criteria such as wheel-ground contact quality. Force distribution coefficient can be also used as a malfunction recovery mechanism as dropping $k_{wi}$ to zero informs the controller to not demand any force from wheel $i$.

The demanded force $^c \bar{f}$ is distributed to the wheels by first finding the minimum unit of force $f_F, f_B, f_L, f_R$ for each wheel group. The demanded forces in the Y-direction are balanced so that they produce zero moment at C,

$$\begin{align*}
K_F f_F + K_B f_B &= f_y \\
L_F f_F - L_B f_B &= 0,
\end{align*}$$  

from which we get,

$$\begin{align*}
f_F &= f_y/(K_F + K_B) \\
f_B &= f_y/(K_B - K_F) \\
K_F &= \frac{L_B f_F}{L_B f_F - L_F f_B}.
\end{align*}$$  

When balanced distribution is not possible because either $K_F = 0$ or $K_B = 0$ the force is distributed by using just distribution coefficient $k_i$,

$$\begin{align*}
f_F &= f_x/K_F \\
n_u &= -L_F f_F
\end{align*}$$  

or

$$\begin{align*}
f_B &= f_x/K_B \\
n_u &= L_B f_B
\end{align*}$$

where $n_u$ is the moment caused by the unbalanced force $f_F$. Forces in X-direction produce the moment $n_x$ and compensate for $n_u$ if it is non-zero,

$$\begin{align*}
K_R f_R + K_L f_L &= f_x \\
L_R f_R - L_L f_L &= n_x + n_u,
\end{align*}$$

from which we get,
would be enough to fulfill these two conditions. In the orientation (For example, in Fig. 3 just wheels 1, 3 and 5 with parallel orientation and a third one with orthogonal conditions: COM must be inside the support area for mediated instead of used (i.e. not used) (i.e. arbitrary distribution of wheels and masses with two wheels)).

A. Holonomic Robot With Omni-Wheels

The additional torque \( \tau_{demwi} = \omega_{wi} f_{wi} \) needed to accelerate (or decelerate) the rotation of wheel \( i \), can easily be calculated with \( \tau_{wi} \), \( \omega_{wi} \) and \( \frac{df}{dt} \). The final torque command to each wheel \( i \) is the sum of \( \tau_{demwi} \) and \( \tau_{rotwi} \).

The transformation of \( f_p, f_M, f_L \) or \( f_R \) into \( \tau_{wi} \) could be delegated to each individual wheel module \( i \). Delegating would distribute a significant amount of computation to the wheel modules themselves and the processing unit responsible for force distribution (Fig. 2) could remain oblivious to each wheel’s radius and rotational inertia.

IV. EXPERIMENTS

The experiment were done using V-REP robot simulator [6]. The tests are highly idealized, but nevertheless prove the main concepts behind the controller. The simulated robots are basically sliding on locked frictionless wheels and the pure forces are added at the wheel-ground contact points. These simulations demonstrate that the force controller works with different robot configurations as long as the environmental forces can be predicted. Adaptive fine-tuning (\( \frac{c f}{a} \)) was not used in any of the experiments.

A. Holonomic Robot With Omni-Wheels

With ideal omni-wheels, the force-controller works with arbitrary distribution of wheels and masses with two conditions: COM must be inside the support area formed by the wheels, and at least three wheels must provide traction, two with parallel orientation and a third one with orthogonal orientation (For example, in Fig. 3 just wheels 1, 3 and 5 would be enough to fulfill these two conditions). In the simulations displayed in Fig. 5 and Fig. 6, the omni-wheels are ideal, i.e. perfect traction in forward motion and zero friction in lateral motion, so environmental force estimation is not used (i.e. \( \frac{c f}{a} = (0, 0)^T \)).

The simulated robot’s configuration is the same as in Fig. 3; darker rectangles are wheels, the lighter diagonal rectangle is the payload, the circle next to it is the COM, the circle at the center is the origin of auxiliary frame A and the third circle is the location of the sensors. The line below the robot shows the path of the selected steering point \( S \) has traveled (the whole path is not shown in Fig. 5). In this simulation, \( k_i \) is simply the internal battery charge of wheel \( i \) and it decreases in direct relation to the amount of force demanded and produced by the wheel. As can be seen in Fig. 5, the controller adapted to the failure of wheels 3, 4 and 7 and maintained constant acceleration. The failure was simulated by setting \( k_{w3} = k_{w4} = k_{w7} = 0 \) at time \( t=1 \).

\[
\begin{align*}
 f_L &= (f_x - \frac{k_p(n_x + n_u)}{L}) \left( K_L + \frac{k_p l_x}{L} \right), \\
 f_R &= \frac{k_p}{K_R} f_L \\
 f_p, f_M, f_L, f_R &\text{ are transformed into the actual demanded force } f_{wi} \text{ for each wheel. Each } f_{wi} \text{ is converted to torque,} \\
 \tau_{demwi} &= \omega_{wi} f_{wi} \tau_{wi}.
\end{align*}
\]

Fig. 6 represents a different configuration of omni-wheels and shows more clearly the distribution of demanded force amongst wheel nodes with different \( k_{wi} \) values. The wheel number increases from top to bottom, and each successive wheel has initially less battery charge than its predecessor. The amount of force demanded from each wheel is directly related to the amount of battery charge it has left, so the wheels in one group (only two groups, L and R in Fig. 6 have members) run out of charge at the same time. This way, the number of wheels capable of producing force is kept as high as possible as long as possible.

B. 3-Wheeled Robot

The controller in the simulated robot of Fig. 7 estimates (unrealistically) perfectly the combined ground-wheel force \( \frac{c f}{E} \) that provides the needed lateral force to keep the robot on a circular path. Because all the wheels are “in front” of the robot’s COM, \( \frac{c f}{E} \) has also moment component at C which the wheels must compensate according to equations (27) and (28). As the robot’s speed increases, so does \( n_u \) the wheels must produce. Apart from keeping the robot upright, the effect of caster wheel is ignored in this simulation by substituting it with a frictionless sphere.
V. CONCLUSION AND FUTURE WORK

This paper has introduced a flexible and extendable force controller that is designed to be suitable for modular robots. Many parts of the controller can be separated to self-contained sub-units that can be solved with a variety of applicable methods. For example, the force distribution part (Fig. 2) does not need to care how $^c f_G$ and $^c f_A$ are formed and the method how they are obtained can be integrated into the controller in almost plug-and-play fashion. The force controller can also be physically distributed across the robot. For example, the formation $k_{wl}$ and some of the calculations needed for $\tau_{wl}$ could be delegated to the wheel modules themselves. This kind of task distribution would automatically scale up the controller’s HW when more wheels are introduced to the robot’s configuration.

The next step in further development of the force controller is to use some pre-existing method to obtain $^c f_G$ and $^c f_A$. After these additions have been tested in simulation the improved force controller will be used on a real robot of Fig. 8 using the MobilityModule wheel modules from Probot Ltd [7] and the communication is done with their RobotCAN [8] protocol over controller area network (CAN).

VI. ACKNOWLEDGEMENT

The authors would like to thank the University of Oulu Graduate School (UniOGs) and Infotech Oulu for making this research possible. We would also like to thank M.Sc Marko Kauppinen for the critique and improvement suggestions received while writing this paper.

REFERENCES

[6] V-REP