Efficient Event-Driven Reactive Control for Large Scale Robot Skin

Florian Bergner, Emmanuel Dean-Leon and Gordon Cheng

***Abstract*— In this work we present a novel efficient event-driven reactive skin controller for large scale robot skin. The novel event-driven controller derives from a standard Jacobian torque controller and fully takes advantage of our multi-modal event-driven robot skin. Event-driven systems only sample, transmit and process information when the novelty of the information is guaranteed. This increases their efficiency in comparison to synchronous systems. We also use the new event-driven controller formulation to design a new synchronous reactive skin controller. We compare both controllers in a comprehensive performance evaluation with our robot TOMM. TOMM has two UR5 robot arms, each covered with 253 multi-modal skin cells. Each skin cell samples 4 different modalities and supports data mode and event mode. The results show that the event-driven reactive skin controller always outperforms the synchronous reference controller while both controllers show exactly the same response. When the robot is not moving then the event-driven controller reduces the CPU usage by 78% in comparison to the synchronous reference controller. When the robot is responding to contacts then the CPU usage reduces by 66%.**

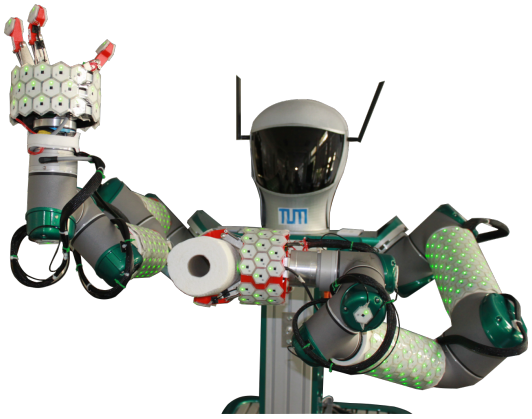

Fig. 1. The robot TOMM [23] with two arms and two grippers covered with skin; the robot holds a paper towel in its left gripper which it uses in our experiment to push the right arm; the right arm is controlled by our novel event-driven tactile reaction controller; it tries to avoid contacts and moves to the right.

I. INTRODUCTION

A. Motivation

Tactile human-robot interaction is essential for collaborative robots in industrial, health-care and household application scenarios and increases the robot's intuitiveness, flexibility and safety. One key element of enabling intuitive interactions is the ability to manually guide the robot by touching it for e.g. in teach-in scenarios [22]. The standard approach for realizing such manual guidance is using force/torque sensors in each robot joint and realizing an interactive controller which applies a dynamic robot model to differ between external and internal torques [15], [16]. However joint/torque sensors are expensive and cannot be easily attached to robots in case of upgrades. Furthermore force/torque sensors can only sense external force/torque stimuli in a cumulative way and can thus not distinguish between multiple contact points per joint. For example antagonistic forces on a joint cancel themselves. Overcoming this problem and enabling multi-contact interactive controllers have contributed to the motivations for developing robot skin [1]–[3], [18]–[20]. However, to fully take advantage of skin in tactile interaction control, the skin has to cover the robot completely. This induces new challenges caused by huge numbers of skin cells in large scale robot skin. The challenges are mainly: 1) solving issues in transmitting huge amounts of tactile information with low-latency and 2) processing huge amounts of tactile information in real-time. Neuromorphic event-driven sensing of tactile information is proposed in the works of [4]–[8] and seems to be a viable solution for reducing redundant information at sensor level. In event-driven systems only novel information has to be transmitted which greatly reduces required transmission bandwidths. This work focuses on exploiting the efficiency of event-driven information to design a novel event-driven reactive skin controller for robots with large scale robot skin.

B. Related Work

The proposed event-driven reactive skin controller is related to standard Jacobian reactive control at torque level and processes event-driven tactile information with low computational costs. To our best knowledge such a tactile control system has not been fully addressed yet. However, tactile robot interaction systems which use force/torque sensors [15], [16] can be considered as state-of-the-art and systems using robot skin have been proposed in the works of [17]–[22]. Nonetheless, all these tactile interaction systems are synchronous systems and process tactile information synchronously. Event-driven sensing and processing is addressed in the works of [7]–[14]. These works demonstrate that event-driven systems show superior efficiency in sampling, transmitting and processing information.

C. Contributions

We propose a novel event-driven reactive skin controller for large scale robot skin. Our control system fully exploits the advantages of event-driven information. The controller

*This work has received funding from the European Communitys Seventh Framework Program (FP7/2007-2013) under grant agreement no. 609206.

All authors are with the Institute for Cognitive Systems, Technische Universität München, Munich, Germany, see http://www.ics.ei.tum.de

only processes tactile information when the novelty of tactile information is guaranteed. Tactile activity correlates directly to the occurrence of events and searches through the complete set of tactile data are no longer necessary. This increases the computational efficiency of the controller and tactile information can be processed inside the control loop at full speed. For the development of the event-driven tactile controller, we separate the Jacobian torque calculation into three parts. One part only depends on static transformations, one only on joint coordinate frames of the robot kinematics and one depends on both. We use these separated formulations also to design a more efficient synchronous tactile interaction controller. We finally compare the performance of both controllers on a real robot through comprehensive experiments and demonstrate the superior performance of our novel event-driven reactive skin controller.

II. SYSTEM DESCRIPTION

A. Robot skin

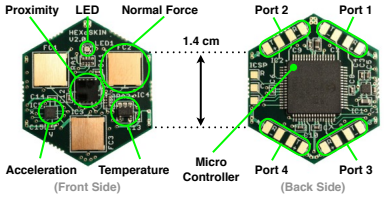

Fig. 2. Robot skin.

Our robot skin [18] is a multi-modal, self-organizing and self-calibrating robot skin. Its hexagonally shaped skin cells are organized in patches (see Fig. 3) and build up a self-organized redundant communication network. Bidirectional communication trees establish connections from each skin cell to the PC and vice versa. The skin cells sense multi-modal tactile stimuli like vibration (3D acceleration sensor), pressure (3 capacitive force sensors), pre-touch (proximity sensor) and temperature (2 temperature sensors). Recently we updated the firmware code of the skin cells which now supports two modes: the data mode and the event mode [7], [8]. In the data mode, skin cells transmit information synchronously with a constant sample rate (e.g. 250 Hz). In the event mode, skin cells transmit information asynchronously in events. The option to switch between modes enables us to easily compare event-driven algorithms with their synchronous state-of-the-art counterparts.

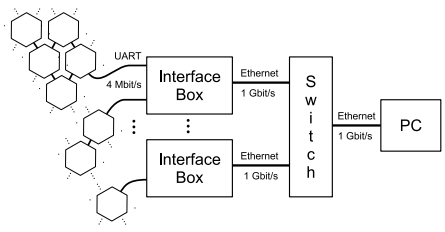

Fig. 3. The skin cell network architecture and interface to the PC.

The skin system is able to self-calibrate [24], [25] and automatically determines static transformations ${}^{i}\mathbf{T}_{k}$ of a skin cell i with respect to its robot joint k (see also Fig. 5). This allows us to define the kinematic chains of every skin cell to a common reference frame. The tactile reactive controller needs these kinematic chains and manual calibration for 253 skin cells would be infeasible.

B. The Tactile Omnidirectional Mobile Manipulator TOMM

We perform our experiments on our robot TOMM [23] see Fig. 1 and 4. TOMM has a mobile omnidirectional base, two UR5 robot arms and two Lacquey grippers. Each arm is covered with 253 skin cells and each gripper with 57 skin cells. TOMM is fully integrated in ROS (Robot OS). Fig. 4 visualizes the control loop of one arm. The green box addresses the reactive skin torque controller of this paper.

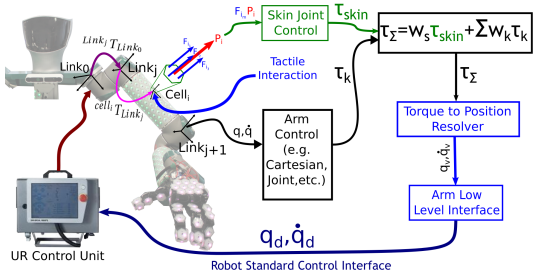

Fig. 4. The control framework on the robot TOMM.

C. Fusing force and proximity values to external force

The skin cells i can only detect external normal forces along their own $\mathbf{z}$-axes. We calculate this external normal force $F_{\mathrm{z},i}$ by fusing the sum of the 3 normalized capacitive forces F_{c} with the normalized proximity value F_{p} (see Algo. 1). α_{c} and α_{p} denote positive gains and $F_{\mathrm{c,th}}$, $F_{\mathrm{p,th}}$ and $F_{\mathrm{z,th}}$ denote noise canceling thresholds. A skin cell i is considered as active as soon as $F_{\mathrm{z}} > 0$.

Algorithm 1 Calculate F_{z} for given F_{c} and F_{p}

1: **if** $F_{\mathrm{c}} < F_{\mathrm{c,th}}$ **then**
2: $\quad F_{\mathrm{c}} := 0$
3: **end if**
4: **if** $F_{\mathrm{p}} < F_{\mathrm{p,th}}$ **then**
5: $\quad F_{\mathrm{p}} := 0$
6: **end if**
7: $F_{\mathrm{z}} := \alpha_{\mathrm{c}}\, F_{\mathrm{c}} + \alpha_{\mathrm{p}}\, F_{\mathrm{p}}$
8: **if** $F_{\mathrm{z}} < F_{\mathrm{z,th}}$ **then**
9: $\quad F_{\mathrm{z}} := 0$
10: **end if**

D. Reactive skin torque controller

We developed a torque based reactive skin controller which avoids contacts. This controller has to map the wrenches of skin cells i to corresponding torques $\boldsymbol{\tau}$ [22]. Fig. 5 depicts that an external wrench $\mathbf{w}_i \in \mathbb{R}^6$ exerted on a skin cell i results in torques τ_l in joints $1, \ldots, k$. These joints we name as active joints. The index k denotes the last active joint. Skin cells i only detect normal external forces $F_{\mathrm{z},i}$ such that the wrench $\bar{\mathbf{w}}_i \in \mathbb{R}^6$ of a skin cell simplifies to

$$\bar{\mathbf{w}}_i^T = \begin{bmatrix} 0 & 0 & F_{\mathrm{z},i} & 0 & 0 & 0 \end{bmatrix}^T \tag{1}$$

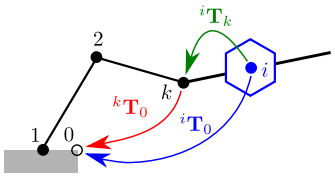

Fig. 5. Kinematic chain from skin cell i on active joint k to the world reference frame 0.

The reactive skin controller maps the wrench $\bar{\mathbf{w}}_i$ of skin cell i to torques τ_l of joints l. Therefore it uses the transposed Jacobian $\mathbf{J}_i^T(\mathbf{q})$ of a skin cell i and the principle of virtual work:

$$\boldsymbol{\tau}_i = \mathbf{J}_i^T(\mathbf{q})\ {}^{i}\bar{\mathbf{w}}_0 \tag{2}$$

${}^{i}\bar{\mathbf{w}}_0$ is the wrench measured by skin cell i with respect to the base coordinate frame 0 and $\boldsymbol{\tau}_i \in \mathbb{R}^{\mathrm{DOFs}}$ is the torque generated by skin cell i. The Jacobian can be easily determined geometrically by applying the laws of physics for translating and mapping linear and angular velocities to joint velocities. For a skin cell i on an active joint k the rows $\mathbf{j}_l^T(\mathbf{q})$, $l \in 1, \ldots, k$ of $\mathbf{J}_i^T(\mathbf{q})$ have to be updated while the rows with index $l > k$ remain zero. Forces detected by a skin cell i do not effect joints after joint k:

$$\mathbf{j}_{l,i}^T(\mathbf{q}) = \left[\left({}^{l-1}\mathbf{z}_0(\mathbf{q}) \times \left[{}^{i}\mathbf{t}_0 - {}^{l-1}\mathbf{t}_0(\mathbf{q})\right]\right)^T \quad {}^{l-1}\mathbf{z}_0^T(\mathbf{q})\right]$$

$$\mathbf{j}_l^T(\mathbf{q}) = \mathbf{0}^T \in \mathbb{R}^6 \qquad \textbf{if} \quad l > k \tag{3}$$

$$\mathbf{J}_i^T(\mathbf{q}) = \left[\mathbf{j}_{1,i}(\mathbf{q}) \quad \ldots \quad \mathbf{j}_{k,i}(\mathbf{q}) \quad \mathbf{0} \quad \ldots \quad \mathbf{0}\right]^T \tag{4}$$

Note that ${}^{l-1}\mathbf{z}_0 \in \mathbb{R}^3$ is the $\mathbf{z}$-axis of coordinate frame $l-1$ with respect to the base coordinate frame 0 and ${}^{i}\mathbf{t}_0 \in \mathbb{R}^3$ is the origin of the coordinate frame of skin cell i with respect to the base coordinate frame 0. The transformation from coordinate frame $l-1$ to 0 is defined by the homogeneous transformation matrix ${}^{l-1}\mathbf{T}_0 \in \mathbb{R}^{4\times4}$ (see Fig. 5). The axes $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ and the translation of the origin $\mathbf{t}$ can be easily extracted from the corresponding transformation matrices $\mathbf{T}$:

$${}^{k}\mathbf{T}_0 = \begin{bmatrix} {}^{k}\mathbf{x}_0 & {}^{k}\mathbf{y}_0 & {}^{k}\mathbf{z}_0 & {}^{k}\mathbf{t}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

E. Cell-wise reactive skin torque control

The easiest way to calculate the total skin torque of all skin cells is to first calculate the torque $\boldsymbol{\tau}_i$ of each skin cell i and then to sum them up. The number of calculations can be reduced by thresholding the external forces $F_{\mathrm{z},i}$. The torque $\boldsymbol{\tau}_i$ is then only calculated whenever the external force is above noise level. See Algorithm 2. However, this algorithm becomes inefficient for large amount of active skin cells. Especially lines 9, 15 and 16 are computationally expensive and contain many redundant calculations. Furthermore, the algorithm has to loop over all skin cells, has to calculate the external force and check if it crossed the threshold. For these reasons we propose a new algorithm which calculates only what is necessary. The new algorithm separates calculations into calculations which have to be processed only for force changes and respectively into calculations which have to be processed only for joint position changes.

Algorithm 2 Cell-wise skin torque calculation

1: **update** ${}^{l}\mathbf{T}_0(\mathbf{q}) \quad \forall l \in 1, \ldots, \mathrm{DOF}$
2: $\boldsymbol{\tau} := \mathbf{0}$
3: **for all** skin cells i **do**
4: **calculate** $F_{\mathrm{z},i}$ using Algo. 1
5: **if not** $F_{\mathrm{z},i} > 0$ **then**
6: **continue**
7: **end if**
8: **map** i **to** k
9: ${}^{i}\mathbf{T}_0(\mathbf{q}) := {}^{k}\mathbf{T}_0(\mathbf{q})\ {}^{i}\mathbf{T}_k$
10: $\mathbf{f}_i := \begin{pmatrix} 0 & 0 & F_{\mathrm{z},i} \end{pmatrix}^T$
11: $\bar{\mathbf{f}}_i := \begin{pmatrix} \mathbf{f}_i^T & 0 \end{pmatrix}^T$
12: ${}^{i}\bar{\mathbf{f}}_0 := {}^{i}\mathbf{T}_0(\mathbf{q})\ \bar{\mathbf{f}}_i$
13: ${}^{i}\bar{\mathbf{w}}_0 := \begin{pmatrix} {}^{i}\bar{\mathbf{f}}_0^T & 0 & 0 \end{pmatrix}^T$
14: **extract** ${}^{i}\mathbf{t}_0$ **from** ${}^{i}\mathbf{T}_0(\mathbf{q})$
15: **calculate** $\mathbf{J}_i^T(\mathbf{q})$
16: $\boldsymbol{\tau}_i := \mathbf{J}_i^T(\mathbf{q})\ {}^{i}\bar{\mathbf{w}}_0$
17: $\boldsymbol{\tau} := \boldsymbol{\tau} + \boldsymbol{\tau}_i$
18: **end for**

F. Component-wise reactive skin torque control

Looking at the l-th component of $\boldsymbol{\tau}$ for a skin cell i on joint k we get:

$$\begin{aligned} \tau_{l,i}(\mathbf{q}) &= \mathbf{j}_{l,i}^T(\mathbf{q})\ {}^{i}\bar{\mathbf{w}}_0 \\ &= \left({}^{l-1}\mathbf{z}_0(\mathbf{q}) \times \left[{}^{i}\mathbf{t}_0 - {}^{l-1}\mathbf{t}_0(\mathbf{q})\right]\right)^T\ {}^{i}\mathbf{f}_0 \\ &= \left(\left[{}^{i}\mathbf{t}_0 - {}^{l-1}\mathbf{t}_0(\mathbf{q})\right]^T \left[{}^{l-1}\mathbf{z}_0(\mathbf{q})\right]_\times^T\right)^T\ {}^{i}\mathbf{z}_0\ F_{\mathrm{z},i} \\ &= \left[A_{l,i}(\mathbf{q}) - B_{l,i}(\mathbf{q})\right]\ F_{\mathrm{z},i} \end{aligned} \tag{6}$$

with

$$\begin{aligned} A_{l,i}(\mathbf{q}) &= {}^{i}\mathbf{t}_0^T \left[{}^{l-1}\mathbf{z}_0(\mathbf{q})\right]_\times^T\ {}^{i}\mathbf{z}_0 \\ &= {}^{i}\hat{\mathbf{t}}_k^T\ {}^{k}\mathbf{T}_0^T(\mathbf{q}) \left[{}^{l-1}\hat{\mathbf{z}}_0(\mathbf{q})\right]_\times^T\ {}^{k}\mathbf{T}_0(\mathbf{q})\ {}^{i}\bar{\mathbf{z}}_k \\ &= {}^{i}\hat{\mathbf{t}}_k^T\ \mathbf{S}_{l,k}(\mathbf{q})\ {}^{i}\bar{\mathbf{z}}_k \end{aligned} \tag{7}$$

$$\begin{aligned} B_{l,i}(\mathbf{q}) &= {}^{l-1}\mathbf{t}_0^T(\mathbf{q}) \left[{}^{l-1}\mathbf{z}_0(\mathbf{q})\right]_\times^T\ {}^{i}\mathbf{z}_0 \\ &= {}^{l-1}\hat{\mathbf{t}}_k^T(\mathbf{q}) \left[{}^{l-1}\hat{\mathbf{z}}_0(\mathbf{q})\right]_\times^T\ {}^{k}\mathbf{T}_0(\mathbf{q})\ {}^{i}\bar{\mathbf{z}}_k \\ &= \mathbf{t}_{l,k}^T(\mathbf{q})\ {}^{i}\bar{\mathbf{z}}_k \end{aligned} \tag{8}$$

Note that this formula already takes the zero torque entries of the wrench ${}^{i}\bar{\mathbf{w}}_0$ in to account. Furthermore, the formulas $\mathbf{S}_{l,k}(\mathbf{q})$ (Eq. 9) and $\mathbf{t}_{l,k}(\mathbf{q})$ (Eq. 10) depend now only on the active joint and on the joint positions $\mathbf{q}$. Thus they only need to be updated on joint changes and, as we will see, on changes of $k_{\max}$. ${}^{i}\bar{\mathbf{z}}_k$ and ${}^{i}\hat{\mathbf{t}}_k$ depend only on a static transformations resulting from the skin calibration. These transformations can be calculated once in advance. The bar above vectors denote 3D vectors which have been extended to homogeneous 4D vectors by appending 0. The hat denotes homogeneous vectors with the standard extension. The cross-product matrix operator $[\cdot]_\times$ produces a skew symmetric matrix and thus the calculation of $\mathbf{S}_{l,k}(\mathbf{q})$ and $\mathbf{t}_{l,k}(\mathbf{q})$ can

be simplified further:

$$\mathbf{S}_{l,k}(\mathbf{q}) = {}^{k}\mathbf{T}_0^T(\mathbf{q}) \left[{}^{l-1}\hat{\mathbf{z}}_0(\mathbf{q})\right]_\times^T {}^{k}\mathbf{T}_0(\mathbf{q}) \tag{9}$$

$$= \begin{bmatrix} 0 & a & b & c \\ -a & 0 & d & e \\ -b & -d & 0 & f \\ -c & -e & -f & 1 \end{bmatrix}$$

$$a = \left({}^{k}\mathbf{x}_0 \times {}^{k}\mathbf{y}_0\right)^T {}^{l-1}\mathbf{z}_0 \qquad b = \left({}^{k}\mathbf{x}_0 \times {}^{k}\mathbf{z}_0\right)^T {}^{l-1}\mathbf{z}_0$$
$$c = \left({}^{k}\mathbf{x}_0 \times {}^{k}\mathbf{t}_0\right)^T {}^{l-1}\mathbf{z}_0 \qquad d = \left({}^{k}\mathbf{y}_0 \times {}^{k}\mathbf{z}_0\right)^T {}^{l-1}\mathbf{z}_0$$
$$e = \left({}^{k}\mathbf{y}_0 \times {}^{k}\mathbf{t}_0\right)^T {}^{l-1}\mathbf{z}_0 \qquad f = \left({}^{k}\mathbf{z}_0 \times {}^{k}\mathbf{t}_0\right)^T {}^{l-1}\mathbf{z}_0$$

$$\mathbf{t}_{l,k}^T(\mathbf{q}) = {}^{l-1}\hat{\mathbf{t}}_k^T(\mathbf{q}) \left[{}^{l-1}\hat{\mathbf{z}}_0(\mathbf{q})\right]_\times^T {}^{k}\mathbf{T}_0(\mathbf{q}) \tag{10}$$

$$= \begin{pmatrix} \bar{a} & \bar{b} & \bar{c} & \bar{d} \end{pmatrix}$$

$$\bar{a} = {}^{k}\mathbf{x}_0^T \left({}^{l-1}\mathbf{z}_0 \times {}^{l-1}\mathbf{t}_0\right) \qquad \bar{b} = {}^{k}\mathbf{y}_0^T \left({}^{l-1}\mathbf{z}_0 \times {}^{l-1}\mathbf{t}_0\right)$$
$$\bar{c} = {}^{k}\mathbf{z}_0^T \left({}^{l-1}\mathbf{z}_0 \times {}^{l-1}\mathbf{t}_0\right)$$
$$\bar{d} = {}^{k}\mathbf{t}_0^T \left({}^{l-1}\mathbf{z}_0 \times {}^{l-1}\mathbf{t}_0\right) + 1$$

Using the results for $\mathbf{S}_{l,k}(\mathbf{q})$ and $\mathbf{t}_{l,k}(\mathbf{q})$ of Eq. 9 and 10 we can symbolically expand $A_{l,i}(\mathbf{q})$ and $B_{l,i}(\mathbf{q})$ of Eq. 7 and 8 and order and re-factorize everything considering the permutation rules of the triple product expansion of the cross-product. We have such a triple product in all the entries of $\mathbf{S}_{l,k}(\mathbf{q})$ and $\mathbf{t}_{l,k}(\mathbf{q})$. This reduces redundant calculations caused by the extended skew symmetric matrix $\mathbf{S}_{l,k}(\mathbf{q})$:

$$A_{l,i}(\mathbf{q}) = \left({}^{i}\mathbf{z}_k \times {}^{i}\mathbf{t}_k\right)^T \begin{bmatrix} d \\ -b \\ a \end{bmatrix} - {}^{i}\mathbf{z}_k^T \begin{bmatrix} c \\ e \\ f \end{bmatrix} \tag{11}$$

$$= {}^{l-1}\mathbf{z}_0^T(\mathbf{q}) \left[{}^{k}\mathbf{A}_0(\mathbf{q}) \left({}^{i}\mathbf{z}_k \times {}^{i}\mathbf{t}_k\right) - {}^{k}\mathbf{B}_0(\mathbf{q}) {}^{i}\mathbf{z}_k\right]$$
$$= {}^{l-1}\mathbf{z}_0^T(\mathbf{q}) \left[{}^{k}\mathbf{A}_0(\mathbf{q}) \, {}^{i}\mathbf{c}_k - {}^{k}\mathbf{B}_0(\mathbf{q}) \, {}^{i}\mathbf{z}_k\right]$$
$$= {}^{l-1}\mathbf{z}_0^T(\mathbf{q}) \, {}^{i}\mathbf{c}_0(\mathbf{q})$$

$$B_{l,i}(\mathbf{q}) = {}^{i}\mathbf{z}_k^T \, {}^{k}\mathbf{R}_0^T(\mathbf{q}) \left({}^{l-1}\mathbf{z}_k \times {}^{l-1}\mathbf{t}_k\right) \tag{12}$$
$$= {}^{i}\mathbf{z}_{\mathrm{r},0}^T(\mathbf{q}) \, {}^{l-1}\mathbf{c}_0(\mathbf{q})$$

with

$${}^{i}\mathbf{c}_0(\mathbf{q}) = {}^{k}\mathbf{A}_0 \, {}^{i}\mathbf{c}_k - {}^{k}\mathbf{B}_0 \, {}^{i}\mathbf{z}_k \tag{13}$$
$${}^{k}\mathbf{A}_0(\mathbf{q}) = \begin{pmatrix} {}^{k}\mathbf{z}_0 \times {}^{k}\mathbf{y}_0 & {}^{k}\mathbf{x}_0 \times {}^{k}\mathbf{z}_0 & {}^{k}\mathbf{y}_0 \times {}^{k}\mathbf{x}_0 \end{pmatrix} \tag{14}$$
$${}^{k}\mathbf{B}_0(\mathbf{q}) = \begin{pmatrix} {}^{k}\mathbf{x}_0 \times {}^{k}\mathbf{t}_0 & {}^{k}\mathbf{y}_0 \times {}^{k}\mathbf{t}_0 & {}^{k}\mathbf{z}_0 \times {}^{k}\mathbf{t}_0 \end{pmatrix} \tag{15}$$
$${}^{i}\mathbf{c}_k = \left({}^{i}\mathbf{z}_k \times {}^{i}\mathbf{t}_k\right) \tag{16}$$
$${}^{i}\mathbf{z}_{\mathrm{r},0}(\mathbf{q}) = {}^{k}\mathbf{R}_0 \, {}^{i}\mathbf{z}_k \tag{17}$$
$${}^{k}\mathbf{R}_0(\mathbf{q}) = \begin{pmatrix} {}^{k}\mathbf{x}_0 & {}^{k}\mathbf{y}_0 & {}^{k}\mathbf{z}_0 \end{pmatrix} \tag{18}$$
$${}^{l-1}\mathbf{c}_0(\mathbf{q}) = \left({}^{l-1}\mathbf{z}_0 \times {}^{l-1}\mathbf{t}_0\right) \tag{19}$$

and

$$\tau_{l,i}(\mathbf{q}) = J_{l,i}(\mathbf{q}) \, F_{\mathrm{z},i} \tag{20}$$
$$= \left[A_{l,i}(\mathbf{q}) - B_{l,i}(\mathbf{q})\right] \, F_{\mathrm{z},i}$$
$$= \left[{}^{l-1}\mathbf{z}_0^T(\mathbf{q}) \, {}^{i}\mathbf{c}_0(\mathbf{q}) - {}^{i}\mathbf{z}_{\mathrm{r},0}^T(\mathbf{q}) \, {}^{l-1}\mathbf{c}_0(\mathbf{q})\right] \, F_{\mathrm{z},i}$$

Algorithm 3 Update $F_{\mathrm{z},i}$ for a given skin cell data packet

1: **get** skin cell id, F_c and F_p **from** data packet
2: **map** id **to** i
3: **calculate** F_z using Algo. 1
4: $F_{\mathrm{z},i} := F_\mathrm{z}$

Algorithm 4 Update $\boldsymbol{\tau}$ for a given joint position $\mathbf{q}$

1: $\boldsymbol{\tau} := 0$
2: **update** ${}^{l}\mathbf{T}_0(\mathbf{q}) \quad \forall l \in 1, \ldots, \mathrm{DOF}$
3: **calculate** ${}^{l-1}\mathbf{c}_0(\mathbf{q}) \quad \forall l \in 1, \ldots, \mathrm{DOF}$
4: **calculate** ${}^{k}\mathbf{A}_0(\mathbf{q}) \quad \forall k \in 1, \ldots, \mathrm{DOF}$
5: **calculate** ${}^{k}\mathbf{B}_0(\mathbf{q}) \quad \forall k \in 1, \ldots, \mathrm{DOF}$
6: **for all** active skin cells i **do**
7: **calculate** $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q}) \, F_{\mathrm{z},i} \quad \forall l \leq k$
8: **compose** $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
9: $\boldsymbol{\tau} := \boldsymbol{\tau} + \boldsymbol{\tau}_i$
10: **end for**

We observe that Eq. 13 - Eq. 19 separate Eq. 20 in to parts which have to be calculated only once per active joint (Eq. 14, 15, 18 and 19, colored in red, containing index k or l), into parts which are only dependent on the static skin calibration (Eq. 16, colored in green, containing only index i) and into parts which are a combination of the former two (Eq. 13, 17 and Eq. 20). These equations which only depend on active joints only depend on terms of the robot kinematics; the axes and translations of the active joint coordinate frames. These equations only need to be updated when the robot state changes, notably when the joint positions $\mathbf{q}$ change. Eq. 16 only depends on static skin information and can be calculated in advance. The remaining equations Eq. 13, 17 and Eq. 20 combine joint and skin transformations and only need to be updated when skin cell i is active or when the joint positions $\mathbf{q}$ change.

$$\tau_{l,i}(\mathbf{q}) = \begin{cases} J_{l,i}(\mathbf{q}) \, F_{\mathrm{z},i}, & l \leq k \ \wedge \ \text{cell } i \text{ is active} \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

Overall the newly proposed component-wise skin torque calculation is computationally less expensive than the cell-wise skin torque calculation because it uses less complex mathematical operations and its separability allows for reducing the redundancy of calculations. This is advantageous for both controller modes, the data mode and the event mode.

G. The skin joint controller in data mode

The skin joint controller in data mode consists of two functions which we present in Algo. 3 and 4. One is a callback function (see Algo. 3) which is called whenever the controller receives a new skin cell data packet containing the normalized sensor values and the skin cell id. This callback function first maps the skin cell id to the corresponding index i and then updates the external force $F_{\mathrm{z},i}$ in the controller memory. The other function (Algo. 4) calculates the skin torques and is called in the update loop of the skin joint controller. This function first updates the kinematic chain such that all transformations ${}^{l}\mathbf{T}_0(\mathbf{q})$ are updated according to the joint position $\mathbf{q}$ of the robot. Then it calculates the torque $\boldsymbol{\tau}_i$ for each active skin cell.

Algorithm 5 Update $F_{\mathrm{z},i}$, and $\boldsymbol{\tau}$ for a given skin cell event

if not force event **and not** proximity event **then**
  return
end if
get skin cell id **from** event
map id **to** i
if force event **then**
  $F_{\mathrm{c},i} :=$ value of event
end if
if proximity event **then**
  $F_{\mathrm{p},i} :=$ value of event
end if
calculate F_{z} using Algo. 1
if $|F_{\mathrm{z},i} - F_{\mathrm{z}}| < F_{\mathrm{z,e,th}}$ **then**
  return
end if
$F_{\mathrm{z},i} := F_{\mathrm{z}}$
if joint k becomes active **then**
  add k **to** active joint list K
  $k_{\mathrm{max},p} := k_{\mathrm{max}}$
  update k_{max}
  for $l \in k_{\mathrm{max},p}, \ldots, k_{\mathrm{max}}$ **do**
    update ${}^{l-1}\mathbf{c}_0(\mathbf{q})$
  end for
  update ${}^{k}\mathbf{A}_0(\mathbf{q})$, ${}^{k}\mathbf{B}_0(\mathbf{q})$
end if
if joint k becomes inactive **then**
  update k_{max}
  remove k **from** active joint list K
end if
$\boldsymbol{\tau} := \boldsymbol{\tau} - \boldsymbol{\tau}_i$
$\boldsymbol{\tau}_i := 0$
if $F_{\mathrm{z},i} > 0$ **then**
  calculate $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q})\ F_{\mathrm{z},i} \quad \forall l \leq k$
  compose $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
end if
$\boldsymbol{\tau} := \boldsymbol{\tau} + \boldsymbol{\tau}_i$

Algorithm 6 Update $\boldsymbol{\tau}$ for a given joint position $\mathbf{q}$

flag := **false**
for $l \in 1, \ldots,$ DOF **do**
  if $|q_{\mathrm{p},l} - q_l| > q_{\mathrm{e,th}}$ **then**
    flag := **true**
    break
  end if
end for
if not flag **then**
  return $\boldsymbol{\tau}$
end if
update ${}^{l}\mathbf{T}_0(\mathbf{q}) \quad \forall l \in 1, \ldots,$ DOF
calculate ${}^{l-1}\mathbf{c}_0(\mathbf{q}) \quad \forall l \in 1, \ldots, k_{\mathrm{max}}$
calculate ${}^{k}\mathbf{A}_0(\mathbf{q}) \quad \forall k \in K$
calculate ${}^{k}\mathbf{B}_0(\mathbf{q}) \quad \forall k \in K$
for all active skin cells i **do**
  $\boldsymbol{\tau} := \boldsymbol{\tau} - \boldsymbol{\tau}_i$
  calculate $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q})\ F_{\mathrm{z},i} \quad \forall l \leq k$
  compose $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
  $\boldsymbol{\tau} := \boldsymbol{\tau} + \boldsymbol{\tau}_i$
end for

H. The skin joint controller in event mode

The skin joint controller in event mode also uses two functions (see Algo. 5 and 6). The callback function (Algo. 5) is executed whenever a skin cell reports capacitive force events or proximity events. The Algo. 5 fuses these events to the corresponding external force $F_{\mathrm{z},i}$. Furthermore, if the change of the external force $F_{\mathrm{z},i}$ exceeds the event threshold $F_{\mathrm{z,e,th}}$ then Algo. 5 stores the external force $F_{\mathrm{z},i}$ in its local controller memory, updates the affected joints and finally calculates the skin torque. The update of the joint related properties only happens whenever a joint becomes active or inactive. To sum up, Algo. 5 triggers all the necessary calculations to map an external force event to the corresponding torque response. Algo. 6 is executed in the update loop of the skin joint controller. However, this algorithm simulates joint events and only triggers further calculations when the change of joint positions $\mathbf{q}$ is large enough (above the joint event threshold $q_{\mathrm{e,th}}$). Otherwise it immediately returns the current skin torque $\boldsymbol{\tau}$. The Algo. 6 uses external forces $F_{\mathrm{z},i}$ and joint related properties which are updated by Algo. 5.

III. EXPERIMENTS

The following demonstration scenarios are part of the European project Factory-in-a-Day.

A. Experimental setup

For a comprehensive performance evaluation of the introduced skin joint controllers (in data/event mode) it is essential to ensure the exact repeatability of the experiment. This includes that both controllers are excited in exactly the same way through all experiments; otherwise the controller response and its load on CPU and network would not be comparable between experiments and modes.

In order to excite the robot repeatedly in exactly the same way we use TOMM's left arm and excite with it its right arm. On the right arm we run our skin joint controllers (see Fig. 1) and on the left arm we execute a Cartesian position controller. The left arm moves along a predefined trajectory and touches the lower part of the right arm with a paper towel held in its left gripper. We provide a video to further illustrate the experiment.

B. Performance evaluation

1) Comparability of experiments and controllers: To prove the comparability of the experiments we have to prove that a repeated execution of the experiment produces the same excitation on the skin and that both controllers respond in the same way. In Fig. 7 e) we observe that the external forces $F_{\mathrm{ext,e}}$ (event mode) and $F_{\mathrm{ext,d}}$ (data mode) are identical for both experiments. The comparability of the controllers in data/event mode is shown in Fig. 7 f). The response in angular joint velocities $\dot{q}$ is identical for both controllers.

2) Controller performance (Idle robot): Fig. 6 depicts the network and CPU usage of the skin driver and the reactive skin controller for both modes. The robot is idle (not moving and not in contact) in phase I. In this phase we observe that the network usage in the skin driver is with 2.5 KB/s for the event mode much lower than in the data mode where it is 1 MB/s. The network usage in data mode is constant in all phases and there is no difference whether the robot is being touched or not. The network usage in the controller is

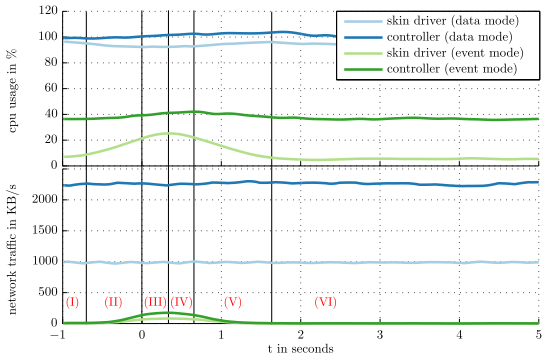

Fig. 6. Comparison of CPU usage and network traffic in data/event mode for controlling one robot arm covered with 253 skin cells.

about 2.25 times higher than in the skin driver. This is caused by the overhead in the communication between ROS nodes. For the idle robot the CPU usage in the skin driver is with 5.5% much lower than in data mode where it is constantly at 95%. The CPU usage in the controllers is not constant for both modes (see Fig. 6 and 7). For the idle robot the CPU usage of the controller in event mode is 36.5% and in data mode 99%. The reduction of the combined CPU usages of skin driver and controller in event mode compared to data mode is from 194% to 42%. The total relative reduction from event mode to data mode is thus 78% for an idle robot. The results clearly show that network and CPU usage are lower for the event-driven system (event mode) in comparison to the synchronous system (data mode) when the robot is idle. The results also show that network and CPU usage in the event-driven system depend on tactile excitation while in the synchronous system only the CPU usage of the controller is not constant. This behavior proves that the event-driven system only transmits and processes novel information and is thus more efficient. The CPU usage of the controller in data mode is not constant because its computational complexity depends on the number of active skin cells.

3) Controller performance (Robot reacting to contact): In this section we analyze the performance of both controllers in different experiment phases (see Fig. 7 and Tab. I):

(I) The controller is idle; the robot is not moving and not excited on its skin

(II) The controller is processing only skin information but has no active skin cell; the robot is not moving

(III) The controller has active skin cells and is processing skin and joint information; the robot starts to respond to the contact

(IV) The controller has less active skin cells but is still processing skin and joint information; the robot continues to respond to the contact

(V) The controller has no longer active skin cells and is processing only skin information; the robot is still moving but is no longer in contact and the joint velocities decrease

(VI) The controller is no longer processing skin information; the robot movement slowly ends

Tab. II shows the CPU usages of the reactive skin controller within the different phases for both modes. As soon as the

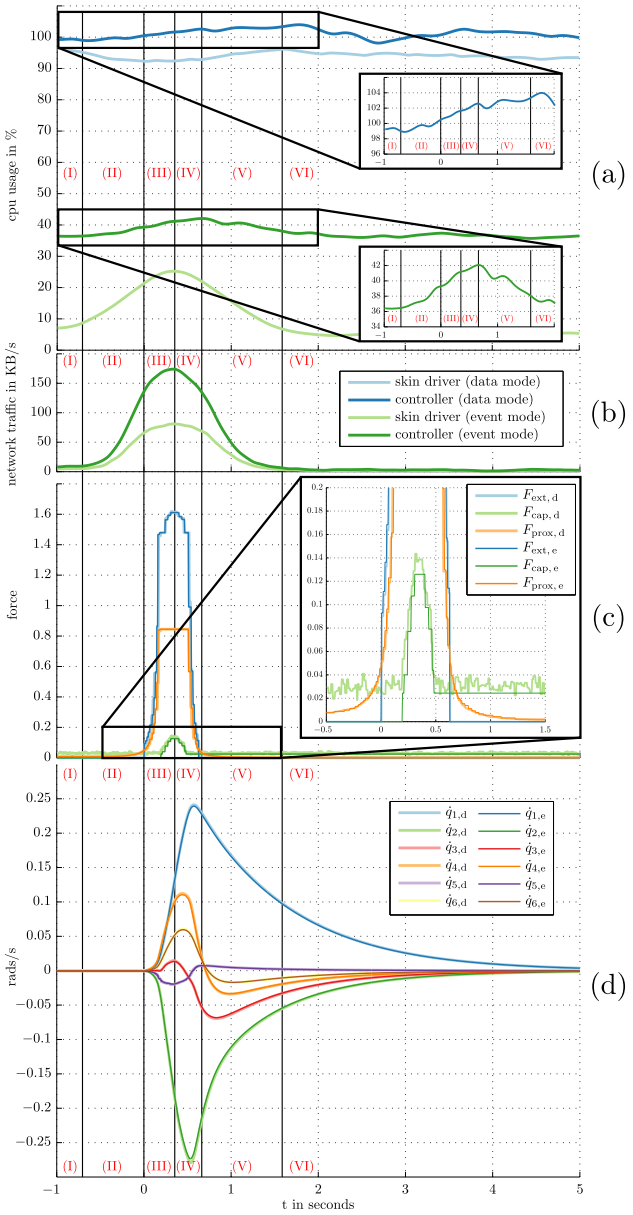

Fig. 7. Comprehensive comparison of controller in data/event mode; all subplots a) - d) are synchronized at $t = 0$ where the controller first detects more than one active skin cell; subplot c) depicts the normalized proximity forces F_{prox}, the normalized force of the capacitive force sensors F_{cap} and the fused external force F_{ext} for one skin cell; with index d for data mode and index e for event mode.

robot is in contact and processes skin information (phase II to phase V) the relative CPU usage of the controller in event mode is increasing faster than in data mode with respect to the CPU usage in the idle phase. This faster relative increase is caused by the additional network usage of the force and proximity events which is close to zero in the idle phase. The network usage in data mode is constant in all phases and thus its contribution to the relative CPU usage cancels out. We compensate the contribution of the network usage to the relative CPU usage in event mode and get changes in relative CPU usage as depicted in Tab. III. The change of relative CPU usage correlates to the computational cost which the reactive skin controller causes when processing skin information of active skin cells. The controller in event

mode outperforms the controller in data mode by a huge margin in all experimental phases. In the worst case (phase III) the controller in event mode requires 60.52% less CPU usage than in data mode. This means that the event-driven controller reduces the CPU usage by at least 59%. The skin driver in event mode causes in worst case a CPU usage of 25%. The CPU usage in data mode is constantly 95%. The combined worst case CPU usage of skin driver and controller is 66.5% in event mode and 199% in data mode. Thus the event-driven system reduces the total system CPU usage by at least 66.6%.

Phase	Mode	network traffic	active skin cells	joints active
I	data	yes	no	no
II	data	yes	no	yes
III	data	yes	yes	yes
IV	data	yes	yes	yes
V	data	yes	no	yes
VI	data	yes	no	yes
I	event	no	no	no
II	event	yes	no	no
III	event	yes	yes	yes
IV	event	yes	yes	yes
V	event	yes	no	yes
VI	event	no	no	yes

TABLE I

Mode	I	II	III	IV	V	VI
data	99%	99.75%	101.05%	102.1%	102.9%	101%
event	36.5%	37.9%	40.2%	41.58%	40.1%	37%

TABLE II

Mode	II	III	IV	V	VI
data	$\sim$ 0%	2.1%	3.1%	3.9%	2%
event	$\sim$ 0%	$\sim$ 0.3%	$\sim$ 2.4%	$\sim$ 2.5%	0.5%

TABLE III

IV. CONCLUSIONS

We discussed in detail how we developed an efficient event-driven reactive skin controller which based on Jacobian torque control. The novel controller efficiently processes events of our event-driven robot skin. We evaluated the reactive skin controller in event mode by comparing it to a reference controller in data mode. We validated the controllers on our robot TOMM and designed an experimental setup which ensures the comparability of repeated experiments. We also proved the comparability of the event-driven controller with its synchronous counterpart. Besides executing different algorithms, both controllers responded to identical tactile stimuli in exactly same way. When the robot is not moving and idle then the event-driven reactive skin controller outperforms the controller in data mode by 78% and when robot responds to contacts then by at least 66%. The event-driven reactive skin controller shows a substantial reduction in CPU usage and paves the way for full body reactive skin controllers for large scale robot skin.

REFERENCES

[1] L. D. Harmon, "Automated Tactile Sensing", in *The International Journal of Robotics Research*, vol. 1, no. 2, pp. 3-32, 1982.

[2] G. Cannata and M. Maggiali and G. Metta and G. Sandini, "IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems", in *4th International Conference on Cognitive Systems (CogSys)*, pp. 434-438, 2008.

[3] M. Strohmayr and D. Schneider, "The DLR artificial skin step I: Uniting sensitivity and collision tolerance", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1012-1018, 2013.

[4] S. Caviglia, M. Valle and C. Bartolozzi, "Asynchronous, event-driven readout of POSFET devices for tactile sensing", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2648-2651, 2014.

[5] S. Caviglia, L. Pinna and C. Bartolozzi, "An event-driven POSFET taxel for sustained and transient sensing", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 349-352, 2016.

[6] W. W. Lee and S. L. Kukreja and N. V. Thakor, "A kilohertz kilotaxel tactile sensor array for investigating spatiotemporal features in neuromorphic touch", in *Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1-4, 2015.

[7] F. Bergner, P. Mittendorfer, E. Dean-Leon and G. Cheng, "Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2124-2129, 2015.

[8] F. Bergner, E. Dean-Leon and G. Cheng, "Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4918-4924, 2016.

[9] M. Neugebauer, and K. Kabitzsch, "A New Protocol for a Low Power Sensor Network", in *IEEE International Conference on Performance, Computing and Communications*, pp. 393-399, 2004.

[10] M. Miskowicz, "Send-on-delta Concept: An Event-Based Data Reporting Strategy", in *sensors*, vol. 6, no. 1, pp. 49-63, January 2006.

[11] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor", in *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566-576, February 2008.

[12] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output", in *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470-1484, October 2014.

[13] C. Bartolozzi, and G. Indiveri, "Selective Attention in Multi-Chip Address-Event Systems", in *Sensors*, vol. 9, no. 7, pp. 5076-5098, 2009.

[14] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow", in *Neural Networks*, vol. 27, pp. 32-37, 2012.

[15] G. Grunwald, G. Schreiber and A. Albu-Schäffer and G. Hirzinger, "Programming by Touch: The Different Way of HumanRobot Interaction", in *IEEE Transactions on Industrial Electronics*, 2003.

[16] D. Massa, M. Callegari and C. Cristalli, "Manual guidance for industrial robot programming", in *Industrial Robot*, 2015.

[17] T. Wösch and W. Feiten, "Reactive Motion Control for Human-Robot Tactile Interaction", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[18] P. Mittendorfer, E. Yoshida, and G. Cheng, "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot", in *Advanced Robotics*, vol. 29, no. 1, pp. 51-67, February 2015.

[19] M. Fritzsche and N. Elkmann and E. Schulenburg, "Tactile sensing: A key technology for safe physical human robot interaction", in *Proceedings of the 6th International Conference on Human-robot Interaction*, pp. 139-140, 2011.

[20] Robot Bosch GmbH, "APAS Intelligent Systems for Man-Machine Collaboration", 2016.

[21] F. Nori and S. Traversaro and J. Eljaik and F. Romano and A. Del Prete and D. Pucci, "iCub whole-body control through force regulation on rigid non-coplanar contacts", in *Frontiers in Robotics and AI*, pp. 1-18, 2015.

[22] E. Dean-Leon, K. Ramirez-Amaro, F. Bergner, I. Dianov, P. Lanillos and G. Cheng, "Robotic Technologies for Fast Deployment of Industrial Robot Systems", in *The 42nd Annual Conference of IEEE Industrial Electronics Society (IECON)*, pp. 6900-6907, 2016.

[23] E. Dean-Leon, B. Pierce, P. Mittendorfer, F. Bergner, K. Ramirez-Amaro, W. Burger and G. Cheng, "TOMM: Tactile Omnidirectional Mobile Manipulator", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, In Press.

[24] P. Mittendorfer and G. Cheng, "3D surface reconstruction for robotic body parts with artificial skins", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[25] P. Mittendorfer, E. Dean-Leon and G. Cheng, "3D spatial self-organization of a modular artificial skin", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.