The robot skin placement problem: a new technique to place triangular modules inside polygons

Fulvio Mastrogiovanni · Xuenan Guo

Received: 30 January 2018 / Accepted: 12 September 2018 / Published online: 13 October 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract
Providing robots with large-scale robot skin is a challenging goal, especially when considering surfaces characterised by different shapes and curvatures. The problem originates from technological advances in tactile sensing, and in particular from two requirements: (i) covering the largest possible area of a robot’s surface with tactile sensors to be able to detect accidental or purposive contacts, and (ii) doing it using cheap, replicable hardware modules to keep production and manufacturing costs low. Given modules of a specific shape, the problem of optimally placing them requires to maximise the number of modules that can be fixed on the selected robot body part. Differently from previous approaches, which are based on methods inspired by computational geometry (e.g. packing), we propose a novel layout design method inspired by physical insights, referred to as Iterative Placement (ItPla), which arranges modules as if physical forces acted on them. A number of case studies from the literature are considered to evaluate the proposed approach.

Keywords Tactile sensing · Robot skin · Optimal placement

1 Introduction
In the past few years, human–robot interaction (HRI) emerged as a major research field, as well as a design paradigm allowing for novel robot behaviours, and considering aspects related to perception, reasoning and action [13,16]. Two research strands can be identified. On the one hand, cognitive HRI aims at understanding how human and robot behaviours can be integrated to obtain advanced forms of assistance or cooperation, e.g. in case of household robots [12,22]. On the other hand, physical HRI investigates how humans and robots can exploit haptic information to carry out a set of tasks where physical contact is essential, e.g. for robot co-workers [5,7].

In physical HRI, special attention is devoted to the use of large-scale robot skin to provide robots with tactile information originating from their whole body. The sense of touch is expected to play a fundamental role and to act as a key enabling technology to implement novel robot behaviours, especially in humanoid robots employed in service scenarios [9,10,14].

Although different technological solutions to deploy large-scale robot skin have been presented (please refer to [11] and the references therein for a comprehensive analysis of the literature), only a few approaches consider general-purpose methods adaptable to different robots and robots’ surfaces. Modular robot skin designs are a good trade-off between cost and quality, specifically in terms of covered robot’s surface [23,26], adaptability of models for data processing [14,17,24], as well as hardware and software infrastructure [28,29]. Among these solutions, ROBOSKIN [20,26] proves to be adaptable to different robot mechanical designs and shapes [6,19,21]. ROBOSKIN modules are triangles of equal side, connected together to form an integrated mesh with just a limited number of entry points. Other technologies exist, which are designed on different bases, such as stretchable electronics [1,8,18,19]. Such approaches typically assume designs based on specific robot body parts, which is out of the scope for the work presented here.

In this paper, we consider the problem of optimally placing robot skin modules on a given robot’s surface, a problem defined in [3] and further developed in [31,32]. We refer to the ROBOSKIN technology and, as a consequence, we con-
sider triangular modules (Fig. 1). However, the technique we present is not specific to the triangular shape, and an extension to different shapes can be derived. In the problem we consider, optimality refers to the actual amount of robot’s surface covered by robot skin, subject to a number of mechanical and design constraints characterising it. Covering as much of a robot’s surface as possible is an important prerequisite to allow for forms of physical human–robot collaboration, taking into account both safety issues and purposive, contact-based robot control [14]. Enforcing a widespread, homogeneous, dense and well-distributed coverage of tactile sensors on a robot body can make robot motions robust with respect to unwanted contacts, as well as more capable of keeping contact with external objects or humans. As a consequence, we identify optimality as the maximisation of the obtained coverage. The typical mathematical formulation induces a NP-HARD class problem, for which a closed-form solution in polynomial time subject to certain conditions has been derived in [32].

Apart from related problems, i.e. point placement [15] and microchip placement [25], only a few techniques have been proposed to address the problem we consider in this paper. Anghinolfi and colleagues first define the robot skin placement problem and solve it using a set of heuristics, which are based on placing a polygon (representing the border of the robot’s surface to cover, which has been appropriately flattened) over an isometric grid, in order to maximise the number of grid’s cells contained by the polygon itself [3]. They show that this formulation is equivalent to fit the highest number of triangles on the robot’s surface. Although they propose seven heuristics (integrated in a multi-start approach), they argue that one seems to always outperform the others. In their approach, since the isometric grid defines the mutual position of triangular modules, the relative displacement is fixed and rigid. Furthermore, all modules in the resulting placement must be strictly located inside the polygon’s border. As a consequence, the approach by Anghinolfi and colleagues does not take into account any soft violation of the geometrical constraints. An extension to the original algorithm has been proposed by Wei et al. [31,32], who adopt a greedy strategy to improve the original results. Their algorithm repeats the polygon placement steps several times. Each time, it gets the largest connected placement part and considers it as part of the final placement, and separates the corresponding area from the polygon’s border. The relative rigid displacement between modules is relaxed, and this leads to better placement results. However, the approach by Wei and colleagues cannot tolerate constraint violations, and because of the adopted greedy strategy, a number of (possibly good) placements may not be generated.

The paper by Wang et al. [30], although not directly related to the problem we consider here, shares some similarities with our work. They tackle a triangle-based packing problem, where an area must be filled with the maximum number of triangles allowed by the area size. According to their approach, triangles enter the area one at a time, and they are positioned considering also the locations of already packed triangles. If no triangles can be added anymore, the packing process is terminated. However, there are important differences with respect to our work: (i) we allow triangles to reposition (according to certain conditions) before the final placement is reached at any time, whereas Wang and colleagues fix the position of each triangle, incrementally; (ii) we set a few rules to move triangles in order to obtain the final placement, whereas Wang and colleagues use strategies inspired by the way humans would perform the packing problem.

The main contribution of this paper is a new physics-based placement algorithm for the automated layout design of large-scale robot skin, referred to as Iterative Placement (ITPLA), which is available open source.1 Given generic robot body parts, a general-purpose process is described to optimally place robot skin modules to cover such body parts. Each module is subject to a number of pseudo-forces, which move it according to positions relative to other modules.

The paper is organised as follows. Section 2 introduces the automated robot skin placement problem, outlines the proposed algorithm, and poses a number of definitions. Section 3 details a simpler version of our problem, which is based on circular modules, to describe basic concepts. Section 4 discusses our solution to the robot skin placement problem for triangular modules. Section 5 completes the whole algorithm with more details. Case studies are discussed in Sect. 6. Conclusions follow.

2 The robot skin placement problem

2.1 Constraints and problem statement

The robot skin placement problem for triangular modules has been introduced in [3]. As previously pointed out, different aspects of the problem have been discussed also in [2,4]. The

---

1 Web: https://github.com/boy2000-007man/ITPLA.
problem originates from novel technological solutions to the development of large-scale robot skin. ROBOSKIN [20,26] is a modular robot skin technology where each module, of triangular shape, has three 3 cm sides and hosts a number of tactile elements (i.e. taxels) representing discrete sensing locations. Triangular modules are mechanically connected with each other at the midpoints of their sides. Such connections can be slightly bent (or even cut) but cannot be stretched nor twisted in an operational arrangement of modules (i.e. a patch). Therefore, triangular modules must not overlap (constraint $C_1$) nor slide ($C_2$) with respect to each other because this would twist the connectors to a breaking point. It is noteworthy that connectors host wires to allow data signals to be exchanged between any two modules.

The whole problem can be divided into three steps, namely (i) body part surface flattening, in which a 2D version of the 3D surface to cover is obtained through a proper topographic mapping in the form of a (possibly distorted) polygon $P$, (ii) optimal body part coverage, which aims at maximising the area of $P$ covered with robot skin, i.e. finding the maximum number of triangles contained in $P$, and (iii) optimal data signals routing in patches, where data buses are routed via connectors to enforce redundancy and fault tolerance [2].

As described in [3], the problem of determining the optimal coverage when fixing robot skin on robot’s body parts can be modelled as finding an appropriate placement of a polygon (representing a flattened version of the surface to cover) over an isometric grid, subject to the two constraints $C_1$ and $C_2$ outlined above, to maximise the number of triangles which can be contained in it. As discussed in [3], this proves to be equivalent to cover the original 3D surface in so far as the distortions induced by the flattening procedure are minimised. In such a model, the isometric grid represents an arrangement of triangles of equal side consistent with the ROBOSKIN layout.

In this paper, we focus only the second step outlined above, i.e. optimal body part coverage. As such we assume that: (i) the whole surface of a robot body part to cover has been already divided into a number of small regions, each one to be independently managed by our algorithm; (ii) each region has already been subject to a flattening process, which gives origin to a corresponding 2D polygon. As described in [2], surface subdivision and the flattening process do play a non-negligible role in obtaining polygons that are well-shaped and not excessively distorted. As a matter of fact, different techniques can be done to that purpose, e.g. preserving such global features of the surface as the overall area, or local characteristics such as local gradients. It is noteworthy that all of these techniques tend to obtain undistorted 2D regions, but either local distortions or cracks are typically present. However, ItPLA does not explicitly take distortion-related or other effects into account. For a comprehensive discussion, the reader is referred to [2] and the literature therein.

### 2.2 Outline of the iterative placement algorithm

In this section, we formalise the problem we address more precisely. As anticipated, optimality refers to the amount of robot’s surface that is actually covered. In Iterative Placement, body part coverage is divided in two steps: first, the algorithm places the highest possible number of triangles inside $P$ (computed based on the ratio between the polygon’s and a triangle’s area); second, it iteratively adjusts triangle’s poses to satisfy constraints $C_1$ and $C_2$.

Before outlining the overall algorithm’s behaviour, a few informal definitions must be introduced. With triangle’s pose, we refer to the location and orientation of a triangle-centred reference frame with respect to an external (i.e. absolute) reference frame. Given a polygon $P$, we refer to a placement as a configuration of any number of triangle poses within $P$. We label as acceptable a placement free from mutual overlaps between triangles, and as stable a placement where triangle poses tend to stabilise as a result of an iterative pose adjustment process aimed at satisfying constraints $C_1$ and $C_2$. Two notions of stability are used in ItPLA. A placement is absolutely stable if poses are not updated in the iterative process, whereas we refer to an approximately stable placement as a placement where each pose varies of a very small amount (i.e. the displacement is below a given threshold), which is negligible in practice. The steps of ItPLA are described as follows (Fig. 2).

1. **Computation of an upper bound to the number of possible triangles inside $P$ and initial randomly generated poses**

   Given a polygon $P$ (Fig. 3a), the upper bound is computed using the ratio between the areas of $P$ and a module. A corresponding number of triangles are placed randomly inside $P$ (Fig. 3b). It is noteworthy that, initially, triangles can overlap with each other as well as with the border of $P$. The final result of the process may vary according to the initial placement. In practice, it is possible to adopt a multi-start approach and select the best placement result.

2. **Iterative adjustment of triangle poses**

   Given the initial placement, ItPLA adjusts poses to obtain an overall acceptable placement. The algorithm proceeds by iteratively arranging triangles using pseudo-forces (i.e. associating each triangle’s reference frame with a velocity vector), therefore making triangles translate and rotate. Pseudo-forces are designed to tend to reduce the overlap between triangles and to properly align them. As a consequence, it can be noticed that solutions are not guaranteed to satisfy the two constraints $C_1$ and $C_2$.

3. **Check whether the current placement is stable**

   After a number of iterations, triangle poses will tend to balance each other and stabilise (Fig. 3c). In ideal conditions (and with a theoretically infinite number of iterations), such
a force balance leads to an absolutely stable placement. However, in order to obtain realistic solutions in a limited amount of time, we deem sufficient to consider placements in which approximate stability is reached. This is a good trade-off between placement stability, solution accuracy and processing time. To this aim, ItPla employs simulated annealing to determine whether the placement is approximately stable according to a metric related to the overlap degree of triangles.

4. **Check whether the obtained stable placement is acceptable** After a stable placement is obtained, ItPla evaluates whether it is acceptable. Acceptability mainly depends on modelled physical constraints between triangular modules. In the reference robot skin technology, constraints are related to mechanical connectors, which are located at the middle of triangular module’s sides. Furthermore, end corners of triangular modules can be cut to a certain extent to accommodate for situations in which triangles overlap with borders of \( P \). Since \( P \) originates from a 3D surface, it is subject to distortions. As a consequence, the algorithm considers the placement to be acceptable when the amount of overlap between neighbouring triangles and borders is below a given threshold.

5. **Removal of overlapping triangles** When a stable placement is not acceptable, a triangle in overlap is chosen and removed (Fig. 3d). Then, a new iterative adjustment process starts considering the remaining triangles until a final acceptable placement is obtained.

6. **Production of acceptable placements** When a stable placement is acceptable, it is shown to the user for evaluation (Fig. 3e). However, if a multi-start approach is employed, ItPla starts again from step 1, until the selected number of algorithm iterations is reached.

### 2.3 Preliminary definitions

The unfolded surface of the robot body part to cover is modelled as a *simple* and closed polygon \( P \), whose boundary \( \partial P \) is characterised by a finite set \( E_P \) of \( \phi \) edges, such that \( E_P = \{e_1, \ldots, e_\phi\} \) (Fig. 3a). A polygon is labelled as *simple* if its edges do not pairwise intersect, with the obvious exception of their vertexes. Since the goal of the robot skin placement is to cover as much of a robot’s surface as possible, it will be convenient to refer to \( A(\cdot) \) as the area of any generic planar shape. \( P \) must be covered with the highest number \( B_P^* \) of modules, such that \( B_P^* \leq B_P \), where \( B_P \) is a theoretical *upper bound* to the number of possible modules fitting in, defined as the floor of the ratio between the areas of \( P \) and a module \( m \), as follows:

\[
\frac{\partial P \ e_3 \ e_2}{e_4 \ P \ e_1}
\]

**Fig. 2** A flowchart of the ItPla algorithm

**Fig. 3** An example of the whole coverage process: a a polygon \( P \); b triangles are randomly placed in \( P \); c an invalid placement is generated; d an overlapping triangle is removed; e a valid placement is obtained
For instance, with respect to Fig. 3b, the upper bound $B_P$ is equal to 3. In this paper, we refer to the set of modules covering $P$ as $S_P = \{m_1, \ldots, m_{B_P}\}$. Given a module $m$, we refer to its pose as $p_m$. A placement $P$ of a polygon $P$ can therefore be expressed as a set of module poses, as follows:

$$P = \{p_{m_1}, \ldots, p_{m_{B_P}}\}. \tag{2}$$

The physical constraints $C_1$ and $C_2$ described above, which are imposed by the target robot skin technology, are modelled in ItPla as rules in the algorithm.

From constraint $C_1$, given a polygon $P$, we have that modules belonging to $S_P$ should not overlap with each other (Fig. 3c). However, since $P$ is a flattened planar representation of a robot body part, it is likely that $P$ is distorted with respect to the original 3D surface, either when unfolding [3] or if surface parameterisation techniques are used [14]. Therefore, results where small overlaps are present may still be acceptable in practice (Fig. 3d). In case even minor overlaps cannot be accepted, the reference technology may still allow one to cut (to a limited extent) a module’s end corner, which may solve a few issues associated with overlapping. In ItPla, we introduce an overlapping threshold $\tau_0$, which is an upper bound to the overlapping degree of any acceptable placement.

From constraint $C_2$, module layouts are hard constraints. However, for the same reasons considered for $C_1$, small violations of these constraints still may lead to acceptable solutions in practice. ItPla minimises misplacements among neighbouring modules, and uses a misplacement threshold $\tau_m$ for acceptable placements.

The two thresholds $\tau_0$ and $\tau_m$ may be tuned to allow for more or less permissive violations of the constraints. A discussion about their effects is present in Sect. 6.4.

### 3 ItPla-Circ: circular modules placement and the role of distances

If we consider the ROBOSKIN layout, we may observe that triangular modules are arranged in an hexagonal pattern. This section considers a set $S_P$ of circular modules to be arranged in an hexagonal structure in order to cover at best the polygon $P$. We focus on circular modules to discuss a few basic properties of ItPla.

It has been demonstrated, first by Gauss and then by Tóth [27], that hexagonal packing is the densest packing in the plane, as it forms a lattice with a packing density $\eta = 0.9068$. Such a packing lattice replicates a number of configurations which can be found in nature, e.g. bees’ honeycombs and the graphite structure [33]. It does make sense to understand how such properties can be adapted to our purposes.

As anticipated in the previous section, ItPla proceeds by iteratively arranging modules using a simulated pseudo-force field, which models attraction and repulsion forces among modules. The algorithm tends to produce stable placements because attraction and repulsion forces tend to equate. In our case, repulsion force is mimicked considering the level of overlap between modules.

While repulsion forces act to separate two neighbouring modules, attraction forces allow them to aggregate. In the algorithm, a repulsion force occurs when modules overlap with each other. The placement resulting from a distributed local effect of attracting and repulsing pseudo-forces is expected to resemble an hexagonal packing. In order to obtain such placements, however, a faithful simulation of attraction and repulsion forces is not necessary. Indeed, as pointed out by Eades [15], it is possible to adopt pseudo-forces, which can be used to compute velocity vectors associated with module poses each time quantum $\Delta t$, whereas real forces would induce accelerations. In our case, this distinction is important because real forces lead to dynamic equilibria, whereas we are interested in static equilibria.

Given the geometric nature of the problem, let us define the overlap and neighbour relationships between modules. In case of circular modules, the concept of pose reduces to their positions. For the overlap, a working definition is straightforward.

**Definition 1** Given two modules $m_i, m_j \in S_P$, let us consider the open subset of $m_i$ as $\varnothing m_i = m_i \setminus \partial m_i$ and $m_j as $\varnothing m_j = m_j \setminus \partial m_j$. An overlap exists between $m_i$ and $m_j$, referred to as $O(m_i, m_j)$ or $O(m_j, m_i)$, when $\varnothing m_i \cap \varnothing m_j \neq \emptyset$.

It is noteworthy that overlaps can occur also between modules and a polygon’s edges.

**Definition 2** Given a module $m_i \in S_P$ and a polygon’s edge $e_j \in E_P$, let us consider the open subset of $m_i$ as $\varnothing m_i$. An overlap exists between $m_i$ and $e_j$, referred to as $O(m_i, e_j)$ or $O(e_j, m_i)$, when $\varnothing m_i \cap e_j \neq \emptyset$.

We define $O_{m_i}$ as the set of all $m_j$ for which an $O(m_i, m_j)$ relationship holds, plus the set of all $e_j$ for which an $O(m_i, e_j)$ relationship holds. In order to quantify the level of overlap between a module $m_i$ and an edge $e_j \in E_P$, it is possible to refer to $P(\cdot)$ as the area of $m_i$ outside $P$, i.e. $A(m_i \setminus P)$. Ideally, the total overlap area $O_P$ associated with a placement $P$ can be computed as:

$$O_P = \frac{1}{2} \sum_{i=1}^{B_P} A(O(m_i, m_j)) + \sum_{i=1}^{B_P} P(O(m_i, e_j)), \tag{3}$$
which can be reduced to:

$$
O_\mathcal{P} = \sum_{i=1}^{\mathcal{E}_\mathcal{P}^*} \mathcal{A}(m_i) - \mathcal{A} \left( \bigcup_{i=1}^{\mathcal{E}_\mathcal{P}^*} m_i \cap \mathcal{P} \right).
$$

(4)

The total overlap area is used to determine whether the associated placement \(\mathcal{P}_\mathcal{P}\) is acceptable. This occurs if \(O_\mathcal{P}\) is lower than or equal to the threshold \(\tau_0\):

$$
O_\mathcal{P} \leq \tau_0.
$$

(5)

In our case, neighbour relationships are tied to the peculiar geometrical nature of the problem. A circular module can have at most 6 neighbours, which are expected to be arranged in an hexagonal structure to enforce the covered area. To this aim, we ideally partition the planar space around a given module \(m_i\) in 6 equal semi-spaces \(s_{i,k}\), with \(k = 1, \ldots, 6\), and we consider as the neighbour module in each semi-space \(s_{i,k}\) the one closest to \(m_i\). Of course, this implies the notion of distance between modules, which is introduced and discussed later. It is noteworthy that, according to this statement, if \(m_i\) is a neighbour of \(m_j\), does not necessarily hold that \(m_j\) is a neighbour of \(m_i\), i.e. the relationship is not reciprocal. An example is shown in Fig. 4. Here, \(m_1\) is a neighbour of \(m_2\), but the opposite does not hold. This is due to the fact that \(m_1\) is the only module belonging to one of the semi-spaces of \(m_2\), whereas both \(m_2\) and \(m_3\) lie in the same semi-space of \(m_1\), and \(m_3\) is closer to \(m_1\). As a consequence, because the interactions caused by pseudo-forces depend on neighbour relationships, they are not expected to be necessarily mutual.

**Definition 3** Given a module \(m_i \in \mathcal{S}_\mathcal{P}\), let us consider the open subset of \(m_i\) as \(\varphi m_i\) and its pose \(p_i\). For each module \(m_j \in \mathcal{S}_\mathcal{P} \setminus \{m_i\}\), let us consider its open subset \(\varphi m_j\) and its pose \(p_j\). For each semi-space \(s_{i,k}\) around \(m_i\), we classify each \(m_j\) as belonging to strictly one semi-space \(s_{i,k}\), i.e. \(m_j \in s_{i,k}\), depending on its pose. Then, for each \(s_{i,k}\), we define \(m_j^*\) as a neighbour of \(m_i\), and we write \(N_k(m_j^*, m_i)\), if and only if:

$$
m_j^* = \arg\min_{m_j} d(p_j, p_i),
$$

(6)

where \(d\) is an Euclidean distance function of module’s poses, i.e. their centres in this case, and \(m_j \in s_{i,k}\).

Figure 5 shows two examples of neighbour relationships. On the left, we notice that only \(N(m_2, m_3)\) holds, whereas \(N(m_1, m_3)\) and \(N(m_4, m_3)\) do not hold since \(m_1\) and \(m_4\) are farther from \(m_3\) than \(m_2\). On the right, both \(N(m_1, m_3)\) and \(N(m_2, m_3)\) hold, since \(m_1\) and \(m_2\) belong to two different semi-spaces.

We consider overlap relationships as an indicator of suboptimal placements. In fact, in the ideal case of acceptable placements, no overlap relationships are present, because all modules keep the proper distance from their neighbours. Henceforth, we assume that only neighbour modules can influence each other through pseudo-forces. This allows IntelPLA to consider only local pseudo-forces, which can be modelled using local information only, such as the current and the theoretically lowest distances between modules.

Figure 6 shows how the lowest distance between any two modules \(m_i\) and \(m_j \in s_{i,k}\) is computed, respectively, for circular and—we anticipate—for triangular modules. For circular modules, the lowest Euclidean distance is obviously independent from module’s orientations and dependent on module’s radii, whereas this is not the case for triangular modules. In order to avoid any module’s shape-specific differences when computing the distance, we adopt a normalised distance, described as follows.

**Definition 4** Given a module \(m_i \in \mathcal{S}_\mathcal{P}\) and a module \(m_j \in s_{i,k}\), we consider the corresponding poses \(p_i\) and \(p_j\). We let \(m_j\) translate (and not rotate) along the segment determined by the centres of \(p_i\) and \(p_j\). The nearest pose \(p_j^*\) such that \(m_j\) can reach \(m_i\) without overlapping determines the closest
The connection between modules must be kept until the repulsive pseudo-force exceeds a parametric amplitude. We adopt a weighting approach to obtain stable placements in the iterative process. Specifically, (9) can be rewritten as follows:

$$\phi_i^d = \frac{\sum_k w_{k,i}^d \cdot f_{k,i}^d}{\sum_k w_{k,i}^d}.$$  \hspace{1cm} (10)

where $w_{k,i}^d$ refers to the weight of $f_{k,i}^d$ on $m_i$.

In order to determine such weights, different functions are equally legitimate. Here, we adopt a simple function depending on the ratio between distances involving modules $m_i$ and $m_k$ (i.e. the neighbour in $s_{i,k}$), and in particular with respect to their poses:

$$w_{k,i}^d = \left(\frac{|p_i p_k^*|}{|p_i p_k|}\right)^n,$$  \hspace{1cm} (11)

where $n$ is a positive integer number. The weighting mechanism in (11) considers the ratio between the theoretically lowest distance $|p_i p_k^*|$ and the current one of modules $m_i$ and $m_k$. When the current and the lowest distances are the same (i.e. modules are connected), then $w_{k,i}^d = 1$. If the actual distance is smaller that the theoretically lowest one (i.e. an overlap occurs), the weighting function increases so that a repulsive pseudo-force $f_{k,i}^d$ caused by the overlapping module is weighted more in (10). Otherwise, if the distance is greater than the theoretically lowest one, which implies that the corresponding module does not violate any constraints, then the corresponding forces are weighted less. As a result, ITPLA first tends to eliminate overlaps, then maintains the established connections, and finally deals with isolated modules.

4 ITPLA-Tri: triangular modules placement and the effects of moments and translation offsets

In the previous section, we discussed the basic principles of ITPLA in the case of circular modules. In order to consider a set $S_P$ of triangular modules covering a polygon $P$, a few differences with respect to the previous case must be discussed.

- The notions of overlap between modules, total overlap and neighbouring do not change, subject to the different concept of Euclidean distance shown in Fig. 6. However, the normalised distance in (7) is used also in the case of triangular modules.
- Considering a triangular module $m_i$, the number of semi-spaces $s_{i,k}$ reduces to 3, because each module can be connected at most to three other modules.
– The pose \( p_i \) must explicitly include the module’s orientation \( \theta_i \), which is expressed with respect to an external (i.e. absolute) reference frame.
– While circular modules can be connected together without defining a specific connection location on their border, ROBOSKIN triangular modules can be only connected to each other through the midpoints of their sides. As a consequence, besides the one described in (10), two other pseudo-forces are designed to play a major role in constraints between neighbour modules. On the one hand, the pseudo-force as attractive when \( e_i \) and \( e_j \) converge to each other, and as repulsive otherwise. Given the contribution of the three semi-spaces around \( m_i \), the overall pseudo-force \( \Phi^i \) is given by:

\[
\Phi^i = \sum_k f^i_{k,i}.
\] (15)

Analogously to (10), stable placements can be enforced by introducing weights, such as:

\[
\Phi^i = \frac{\sum_k w^i_{k,i} \cdot f^i_{k,i}}{\sum k w^i_{k,i}}.
\] (16)

where \( w^i_{k,i} \) refers to the weight of \( f^i_{k,i} \) on \( m_i \). Weights can depend on the angular difference between the two modules, for example:

\[
w^i_{k,i} = 1 + \Delta \theta_i, k^2.
\] (17)

Now we define the translation offset between two neighbour triangular modules.

**Definition 6** Given two triangular modules \( m_i \) and \( m_j \) such that \( N(m_i, m_j) \), let us define \( e_i \) as the side of \( m_i \) closest to \( m_j \), and \( e_j \) as the side of \( m_j \) closest to \( m_i \), i.e. \( e_i \) and \( e_j \) face each other. Then, the angular difference \( \Delta \theta_i, j \) between \( m_i \) and \( m_j \) can be computed as:

\[
\Delta \theta_i, j = \theta_i - \theta_j + \frac{\Pi}{3} \cdot (n_{e_i} - n_{e_j}) - \frac{\Pi}{2},
\] (12)

where \( n_{e_i} \) and \( n_{e_j} \) are numbers associated with modules’ sides \( e_i \) and \( e_j \).

An example is shown in Fig. 7. In order to align two triangular modules, we can use (12) to distribute the variation in modules orientation between \( m_i \) and \( m_j \) (i.e. their moments \( \tau_i \) and \( \tau_j \)), as follows:

\[
\tau_i = \tau_j = \frac{\Delta \theta_i, j}{2},
\] (13)

where the scaling factor \( \frac{1}{2} \) is applied so that both \( m_i \) and \( m_j \) rotate of the same amount to align their closest sides \( e_i \) and \( e_j \). Of course, other scaling factors can be chosen. It is noteworthy that triangular modules are assumed to rotate around an axis which corresponds to the module’s centre of mass. Here, we assume an equal distribution of angular variations.

As in the case of circular modules, a module \( m_i \) is influenced by the pseudo-forces exerted by its neighbours. Given a module \( m_k \) in a semi-space \( s_{i,k} \), we refer to \( f^i_{k,i} \) as the pseudo-force acting on \( m_i \) related to the rotation induced by \( m_k \), as:

\[
f^i_{k,i} = \chi \cdot \frac{\Delta \theta_i, k}{2},
\] (14)

where \( \chi \) is a positive real number. In this case, we consider the pseudo-force as attractive when \( e_i \) and \( e_k \) converge to each other, and as repulsive otherwise. Given the contribution of the three semi-spaces around \( m_i \), the overall pseudo-force \( \Phi^i \) is given by:

\[
\Phi^i = \sum_k f^i_{k,i}.
\] (15)

Analogously to (10), stable placements can be enforced by introducing weights, such as:

\[
\Phi^i = \frac{\sum_k w^i_{k,i} \cdot f^i_{k,i}}{\sum k w^i_{k,i}},
\] (16)

where \( w^i_{k,i} \) refers to the weight of \( f^i_{k,i} \) on \( m_i \). Weights can depend on the angular difference between the two modules, for example:

\[
w^i_{k,i} = 1 + \Delta \theta_i, k^2.
\] (17)

Now we define the translation offset between two neighbour triangular modules.

**Definition 5** Given two triangular modules \( m_i \) and \( m_j \) such that \( N(m_i, m_j) \), let us define \( e_i \) as the side of \( m_i \) closest to \( m_j \), and \( e_j \) as the side of \( m_j \) closest to \( m_i \), i.e. \( e_i \) and \( e_j \) face each other. Then, the angular difference \( \Delta \theta_i, j \) between \( m_i \) and \( m_j \) can be computed as:

\[
\Delta \theta_i, j = \theta_i - \theta_j + \frac{\Pi}{3} \cdot (n_{e_i} - n_{e_j}) - \frac{\Pi}{2},
\] (12)
The situation is depicted in Fig. 8. Analogously to what we do in (13), we can distribute equally the offset reduction to \( m_i \) and \( m_j \), by imposing:

\[
t_i = t_j = \frac{\Delta T_{i,j}}{2}.
\]

(19)

Given a module \( m_k \) in a semi-space \( s_{i,k} \), we refer to \( f^T_{k,i} \) as the pseudo-force acting on \( m_i \) related to the translation offset induced by \( m_k \), as:

\[
f^T_{k,i} = \chi_T \cdot \frac{\Delta T_{i,k}}{2},
\]

(20)

where \( \chi_T \) is a positive real number. The pseudo-force is always attractive whether \( \Delta T \) reduces or increases. Given the contribution of the three semi-spaces around \( m_i \), the overall pseudo-force \( \Phi^T_i \) can be computed as:

\[
\Phi^T_i = \sum_k f^T_{k,i}.
\]

(21)

Also in this case we introduce weights to enforce stable placements, as follows:

\[
\Phi^T_i = \sum_k w^T_{k,i} \cdot f^T_{k,i},
\]

(22)

where:

\[
w^T_{k,i} = 1 + \Delta T_{i,k}^2.
\]

(23)

The total misplacement \( \mathcal{M}_P \) associated with a placement \( \mathcal{P}_P \) can be computed as:

\[
\mathcal{M}_P = \frac{1}{2} \sum_{i=1}^{B^P} \sum_{j=1, j \neq i}^{B^P} \mathcal{L}(\Delta T_{i,j}),
\]

(24)

where \( \mathcal{L}(\cdot) \) denotes the length of a line segment. The total misplacement is used to determine whether the associated placement \( \mathcal{P}_P \) is acceptable. This occurs if it is lower than or equal to the threshold \( \tau_m \):

\[
\mathcal{M}_P \leq \tau_m.
\]

(25)

Once these two pseudo-forces are introduced, we can define the overall pseudo-force acting on a triangular module \( m_i \) simply by summing up the contributions of distances (10), moments (16) and offsets (22), as follows:

\[
\Phi_i = \Phi^d_i + \Phi^T_i + \Phi^T_i.
\]

(26)

Figure 9 shows an example of the effects of \( \Phi_i \) on two triangular modules. It is possible to note that distances, moments and translation offset alignments contribute to the mutual position of two modules \( m_i \) and \( m_j \).

5 ITPLA: iterative placement

Having devised the main roles of pseudo-forces, we can now introduce the whole ITPLA algorithm. It is necessary to determine how the algorithm deals with the effects of the polygon’s border on modules, how stable placements are determined, and how modules in overlap (which cannot be accommodated) are removed from the placement.

5.1 The influence of the polygon’s border on modules

In this section, we describe how a polygon’s border influences the poses of the modules inside it. As we anticipated, the border constitutes a hard constraint, since no overlap is allowed with its edges. However, if the algorithm is used to cover a robot’s body part using the ROBOSKIN technology, a small portion of triangular module corners can be cut, and in practice we can allow module’s corners to overlap with polygon’s edges [3]. This is not true in general, though.

Let us define first the neighbour relationship between modules and edges.

**Definition 7** Given a module \( m_i \in \mathcal{S}_P \) and an edge \( e_j \in \mathcal{E}_P \), let us consider the open subset of \( m_i \) as \( \mathcal{S}_m \), its pose \( p_i \), and let us define \( c_j \) as the midpoint of \( e_j \). For each semi-space \( s_{i,k} \) around \( m_i \), we classify \( e_j \) as belonging to strictly one semi-space \( s_{i,k} \), such that \( e_j \in s_{i,k} \), depending on the position of \( c_j \). Then, for each \( s_{i,k} \), we define \( e^*_{j,i} \) as a neighbour of \( m_i \), and we write \( N(e^*_{j,i}, m_i) \), such as:
where $d$ is the Euclidean distance and $e_j \in s_{i,k}$.

Henceforth, we refer to $\mathbb{N}_{m_j}$ as the set of all $m_j$ for which a $N(m_j, m_j)$ relationship holds, plus the set of all $e_j$ for which a $N(e_j, m_j)$ relationship holds.

Given a polygon $P$, each edge $e_j \in \mathbb{E}_P$ contributes to the pose $p_i$ of a neighbour module $m_i$, both with an induced attraction or repulsion, as well as with a moment pseudo-force. In order to define how an edge $e_j$ attracts or repels a module $m_i$, we must define the notion of distance between them. We recur to the notion of generalised distance previously introduced in Definition 4.

**Definition 8** Given a module $m_i \in \mathbb{S}_P$ and a polygon’s edge $e_j \in \mathbb{E}_P$, we consider the pose $p_i$ and the point $c_j$ of $e_j$ where $p_i$ can be projected along the normal direction. We let $m_i$ translate (and not rotate) along the segment determined by $p_i$ and $c_j$. The nearest location $p_i^\ast$ such that $m_i$ can reach $e_j$ without overlapping determines the closest location to $e_j$. Then, the normalised distance $\overline{d}$ between $m_i$ and $e_j$ can be computed as follows:

$$\overline{d}(p_i, e_j) = \left( \frac{|p_i c_j|}{|p_i^\ast c_j|} - 1 \right) \cdot \frac{d(p_i^\ast, c_j)}{|p_i^\ast c_j|}. \quad (28)$$

This notion of distance can be visualised as a particular case of what happens in Fig. 6, where the edge $e_j$ acts as if it were a triangular module’s edge.

The attractive or repulsive pseudo-force originating from $e_j$ and acting on $m_i$ depends on the distance $\overline{d}$, as follows:

$$f_{j,i}^{P,d} = \chi_{P,d} \cdot \overline{d}(p_i, e_j), \quad (29)$$

where $\chi_{P,d}$ is a positive real number. It is noteworthy that, if the actual distance between $m_i$ and $e_j$ is lower than $|p_i^\ast c_j|$, the distance is negative (i.e. there is overlap), then $m_i$ and $e_j$ are subject to a repulsion force; otherwise, to an attractive force. The overall weighted pseudo-force $\Phi_i^{P,\tau}$ on $m_i$ due to polygon’s edges can be computed as:

$$\Phi_i^{P,\tau} = \sum_j w_{j,i}^{P,\tau} \cdot f_{j,i}^{P,\tau}, \quad (30)$$

where $w_{j,i}^{P,\tau}$ refers to the weight of $f_{j,i}^{P,\tau}$ on $m_i$, i.e. the contribution of $e_j$ related to its distance with respect to the module $m_i$.

As far as the moment pseudo-force is concerned, we can reuse the notion of angular difference of Definition 5, treating again the edge $e_j$ as if it were the edge of a triangular module.

**Algorithm 1** GenerateNextPlacement()

**Require:** a polygon $P$, a set of modules $\mathbb{S}_P$, the current placement $P^\ast_P$, the time quantum $\Delta t$

**Ensure:** a new placement $P'_P$

for all $m_i \in S_P$ do

Compute $\Phi_i^{P,d}$

end for

Compute $\Phi_i^\tau$

end for

The rotation induced by $e_j$ on $m_i$ is given by the following pseudo-force:

$$f_{j,i}^{P,\tau} = \chi_{P,\tau} \cdot \Delta \theta_{i,j}, \quad (31)$$

where, differently from (13), all the rotation is induced on $m_i$. The overall weighted pseudo-force $\Phi_i^{P,\tau}$ on $m_i$ due to polygon’s edges is computed as:

$$\Phi_i^{P,\tau} = \sum_j w_{j,i}^{P,\tau} \cdot f_{j,i}^{P,\tau}, \quad (32)$$

Analogously to what we do in (30), we introduce weights to balance the contributions of different edges. As a result, (26) can be updated as:

$$\Phi_i = \Phi_i^d + \Phi_i^\tau + \Phi_i^f + \Phi_i^{P,d} + \Phi_i^{P,\tau}, \quad (33)$$

where we can identify the contributions of modules and edges, respectively, to a module’s translation and rotation.
Algorithm 2 GenerateStablePlacements()

Require: a polygon \( P \), a set of modules \( \mathcal{S}_P \), the number of modules \( \mathcal{B}_P \), the time quantum \( \Delta t \), the desired number of stable solutions \( \tau_E \)

Ensure: the set of acceptable candidate placements \( \mathcal{E}_P \)

\[
\mathcal{E}_P \leftarrow \{\emptyset\}
\]

\[
i \leftarrow 0
\]

\[
\mathcal{P}_P \leftarrow \text{GenerateInitialPlacement}(P, \mathcal{S}_P, \mathcal{B}_P)
\]

\[
\text{while } i < \tau_E \text{ do}
\]

\[
\mathcal{P}_P \leftarrow \text{GenerateNextPlacement}(P, \mathcal{S}_P, \mathcal{B}_P, \Delta t)
\]

\[
\Delta E \leftarrow \alpha_\mathcal{O}(\mathcal{O}_P^0 - \mathcal{O}_P^i) + \alpha_\mathcal{M}(\mathcal{M}_P^0 - \mathcal{M}_P^i)
\]

\[
\text{if } e^{-\frac{\Delta E}{\tau_0}} < \text{random}(0, 1) \text{ then}
\]

\[
\mathcal{E}_P \leftarrow \mathcal{E}_P \cup \mathcal{P}_P
\]

\[
i \leftarrow i + 1
\]

\[
\text{end if}
\]

\[
\mathcal{P}_P^c \leftarrow \mathcal{P}_P
\]

\[
\text{end while}
\]

5.2 Generation of new placements

Each placement step requires computing all the pseudo-forces, and then to use (33) to translate and rotate each module (Algorithm 1). The step assumes the availability of the polygon \( P \) associated with the robot body part to cover (in particular, its edges in \( \mathcal{E}_P \)), the set of modules \( \mathcal{S}_P \) constituting the current placement \( \mathcal{P}_P^c \), and the time quantum \( \Delta t \).

For each module \( m_i \), GenerateNextPlacement() proceeds by computing all the pseudo-forces, respectively, \( \Phi_i^t \) using (10) in line 5, \( \Phi_i^r \) using (16) in line 9, \( \Phi_i^{p.r} \) using (22) in line 13, \( \Phi_i^{p.d} \) using (30) in line 17 and \( \Phi_i^{p.t} \) using (32) in line 21. Then, a resultant pseudo-force for the translation component \( \Phi_i^{tra} \) is computed in line 22, and for the rotation component \( \Phi_i^{rot} \) in line 23. Finally, two functions, namely TranslateModule() and RotateModule() are used to generate the next placement \( \mathcal{P}_P^i \) (lines 26 and 27). In these two functions, pseudo-forces (i.e. velocity vectors) are transformed in displacement by time multiplication for the time quantum \( \Delta t \).

5.3 Generation of stable placements

When a new polygon \( P \) is given, ItPLA first computes with (1) the upper bound \( \mathcal{B}_P \) to the number of modules to use. ItPLA tries to find a stable placement \( \mathcal{P}_P \), whose acceptability must be determined. If the total overlap area \( \mathcal{O}_P \) exceeds \( \tau_0 \) or the total misplacement \( \mathcal{M}_P \) exceeds \( \tau_m \), a module to remove must be chosen (see next section), and the algorithm keeps iterating on possible solutions. Determining whether \( \mathcal{P}_P \) is acceptable is a crucial step in the iteration process: on the one hand, removing a module implicitly prunes the solutions space, therefore making possibly good solutions unreachable; on the other hand, not removing a module can lead ItPLA to bias on a specific portion of the solutions space, which may not contain the best solutions.

We employ an adaptive simulated annealing approach to generate the set of acceptable candidate solutions, from which to select the best one. In our case, as described in Algorithm 1, the iterative generation of the next solution (i.e. a placement \( \mathcal{P}_P^m \)) depends on the polygon \( P \), the set of modules \( \mathcal{S}_P \) and the current solution \( \mathcal{E}_P \) through the combined effect of pseudo-forces. Differently from the standard simulated annealing procedure, where the acceptability of a solution is subject to a certain randomness degree, in our case we can tie it also to the total overlap area \( \mathcal{O}_P \) and the total misplacement \( \mathcal{M}_P \), which are related to the specific placement.

GenerateStablePlacements() (Algorithm 2) assumes a polygon \( P \), the number of modules \( \mathcal{B}_P \) used to generate the current coverage, and the time quantum \( \Delta t \), to forward to GenerateNextPlacement() (Algorithm 1), as well as the number of desired stable solutions \( \tau_E \). At the beginning, the set of solutions \( \mathcal{E}_P \) and the solution index \( i \) are initialised (lines 1 and 2). The function GenerateInitialPlacement() is called once to generate a random initial placement of modules, i.e. the current placement \( \mathcal{P}_P^c \) and the associated set of modules \( \mathcal{S}_P \) (line 3). Then, at least \( \tau_E \) iterations of lines 5–12 are executed. For each iteration, the GenerateNextPlacement() function is called, which generates a new placement \( \mathcal{P}_P^i \) (line 5). The two total overlaps (respectively, for the new and current placements) \( \mathcal{O}_P^i \) and \( \mathcal{O}_P^c \), as well as the two total misplacements \( \mathcal{M}_P^i \) and \( \mathcal{M}_P^c \) are computed (line 6). Their weighted difference \( \Delta E \) is computed (line 7). Parameters \( \alpha_O \) and \( \alpha_M \) can be chosen to weight differently area overlap and misplacements. When the total overlap and the total misplacement tend to stabilise, then \( \Delta E \) decreases, leading to a stable solution with a given probability. If this happens, the placement \( \mathcal{P}_P^i \) is added to the set of stable solutions \( \mathcal{E}_P \) and \( i \) is increased (lines 9 and 10). Otherwise, the new solution becomes the current one and a new placement for the index \( i \) is determined (line 12). It is noteworthy that the annealing effect is tied to both total area and total misplacement using a function \( F \): the lower their values, the greater the annealing effect (line 8).

5.4 Generation of acceptable placements

With the generation of acceptable placements, we fully implement the ItPLA procedure, as outlined in Fig. 2. ItPLA (Algorithm 3) assumes as input a polygon \( P \), the time quantum \( \Delta t \) for the iterative procedure and the number of stable solutions to choose from. It produces an acceptable placement \( \mathcal{P}_P^c \) that can be evaluated by a human designer.

First of all, the upper bound to the number of modules is computed using (1) in line 2. A number of steps are executed until an acceptable placement \( \mathcal{P}_P^c \) is found (lines 5–14). The
GenerateStablePlacements() function is executed to generate a set $\mathcal{E}_P$ of stable placements (line 6). When the set of stable solutions $\mathcal{E}_P$ is determined, the best candidate for acceptability $P^+_p$ can be selected (line 7). Different strategies may be involved. For instance, the solution minimising a weighted sum of total overlap $\mathcal{O}_P$ and total misplacement $\mathcal{M}_P$ can be selected, as follows:

$$P^+_p = \arg \min_{\mathcal{E}_P} \mathcal{E}_P.$$  \hspace{1cm} (34)

Obviously, other approaches can be preferred as well. These can be related to the peculiar geometry of the problem, distortions in $P$, adopted technology and common sense. It is noteworthy that the selection of thresholds $t_o$ and $t_m$ for, respectively, the total overlap and misalignment parameters depend cannot be a priori mathematically defined, but can be set by experience. As an example, Table 1 in the next section shows how different acceptable placements are produced, with a non-negligible value for $\mathcal{O}_P$.

If $P^+_P$ meets the requirements related to overlap and misplacement, then an acceptable solution is found (line 9) and the algorithm exits. Otherwise, it is necessary to remove one module from the corresponding set $\mathcal{S}_P$. To this aim, in general we induce a total order on the pairs $(|\mathcal{O}_m|, |\mathcal{N}_m|) \in \mathbb{R}^2$, where $m_i \in \mathcal{S}_P$.

**Definition 9** Given two modules $m_i$ and $m_j \in \mathcal{S}_P$, we define a lexicographic order on $\mathcal{S}_P$, as we define it as $(\mathcal{S}_P, \succeq)$, such that:

$$(|\mathcal{O}_{m_i}|, |\mathcal{N}_{m_i}|) \succeq (|\mathcal{O}_{m_j}|, |\mathcal{N}_{m_j}|)$$

if an only if:

1. $|\mathcal{N}_{m_i}| < |\mathcal{N}_{m_j}|$,
2. $|\mathcal{N}_{m_i}| = |\mathcal{N}_{m_j}|$ and $|\mathcal{O}_{m_i}| \geq |\mathcal{O}_{m_j}|$.

Since we want to remove the module $m_i^* \in S^+_P$ which has the lowest number of neighbours and an above-than-average overlap, RemoveModuleFrom() selects one such module such that (i) $m_i^* \in S_P$ and (ii) $O_{m_i}^* \geq \text{avg}(O_{m_j})$ (line 12). When the module is removed, a new set $\mathcal{S}_P'$ is obtained and the algorithm iterates.

### 6 Use cases

This section introduces first a few notes about the implementation of the algorithm, and then presents three uses cases involving different robot body parts, one from the iCub robot, and two from a Schunk manipulator.

#### 6.1 Implementation

ItPla has been implemented both in C++ and Python to broaden its users base.

The C++ version integrates the CGAL computational geometry library\textsuperscript{2} and Box2D\textsuperscript{3} to simulate polygon motions. This version also includes a graphical user interface developed using the QT libraries.\textsuperscript{4} The C++ version can be considered as a sort of mock-up: the runtime environment is not easy to set-up, it may be difficult to add new features because of poor architectural design choices, and the computation of minimum distances is just an approximation. As far as the Python version is concerned, the algorithm natively computes 2D calculations as well as physics simulations, and integrates a Python-based graphical user interface. It uses the pip package manager, and integrates such libraries as NumPy,\textsuperscript{5} Box2D and Pygame.\textsuperscript{6} With respect to the C++ version, it has a number of advantages: (i) setting up the environment is quite easy; (ii) the code is simpler and more clear; (iii) the computation of the lowest distance is accurate; (iv) the number of lines of code is around 200 versus more than 1000 of the C++ version. As anticipated in Sect. 1, both versions are available open source.

#### 6.2 Use case 1: the iCub’s left hip

This use case illustrates how ItPla behaves using the simplified version of the iCub’s left hip used in \cite{[3]}.

Figure 10 shows a few significant steps in the iteration process. First of all, given a flat representation of the hip surface as a polygon, the theoretically maximum number of
triangular modules is computed using (1). As a consequence, a certain amount of triangular modules are generated and randomly placed inside the polygon, as shown in Fig. 10a. Obviously enough, a triangle sides can actually stick out from the polygon’s area. Then, the iteration phase of the algorithm starts, and the modules tend to distribute uniformly as per effect of the attractive and repulsive forces, leading to the situation in Fig. 10b, as described in (33). In this case, a part from a few notable exceptions, all modules turn to be uniformly distributed. In Fig. 10c, it is possible to see how pseudo-forces tend to remove overlaps, although some of them seem to be unavoidable. Since a stable but unacceptable (i.e. it contains overlaps) placement is obtained, ItPla needs to remove one triangular module, which is done as shown in Fig. 10d. As discussed above, the removed module is the one with the lowest number of neighbours and the highest overlap area. Similar removal operations are performed in the situations depicted in Fig. 10f, h, k. The acceptable placement shown in Fig. 10l is finally obtained, which terminates the algorithm. It is noteworthy that, with respect to the results in [3], and in particular those obtained using the best performing heuristic, the number of triangular modules that are accommodated inside the polygon raises from 9 to 10, which amounts to a 11.11% increase in overall coverage. More details can be found in Table 1, where the row termed LH reports results for this polygon.

The trend in both the number of modules and the overlap area associated with the process described above is shown in Fig. 11. It is possible to observe that the number of modules decreases as per effect of module removals, whereas the overlap area tends to stabilise and sometimes to increase during the iterative procedure. This phenomenon is caused by the employed weighting mechanism. In ItPla, pseudo-force effects are computed taking into account the amount of overlap multiplied by a gain, i.e. the weight, which means that such functions produce gain-dependent results, i.e. their effect is uniquely defined only subject to a specific choice of gains. However, this does not affect the procedure since the overlap area is used only for evaluations, therefore having little influence on the final results.

![Fig. 10](image)

The ItPla’s evolution process for the iCub robot’s left hip

---

Table 1 Results of placement procedures for different polygons, i.e. tr1_1, tr1_2, tr1_4, tr1_5, and LH

| P      | B_P | Avg  | Min  | Max  |
|--------|-----|------|------|------|
| tr1_1  | 4   | 3.77 | 2.60 | 3.90 |
| tr1_2  | 16  | 17.41| 14.29| 19.49|
| tr1_4  | 14  | 9.22 | 7.79 | 10.39|
| tr1_5  | 22  | 16.89| 10.39| 20.78|
| LH     | 14  | 11.56| 10.39| 14.29|

---

Intelligent Service Robotics (2019) 12:27–43
6.3 Use cases 2 and 3: a Schunk manipulator’s link

The second and third use cases are related to the coverage of a Schunk manipulator’s link, as shown in Fig. 12. The whole link’s surface has been a priori divided into five different parts, each one subject to a different flattening and coverage process. As it can be seen in the figure, the five surfaces are referred to as $tr_1_1, \ldots, tr_1_5$. Here, we focus only on $tr_1_2$ and $tr_1_5$, which are characterised by challenging irregular shapes. We do not consider $tr_1_3$ because its area is too small, neither do we consider $tr_1_1$ nor $tr_1_4$ because their flat shape is similar to a rectangle, which would be trivial to cover. However, synthetic results related to all pieces except for $tr_1_3$ are present in Table 1.

Figure 13 shows a sequence of snapshots related to the coverage of $tr_1_2$. As in the previous case, the number of triangular modules which—in principle—would cover at best the polygon is computed, and the corresponding number of modules is randomly placed within the polygon (Fig. 13a). As soon as the iterative process starts, we can observe that the leftmost and the rightmost regions of the polygon become soon well-covered, whereas overlaps occur in the central region (Fig. 13c). The algorithm alternates iterations necessary to stabilise the placement, which can be shown in Fig. 13c, f, h, with events necessary to remove modules, as shown in Fig. 13d, e, i. As a comparison with the results obtained using the approach described in [3], ITPLA generates two different patches (instead of one) for a total of 15 triangular modules in the best case instead of 11 obtained using the best performing heuristic, thereby obtaining a 36.36\% coverage increase rate.

Figure 14 shows a sequence of snapshots related to the coverage of $tr_1_5$. Analogously to what happens in other cases, the iterative process alternates phases where pseudo-forces arrange triangular modules towards a stable placement, with others where one module is removed to decrease the overall overlap area. In the case of Fig. 14, we observe an increase in coverage of about 20\% (from 15 to 18 modules) in the best case generated by our algorithm with respect to the baseline solution presented in [3].

Table 1 reports results for all polygons except for $tr_1_3$, which is too small. For each polygon, ITPLA has been run 10 times with a different initial bootstrap in terms of triangular module poses. The second column reports the theoretical upper bound $B_P$, whereas the third column specifies different evaluation metrics for placements, namely the covered area $A(S_P)$, the ratio $UR$ between the uncovered area $1 - A(S_P)$ and the polygon area $A(P)$, the number of triangular modules $B^*_P$ in the resulting patch, the time to obtain a solution, and the total overlap $O_P$. We report average, minimum and maximum values (third, fourth and fifth columns). First of all, we report a significant variability in the size $B^*_P$ of the resulting placement depending on the initial random poses. For example, considering $tr_1_5$ we obtain placements ranging from 8 to 18 modules. As a trend, we can observe that the higher $B_P$ the wider this variability. Then, it is also possible to appreciate a positive correlation between $B^*_P$ and the time it takes to obtain the placement (as well as its variability). This is not surprising because the number of local pseudo-forces affecting the process linearly increases with the number of triangular modules. Finally, best cases are characterised by low values for the total overlap $O_P$, which can reach zero in certain cases. However, the average values are typically non-negligible.

6.4 Discussion

The overall algorithm behaviour discussed in the previous sections leads us to a number of considerations about both the results and the possible follow-ups.
ItPla is able to generate placements where multiple patches are present. Patches are distributed on the polygon in order to maximise the number of contained modules. Attractive pseudo-forces seem to be appropriate to guarantee an intra-patch connectivity, whereas repulsive pseudo-forces are involved in separating patches. Although in our experiments parameter $\chi_d$ in (8) has been kept constant, such an attraction or repulsion mechanism can be modulated also considering the distance between modules. Currently, no explicit threshold has been set to the influence of modules on each other, subject to the fact they are neighbours. For different patches to be generated, a non-negligible effect is played by the polygon boundaries. This can be observed in the example involving $tr1_2$. In this case, the polygon is clearly divided in two main regions: the leftmost part is narrower than the rightmost one, which causes an unsolvable overlap situation at the junction of the two regions. Two patches are also generated in the polygon associated with the iCub’s left hip, where the top left corner cannot accommodate a protrusion of the main patch. An irregular (although convex) shape like the one associated with $tr1_5$ originates approximately one patch.

The patches generated by ItPla are not constrained to share the same regular grid structure, as it is required in the approach proposed in [3]. Because of attractive pseudo-forces, triangular modules forming the same patch tend to be aligned along the principal directions of a regular isometric grid. However, because of repulsive pseudo-forces and the effect of polygon edges, different patches become aligned along different directions. Therefore, each patch is aligned with respect to a different regular isometric grid. This effect allows ItPla to be quite flexible in the process of placements generation and patches formation, and it is the main reason why it produces results better than those obtained by the baseline approach discussed in [3], above all in such cases as $tr1_2$, where polygons are characterised by strong irregular shapes. In fact, irregular shapes are typically characterised by protrusions aligned with directions requiring the coverage algorithm to try different local alignments. ItPla supports such multiple alignments, whereas other methods over constrain the overall set of patches.

As it is possible to observe from the various snapshots in Figs. 10, 13 and 14, the overall overlap area is not a monotone decreasing function of the iteration procedure.
On the contrary, as per effect of attractive and repulsive pseudo-forces, it can increase as long as the placement gets stable. This behaviour can be observed also in Fig. 11 for the specific case of the iCub’s left hip. However, the overlap is reduced from now and then when a module is removed. Module removals are of the utmost importance for the generation of stable and acceptable placements, as they drastically impact on the overlap area. Furthermore, by initialising the number of modules to the theoretical upper bound subject to the polygon’s area, we enforce (although this cannot be theoretically guaranteed) the maximum coverage.

Results exemplified in Table 1 seem to suggest that the computational performance of ItPla does not depend straightforwardly on the maximum theoretical number of modules $B_P$, whereas a positive correlation seems to be present with the number of modules in the optimal patch $B^*_P$, and the related covered area $A(S^*_P)$, on average. This is not surprising, because $B_P$ is a theoretical upper bound which does not take into account the polygon shapes, and therefore cannot explicitly cope with the presence of regions difficult to cover. The computational performance of our algorithm depends on the effective polygon’s surface that is possible to cover. From the table, it is also possible to observe a non-negligible dependency on the initial (random) placement of modules, which is reflected by the various results obtained in terms of covered area, number of used modules, and time required to obtain the final placement. As a consequence, an interesting possibility is to pose a probability prior on the initial pose assignment to enforce better placements.

Other module’s shapes are possible, subject to the fact that they are radially symmetrical at least with respect to certain orientations. For instance, circular modules are radially completely symmetrical, whereas triangular modules are radially symmetrical with respect to three equidistant radii. Since many of the ItPla’s properties are based on the abstract circular shape, from it many other shapes can be derived. One notable example would be the use of hexagonal shapes (which are radially symmetrical with respect to six radii), since a modular robot skin based on such a shape has been proposed in [17]. Obviously enough, distance information as well as the effects of pseudo-forces should be modified to reflect the new mechanical constraints associated with the employed technology. It is noteworthy that ItPla by its nature is focused in robot skin technology based on replicable basic modules. As such, it does not consider other technologies designed on different bases, such as stretchable electronics [1,8,18,19]. However, should the underlying technology support modules whose mechanical connections can be misplaced (for instance, when bending connectors are used), ItPla may support it by allowing for high thresholds in misplacements.

Although we did not discuss it in the paper, and subject to meeting the possible associated mechanical constraints, module sizes need not to be uniform, as it is instead implicitly posed by the authors of [3] with the introduction of an isometric grid. Modules with different sizes in the same placement are allowed in ItPla. Such capability may prove to be useful in covering certain large-scale robot surfaces, where large regions would be covered by big modules and small regions by modules of a reduced size.

7 Conclusions

In this paper, we present and discuss ItPla, an algorithm for the design of layouts related to the so-called robot skin placement problem. The robot skin placement problem is a central issue to provide robots with the sense of touch, and in particular to attain robots with large-scale tactile sensing capabilities, which is fundamental in a wide range of service applications.

Differently from algorithms previously discussed in the literature, which are based on purely geometrical considerations, ItPla produces placements by the combined effect of a number of pseudo-forces. Pseudo-forces locally translate and rotate modules to fit them inside the area of a polygon representing the robot’s surface to cover. The algorithm is characterised by a number of advantages when compared to previous approaches, namely: generating placements where multiple patches are present, which are not constrained to be aligned with respect to the same isometric grid, as well as accommodating for different module’s shapes and sizes. Given these features, ItPla proves to behave better than previous algorithms in the performance indicator of area coverage percentage. The algorithm is available open source.

References

1. Alirezaei H, Nagakubo A, Kuniyoshi Y (2007) A highly stretchable tactile distribution sensor for smooth surfaced humanoids. In: Proceedings of the 2007 IEEE-RAS international conference on humanoid robots (HUMANOIDS 2007), Pittsburgh, PA, USA
2. Anghinolfi D, Cannata G, Mastrogiavanni F, Nattero C, Paolucci M (2012) Heuristic approaches for the optimal wiring in large scale robotic skin design. Comput Oper Res 39(11):2715–2724
3. Anghinolfi D, Cannata G, Mastrogiavanni F, Nattero C, Paolucci M (2013) On the problem of the automated design of large-scale robot skin. IEEE Trans Autom Sci Eng 10(4):1087–1100
4. Anghinolfi D, Cannata G, Mastrogiavanni F, Nattero C, Paolucci M (2014) Application and experimental validation of pheromone design in ant colony optimisation: the problem of robot skin wiring. Appl Artif Intell 28(3):292–321
5. Argall BD, Billard A (2010) A survey of tactile human–robot interactions. Robot Auton Syst 58(10):1159–1176
6. Baglini E, Youssefi S, Mastrogiovanni F, Cannata G (2014) A real-time distributed architecture for large-scale tactile sensing. In: Proceedings of the 2014 IEEE/RSJ international conference on intelligent robots and systems (IROS 2014), Chicago, IL
7. Beetz M, Bartels G, Albu-Schäffer A, Bálint-Benczédi F, Belder R, Beißler D, Haddadin S, Maldonado A, Mansfeld N, Wiedemeyer T, Weitschat R, Worch J-H (2015) Robotic agents capable of natural and safe physical interaction with human co-workers. In: Proceedings of the 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS 2015), Hamburg, Germany
8. Bhattacharjee T, Jain A, Vaish S, Killpack MD, Kemp CC (2013) Tactile sensing over articulated joints with stretchable sensors. In: Proceedings of the 2013 world haptics conference (WHC 2013), Daejeon, South Korea
9. Cannata G, Denei S, Mastrogiovanni F (2010) Tactile sensing: steps to artificial somatosensory maps. In: Proceedings of the 2010 IEEE international symposium on robot and human interactive communication (RO-MAN 2010), Viareggio, Italy
10. Dahiya RS, Metta G, Valle M, Sandini G (2010) Tactile sensing— from humans to humanoids. IEEE Trans Robot 26(1):1–20
11. Dahiya RS, Valle M (2013) Robotic tactile sensing. Springer, Berlin
12. Darvish K, Wanderlingh F, Bruno B, Simetti E, Mastrogiovanni F, Casalino G (2018) Flexible human–robot cooperation models for assisted shop-floor tasks. Mechatronics 51:97–114
13. Dautenhahn K (2007) Socially intelligent robots: dimensions of human–robot interaction. Philos Trans R Soc B Biol Sci 362(1480):679–704
14. Denei S, Mastrogiovanni F, Cannata G (2015) Towards the creation of tactile maps for robots and their use in robot contact motion control. Robot Auton Syst 63:293–308
15. Eades P (1984) A heuristic for graph drawing. Congr Numerantium 42:149–160
16. Goodrich MA, Schultz AC (2007) Human-robot interaction: a survey. Found Trends Hum Comput Interact 1(3):203–275
17. Kaboli N, Long A, Cheng G (2015) Humanoids learn touch modalities identification via multi-modal robotic skin and robust tactile descriptors. Adv Robot 29(21):1411–1425
18. Kramer R, Majidi C, Wood RJ (2011) Wearable tactile keypad with stretchable artificial skin. In: Proceedings of the 2011 IEEE international conference on robotics and automation (ICRA 2011), Shanghai, China
19. Loi A, Basiricò L, Cosseddu P, Lai S, Barbaro M, Bonfiglio A, Maiolino P, Baglini E, Denei S, Mastrogiovanni F, Cannata G (2013) Organic bendable and stretchable field effect devices for sensing applications. IEEE Sens J 13(12):4764–4772
20. Maiolino P, Maggiali M, Cannata G, Metta G, Natale L (2013) A flexible and robust large scale capacitive tactile system for robots. IEEE Sens J 13(10):3910–3917
21. Maiolino P, Galantini F, Mastrogiovanni F, Gallone G, Cannata G, Carpi F (2015) Soft dielectrics for capacitive sensing in robot skins: performance of different elastomer types. Sens Actuators A Phys 226:37–47
22. Matarić M (2006) Socially assistive robotics. IEEE Intell Syst 5(3–4):81–83
23. Mittendorfer P, Yoshida E, Cheng G (2015) Realising whole-body tactile interactions with a self-organising, multi-modal artificial skin on a humanoid robot. Adv Robot 29(1):51–67
24. Muscari L, Seminara L, Mastrogiovanni F, Valle M, Capurro M, Cannata G (2013) Real-time reconstruction of contact shapes for large-area robot skin. In: Proceedings of the 2013 IEEE international conference on robotics and automation (ICRA 2013), Karlsruhe, Germany
25. Quin N, Breuer MA (1979) A force directed component placement procedure for printed circuit boards. IEEE Trans Circuit Syst 26(6):377–388
26. Schmitz A, Maiolino P, Maggiali M, Natale L, Cannata G, Metta G (2011) Methods and technologies for the implementation of large-scale robot tactile sensors. IEEE Trans Robot 27(2):297–312
27. Töth GF (1972) Lagerungen in der Ebene, auf der Kugel und in Raum. Springer, Berlin
28. Youssefi S, Denei S, Mastrogiovanni F, Cannata G (2015) A real-time data acquisition and processing framework for large-scale robot skin. Robot Auton Syst 68:86–103
29. Youssefi S, Denei S, Mastrogiovanni F, Cannata G (2015) Skinware 2.0: a real-time middleware for robot skin. SoftwareX 3:6–12
30. Wang R, Qi X, Luo Y, Dong J (2014) A new quasi-human algorithm for solving the packing problem of unit equilateral triangles. Abstr Appl Anal 2014(308474):1–8
31. Wei X, Joneja A, Tang K (2015) An improved algorithm for the automated design of large-scale robot skin. IEEE Trans Autom Sci Eng 12(1):372–377
32. Wei X, Zhao B, Joneja A, Xi J (2017) On the polygon containment problem on an isometric grid. IEEE Trans Autom Sci Eng 14(2):1075–1083
33. Williams R (1979) The geometrical foundation of natural structure: a source book of design. Dover Publications Inc, New York

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.