On the Problem of the Automated Design of Large-Scale Robot Skin

Davide Anghinolfi, Giorgio Cannata, Fulvio Mastrogiovanni, Cristiano Nattero, and Massimo Paolucci

***Abstract*—This paper describes automated procedures for the design and deployment of artificial skin for humanoid robots. This problem is challenging under different perspectives: on the one hand, different robots are characterized by different shapes, thereby requiring a high degree of *skin customization*; on the other hand, it is necessary to define optimal criteria specifying how the skin must be placed on robot parts. This paper addresses the problem of optimally covering robot parts with tactile sensors, discussing possible solutions with reference to a specific artificial skin technology for robots, which has been developed in the past few years. Results show that it is possible to automate the majority of the required steps, with promising results in view of a future complete automation of the process.**

***Note to Practitioners*—The research activities described in this paper have been carried out to tackle a novel problem in Robotics. Since different robots have different shapes (in terms of covers) and since—even narrowing the analysis to a single robot—different body parts of the *same* robot have different shapes, how can we automate the procedure of designing and deploying sensitive skin for robots? Sensitive skin for robots is a fundamental means for achieving a high degree of physical human-robot interaction in real-world scenarios. The design of sensitive skin for robots is difficult: modular, scalable and dependable solutions (both at the hardware and software levels) are needed, which are in contrast with the need of deploying the actual sensitive skin on a variety of surfaces. In this paper, we formally characterize the robot skin design process as a geometric placement problem, in an effort to obtain a tradeoff between skin modularity and skin customization for specific robot surfaces.**

***Index Terms*—Automated design, heuristics, robot skin.**

I. Introduction

DURING the past few years, many approaches for developing arrays of tactile sensors have been presented. The research has been driven by needs related to small-scale tactile arrays [1], the bio-inspired design of tactile elements [2], or human-robot interaction [3].

When considering the robot skin *as a whole*, novel problems must be taken into account, such as skin calibration [4], [5] and representation [6], infrastructure and networking [7], [8], as well as design and manufacturing [9]. This paper focuses on a specific aspect of the design process, namely, obtaining the highest *quality of skin coverage with respect to the robot shape*. As a matter of fact, when large-scale skin-like sensors must be assembled on robot body parts, it is necessary to define parameters for measuring coverage quality. These parameters range from covering as much of a robot surface area as possible to finding the *best* position of skin *patches* (i.e., independent sensing surfaces covering robot body parts) with respect to specific optimality criteria, such as the likelihood for a robot body part to be in contact with the environment, the fault tolerance capability associated with a specific skin patch or even the skin weight.

In principle, the problem could be easily *solved* by designing custom skin patches for every possible robot body part of every possible robot that needs coverage. The high manufacturing and set-up costs associated with these *unique pieces* of skin patches may prevent the widespread adoption of skin-like systems on robots. For example, let us consider the different patches covering various parts of the iCub robot (Fig. 1): each patch is characterized by its own shape and peculiarities. It is necessary to find a reasonable tradeoff between the optimality of the coverage results and such design principles as *modularity*, *scalability*, and *wiring complexity* [9], [10]. Modularity and scalability refer to the actual physical organization of basic skin modules that are independent in terms of sensing capabilities, data processing and communication. As a matter of fact, modules can enforce both skin coverage (since they can be replicated to fit specific local surface shapes) and fault-tolerance (given their independence). Furthermore, modularity allows for a very high scaling factor, which is a fundamental prerequisite for covering large robot body areas with low manufacturing and set-up costs. Scalability is a major issue in strongly embedded systems such as humanoid robots. For typical designs, cost and sensor complexity grow quadratically with respect to the sensing surface [11], and also wiring complexity is subject to physical constraints. Large-scale skin-like systems are expected to cover up to 1 m^2 body surface, whereas tactile data processing can occur in a hybrid setting, e.g., distributed microcontrollers managing skin modules exchanging information with centralized master nodes [7]. Scalability is a feasible strategy to reduce design and manufacturing costs by focusing on cheap, easy to manufacture modules that can be replicated throughout the surface as needed. On the one hand, issues related to wiring complexity in terms of number of wires and the integration with fieldbus communication channels must be—whenever possible—minimized, whereas on the other hand wiring can be used to enforce fault-tolerance through redundancy or smart assignment

Manuscript received June 20, 2012; revised November 16, 2012; accepted January 28, 2013. Date of publication April 11, 2013; date of current version October 02, 2013. This paper was recommended for publication by Associate Editor D. Popa and Editor A. Bicchi upon evaluation of the reviewers' comments. This work was supported in part by the European Community's Seventh Framework Program (FP7/20072013) under Grant 231500 (project ROBOSKIN) and Grant 288553 (project CloPeMa).

The authors are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, 16145 Genoa, Italy (e-mail: davide.anghinolfi@unige.it; giorgio.cannata@unige.it; fulvio.mastrogiovanni@unige.it; cristiano.nattero@unige.it; massimo.paolucci@unige.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2013.2252617

of modules to microcontrollers [12], [40]. Finally, when a network of microcontrollers is involved, automated solutions for identification, configuration and firmware management (e.g., ID assignment) must be provided [13].

The main contribution of this paper is a technique for the automated layout design of large-scale robot skin. In particular, given generic robot body parts, we discuss a general-purpose process to optimally place modular skin patches to cover such body parts. In this case, optimality refers to the maximization of the amount of robot body surface that is effectively covered at the end of the process. In principle, although the techniques introduced in the paper can be easily adapted to other skin layouts and modular designs, the discussion is grounded with respect to a previously developed artificial skin [9], [14], where basic elements are (interconnected) triangular modules, which has been developed in the context of the FP7 EC-funded ROBOSKIN project.[1] Specifically, we formally state the automated skin layout design as a polygon placement problem and, on the basis of this theoretical framework, we provide necessary and sufficient conditions to produce feasible placements and we propose a heuristic approach to obtain suboptimal placements.

This paper is organized as follows. Section II introduces relevant literature. Section III describes the current robot skin structure and introduces the main steps of the designed procedure. The placement problem is discussed in Section IV and Section V. Next, implementation details and experimental results are discussed. Conclusions follow.

II. Related Work

In the context of the tactile sensing literature, the topic of *optimal skin coverage* has not been addressed in a principled way, even if many related aspects are sparsely considered. The following paragraphs report about related work that—in different ways—introduces relevant concepts, ideas and requirements.

The importance of fabrication parameters such as the actual array shape, the thickness of the material, the effect of thermal conditioning and the size of a patch with respect to tactile data processing time has been pointed out by many authors, e.g., [15]–[17]. Similarly, the use of skin-like sensors in the context of system-level architectures is discussed by Cheung *et al.* [18]–[20]. Issues related to modularity and scalability are discussed with respect to motion planning and control, with a specific emphasis on the tradeoff between the density of tactile elements and the computational time needed to process tactile information. However, these considerations do not lead to general techniques to automate the skin customization process.

Cameron exploits techniques borrowed from statistical learning theory to determine sensor locations minimizing the amount of tactile data needed to perform shape recovery [21]. Although the important notion of optimally placing a sensor in the context of a specific task is introduced, the study is rather theoretical because it does not discuss the effects of optimal configurations on system-level design issues. Johnston *et al.* [22] present the coverage of an entire robot hand, considering a number of system-level requirements. However, the proposed solutions are completely manual: skin patches are specifically suited for the considered hand mechanical structure.

[1]Please refer to the official website at www.roboskin.eu.

A full-body suit for humanoid robots is discussed in [23], which is based on conductive fabric. This is one of the first examples where problems related to modularity, scalability, wiring complexity and data processing are considered in a principled way. The overall skin is divided into smaller patches, roughly corresponding to robot body parts. However, no specific focus is devoted to find optimal tradeoffs between patches and actual tactile-based tasks.

Um and Lumelsky [24]–[26] consider each patch as an independent module characterized by a number of properties related to configuration, interconnection and data acquisition. Such modularity is back-propagated to the communication infrastructure as well. Patches can be reconfigured to fit different robot shapes: however, neither optimality conditions about skin placement nor best practices for wiring and data acquisition are described.

Force detectable surface covers have been introduced by Iwata *et al.* [27]. Information originating from both resistive and force sensors to correlate pressure information and the exerted force is exploited. Although the presented system can be used to cover large surfaces, the actual design is strictly dependent on the actual robot surface. However, a principled discussion about design issues is not carried out.

Ohmura *et al.* [28], [29] have presented design and implementation results for a conformable and modular large-scale skin. Based on a *H-like* shape, the skin is made up of a number of interconnected modules that can be cut, bent or self-overlap to better cover selected robot body parts. Coverage experiments related to a full humanoid robots are presented. Although modularity and scalability issues are tackled in depth, the skin deployment is still a manual operation: no algorithm is provided that is able to automate the overall process.

System-level engineering problems have been considered in [30], where a multimodal sensitive skin is introduced. Modules roughly correspond to different morphological robot body parts. However, the actual design is neither easily scalable nor automated, since skin coverage is manually defined.

A 272 taxels array for the ARMAR-III robot has been discussed by Goger *et al.* [13], [31]. The authors clearly describe the need for a modular structure, recognizing the importance of the wiring problem as well. The array is a square matrix built on a flexible printed circuit board (fPCB in short) to be bent across surfaces of single curvature. In order to conform the array shape to different robot body parts, the matrix can be cut *manually* along predefined lines.

Yoshikai and collaborators presented a soft stretchable knit sensor [32]. The overall sensing structure is embedded within a cardigan-like soft stretchable exterior. All the layers of the conductive fabric are knitted for improved stretchability. Although different patches for humanoid robot arms, legs and torso have been produced, the process is manual and sensor placement is not guided by any optimization procedure.

Shimojo *et al.* [33] describe design and development issues of a high speed tactile matrix able to cover surfaces of arbitrary curvatures. The matrix is arranged as a net, where only nearby taxels are connected through wires. Modularity issues are not properly addressed, since the shape of a patch must be specifically designed *by hand*. Scalability is enforced by the smart

wiring infrastructure: only four wires actually constitute the interface of the patch.

The Hex-o-skin design [5], [34] exploits hexagonal interconnected robot skin modules that can be applied to robot surfaces. Tests have been shown where the skin is assembled on robot manipulators. However, there is no evidence of any principled (i.e., automated) skin layout customization method.

Very recently, Buchan and colleagues [35] have proposed a robot skin design which exploits printable electronics. The resulting skin layout can be cut along a number of predefined lines in order to conform to the robot shape. A simple example is provided and a proof-of-concept design for a small biomimetic robot is discussed, which is done by hand.

Summarizing, from the analysis of the literature it emerges that the process for actually deploying large-scale robot skin *is still largely manual*, given that skin patches are usually arranged *ad hoc* on different robot body parts.

III. The Problem of Robot Surface Coverage

A. Large-Scale Modular Artificial Skin for Humanoids

The proposed skin system comprises a number of capacitance-based tactile sensors (i.e., *taxels*) distributed throughout a surface. The transduction mechanism is rather straightforward: a soft dielectric is placed amidst two electrodes, which are the plates of the capacitors [9], [14]. When a force is applied, the distance between the electrodes changes as a consequence of the varied pressure. This entails a change in the overall capacitance, which inversely depends on the distance between the plates.

As shown in Fig. 2, the transducer's lower plates are hosted by flexible PCBs, which are shaped as triangular modules of 3 cm side. The triangular shape of the PCB has been selected to better conform to generic smooth curved surfaces.

Each module includes 12 taxels (round golden pads in Fig. 2) as well as the ancillary electronics (i.e., a capacitive to digital converter chip, CDC in short) to obtain the related pressure measurements, which are forwarded through a serial I^2C communication bus. Each CDC chip can be assigned with a 2 bit address along the bus line, thus up to four modules can be managed by the same I^2C bus. Each triangle side is provided with a communication port. In particular, one port is considered an *input*, whereas the other two are *output* ports. The rationale is that it is possible to route an input signal to two nearby triangles, thereby avoiding to use dedicated cabling for each triangular module.

Each patch is associated with an external microcontroller sampling all the tactile data from different I^2C lines. Currently, four bus lines can be managed by a microcontroller, which implies that a patch can be constituted by up to 16 triangles. However, each microcontroller is associated with an entry point in the patch: all the four I^2C bus lines enter from one specific module and are then differently routed to reach the assigned modules.

The designed modular structure allows to cover robot body parts with different shapes, since triangular modules can be arranged to fit different surfaces. Furthermore, triangle corners can be cut to a smaller extent to better conform to the surfaces themselves. It is noteworthy that the main focus of this design are large-scale robot body surfaces, such as the palm, the forearm, the arm, the torso or the legs of a humanoid robot (Fig. 1), although a variation has been exploited in [36] to build a sensing fingertip for the iCub humanoid robot. Despite the fact that in this paper we assume all the triangular modules to be identical as far as number of taxels, dielectric material, and electronics are concerned, it is possible to apply the discussed techniques also when these features differ (e.g., different number of taxels per triangular module or different dielectric material for obtaining a given sensitivity [37]) subject to the fact that the triangular shape is preserved. This assumption allows us to model skin coverage as a problem of polygon placement over a regular isometric grid. If one wanted to extend these algorithms to different module shapes (e.g., hexagonal shapes as in [34]), the procedure should be modified to consider a different regular grid (i.e., hexagonal instead of isometric).

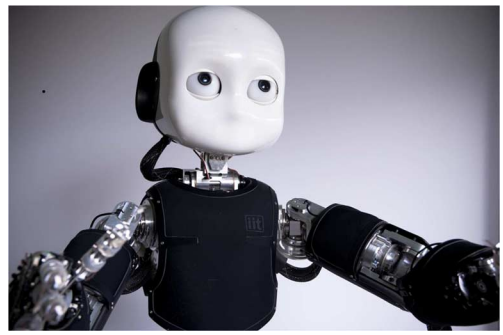

Fig. 1. The different skin patches that cover the iCub torso, upper and lower forearms and arms (courtesy of the Italian Institute of Technology).

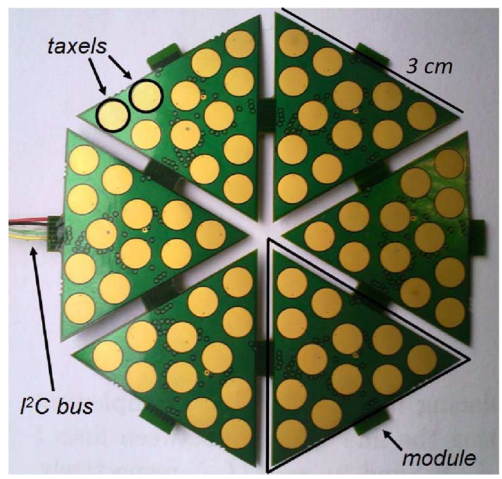

Fig. 2. A small skin patch made up of 6 triangular modules organized as a hexagon.

B. The Automated Skin Placement Process

Triangular modules can be modeled as triangles of equal side. Given the skin layout in Fig. 2, it is possible to think about a large sensing structure as an isometric grid where the single element is the triangular module. On these premises, a comprehensive approach for the overall skin deployment process can be split into three steps (Fig. 3).

1) *Body part surface unfolding*. The process starts from robot CAD models. Given such a model, it is necessary to extract the surface of the robot body part to cover, which results in

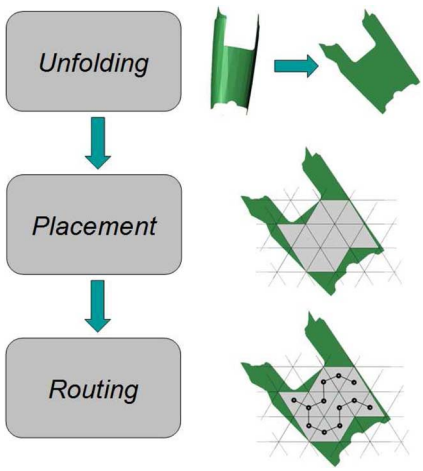

Fig. 3. The overall skin deployment process. Step 1) Unfolding. Step 2) Placement. Step 3) Routing.

a 3D mesh. Then, the mesh is unfolded in order to obtain a 2D representation that can be placed on a triangular grid (Fig. 3 on the top), in the form of a polygon P.

2) *Optimal body part coverage*. Once a polygon P has been obtained as a result from the first step, the actual placement problem over a triangular isometric grid G can be set up (Step 2 in Fig. 3). The polygon is moved over the grid in order to maximize the number of fully covered modules. It is straightforward to show that this is equivalent to determining the modules configuration best fitting the polygon surface. As anticipated, the optimality criteria that we want to maximize is the amount of covered area. The result is a set of connected modules that constitute the optimal skin patch $\mathbb{P}$ for the robot body part.
3) *Optimal bus routing in patches*. Given a patch $\mathbb{P}$, the optimal routing of I^2C buses across triangular modules must be defined (Step 3 in Fig. 3). In particular, it is necessary to associate bus lines to modules, as well as defining the cable routing.

Given the huge amount of work done in polyhedra unfolding [38], a freely available implementation of a state-of-the-art technique has been adopted [39]. Furthermore, with reference to the introduced skin system, this paper is focused on Step 2 in Fig. 3, i.e., the process of finding the optimal placement of skin patches over generic robot surfaces. A principled treatment of Step 3 is out of the scope of this paper. The interested reader is referred to relevant literature, e.g., [12] and [40].

It is noteworthy that the use of an isometric grid is a reasonable approximation under certain conditions. As a matter of fact, two possible sources of uncertainty must be considered. On the one hand, the 2D flat representation produces either distortions or cracks in the 2D polygon P. This leads to an inaccurate representation of the body part in 2D and, in general, may lead to bad coverage results. However, as discussed in Section IV-B, we argue that a first *manual* subdivision of the body surface to cover can minimize this problem to a great extent. On the other hand, the optimal skin patch $\mathbb{P}$ requires to be able to physically *cut* PCB connections in the real skin hardware in order to obtain the desired patch shape. To this aim, the underlying hardware must support the online reconfiguration of the data buses across the various modules [12], [34], [35], [40], which is a reasonable assumption in modular robot skin layouts since it enforces fault tolerance.

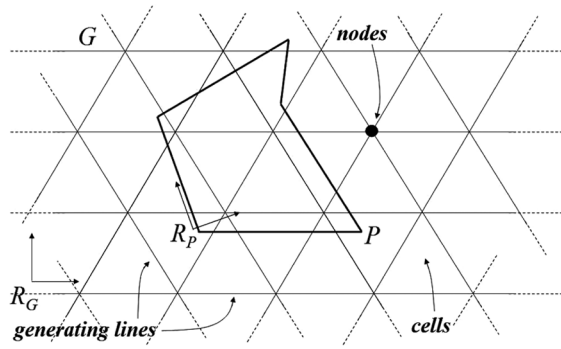

Fig. 4. A grid modeling triangular modules.

IV. Optimal Placement of Skin Patches Over a Grid

A. Problem Formalization and Definitions

With respect to the introduced robot skin technology, we model each triangular module as a *triangle* c. We refer to S_c and N_c as, respectively, the sets of sides and vertexes of c. All the triangular modules lie on a planar surface $\mathcal{S}$, and induce an *isometric grid* G (see Fig. 4) that replicates the physical arrangement of skin modules in patches [41].

Although a grid G is made up of triangular modules, it is useful to consider it as being *generated* by three distinct sets of parallel lines, respectively oriented at 0°, 60°, and 300°. We refer to each of these lines as a *generating line* l. Therefore, a grid can be thought of as a set $G = \{l_1, \ldots, l_n\}$ of n generating lines. We denote with MS_l the set of module sides composing the generating line l.

Unfolded robot surfaces are modeled as *simple* and *closed* polygons P, each one characterized by a finite set E_P of line segments and a finite set V_P of vertexes. A polygon is *simple* if its edges do not intersect with each other. Since the parameter we want to maximize is the coverage area, we indicate with $A(\cdot)$ the area of a generic planar surface.

Definition 1: Given a grid G with an associated reference frame R_G and a polygon P with an associated reference frame R_P, a *placement* $\rho(P)$ is a transformation of R_P with respect to R_G, which preserves internal angles, mutual distances between vertexes in V_P and area $A(P)$ (see Fig. 4).

A placement $\rho(P)$ is uniquely defined using four parameters (x, y, θ, v). Given a position for P (either as a vertex $v \in V_P$ or not), x and y are the horizontal and vertical displacements of R_P, whereas θ is the orientation of R_P with respect to R_G.

Definition 2: Given a grid G, a polygon P, and a placement $\rho(P)$, a *patch* $\mathbb{P}$ is the set of triangular modules *entirely contained* in P.

The polygon P is the result of the unfolding procedure of a surface M corresponding to a given robot body part. A skin patch for M is made up of all the modules contained within P (see Fig. 5). Since a reasonable measure of coverage is the number of skin modules that can be assembled on M, then the

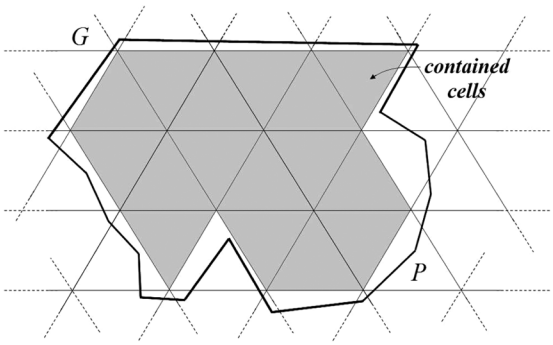

Fig. 5. A patch (i.e., gray modules) obtained as a placement of P over G.

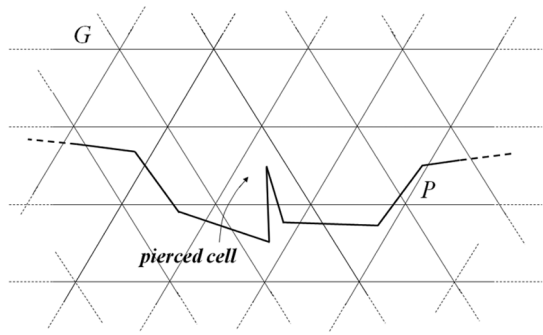

Fig. 7. Polygon P pierces a grid module.

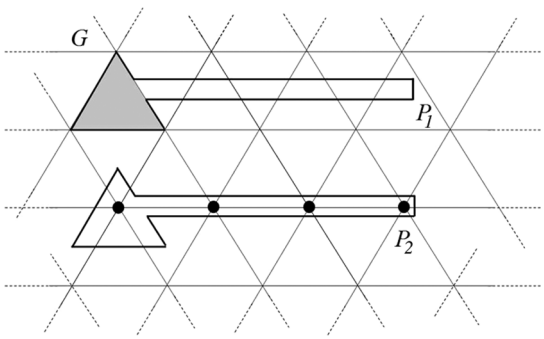

Fig. 6. Top: module containment. Bottom: node containment.

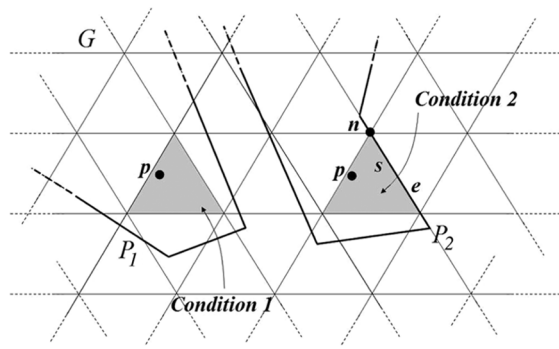

Fig. 8. Left: Condition 1. Right: Condition 2.

number of triangular modules entirely contained in P must be maximized.

Definition 3: Given a grid G, a polygon P, and a placement $\rho(P)$, the *objective function* $\mathfrak{C} : \rho(P) \longrightarrow \mathbb{N}$ measures the number of triangular modules c entirely contained in P.

Definition 4: A placement $\rho^*(P)$ is said to be *optimal* if, for each allowed placement $\rho(P) \neq \rho^*(P)$, it holds that the number of triangular modules entirely contained in P is the lowest.

In the following paragraphs, we refer to $\mathbb{P}^*$ as the *optimal patch* obtained by applying $\rho^*(P)$.

Remark: In principle, since P may be either convex or concave, it is not sufficient to adopt commonly used algorithms for solving the *maximum point containment* problem [42], [43]. As a matter of fact, a placement $\rho^*(P)$ maximizing the number of modules entirely contained is different from a placement where the number of vertexes is maximized. Fig. 6 on top shows a related example where polygon P_1 contains one module but no vertexes (other than those of the module itself), whereas the polygon on the bottom contains no modules but four vertexes. □

In order to check for the containment of a module c, testing whether its nodes lie inside the polygon P is not sufficient: as it is shown in Fig. 7, a concave polygon P might "pierce" c. In order to better characterize the behavior of the proposed placement algorithm, it is necessary to formally introduce the notion of *entire containment*. To this aim, we refer to ∂P as the *border* of the polygon P, i.e., the set of segments that limit the polygon area, and to $cl(P)$ as the *closure* of P, i.e., as the union of the polygon area P and border ∂P.

Definition 5: Given a polygon P and its closure $cl(P)$, a module c is said to be *entirely contained* in P, and we indicate this with $c \in P$, if all the sides $s_i \in S_c$ and the nodes $n_i \in N_c$, $i = 1, \ldots, 3$, lie within $cl(P)$.

It is noteworthy that, in Definition 5, the typical set terminology is used. As a matter of fact, a generic triangular module c and a polygon P are sets in $\mathbb{R}^2$. As such, we can refer to their *containment* relationships as *inclusion* relationships between sets. In order to test for the previous condition to hold, it is convenient to check for two separate conditions.

Condition 1: The first condition is related to modules that are entirely contained in a polygon P. If the edges of a module c have no intersection with the polygon border ∂P, c can be either inside or outside the polygon. Furthermore, if at least one node of c is inside P, then c is entirely contained, or *vice versa* (see the left triangle in Fig. 8). In other words, given a polygon P and a module c: 1) there is no side $s_i \in S_c$ intersecting the polygon border ∂P and 2) at least one point $p \in c$, typically a node $n \in N_c$, lies inside P.

Condition 2: The second condition describes a situation where a module c *touches* the polygon border ∂P but the edges of P do not pierce c. In this case, a side can touch ∂P either from the inside or the outside: this can be verified by checking that at least one point inside c is inside $cl(P)$ as well. It is noteworthy that modules *touching* ∂P from the inside are also added as elements of $\mathbb{P}$ (see the right triangle in Fig. 8). In other words, given a polygon P, let us consider an edge $e \in E_P$ and a vertex $v \in V_P$; for a module c: 1) at least one side $s \in S_c$ has a non-empty intersection with ∂P; 2) all the module nodes $n \in \mathbb{N}(c)$ are inside $cl(P)$; 3) a point $p \in c$ exists such that $p \in P$; 4) for all polygon edges e intersecting c one of the following conditions holds: a) e is parallel to s, or b) the intersection point between e and s is v.

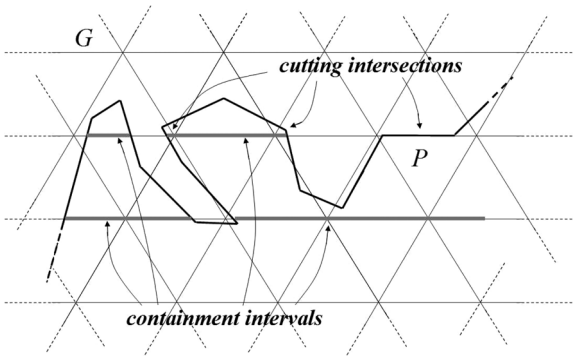

Fig. 9. Cutting intersections and containment intervals.

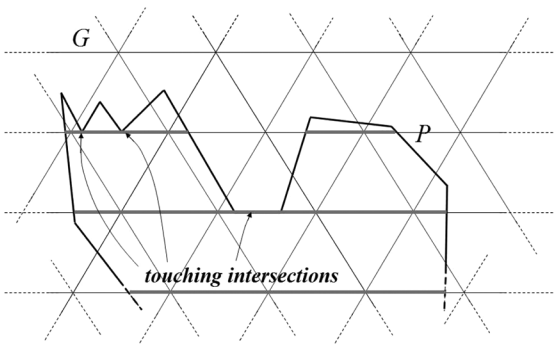

Fig. 10. Touching intersections do not contribute to containment intervals.

From the two previous conditions, it emerges the need for defining more precisely the concept of *intersection*.

Definition 6: Given a grid G and a polygon P, a generating line $l \in G$ divides the plane into two half planes. Let us consider the set $\mathbb{I}$ of all the intersections between P and l, i.e., $\mathbb{I} = \{i_1, \ldots, i_g\}$. Given the generic intersection i, let us call $v_{i-} \in V_P$ and $v_{i+} \in V_P$ the vertexes of P before and after i, scanning ∂P in a chosen order, (e.g., clockwise). If v_{i-} and v_{i+} belong to the same half plane, i^t is said to be a *touching intersection*. If they belong to different half planes, i^c is a *cutting intersection*.

Henceforth, we will indicate with $\mathbb{I}^t$ the set of touching intersections and with $\mathbb{I}^c$ the set of cutting intersections. It must be pointed out that also cases such as that of Fig. 9 on the right hand side can be classified as cutting intersections. If we ideally shrink the segment of the polygon P lying on the generating line to a single point, we can observe that the vertexes preceding and following it belong to different half planes (with respect to the same generating line), which is exactly the condition for a cutting intersection. On the contrary, if we refer to Fig. 10 and we ideally shrink the segment lying on the generating line to a single point, we can notice that both the preceding and following vertexes belong to the same half plane, which is the condition for a touching intersection.

It is possible to exploit cutting intersections to determine whether a given line segment of a generating line is inside the polygon.

Definition 7: Given a grid G, a polygon P and a generating line $l \in G$ such that $\mathbb{I} \neq \emptyset$, then a *containment interval* ci is a line segment entirely contained in P such that $ci \subset l$.

Theorem 1: Given a grid G, a polygon P, and a generating line $l \in G$ such that $\mathbb{I} \neq \emptyset$, each containment interval $ci_{j,j+1}$ is defined by a pair (i^c_j, i^c_{j+1}) of cutting intersections such that $j \bmod 2 = 1$.

Proof: Let us consider the subset $\mathbb{I}^c \subseteq \mathbb{I}$ made up of all the cutting intersections i^c_j. Since l is an oriented line, we can define a partial ordering $U_{\preceq}$ on the elements of $U_{\preceq}(\mathbb{I}^c) = \{i^c_1, \ldots, i^c_j, i^c_{j+1}, \ldots, i^c_g\}$. Since P is assumed simple, it is possible to define an ordered sequence of pairs (i^c_j, i^c_{j+1}), $j = 1, \ldots, (g-1)$, each one representing line segments lying on l alternatively inside and outside P, with the first pair (i^c_1, i^c_2) inside. Since the condition $j \bmod 2 = 1$ identifies odd pairs, these correspond to containment intervals. ■

From the previous Definition, it follows immediately that containment intervals $ci_{j,j+1}$ are implicitly defined as pairwise pairs of cutting intersections (see Fig. 9). It can be easily shown that, for each polygon, if $|\mathbb{I}^c| > 0$ then $|\mathbb{I}^c|$ is even. In particular, given a point $p \in l$, it holds that $p \in ci_{j,j+1} \Rightarrow p \in P$.

Definition 8: Given a grid G, a polygon P, and a generating line $l \in G$ such that $\mathbb{I}^c \subseteq \mathbb{I} \neq \emptyset$, $CI(\mathbb{I}^c_{\preceq}) = \{ci_{1,2}, \ldots, ci_{g-1,g}\}$ is defined as the set of all the containment intervals for the generating line l.

As shown in Fig. 10, the set $\mathbb{I}^t$ of touching intersections is not useful to define containment intervals. It is now possible to formally define the containment for a module side.

Theorem 2: Given a grid G, a polygon P, a module $c \in G$, and a generating line $l \in G$, a module side $s \in S_c$ or a module node $n \in N_c$ is *contained* in P if it belongs either to a containment interval $ci_{j,j+1}$ or to a polygon edge e such that $e \subset l$ under the current placement $\rho(P)$.

Proof: A module side s or a node n are considered contained in P either if they lie inside P or on its border ∂P. In the first case, if a containment interval ci exists such that $s \in ci$ or $n \in ci$ then s and n are contained in P for Theorem 1. The second case is straightforward. ■

This theorem is the core result which allows us to discriminate triangles that are located inside a given polygon P, which will be used in the following section. For the following discussion, it is important to introduce also the concepts of *contact* and *stable placement*.

Definition 9: Given a grid G and a polygon P, a *contact* or *event* is a placement $\rho(P)$ with respect to G such that [44], [45]: *Type 1*) given a module c and an edge $e \in E_P$, a node $n \in N_c$ *touches* e; *Type 2*) given a module c and a vertex $v \in V_P$, v *touches* a side $s \in S_c$.

Definition 10: A placement $\rho(P)$ is *stable* if it exhibits three distinct and independent contacts of any type [44].

According to the last classical definition, a contact is defined using generic *segments* and *points*. Therefore, in the case of Type 2 contacts, two distinct contacts happen: this property is used to simplify the generation of stable placements as described in Section V.

B. Evaluation of the Objective Function

In this section, we describe an algorithm that, given a grid G, a polygon P and a generic placement $\rho(P)$ over G, produces a patch $\mathbb{P}$ as well as the associated value F for the objective function $\mathfrak{C}$. Many general purpose algorithms exist [46], [47]

to solve standard polygon containment problems. However, the problem of placing a polygon over a grid allows for more efficient—yet peculiar—solutions.

Algorithm 1: Evaluate$\mathfrak{C}(\cdot)$

Require: a grid G and a placement $\rho(P)$

Ensure: a value F, a patch $\mathbb{P}$

1: {Initialize the value of the objective function F and the number of contained sides ns_j for each module c_j to 0}
2: {Initialize the patch $\mathbb{P}$ as empty}
3: **for all** generating lines l **do**
4: {Determine the ordered set CI of containment intervals as in Definition 8}
5: **for all** containment intervals $ci \in CI$ **do**
6: **for all** the module sides s which lie on l and overlap with ci **do**
7: **for all** modules c_j the side s belongs to **do**
8: {Increase the number of contained sides for c_j}
9: $ns_j \leftarrow ns_j + 1$
10: **end for**
11: **end for**
12: **end for**
13: **end for**
14: **for all** the module sides c_j **do**
15: {c_j is contained if and only if all of its sides s (three for triangular modules) are contained according to Definition 5}
16: **if** $ns_j = 3$ **then**
17: {Update the value N of the objective function}
18: $F \leftarrow F + 1$
19: {Update the patch $\mathbb{P}$}
20: $\mathbb{P} \leftarrow \mathbb{P} \cup c_j$
21: **end if**
22: **end for**
23: **return** F, $\mathbb{P}$

Algorithm 1 describes this procedure. As an initialization step, the value F of the objective function and the number of sides ns of each triangular module that are contained in the polygon are set to 0 (line 1). Then, the patch $\mathbb{P}$ is set as empty (line 2). Each placement $\rho(P)$ triggers a number of operations on generating lines l (lines 3–13). First, the set of all the cutting intersections $\mathbb{I}^c$ is found, by determining all the intersections between l and edges of P, as pointed out in Definition 6. Then, the set is ordered according to the direction associated with l, i.e., from *left* to *right*. Then, the set of all the containment intervals CI is determined by considering pairs of cutting intersections as described by Definition 8 (line 4). These define parts of l lying inside P. In order to determine this set, Theorem 1 is applied to every couple of cutting intersections and the related condition evaluated. Then, it is necessary to relate containment intervals with module sides. As it has been stated in Theorem 2, a module side s is inside P if it lies on a containment interval or on ∂P. As a consequence, for each containment interval ci and s lying on l, the mutual overlap is checked (line 6): if they overlap then s is inside P and the number of contained sides for the related module is updated (line 9). If, for a given module c_j (line 14), this number sums up to the number of cell sides (e.g., three in the case of triangular modules), then c_j is inside P: as a consequence, F is incremented (line 18) and c_j is added to the current patch (line 20).

Remark: It is possible to characterize the computational complexity of Algorithm 1. Note that the grid G is theoretically infinite in its size. However, a preprocessing stage occurs to avoid exploring all the generating lines in the grid, by creating a suitable *bounding box* around the polygon P and considering all the generating lines intersecting the bounding box. Furthermore, the overall set of generating lines is divided into three disjoint sets of parallel lines on the basis of their orientation, i.e., 0°, 60°, and 300°. Therefore, according to the worst case analysis, Algorithm 1 runs in

$$O\left(n \times |E_P| \times |MS^*|^2\right) \tag{1}$$

where n is the number of generating lines l, $|E_P|$ is the number of polygon edges and $|MS^*|$ corresponds to

$$|MS^*| = \arg\max_l \{|MS_l|\}. \tag{2}$$

In particular, $|MS^*|$ models the worst case for the two inner loops of Algorithm 1, and it corresponds to the number of module sides lying on a generating line. Worst cases are related to the presence of only one containment interval for a generating line (see Fig. 10 on the bottom), where all the lying modules must be checked. □

C. Bounds and Optimality Conditions

As discussed in the previous section, the periodicity of the grid G allows for a number of simplifications in the exploration strategy of possible placements $\rho(P)$. Two of them are of the utmost importance in the considered scenario.

First, it is not necessary to assign all the possible values to decision variables (x, y, θ, v) to find the optimal placement. On the one hand, as it has been demonstrated in [45], x and y are exhaustively explored by considering possible positions for a given $v \in V_P$ inside a single module c. On the other hand, given the peculiar triangular structure, for the rotational component it is sufficient to explore values of $\theta \in [0, 60)°$.

Second, it has been shown in [44] that, if an optimal placement $\rho^*(P)$ exists, then an equivalent *stable placement* $\rho_s^*(P)$ (in the sense of Definition 10) can be found, such that the optimal value of the objective function is preserved. This insight

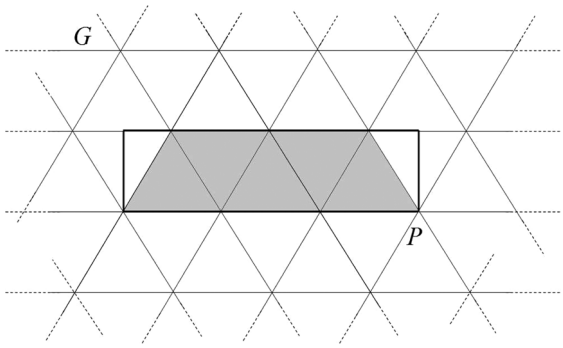

Fig. 11. Theoretical upper bounds versus placement bounds.

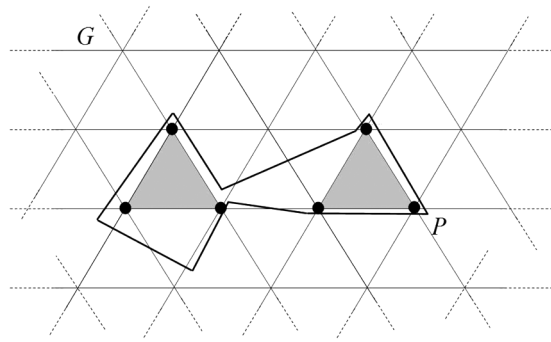

Fig. 12. A patch made up of disjoint triangles.

can further reduce the search space, because it implies that only stable placements must be searched for.

With respect to this formalization, it is possible to define theoretical lower and upper bounds for the values assumed by $|\mathbb{P}|$. It is straightforward to verify that the upper bound B_u can be determined as:

$$B_u = \left\lfloor \frac{A(P)}{A(c)} \right\rfloor \tag{3}$$

where c is a generic module and A is the area of a module or a polygon. The lower bound $B_l = 0$ is related to the case where $A(P) < A(c)$ (i.e., the area of the surface to cover is smaller than the single module).

Two considerations can be made. The first refers to the fact that the theoretical value of B_u can be unreachable for the physical configuration of the triangular grid. Fig. 11 shows an example where the $B_u = 6$, whereas the best stable placement can produce an optimal patch $\mathbb{P}^*$ where $|\mathbb{P}^*| = 5$. The second is related to the difference between the nominally designed skin layout for P and the result of placing the physical sensors on the robot body part M: a slight displacement between designed and real locations can be present due to skin mechanical stress induced by conformance to varying curvatures as well as for the unfolding procedure, thereby requiring a further calibration procedure [4], [6].

V. Placement Algorithms

In principle, a theoretical *naive* algorithm for *optimally* solving the *placement problem* can be easily designed, specifically in the form of an iterative procedure. Given a grid G and a polygon P, and assuming that the surface area to cover is larger than a module c, the algorithm considers all the (not necessarily connected) subsets $G_{F,i} \subseteq G$ formed by F modules, with F (i.e., the value of the objective function $\mathfrak{C}$) initialized to 1 and incremented—theoretically—up to the upper bound B_u. Given F, the algorithm defines all the possible combinations $P_{F,i}$, for which it is necessary to explore all the stable placements until a suitable one is found, such that $P_{F,i}$ is contained within P. If such a placement exists, then F can be incremented, and the whole process is repeated. Otherwise, $P_{F,i}$ defines the optimal patch.

The generic subset $G_{F,i}$ forms a set of polygons with $|V_{P_{F,i}}|$ vertexes, amounting to $|V_{P_{N,i}}| = 3F$ in the worst case. As shown in Fig. 12, this can happen for instance when in P two main regions can be clearly identified, which are connected by a thin surface that cannot be covered. In the figure, the solution patch is composed of two distinct and separate modules: the total number of vertexes amounts to $|V_{P_{N,i}}| = 3 \times 2 = 6$. According to [44], this algorithm runs in

$$O\left(|V_P|^3 \times \left|V_{G_{F,i}}\right|^3 \times \left(|V_P| + \left|V_{G_{F,i}}\right|\right) \times \log\left(|V_P| + \left|V_{G_{F,i}}\right|\right)\right).$$

Slightly faster algorithms have been designed [48], but they are still characterized by a high computational complexity, for instance

$$O\left(|V_P|^3 \times \left|V_{G_{F,i}}\right|^3\right). \tag{4}$$

However, if h is the number of considered modules in G,[2] this series of evaluations must be repeated

$$\sum_{F=1}^{\mathbb{C}(\rho^*(P))} \binom{h}{F} \tag{5}$$

times. Therefore, although the complexity of the exact Algorithm is output-dependent, it is nevertheless *exponential*.

These considerations led us to the development of a number of placement heuristics, which have been experimentally validated. However, it is necessary to point out that this problem is fundamentally different from the polygon containment problem described in [44]. On the one hand, the containment problem assumes that, given two polygons P_1 and P_2, it is possible to find an optimal stable placement $\rho_s(P_1)$ such that P_1 is entirely contained within P_2. On the other hand, the problem described in this paper is to place a polygon P over a grid such that the number of contained modules is maximized. In principle, it is possible to consider the resulting patch as a polygon to be placed inside P. Obviously, this cannot be done in practice because it would be necessary to know beforehand which modules constitute the patch. As a consequence, it is not possible to guarantee that the proposed heuristics are able to find stable placements, because the shape of the patch is not *a priori* known.

The presented heuristics have been designed to enforce contacts between the polygon P and module sides or nodes, with the aim of facilitating the presence of three distinct contact points, either of *Type 1* or *Type 2*. As pointed out in Section IV-A, it is necessary to decide values for the decision

[2]The grid size is bounded since a bounding box around P is built.

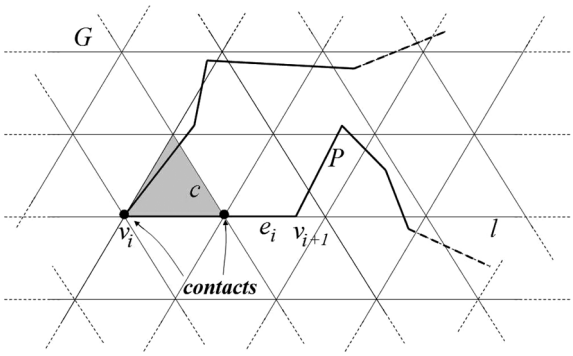

Fig. 13. A possible placement produced by heuristic R_2.

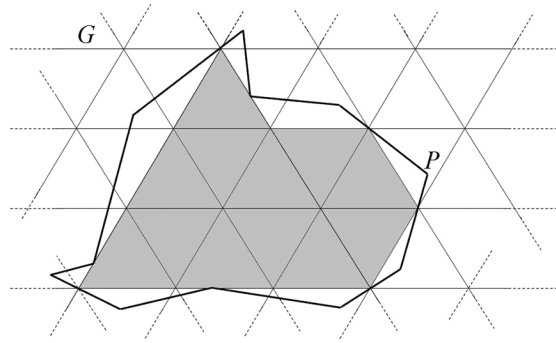

Fig. 14. An optimal placement that cannot be obtained with heuristics R_2, R_3, and R_4.

variables (x, y, θ, v). A set $\mathfrak{R} = \{R_1, \ldots, R_7\}$ of seven different methods has been designed to generate a suitable set of placements $\rho_i(P)$. In particular, decision variables are assigned using the following strategies.

- R_1 uses a Monte Carlo method.
- R_2 selects each polygon vertex in sequence and aligns polygon edges having the next and the preceding vertices in the sequence as end point to a selected generating line.
- R_3 selects pairs of (possibly non consecutive) polygon vertexes in order to align the segment having them as end points to a selected generating line.
- R_4 extends R_3 allowing the polygon to translate aligning each polygon vertex in sequence with respect to a grid node.
- R_5 extends R_2 allowing the polygon to translate to any randomly chosen point in the grid space.
- R_6 is complementary to R_5 in that it randomly selects the orientation for the polygon.
- R_7 extends R_3 allowing the polygon to translate to any randomly chosen point in the grid space.

With respect to the described strategies, random generation adheres to uniform distributions, where $\theta \in [0, 60)^\circ$ and pairs (x, y) are uniformly generated within module areas. A detailed discussion of the proposed strategies is beyond the scope of the paper. However, the most promising strategy is discussed in Appendix A.

Placements produced by heuristics R_2, R_3, and R_4 tend to be *stable* in the sense of Definition 10. Although the described heuristics prove to work well in realistic cases (see Section VI), they are not guaranteed to generate all the possible stable placements, since the placement takes only the mutual geometric relationship between the polygon and a single module into account. For instance, Fig. 13 represents a possible placement obtained with R_2 (considering v_i and the edge e_i): it depicts a case where P is stable with respect to module c, but nothing can be said about the stability of the obtained patch $\mathbb{P}$: as a matter of fact, c is not even contained in P.

Fig. 14 depicts a case that cannot be handled by the described heuristics: a polygon in its best placement over a module neither has vertexes on grid nodes, nor diagonals are parallel to module sides, and therefore to grid lines. Placements produced by R_5, R_6, and R_7 are aimed at overcoming this drawback by exploiting a limited use of Monte Carlo technique. The reason is that the case in Fig. 14 requires the polygon to be tightly *packed* among contacts, a case that cannot be managed by R_2, R_3, and R_4 since they require vertexes to correspond to grid nodes.

Algorithm 2: Determine$\mathbb{P}^*(\cdot)$

Require: a grid G, a polygon P and a set $\mathfrak{R} = \{R_1, \ldots, R_7\}$

Ensure: a suboptimal placement ρ^*, a suboptimal patch $\mathbb{P}^*$

1: {Initialize the optimal value of the objective function $\mathfrak{C}^*$ to 0}
2: {Initialize the optimal placement ρ^* as $null$}
3: {Initialize the optimal patch $\mathbb{P}^*$ as $empty$}
4: **for all** the considered heuristics R_i **do**
5: {Produce a number of placements Ξ_{R_i}}
6: **for all** produce placements ρ_j **do**
7: {Obtain the corresponding F_j and $\mathbb{P}_j$}
8: $(F_j, \mathbb{P}_j) \leftarrow \text{Evaluate}\mathfrak{C}(\rho_j)$
9: **if** F_j is greater that the current value of the objective function a better placement has been found **then**
10: {Update the best placement ρ^*}
11: $\rho^*(P) \leftarrow \rho_j(P)$
12: {Update the best objective function value $\mathfrak{C}^*$}
13: $\mathfrak{C}^* \leftarrow F_j$
14: {Update the optimal patch $\mathbb{P}^*$}
15: $\mathbb{P}^* \leftarrow \mathbb{P}_j$
16: **end if**
17: **end for**
18: **end for**
19: **return** $\rho^*(P)$, $\mathfrak{C}^*$, $\mathbb{P}^*$

Algorithm 2 describes the generic placement procedure taking all the described strategies into account. For each strategy, the Algorithm produces a set of possible placements, each one evaluated through Algorithm 1. We recall that Algorithm 1 exploits the theoretical results of Theorem

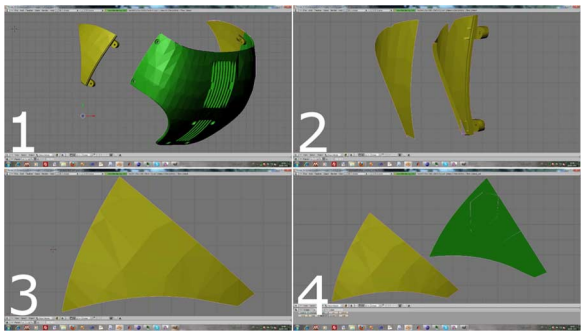

Fig. 15. Steps involved in the unfolding procedure: 1) separation of the left hip; 2) extraction of the surface; 3) mesh simplification; 4) unfolding.

2 in order to find the number of triangles entirely contained within the polygon P. The result is the suboptimal placement $\rho^*(P)$, the associated value of the objective function $\mathfrak{C}^*$ and the suboptimal patch $\mathbb{P}^*$. For each possible strategy R_i in $\mathfrak{R}$ (line 4), the set Ξ_{R_i} is initialized, which contains a number of placements generated using the related heuristic (line 5). The actual number of generated placements depends on the specific heuristic. Placements ρ_j are then evaluated using Algorithm 1, which produces a value F_j for the objective function and a possible patch $\mathbb{P}_j$ (line 8). If the value is greater than the current best one, it is selected as the new best value: as a consequence, the current best placement ρ^* and patch $\mathbb{P}^*$ are updated (lines 10–15). When all the heuristics have been evaluated, the procedure returns the final best parameters.

VI. Implementation and Experimental Evaluation

The automated design procedure described in the previous sections has been implemented using off-the-shelf and custom tools. The unfolding procedure has been implemented adopting the *Unfolder* script provided by Blender [39], an open source software application for 3D graphics. The placement algorithms have been coded in C++ using external libraries provided by the CGAL package [49], an open source tool for the development of computational geometry algorithms. All the tests have been performed on the CAD models of various body parts of the *iCub* robot [50], a well-known open source robotic platform developed by the Italian Institute of Technology. In particular, the experimental results reported here are an illustrative subset of the results, specifically related to five robot parts, namely the *left hip* (LH), the lower side of the *left upper arm* (LUA), the *lower torso* (LT), the *right upper forearm* (RUF) and the *upper torso* (UT). These body parts have been selected because they constitute a representative example in terms of surface area and different curvatures.

In the following paragraphs, we describe specific experiments and results for the first two steps of the overall process in Fig. 3.

A. Step 1: Unfolding

Fig. 15 depicts the steps involved in the unfolding procedure, specifically for LH. As anticipated, the *unfolder* script provided by Blender is adopted and applied to the surface of various robot covers. This is the only step requiring manual operation. In Fig. 15-1, the CAD model of the hip (in yellow) is

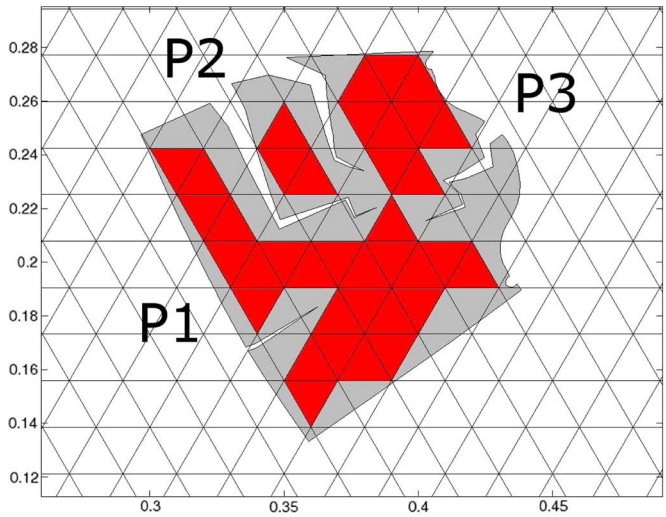

Fig. 16. The right upper forearm unfolded: three different patches are needed to cover the surface because of cracks.

isolated from the main structure (in green). Since the unfolding procedure does not distort the original 3D mesh, the flattened mesh is likely to be characterized by a number of *cracks* (i.e., spurious concavities in the resulting polygon). As a matter of fact, no method exists to flatten a general 3D manifold into 2D space without any distortion or crack. In this approach, a method occasionally producing cracks is preferable, since distortions introduced by *surface parametrization* techniques [51] are likely to cause a non homogeneous metric in the grid. However, cracks limit the search for good placements, since spurious concavities prevent the patch to uniformly cover the polygon. For instance, in the case of RUF (see Fig. 16), three different patches are produced by the placement procedure, which is due to the presence of cracks in the unfolded polygon.

In order to minimize the occurrence of cracks, experience suggests to preliminary cut big 3D meshes in smaller ones. This is a reasonable step if one considers that the resulting patch must be folded back on the robot body part: *tailor-like* cuts can be applied to surface areas characterized by high curvatures, where the patch could be hardly folded anyway. An example of preliminary purposive cuts is given in Fig. 17, where the *upper forearm* is divided into two parts (i.e., *forearm* and *shoulder*, with a bending radius of about 5 cm, roughly the limiting bending radius of the considered technology) and then the unfolding process is separately performed.

Going back to the overall procedure, Fig. 15-2 depicts the surface extraction procedure. The surface (in the left hand side) is extracted from the 3D model (in the right-hand side). This is necessary because the unfolding process strictly operates on discrete 2D manifolds (although in 3D space). Although a tedious process, this step allows to discard those elements in the CAD model that are irrelevant for the coverage problem. In Fig. 15-3, the extracted mesh is depicted. Finally, Fig. 15-4 represents the 3D as well as the corresponding 2D surface (in dark green). Remaining small cracks can be eliminated by hand without affecting the overall polygon shape. It is straightforward to extract the polygon boundary, to be further used by the placement step.

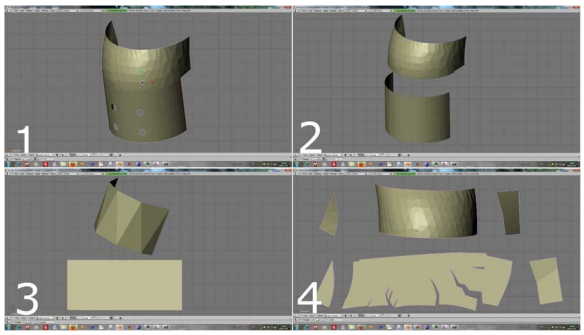

Fig. 17. Preliminary cuts can be applied: 1) whole upper arm; 2) arm separated from shoulder; 3) the flattened arm; 4) the flattened shoulder.

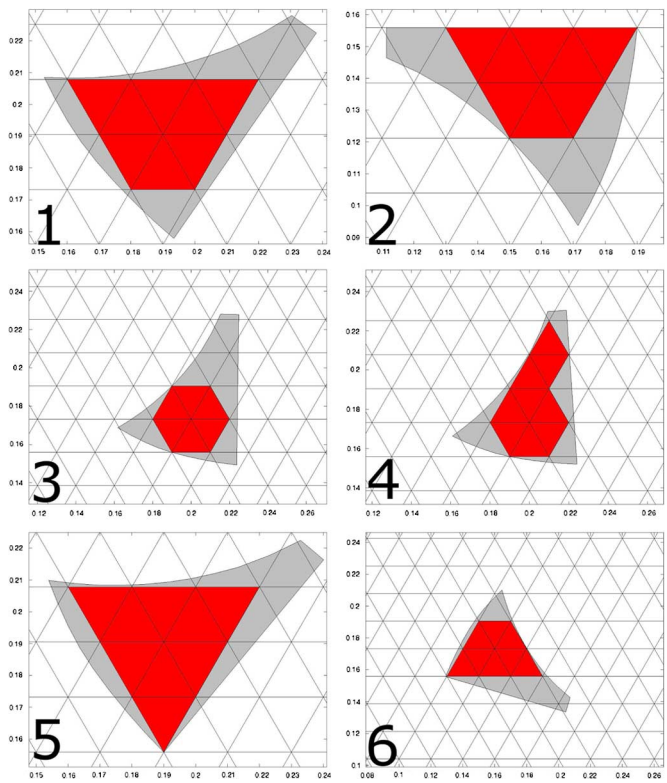

Fig. 18. Placements obtained through heuristics R_1 to R_7 on LH. R_2, and R_5 lead to the same placement solution (also for other robot parts).

B. Step 2: Placement

The heuristics discussed in Section V have been applied to all the *iCub* robot covers. Specifically, each cover has undergone an unfolding procedure (as discussed in the previous section) and then the related polygon P has been placed on the isometric grid G according to the envisaged heuristics.

Fig. 18 shows qualitative results of the seven heuristics when applied to the *left hip* cover (since heuristics R_2 and R_5 produce the same results, the top right graph in Fig. 18 represents both), whereas Table I shows numerical data about few representative pieces. The table reports for each considered robot surface (i.e., LH, LUA, LT, RUF, and UT) four rows, each one reporting the value of one parameter, respectively, the *covered area* $A(\mathbb{P}^*)$ (CA), the *ratio between the uncovered area and the polygon area* $1-(A(\mathbb{P}^*)/A(P))$ (UR), the *value of the objective function* $|\mathbb{P}^*|$ (OF), and the time (t, in seconds) for each heuristic. The last

TABLE I
RESULTS OF THE PLACEMENT PROCEDURES FOR FIVE POLYGONS: LH, LUA, LT, RUF, AND UT

P		R_1	R_2	R_3	R_4	R_5	R_6	R_7
LH	CA	31.12	31.12	23.34	35.01	31.12	31.12	31.12
	UR	45%	45%	59%	39%	45%	45%	45%
	OF	8	8	6	9	8	8	8
	t	9	2	2	34	17	2	17
LUA	CA	35.01	31.12	31.12	38.9	35.01	27.23	35.01
	UR	74%	77%	77%	72%	74%	80%	74%
	OF	9	8	8	10	9	7	9
	t	172	154	159	11726	5925	108	5925
RUF	CA	132.26	143.93	143.93	147.82	143.93	132.26	143.93
	UR	52%	48%	48%	47%	48%	52%	48%
	OF	34	37	37	38	37	34	37
	t	206	301	316	42795	306	229	21256
UT	CA	517.37	536.82	536.82	552.38	536.82	536.82	548.49
	UR	31%	29%	29%	27%	29%	29%	27%
	OF	133	138	138	142	138	138	141
	t	290	359	371	41773	348	291	20939
LT	CA	894.7	898.59	894.7	902.48	898.59	898.59	898.59
	UR	15%	14%	15%	14%	14%	14%	14%
	OF	230	231	230	232	231	231	231
	t	37	9	9	157	9	8	87
avg	UR	44%	45%	46%	39%	42%	44%	42%
stdev		22%	25%	25%	22%	23%	25%	23%
avg	t	142.8	165	171.4	19297	1321	127.6	9644.8
stdev		117.91	163.68	170.27	21517.72	2578.54	129.85	10726.86

two lines of the table show the average and standard deviations for the values of the obtained ratios (R) and for the time required by the heuristics.

The *left hip* LH is a polygon where $|V_{LH}| = 20$, $A(LH) = 57\ \mathrm{cm}^2$ and the theoretical upper bound is $B_u(LH) = 14$ (given that the area of each module c is $A(c) = 3.89\ \mathrm{cm}^2$). It must be pointed out that for LH heuristics R_2 and R_5 provide not only the same numerical results (see Fig. 18-2/5), but the very same placement. As already pointed out, the first heuristic is completely random, the following three are completely deterministic, whereas the last three introduce some randomness in defining the decision variables.

Heuristic R_1 (see Fig. 18-1) adopts a Monte Carlo approach and it is used only for a comparative analysis with respect to the other methods. However, it can be noticed that it statistically tends to generate patches covering the central area of the considered body part. Although this method does not provide the best coverage in the sense defined in Section III (i.e., $|\mathbb{P}^*| = 8$), it is nonetheless useful in many practical applications, where only a *generic coverage* is required.

Heuristic R_2 (see Fig. 18-2) exploits the alignment between the polygon and the grid. The alignment can have only a local effect, since the placement algorithm does not assume the complete knowledge of the polygon shape. As a consequence, the numerical solutions are very similar to those generated by R_1 (in some cases even identical in a numerical perspective). Heuristic R_3 (see Fig. 18-3) tends to generalize R_2 by considering two different non contiguous vertexes. It turns out that, on average, coverage quality is rather poor. In this case, it is even poorer that R_1 ($|\mathbb{P}^*| = 6$ with respect to $|\mathbb{P}^*| = 8$ on LH). This strategy is further generalized in heuristic R_4 (see Fig. 18-4). As it can be noticed, the obtained placement slightly differs from that obtained by R_3. However, the number of entirely contained modules is greatly increased. This reflects the fact that different axes

between polygon vertexes are considered, thereby mitigating its lack of symmetry. Heuristic R_5 (see Fig. 18-2) is aimed at extending R_2 by removing the constraint that the chosen vertex must be part of the polygon. However, as already pointed out, it is often the case that generated placements are similar to those obtained by R_2, which confirms the intuition that R_2 locally maximizes the chance of including more modules. A similar mechanism is implemented in R_6, which is parameterized in the orientation. When dealing with asymmetric pieces (like the *left hip*), this heuristic can give better results than others (e.g., heuristic R_3), as shown by Fig. 18-5. Finally, heuristic R_7 (see Fig. 18-6) is aimed at considering both these aspects into account, allowing random parameters both for displacement and orientation and obtaining a tradeoff between the two.

Considering again Table I, it is possible to evaluate the different heuristics. The main result is that R_4 outperforms all the other heuristics in terms of the desired parameter, i.e., coverage, followed by heuristic R_7, independently from the actual polygon size. As far as computational time is concerned, it is possible to observe from Table I that heuristic R_4 needs significantly more time than other heuristics, often for a limited increase of the amount of area that is effectively covered with robot skin. However, we argue that heuristic R_4 is still preferable on average given the offline nature of all the customization procedure.

VII. Conclusion

This paper discusses a number of issues related to the automated design and deployment of artificial skin for humanoid robots. On the one hand, the availability of large-scale patches of tactile sensors allows for an unprecedented possibility in extending robot cognitive capabilities. On the other hand, different robots need different skin layouts (because of their physical differences), thereby introducing the problem of skin *customization*.

As a matter of fact, a completely custom solution proves to be very expensive as manufacturing and setup costs are concerned. Automated techniques are needed at different levels. Procedures to optimally cover the required body parts as well as to define how tactile data can be conveyed for either decentralized or centralized processing are of the utmost importance. With respect to a specific robot skin technology, the paper is aimed at providing (still incomplete) solutions in this direction. On the one hand, the techniques that have been developed can be generalized to different skin layouts, subject to their modular and scalable nature, which are anyway desirable features for large-scale robot skin systems. On the other hand, the current process is still limited in different respects: (i) the flattening step introduces distortions or cracks in the 2D representation, which must be manually corrected and can lead to bad coverage results; (ii) being based on heuristics, the result of the placement step is still suboptimal and manual refinements may be occasionally needed for some body part shapes; (iii) among all the possible optimality criteria, we decided only to maximize the amount of body surface covered with robot skin, whereas a tradeoff could be set between this and other criteria, such as the skin weight, the likelihood of contacts in a given location, etc; and (iv) the current procedure is only suitable to be applied to robot skin designs based on basic modular components (e.g., [5], [34], [35] or triangular components in our case [9], [14]), whereas other approaches (e.g., [52], [53]) would require a completely different customization process.

The developed procedure has been validated using a number of body parts of the iCub humanoid robot. Although not exhaustive, the selected robot body parts constitute a comprehensive example of typical real-world robot surfaces. As a consequence, it is possible to argue that the presented results (in terms of covered surface areas and computational time) can be taken as a representative example of real-world cases. In particular, the results show that it is possible to automate two of the different steps usually involved in designing and deploying robot skin to a great extent. However, there are specific processing stages that still require *smarter* solutions to be defined as completely automated. These topics are subject to current investigation.

Appendix A
Heuristic R_4

As previously described, each proposed heuristic is modeled as an algorithm $R_i(G, P)$, $i = 1, \ldots, 7$, producing a set of placements Ξ_{R_i}. As discussed in Section VI, heuristic R_4 outperforms any other envisaged heuristic in both realistic and stress cases.

Algorithm 3: Heuristic R_4

Require: a grid G, a polygon P

Ensure: Ξ_{R_4}

$l \in G$

$n = (x_n, y_n) \in l$

$\Xi_{R_2} = \emptyset$

for all (i, j) s.t. $v_i, v_j \in V_P, i < j$ **do**

  $e_{i,j} = (v_i, v_j)$

  for $k = 1, \ldots, |V_P|$ **do**

    $(x_k, y_k) \leftarrow (x_n - x_{v_k}, y_n - y_{v_k})$

    $\theta_{i,j} \leftarrow \widehat{le_{i,j}}$

    $\rho_{i,j,k}(P) = (x_k, y_k, \theta_{i,j}, v_i)$

    $\Xi_{R_4} \leftarrow \Xi_{R_4} \cup \rho_{i,j,k}(P)$

  end for

end for

return Ξ_{R_4}

Heuristic R_4 is a strategy where, given a generating line $l \in G$ and a grid node $n = (x_n, y_n) \in l$, for each pair of polygon vertexes $v_i, v_j \in V_P$ such that $i < j$, are generated $|V_P|$ placements. Therefore, the total number of generated placements is

$$|V_P| \times \frac{(|V_P| - 1)}{2}.$$

Each pair (v_i, v_j) define a segment $e_{i,j}$ connecting the two vertexes (line 5). Each placement assumes the form $\rho_{i,j,k}(P) = (x_k, y_k, \theta_{i,j,k}, v_i)$ (line 9), where $x_k = x_n - x_{v_k}$ (respectively, $y_k = y_n - y_{v_k}$) is the difference between the x (respectively, y) coordinate of the chosen grid node n and the specific polygon vertex v_k (line 7), and $\theta_{i,j} = \widehat{le_{i,j}}$ is the angle necessary to rotate $e_{i,j}$ in order to lie on l. Finally, the iteration over all the vertexes v_k (line 6) allows to select different pivots for such a rotation.

REFERENCES

[1] M. Lee and H. Nicholls, "Tactile sensing for mechatronics: A state of the art survey," *Mechatronics*, vol. 9, no. 1, pp. 1–31, Feb. 1999.

[2] R. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing: From humans to humanoids," *IEEE Trans. Robotics*, vol. 26, no. 1, pp. 1–20, Feb. 2010.

[3] B. Argall and A. Billard, "A survey of tactile human-robot interactions," *Robot. Autom. Syst.*, vol. 58, no. 10, pp. 1159–1176, Oct. 2010.

[4] G. Cannata, S. Denei, and F. Mastrogiovanni, "Towards automated self-calibration of robot skin," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'10)*, Anchorage, AK, USA, May 2010.

[5] P. Mittendorfer and G. Cheng, "3d surface reconstruction for robotic body parts with artificial skins," in *Proc. IEEE-RSJ Int. Conf. Intell. Robot. Syst. (IROS'12)*, Vilamoura, Portugal, 2012.

[6] G. Cannata, S. Denei, and F. Mastrogiovanni, "Tactile sensing: Steps to artificial somatosensory maps," in *Proc. IEEE Int. Symp. Robot Human Interactive Commun. (RO-MAN 20\\\'10)*, Viareggio, Italy, Sep. 2010.

[7] E. Baglini, G. Cannata, and F. Mastrogiovanni, "Design of an embedded networking infrastructure for whole-body tactile sensing in humanoid robots," in *Proc. IEEE Conf. Humanoid Robot. (HUMANOIDS'10)*, Nashville, TN, USA, Dec. 2010.

[8] I. Kiyoto, S. Makoto, and O. Kenichi, "Scalable robotic-hand control system based on a hierarchical multi-processor architecture adopting a large number of tactile sensors," in *Proc. IEEE-RSJ Int. Conf. Intell. Robot. Syst. (IROS.12)*, Vilamoura, Portugal, 2012.

[9] A. Schmitz, P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta, "Methods and technologies for the implementation of large-scale robot tactile sensors," *IEEE Trans. Robotics*, vol. 27, no. 3, pp. 389–400, 2011.

[10] A. Nagakubo, H. Alirezahei, and Y. Kuniyoshi, "A deformable and deformation sensitive tactile distribution sensor," in *Proc. IEEE Int. Conf. Robot. Biomimet. (ROBIO'07)*, Sanya, China, Dec. 2007.

[11] H. Shinoda and E. So, "Hybrid resistive tactile sensing," *IEEE Trans. System, Man, Cybern.—Part B*, vol. 32, no. 1, pp. 57–64, Feb. 2002.

[12] D. Anghinolfi, G. Cannata, F. Mastrogiovanni, C. Nattero, and M. Paolucci, "A heuristic approach for the optimal wiring in large scale robotic skin design," *Comput. Oper. Res.*, vol. 39, no. 11, pp. 2715–2724, 2012.

[13] D. Goger and H. Worn, "A highly versatile and robust tactile sensing system," in *Proc. IEEE Conf. Sens. (SENSORS'07)*, Atlanta, GA, USA, Oct. 2007.

[14] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An embedded artificial skin for humanoid robots," in *Proc. IEEE Int. Conf. Multi-Sensor Fusion and Integr. (MFI'08)*, Seoul, Korea, Aug. 2008.

[15] D. Pirolo and E. Kolesar, "Piezoelectric polymer tactile sensor arrays for Robotics," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON'89)*, Dayton, OH, USA, May 1989.

[16] R. Reston and E. Kolesar, "Robotic tactile sensor array fabricated from a piezoelectric polyvinylidene fluoride film," in *Proc. 1990 IEEE Nat. Aerosp. Electro. Conf. (NAECON'90)*, Dayton, OH, USA, May 1990.

[17] C. Dyson, N. Yauilla, and E. Kolesar, "Object imaging with a piezoelectric robotic tactile sensor," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON'93)*, Dayton, OH, USA, May 1993.

[18] E. Cheung and V. Lumelsky, "Proximity sensing in robot manipulator motion planning: System and implementation issues," *IEEE Trans. Robot. Autom.*, vol. 5, no. 6, pp. 740–752, Dec. 1989.

[19] E. Cheung and V. Lumelsky, "Development of sensitive skin for a 3D robot arm operating in an unknown environment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'89)*, Scottdale, AZ, USA, May 1989.

[20] E. Cheung and V. Lumelsky, "A sensitive skin system for motion control of robot arm manipulators," *Robot. Autom. Syst.*, vol. 10, no. 1, pp. 9–32, Jan. 1992.

[21] A. Cameron, "Optimal tactile sensor placement," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'89)*, Scottdale, AZ, USA, May 1989.

[22] D. Johnston, P. Zhang, J. Hollerbach, and S. Jacobsen, "A full tactile sensing suite for dextrous robot hands and use in contact force control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'96)*, Minneapolis, MI, USA, May 1996.

[23] M. Inaba, Y. Hoshino, K. Nagasaka, T. Ninomiya, S. Kagami, and H. Inoue, "A full-body tactile sensor suit using electrically conductive fabric and strings," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'96)*, Osaka, Japan, Nov. 1996.

[24] D. Um, B. Stankovic, K. Giles, T. Hammond, and V. Lumelsky, "A modularized sensitive skin for motion planning in uncertain environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'98)*, Leuven, Belgium, May 1998.

[25] D. Um and V. Lumelsky, "Fault tolerance via component redundancy for a modularized sensitive skin," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'99)*, Detroit, MI, USA, May 1999.

[26] D. Um and V. Lumelsky, "Fault tolerance via analytic redundancy for a modularized sensitive skin," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'99)*, Kyonju, South Korea, Oct. 1999.

[27] H. Iwata, H. Hoshino, T. Morita, and S. Sugano, "Force detectable surface covers for humanoid robots," in *Proc. IEEE/ASMA Int. Conf. Adv. Intell. Mechantronics (AIM'01)*, Como, Italy, Jul. 2001.

[28] Y. Ohmura, Y. Kunyoshi, and A. Nagakubo, "Conformable and scalable tactile sensor skin for curved surfaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'06)*, Orlando, FL, USA, May 2006.

[29] Y. Ohmura and Y. Kunyoshi, "Humanoid robot which can lift a 30 kg box by whole body contact and tactile feedback," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'07)*, San Diego, CA, USA, Oct. 2007.

[30] W. D. Stiehl and C. Breazeal, "A sensitive skin for robotic companions featuring temperature, force and electric fields sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'06)*, Beijing, China, Oct. 2006.

[31] D. Goger, N. Gorges, and H. Worn, "Tactile sensing for an anthropomorphic robot hand: Hardware and signal processing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'09)*, Kobe, Japan, May 2009.

[32] T. Yoshikai, H. Fukushima, M. Hayashi, and M. Inaba, "Development of soft stretchable knit sensor for humanoids whole-body tactile sensitivity," in *Proc. IEEE Conf. Humanoid Robot. (HUMANOIDS'09)*, Paris, France, Dec. 2009.

[33] M. Shimojo, T. Araki, A. Ming, and M. Ishikawa, "A high speed mesh of tactile sensors fitting arbitrary surfaces," *IEEE Sensors J.*, vol. 10, no. 4, pp. 822–831, Apr. 2010.

[34] P. Mittendorfer and G. Cheng, "Humanoid multimodal tactile-sensing modules," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 401–410, 2011.

[35] A. Buchan, J. Bachrach, and R. Fearing, "Towards a minimal architecture for a printable, modular, and robust sensing skin," in *Proc. IEEE-RSJ Int. Conf. Intell. Robot. Syst. (IROS'12)*, Vilamoura, Portugal, 2012.

[36] A. Schmitz, M. Maggiali, L. Natale, B. Bonino, and G. Metta, "A tactile sensor for the fingertips of the humanoid robot icub," in *Proc. IEEE-RSJ Int. Conf. Intell. Robot. Syst. (IROS'10)*, Taipei, Taiwan, 2010.

[37] T.-H.-L. Le, P. Maiolino, F. Mastrogiovanni, G. Cannata, and A. Schmitz, "A toolbox for supporting the design of large-scale capacitive tactile systems," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (HUMANOIDS 2011)*, Bled, Slovenia, 2011.

[38] E. Demaine and J. O'Rourke, *Geometric Folding Algorithms*. Cambridge, MA, USA: Cambridge Univ. Press, 2007.

[39] Official Blender Website. [Online]. Available: http://www.blender.org

[40] C. Nattero, D. Anghinolfi, M. Paolucci, G. Cannata, and F. Mastrogiovanni, "Experimental analysis of different pheromone structures in ant colony optimization for robotic skin design," in *Proc. Int. Fed. Conf. Comput. Sci. Inform. Syst. (FedCSIS'12)*, Wrocland, Poland, 2012.

[41] M. Gardner, *Knotted Doughnuts and Other Mathematical Entertainments*. New York, NY, USA: W. H. Freeman, 1986.

[42] M. D. G. Barequet and Y. Scharf, "Covering points with a polygon," *Computational Geometry: Theory and Applications*, vol. 39, no. 3, pp. 143–162, 2008.

[43] M. Dickerson and D. Scharstein, "Optimal placement of convex polygons to maximize point containment," in *Proc. 7th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA'96)*, Atlanta, GA, USA, Jan. 1996.

[44] B. Chazelle, "The polygon containment problem," in *Advances in Computing Research*, F. Preparata, Ed. Greenwick, CT, USA: JAI Press, 1983.

[45] D. Lessard, "Optimal polygon placement on a grid," M.S. thesis, Ottawa-Carleton Institute for Computer Science, Ottawa, ON, Canada, 2000.
[46] M. Shamos, "Geometric complexity," in *Proc. 7nth Annu. ACM Symp. Theory Comput. (STOC'75)*, New York, NY, USA.
[47] J. Bentley and T. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Computers*, vol. 100, no. 28, pp. 643–647, 1979.
[48] F. Avnaim and J. Boissonnat, "Simultaneous containment of several polygons," in *Proc. 3rd Annu. Symp. Comput. Geometry (SGC'87)*, Waterloo, ON, Canada, Jun. 1987.
[49] Official CGAL Website. [Online]. Available: http://www.cgal.org
[50] Official iCub Website. [Online]. Available: http://www.icub.org
[51] M. Floater and K. Hormann, "Surface parameterization: A tutorial and survey," in *Advances in Multiresolution for Geometric Modelling*, ser. Mathematics and Visualization, N. Dodgson, M. Floater, and M. Sabin, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 157–186.
[52] D. Tawil, D. Rye, and M. Velonaki, "Improved image reconstruction for an EIT-based sensitive skin with multiple internal electrodes," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 425–435, 2011.
[53] D. Tawil, D. Rye, and M. Velonaki, "Touch modality interpretation for an EIT-based sensitive skin," in *Proc. 2011 IEEE Int. Conf. Robot. Autom. (ICRA'11)*, Shanghai, China, 2011.

Davide Anghinolfi is a Postdoctoral Researcher at the University of Genoa, Genoa, Italy. He deals with combinatorial optimization problems and metaheuristic approaches that solve them. The main applications faced are scheduling in a manufacturing environment, assignment of containers to the locations in a ship, intermodal transport, and optimization of a distributed supply chain.

Giorgio Cannata was born in Genoa, Italy, in 1963. He received the Laurea degree in electronic engineering from the University of Genoa, Genoa, Italy, in 1988.

In 1988, he joined the Naval Automation Institute, Italian National Research Council, as a Researcher, where he worked in the area of underwater robotics. From 1995 to 1998, he was with the Department of Communication, Information and System Sciences (DIST), University of Genoa, and where he was an Assistant Professor and became Associate Professor in 1998. His current research interests include humanoid robotics, tactile sensor technologies, bio-inspired robotics, automatic control systems and control architectures for robotic and mechatronic systems, robot manipulation, and robot control theory.

Fulvio Mastrogiovanni received the Degree in computer science engineering (Hons.) and the Ph.D. degree in robotics from the University of Genoa, Genoa, Italy, in 2003 and 2008, respectively.

He was a Visiting Professor at the Asian Institute of Technology, Thailand (2010), Jiao Tong University, China (2012), and the Karlsruhe Institute of Technology, Germany (2013). He is an Assistant Professor of Robot Architectures at the University of Genoa. He is coauthor of more than 70 peer-reviewed publications in international journals or conferences, and he is co-editor of one book. His main interests include perception and cognitive representation processes, reasoning, sensory-motor strategies, human behavior understanding, and human-robot interaction.

Prof. Mastrogiovanni received the Best Paper Award at DARS 2008, IEEE RO-MAN 2010, and the IFSA Award in 2013. He served as Automation Information Co-Chair for IEEE CASE 2012, EU Program Chair for IEEE RO-MAN 2013, and Program Chair for URAI 2013.

Cristiano Nattero was born in Genoa, Italy, in 1978. He received the M.Sc. degree in computer science and the Ph.D. degree in mathematical engineering and simulation in 2013, both from the University of Genoa.

His interests are in hybrid techniques for combinatorial optimization problems and applications, especially to routing and scheduling. He is also interested in sustainability issues.

Dr. Nattero received the 2012 International Fuzzy System Association (IFSA) Award for Young Scientist in Artificial Intelligence.

Massimo Paolucci received the Ph.D. degree in electronic and computer science from the University of Genoa, Genoa, Italy, in 1990.

He is an Assistant Professor with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS), University of Genoa. He teaches methods and models for decision support in the degree courses in computer engineering and management engineering. His research activities are focused on the following topics: metaheuristic and matheuristic algorithms for combinatorial optimization problems, planning and scheduling, decision support systems and multicriteria methods. Reference fields of application are logistics, particularly intermodal logistics and shipping, and manufacturing. Moreover, he also dealt with research activities related to multiagent systems, information systems, and databases.

Prof. Paolucci is a member of the Italian Excellence Center on Integrated Logistics (CIELI), and the Italian Inter University Center for Operations Research (CIRO).