Computational Object-Wrapping Rope Nets

JIAN LIU and SHIQING XIN, Shandong University
XIFENG GAO, Florida State University and Tencent America
KAIHANG GAO, Shandong University
KAI XU, National University of Defense Technology
BAOQUAN CHEN, Peking University
CHANGHE TU, Shandong University

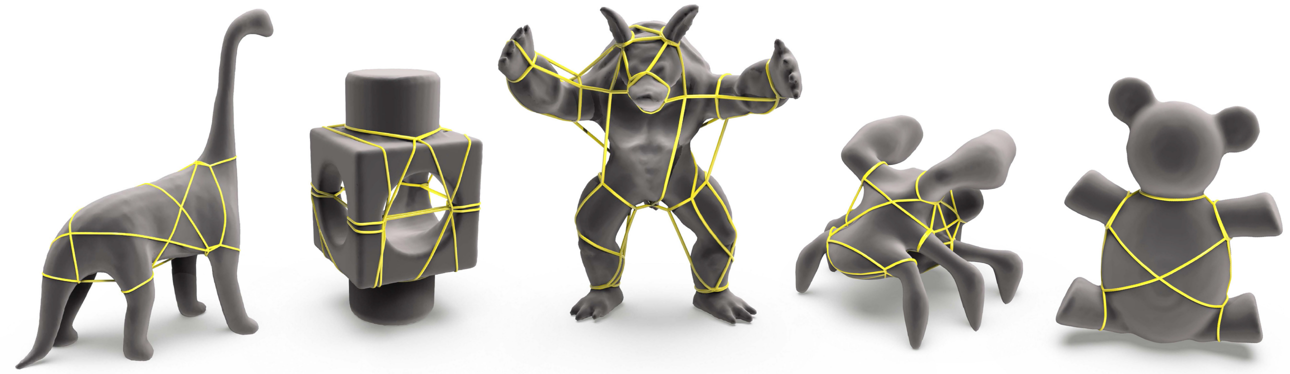


Fig. 1. Example object-wrapping rope nets generated from input 3D surfaces using our fully automatic pipeline.

Wrapping objects using ropes is a common practice in our daily life. However, it is difficult to design and tie ropes on a 3D object with complex topology and geometry features while ensuring wrapping security and easy operation. In this article, we propose to compute a rope net that can tightly wrap around various 3D shapes. Our computed rope net not only immobilizes the object but also maintains the load balance during lifting. Based on the key observation that if every knot of the net has four adjacent curve edges, then only a single rope is needed to construct the entire net. We reformulate the rope net computation problem into a constrained curve network optimization. We propose a discrete-continuous optimization approach, where the topological constraints are satisfied in the discrete phase and the geometrical goals are achieved in the continuous stage. We also develop a hoist planning to pick anchor points so that the rope net equally distributes the load during hoisting. Furthermore, we simulate the wrapping process and use it to guide the physical rope net construction process. We demonstrate the effectiveness of our method on 3D objects with varying geometric and topological complexity. In addition, we conduct physical experiments to demonstrate the practicability of our method.

CCS Concepts: • **Computing methodologies → Shape analysis;**

Additional Key Words and Phrases: Euler tour theorem, rope net, hoisting

ACM Reference format:
Jian Liu, Shiqing Xin, Xifeng Gao, Kaihang Gao, Kai Xu, Baoquan Chen, and Changhe Tu. 2021. Computational Object-Wrapping Rope Nets. *ACM Trans. Graph.* 41, 1, Article 6 (August 2021), 16 pages.
https://doi.org/10.1145/3476829

1 INTRODUCTION

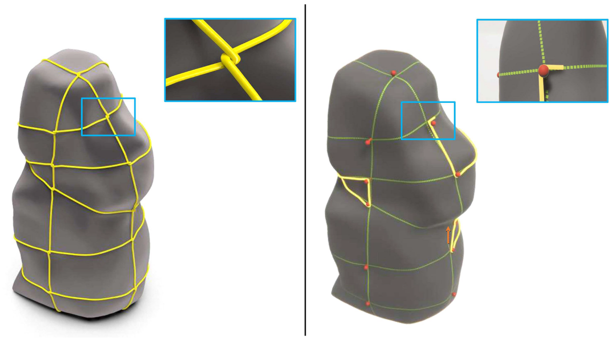
Wrapping objects with rope nets finds many applications such as packing, hoisting, and transportation, among others. For example, when hoisting and carrying a sculpture, tying it up with ropes and then lifting up the ropes is historically a common practice and remains to be an economic, safe, and widely adopted solution today [Fu et al. 2017; Nets4You 2019; Sageman-Furnas et al. 2019; US Netting 2019; Wan et al. 2020]. Figure 2 shows a few real-world examples of tightly wrapped rope nets. However, planning and tying up such object-wrapping rope nets, with the requirements of tightness, load balance, and simplicity, is by no means an easy exercise. It heavily relies on human experience and can quickly frustrate a novice practitioner.

In this work, we study the problem of object-wrapping rope nets from both geometric and physical modeling points of views. We propose a computational method for designing rope nets

This work was supported in part by NSFC (61772318, 61772016, 906 61532003, 62002376) and National Key Research and Development 907 Program of China (2018AAA0102200).
Authors. addresses: J. Liu, S. Xin (corresponding authors), K. Gao, and C. Tu (corresponding authors), Shandong University; emails: {xinshiqing, chtu}@sdu.edu.cn; X. Gao, Florida State University, Tencent America; K. Xu, National University of Defense Technology; B. Chen, Peking University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
0730-0301/2021/08-ART6 $15.00
https://doi.org/10.1145/3476829

Fig. 2. Examples of the object-wrapping rope net applications: Lifting the furniture (left, obtained from internet public domain) and wrapping the statue (right, courtesy of Shunk-Kender).



(a) Rope net structure (b) Visual guidance

Fig. 3. (a) Our proposed new style (top right) can provide stronger force support for fastening the rope net on the object while remaining simple. (b) An illustration tool is provided to guide the physical rope net construction. The pins (colored in red) are needed to perform the assembly in practice.

satisfying a few practical requirements. The main factor to be considered is safeness. We require that the rope net tightly wraps the object and evenly distributes load for safety under the hoisting scheme. Although tight wrapping confines the object movement within the rope net as much as possible, load balancing prevents the object and the rope from breaking. We also hope that the rope net can be composed by a *single* rope and the knotting is as easy as possible. The last aspect is economy. We stipulate that the total length of the rope is minimized.

Given a 3D object represented by a surface mesh, we introduce a simple and robust approach to generate an object-wrapping rope net satisfying the preceding requirements. A rope net is composed of knots and curve edges wired over the surface. Our method is based on two *key design principles*. First, the rope should sling over prominent geometric or topological features of the object surface, such as concave geometric features, forks, and branches, to ensure a safe and reliable tying [Johnson 2016]. Second, we hope the rope net could be composed with a *single* rope.

To make the rope net construction aware of geometric and topological features, we opt to start with a set of key loops induced by the segmentation boundaries of **Shape Diameter Function (SDF)** of the object [Shapira et al. 2007]. SDF is proven to capture well the prominent geometric and topological features of a 3D shape. In achieving single-rope composition, our key observation is that a rope net can be composed with a single rope if each node has a degree of 4 (e.g., has four incident curve edges). This observation has a rigorous theoretical guarantee based on the Euler tour theorem [Bondy and Murty 1976].

To form an evenly distributed rope net over the surface, the sparse key loops are connected with *assistant curves* while satisfying the 4-degree principle. To this end, we build a cross field constrained by the key loop directions. The assistant curves are then constructed from the field directions perpendicular to the key loops.

The preceding process amounts to a discrete-continuous optimization of both the topology and the geometry of the curve network. Although the discrete phase improves the topology of the curve network by selecting proper assistant curves, the continuous stage optimizes the curve geometry via altering the node positions and curve shapes.

In particular, in the discrete step, we design a dedicated algorithm to compute a sparsely distributed, coarse 4-degree initial curve network over the 3D surface through solving a mixed-integer programming problem. Starting with this initial curve network, we conduct an alternating optimization of all node positions to tightly fasten every curve edge by minimizing the length of all curve edges. During this process, a curve edge may leave the surface but is constrained to never penetrate the surface.

In realizing hoisting, we compute anchor points over the curve network so that the rope net is load balanced when being lifted. We develop a hoist planning method that chooses suitable anchors from a set of candidates to meet safety requirements [Johnson 2016]. It minimizes the stretching stress of all curve edges subject to the constraint of the yielding tolerance of rope. Furthermore, when the rope net is too sparse to come up with a suitable hoisting plan, we opt to re-optimize the curve network to add some *reinforcement loops* by tracing through the regions where the stress violates the yielding tolerance. Hence, our method alternates between curve network generation and anchor point selection until a valid hoisting plan is found.

For rope tying, we adopt a twisting knot, a simple and effective knot type for rope net composition (top right of Figure 3(a)). A twisting knot is physically firm while being easy to tie and material saving [Patil et al. 2020]. Note that the one-rope composition property still holds for this twisting knot type based on a rope-able Euler cycle (see Section 5.3 for the proof). In addition, we provide an illustration tool to demonstrate how to compose the rope net intuitively (see Figure 3(b)).

We demonstrate the efficacy of our method through computing rope nets for a variety of 3D objects with complex shapes and topology, and quantitatively evaluating them with a series of metrics. We also conduct physical experiments and present a prototype application in flexible hoisting to show the practical usage of our rope net generation. To sum up, the contributions of our work include the following:

- We solve a new problem of computational object-wrapping rope nets with a series of practical constraints such as safeness and economy.

- We propose a formulation of the rope net problem based on curve network optimization.
- We devise a discrete-continuous optimization process that optimizes both the topology and the geometry of the curve network.
- We provide an illustration tool to guide users for rope net composition and conduct extensive evaluations with not only simulation but also physical experiments.

2 RELATED WORK

We first review the caging literature, which is highly related to our rope net wrapping. Then, we review state-of-the-art quad-meshing approaches since the layout of a quad mesh shares similarities with our rope net structure.

3D caging. Caging is to restrict the moving space of a target so that it will not escape [Diankov et al. 2008]. It has been an important topic in robotic research and is often addressed together with grasping [Diankov et al. 2008; Rodriguez et al. 2011; Wan et al. 2012, 2013]. Caging and grasping of a 2D object have been thoroughly studied, and we refer the interested reader to the work of Makita and Wan [2017] for a complete survey.

Unlike the 2D caging, the 3D caging problem has no complete analysis. The main reason is that it is challenging the high dimensionality without considering mechanical implementation. Several works attempt to tackle the problem using shape analysis and geometry processing methods. Through the usage of the topological characteristics of loops of both the objects and the robotic hands, approaches presented by Dey et al. [2010], Pokorny et al. [2013], and Stork et al. [2013] plan to cage and grasp on objects with holes. The method in the work of Zarubin et al. [2013] employs geodesic balls computed on the target surface to determine the caging regions. They propose circle caging and sphere caging. Whereas circle caging allows robot hands to grasp the thin part of an object by computing closed curves wrapping around the object, sphere caging lets the robot hand wrap a solid part of the object. However, they do not consider the topological structure of the target. In contrast, Kwok et al. [2016] compute a topological Reeb graph over the 3D surface and extract iso-value rings based on the geodesic field for the object rope caging. To deal with the common issue of these methods that the designed caging is oblivious to the relative size between the target object and the gripper, Liu et al. [2018] present a method to compute feasible caging grasp that can form relative-scale-aware caging loops encompassing multiple handles.

In our work, we propose to immobilize a 3D target by computing a tight rope net wrapping over the surface. Unlike caging that allows object moving and reorienting inside the caging space, our rope net is tightly fastened on the surface, ensuring the safety and effortless operation during hoisting big and heavy objects.

Quad layout. Quad meshing has been researched for more than two decades [Bommes et al. 2013b], and many approaches have been proposed. Among the many goals of quad-mesh generation and processing methods, generating a quad mesh with a coarse and feature-aligned quad layout is one of the most desired ones [Tarini et al. 2011a]. Quad meshes can be created either through user interactions [Bommes et al. 2008; Campen and Kobbelt 2014; Marcias et al. 2015; Takayama et al. 2013], semi-automatic methods [Ji et al. 2010; Tierny et al. 2012; Tong et al. 2006], or fully automatic approaches [Bommes et al. 2011; Campen et al. 2012; Razafindrazaka et al. 2015; Tarini et al. 2011a; Zhang et al. 2015]. By tracing edge flows from irregular vertices of a quad mesh, a coarser quad layout than the quad mesh can be constructed. The dual graph of this quad layout could be embedded into our pipeline for initializing our rope net since it satisfies our topology requirement that every node has a valence of 4. However, the layouts extracted from quad meshes generated from existing approaches either are overly dense, which is impractical for physically composing the rope net, or require user interactions and are limited to specific types of shapes. Moreover, all quad-meshing methods optimize the singularities of a quad mesh that greatly affect its layout structure to be in high curvature regions. However, the dual graph of the layout may not capture the features well. It may lead to an unstable rope net. In our work, we propose a simple and effective initial rope net generation approach that directly addresses our object-wrapping goal, sidestepping the usually enforced complex geometrical constraints and misalignment issues during the typical quad meshing. Later, Figures 24 and 25 demonstrate the advantages of generating rope nets initialized by our method over representative quad-meshing approaches.

3 OVERVIEW

We first introduce the basic definitions of the rope net, then state our objectives, and finally give a high-level overview of our method.

Rope net. A rope net $\mathcal{R} = (\mathcal{V}, \mathcal{E})$ wrapping around a 3D surface model $\mathcal{M}$ consists of a set of nodes, $\mathcal{V}$, and a set of curve edges, $\mathcal{E}$ (see Figure 3).

Objectives. Our input includes a 3D surface model $\mathcal{M}$, the center of gravity and weight of the model, and a rope with the maximum stretching stress λ. Our method aims to generate an object-wrapping rope net used for hoisting by satisfying the following objectives:

- *Simplicity*: The rope net should be cost effective (e.g., short total length and a small number of nodes and could be composed with a single rope).
- *Tightness*: The rope net should be tight enough to both immobilize the object and not slide when lifted from any place of the rope net (see Figure 5).
- *Load balance*: To reduce bearing pressure, the rope net should equally distribute the load during lifting.

To generate a rope net that can be used for hoisting while satisfying the preceding goals, we design our approach by obeying the principle that tackles one goal at a time and once a goal is solved, the problem will not appear again. For example, we first ensure the simplicity of the to-be-generated rope net by generating an initial rope net that can be composed using a single rope (Section 4), then achieve the tightness of the rope net on the object by simulating the tightening process via a geometric optimization (Section 5), and finally guarantee the load balance of the rope net through stress analysis and reinforcing weak regions of the rope net. In our work, we make the following assumptions to allow the

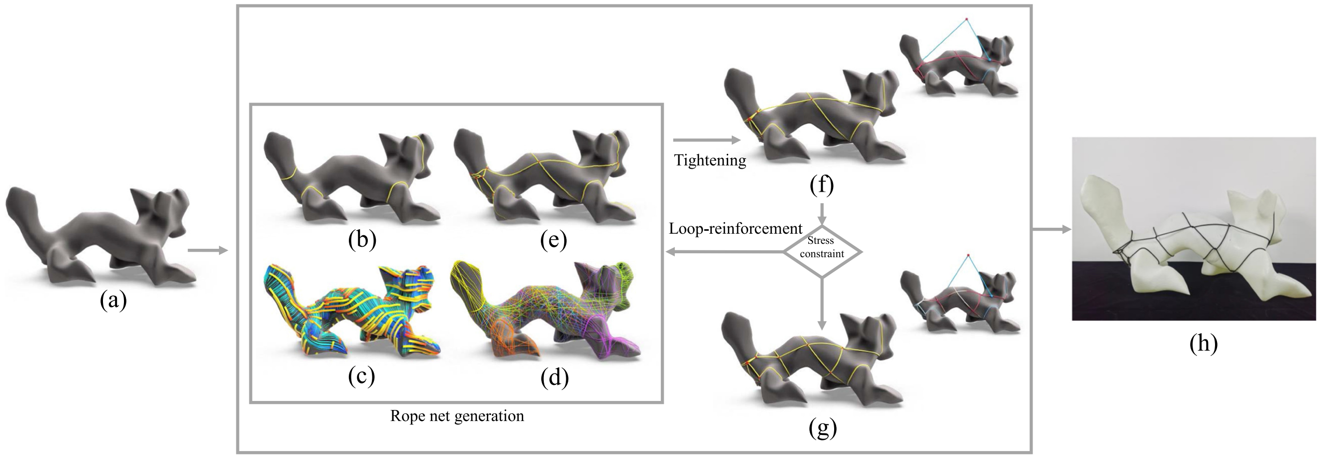


Fig. 4. Overview of our algorithm. Given an input model (a), we generate a rope net step by step. We first capture some key loops (b) that induce a cross field (c). Then, we construct a sufficiently large candidate curve set (d) and take a suitable curve subset as the initialization (e). After that, we refine it to a tight rope net (f) and reinforce the net when necessary (g). The final physically composed result is shown in (h).

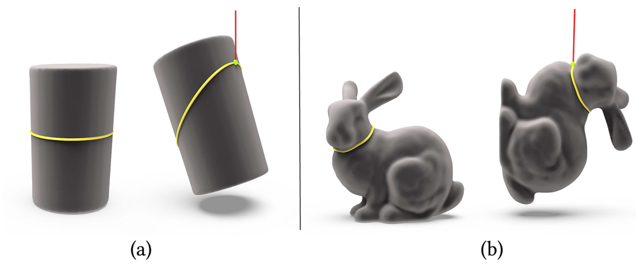


Fig. 5. A tight rope net we target that is able to tightly secure the object without slipping during lifting. The geodesic loop (a) is unstable (easy to slip), although it can be used to cage the object. A stable loop (b) wraps tightly around the object and does not slide during lifting.

solving of the rope net generation problem to be tractable: (i) materials are uniformly distributed throughout the object and the rope, and (ii) the object to be hoisted is strong enough to support forces from the rope net.

Pipeline. Starting from the input object (Figure 4(a)), our approach first relies on a mixed-integer programming to generate an initial rope net that topologically satisfies the one rope construction property and geometrically captures critical regions to immobilize the object (Figure 4(b)–(d), Section 4). After that, we perform a tightening step of the rope net to achieve the tightness objective while avoiding any penetrations (Figure 4(f), Section 5), which is followed by a hoisting planning step (Figure 4(g), Section 5.2). We look for a suitable hoisting plan if (i) there are anchor points satisfying safety standards in lifting operation [Johnson 2016] and (ii) the stress limit of the rope net is not violated when performing the mechanics analysis of the rope net under the specific hoisting configuration. If no such plan exists, we locate the weakest curve edges that violate the stretch stress limit, and add *reinforcement loops* through them as key loops, and iterate the aforementioned steps until a suitable hoisting plan is found. We provide an intuitive user interface (Section 5.3, the attached video) to guide the assembly process of the resulting rope net in practice (Figure 4(h)).

4 ROPE NET GENERATION

In this section, we generate an initial rope net that satisfies the topology constraint—that is, it can be composed by one rope while wrapping key regions of the object that provides a good starting position for achieving both simplicity and load balancing for further steps. Our rope net is computed by solving a mixed-integer optimization on a curve network. Intuitively, the curve network is composed of two types of curves: key loops that are critical to immobilize the 3D object, and assistant curves that connect key loops to ensure the correct topology of the rope net. Given a surface model, from a view of shape analysis, we first compute key loops (Figure 4(b)) that form a subset of curves of the to-be-constructed rope net, and then we connect these key loops by inserting directional field (Figure 4(c)) guided assistant curves that are possibly redundant (Figure 4(d)), and finally we construct the rope net (Figure 4(e)) by solving a mixed-integer optimization to remove redundant assistant curves.

4.1 Key Loops

As discussed earlier, our rope net aims to immobilize the object so that it does not slip during lifting. From the viewpoint of hoisting in practice [Johnson 2016], it suggests that the ropes should be tied to the critical wrapping regions (e.g., concave geometric features, forks, branches) on the surface of an object, which can help to secure the object tightly. Motivated by this finding, we consider wrapping these regions with key loops as critical components of the rope net.

To efficiently compute the key loop wrapping the critical regions described earlier, we use the SDF-guided mesh partition method [Shapira et al. 2007]. This is because, first, the SDF is defined on facets of the mesh that measures the local object diameter. It is used to intrinsically distinguish thin and thick object parts. Thus, it can generate key loops lying at regions with

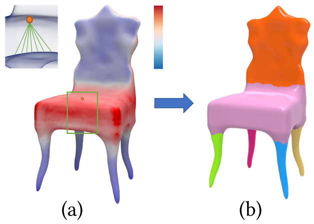


Fig. 6. An example of the SDF-guided mesh partition.

concave geometric features, forks, or branches that align well to the goal of identifying critical wrapping regions. Second, this method is also proposed to extract the skeleton of a 3D object, of which any segmentation can infer a branch of the object. Hence, it can help to generate key loops at the critical wrapping regions described earlier.

To obtain the SDF-based key loops, we followed the partitioning algorithm described in the work of Shapira et al. [2007]. For each face of the mesh, the approach first calculates its SDF value to be the weighted average of the penetration depths of all rays contained inside an inward cone (Figure 6(a)). The weight for a ray is the inverse of the angle between the ray and the center of the cone. Next, the partitioning approach fits k Gaussian distributions to the distribution of the SDF values of the facets and, finally, finds the actual partitioning into m clusters using an alpha-expansion graph cut algorithm [Boykov et al. 2001] by considering local mesh geometric properties. Note that k represents the number of levels of a segmentation, which is different from m. A large value of k can result in many small segments of the mesh. In our implementation, the default value $k = 5$ is used.

After running the mesh partition algorithm, the boundary edges of the segmentation form polyline loops. For a polyline loop, it is considered as a key loop if it does not share edges with all the other polyline loops. Otherwise, we choose the minimal loop (the one with the shortest length) from those polyline loops that share with edges as the key loop. Thus, the selected polyline loops are denoted as the key loop set $\mathcal{L}$ (see Figure 6(b)). Note that small loops may occur due to noises or thin shape features. Since small loops are not helpful to the rope net design, we filter out these loops from $\mathcal{L}$ if their lengths are shorter than 0.005 (the input model is scaled uniformly into a $1 \times 1 \times 1$ box).

4.2 Assistant Curves

After extracting the key loops $\mathcal{L}$, we now generate another set of curves $\mathcal{S}$, which are referred to as assistant curves, to connect the key loops. Our to-be-constructed rope net will be composed of all the key loops and some selected assistant curves.

According to the hoisting operation [Johnson 2016], large contact areas between the rope net and the object can greatly reduce the stresses on the rope net during hoisting. Based on the empirical observation, the orthogonal rope net is commonly used in daily life. The rational behind is that the orthogonal rope net can help distribute the force evenly, which is very useful for hoisting. Therefore, our assistant curves are trajectories traced on the surface of the object. We employ an optimized cross field that aligns directions of geometric features on the surface to guide the curve tracing. The

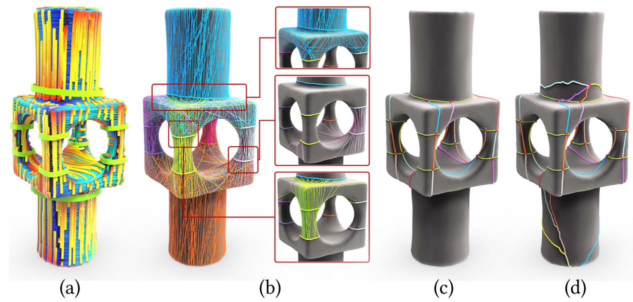


Fig. 7. The workflow of the rope net generation. (a) The cross field aligned with the directions of the key loops (plotted by green circles). (b) The field-guided geodesic curves (plotted by various colors) traced from both two sides of the key loops (colored the same as their curves). (c) We obtain the nodes (red dots) and the curve edges (colored in various colors) between them to compose the rope net by solving a mixed-integer optimization. (d) To guarantee 4-degree connectivity for the nodes on the ending loops, we use a simple linking operation to connect them with random sampling points through Dijkstra shortest paths on the surface model. The obtained curves and the nodes compose the rope net.

efficient instant meshes algorithm [Jakob et al. 2015] is used to generate the directional field. To adapt to the preceding requirement—that is, generating assistant curves that are perpendicular to key loop directions so that contact areas between the rope net and the object can be increased and forces imposed on the rope net can be distributed over different directions [Johnson 2016], we generate a key loop direction constrained cross field (Figure 7(a)). Specifically, for each vertex p, we use a key loop dependent weight instead of their original one: $w(p) = exp(-d^2/2)$, where d is the geodesic distance between p and its nearest key loop. This weight encourages p to get its direction from its most relevant key loop.

After computing the direction field, we can now trace assistant curves. We start tracing from seed points uniformly sampled on the key loops. Note that to provide an enough amount of candidate assistant curves for the rope net generation, the sampling should be dense. In our experiments, we use the average edge length of the surface mesh as the step size for the seed point sampling. For each seed point at each key loop l, we use the field-guided tracing approach [Pietroni et al. 2016] to trace two curves with opposite directions (colored the same as their key loop) that are both perpendicular to l (Figure 7(b)). Therefore, we have two sets of curves l^+ and l^-, one for each side of l. We filter out traced curves that are not ended at any key loop or intersect themselves during tracing, or both, and denote all the remaining ones as the assistant curve set $\mathcal{S}$.

So far, the curve network formed by the key loops and the assistant curves is not necessarily simple, and neither satisfies the topology constraints of the desired rope net. By taking the curve network as the initialization, we next present a mixed-integer programming approach to generate the rope net.

4.3 Mixed-Integer Optimization

As the 2D illustration in Figure 8, the initial node set $\mathcal{V}$ consists of the intersection points (blue dots) between the assistant curves and the intersection points (orange dots) on the key loops. The

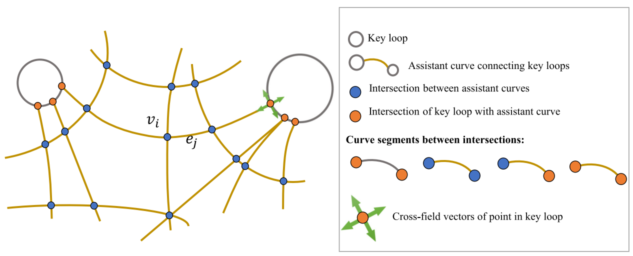


Fig. 8. Illustration of initialization of the curve edges and the nodes. The nodes are initialized with the intersections (blue dots) between the assistant curves (yellow curves) connecting with the key loops (gray circles) and the intersections (orange dots) of the key loops with the assistant curves. The curve edges are initialized with the curve segments between these intersections.

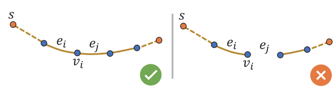


Fig. 9. Two adjacent nodes need to be consecutive in geometry.

initial curve edge set $\mathcal{E}$ consists of the curve segments between these intersections.

For each curve segment, we define an indicator function (v_i, e_j) : $\mathcal{V} \times \mathcal{E} \rightarrow \{0, 1\}$ to represent its existence in the rope net. $(v_i, e_j) = 1$ indicates that the curve segment e_j, of which one endpoint is v_i, is used to compose the rope net. If $(v_i, e_j) = 0$, that means we do not compose the rope net with e_j. Thus, we turn the rope net generation into the problem of removing curve segments from the curve network, which can be formulated as

$$\begin{gathered} \max_{v_i, e_j} E(v_i, e_j) \\ \text{s.t. } v_i, e_j \text{ satisfies the constraints of rope net } (3-9). \end{gathered} \tag{1}$$

Intuitively, we prefer to select long curve segments to reduce the rope net complexity since the longer each segment is, the fewer nodes or knot operations needed to construct the rope net. Hence, the objective function will be formulated as linear energy with variable (v_i, e_j):

$$E(v_i, e_j) = \sum_{i,j} (v_i, e_j) \cdot \|e_j\|, \tag{2}$$

where $\|e_j\|$ denotes the length of the curve segment e_j. Next, we introduce the topology constraints and the sparsity constraints to ensure the reliability of the rope net.

Topology constraint I. The rope net should be a connected graph to be possibly constructed by a single rope (Figure 9). Hence, for an assistant curve s, two consecutive curve segments $e_i, e_j \subset s$ that share one of their endpoints have the constraint $(v_i, e_i) = (v_i, e_j)$, where v_i denotes their shared endpoint. For the two endpoints v_i and v_j of a curve segment $e_j \in \mathcal{E}$, we have a constraint $(v_i, e_j) = (v_j, e_j)$. However, every curve segment e_j on the key loop $l \in \mathcal{L}$ has $(v_i, e_j) = 1$ since all key loops need to be present in the rope net. To sum up, we have the connectivity constraints as follows:

$$\begin{gathered} (v_i, e_j) = (v_j, e_j), \ \forall\, e_j \in \mathcal{E}, \text{ where } v_i \text{ and } v_j \text{ are endpoints of } e_j; \\ (v_i, e_i) = (v_i, e_j), \ \forall\, e_i, e_j \subset s \in \mathcal{S}, \text{ where } v_i = e_i \cap e_j; \\ (v_i, e_j) = 1, \text{ where } e_j \text{ is on the key loop } l, \ \forall\, l \in \mathcal{L}. \end{gathered} \tag{3}$$

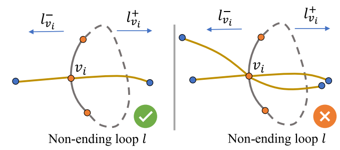


Fig. 10. The constraint of a node on the non-ending loop.

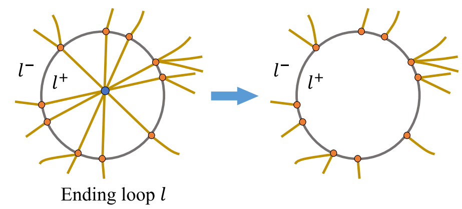


Fig. 11. Illustration of the ending loop. Left: We call a key loop an *ending loop* if all assistant curves on one side of it have endpoints both on the same key loop. Since the assistant curves are generated based on the direction field, they often converge at same intersection point at the ending regions of the branches. Right: For easy post-processing to meet the 4-degree requirement, we remove these curves and connect their nodes back in the post-processing step.

Topology constraint II. A node of the curve network may have an arbitrary valence (i.e., the number of adjacent segments), especially at the ending region of the branches. Since the assistant curves are generated based on the direction field, they often converge at the same intersection point. It is difficult to precisely achieve the 4-degree property for each node during optimization with a consistent topology constraint for all of them. Therefore, we divide the nodes into three cases and impose different constraints for all of them. The constraints are to ensure that the optimized rope net can be easily post-processed to meet the 4-degree requirement.

Case 1. For any node v_i on an assistant curve, we set the constraint

$$0 \leq \sum_j (v_i, e_j) \leq 4, \text{ if } v_i \text{ is on an assistant curve.} \tag{4}$$

After optimization, its valence will be either 0 (not selected), 2 (to be removed by merging its two adjacent curve segments), or 4 (to be preserved as a 4-degree node), due to both constraints Equation (4) and Equation (3).

Case 2. A key loop is called a *non-ending loop* if all connected assistant curves have the two endpoints on different key loops. For a node v_i on a non-ending loop, it should have the same number of curves from each side and at most one from each side (Figure 10). Thus, we set the constraint

$$\begin{gathered} 0 \leqslant \sum_{e_j \in l^-_{v_i}} (v_i, e_j) = \sum_{e_j \in l^+_{v_i}} (v_i, e_j) \leqslant 1, \\ \text{if } v_i \text{ is on a non-ending loop } l, \end{gathered} \tag{5}$$

where $l^-_{v_i}$ and $l^+_{v_i}$ denote the sets of curve segments connecting v_i located at the left and right sides of l, respectively. After optimization, the nodes on the non-ending loops will have the degree being either 2 (to be removed by merging its two adjacent curve segments) or 4 (preserved as a 4-degree node).

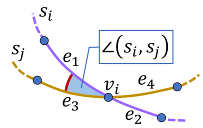


Fig. 12. The angle constraint of the node.

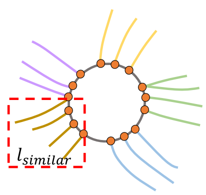


Fig. 13. Similar assistant curves (denoted by same color).

Case 3. For a node v_i on an ending loop, some of the connected curves will have both endpoints on the same key loop (left side of Figure 11). Therefore, we remove these curves from $\mathcal{S}$ before optimization and reconnect their nodes in the post-processing step. During optimization, we formulate the constraint as:

$$2 \leqslant \sum_j (v_i, e_j) \leqslant 3, \quad \text{if } v_i \text{ is on an ending loop.} \tag{6}$$

After optimization, the nodes on the ending loops will have degrees either 2 (to be removed by merging its two adjacent curve segments) or 3 (to be preserved as a 4-degree node with the connected-back assistant curves).

Sparsity constraint I. To make the rope net as simple as possible, we need to limit the number of assistant curves passing through an ending loop by the constraint

$$\sum_{v_i \in l, e_j \subseteq s} (v_i, e_j) = 2k^*, \; k_1 \leqslant k^* \leqslant k_2, \text{ if } l \text{ is ending loop,} \\ \forall\, s \in l^+ \cup l^- \; (l^+ \; or \; l^- = \emptyset), \; k_1 \leqslant k_2 \in \mathbb{N}, \tag{7}$$

where the parameters k_1 and k_2 indicate the minimum and maximum number of assistant curves allowed for one curve set l^+ or l^-, respectively. In our implementation, we set $k_1 = 2$ and $k_2 = 3$. Thus, the number of nodes on an ending loop is either four or six.

Sparsity constraint II. The nodes of the rope net should be cross-like so that the curve edges of the rope net are not too close together after tightening the rope net. Hence, if the angle between two assistant curves s_i and s_j at v_i (Figure 12) is smaller than a threshold θ ($pi/3$ is used in the implementation), we restrict the curve segments that are either on s_i or s_j as follows:

$$0 \leqslant \sum_j (v_i, e_j) \leqslant 2, \; \forall\, e_j \subset s_i \cup s_j, \quad \text{if } \angle(s_i, s_j) < \theta. \tag{8}$$

Sparsity constraint III. Nearby assistant curves with similar shapes should be clustered as one. Note that to improve efficiency, we cluster nearby assistant curves that connect to the same key loop and lie on the same side of that loop into one group (Figure 13).

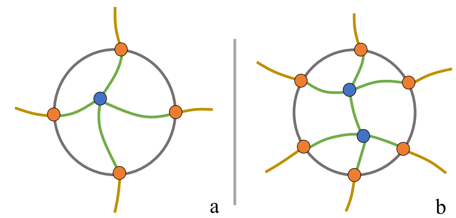


Fig. 14. Post-processing of the ending loop.

At most one assistant curve from each group can be used in the rope net. This results in the following restriction:

$$0 \leqslant \sum_{v_i \in l, e_j \subseteq s} (v_i, e_j) \leqslant 1, \; \forall\, s \in l_{\text{similar}}. \tag{9}$$

To cluster the similar assistant curves, we use the K-Means method. In the implementation, all the assistant curves are converted into 2D embedding by the classical multidimensional scaling (CMDS) algorithm [Kong et al. 2019]. The distance between the assistant curves is calculated based on the discrete Frechet distance metric [Eiter and Mannila 1994]: $d_{Frechet}(e_i, e_j) = \min\{\|\Gamma\| \mid \Gamma$ is a coupling between the curve e_i and curve $e_j\}$. The K-Means method takes the distance measure and the 2D points as inputs, and outputs the clustering results of assistant curves. To determine the optimal number of clusters of the K-Means method, we also use the Elbow method [Ketchen and Shook 1996], which is a fundamental step in cluster analysis.

Up to now, we have the necessary topology and sparsity constraints for our optimization. We compute the curve segments that compose the rope net by solving the constrained integer linear programming (CILP) problem using the work of Achterberg [2009]. After merging the adjacent selected curve segments of the 2-degree nodes, the rope net then has the remaining nodes with degree 3 or 4 (see Figure 7(c)).

Post-processing. To further obtain the 4-degree rope net (see Figure 7(d)), we add back the curves with endpoints on the same ending loop. According to Equation (7), the number of nodes on the ending loop l is either four or six. If l has four nodes (Figure 14(a)), we connect them (orange dots) with one sampling point (blue dot) by Dijkstra's shortest paths (green curves) on the surface. When six nodes are selected on l (see Figure 14(b)), we sample two different points to connect them by geodesic paths as well. Each sampling point connects with three nodes along the clockwise direction of the ending loop. Then we connect the two sampling points with a geodesic path. Figure 7(d) shows the result after the post-processing step. Thus, all the 4-degree nodes and the obtained curves connected to them form the rope net.

Discussion. The optimization may fail in extreme cases when the assistant curves are very sparse. For example, if only a few assistant curves cross through a key loop placed at the region like a handle, the sparsity constraints of Equation (9) will conflict with feasibility. However, our algorithm works well for most of our test examples because we sample a dense curve network before optimization.

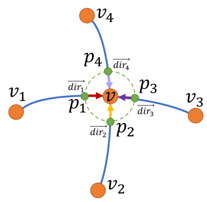


Fig. 15. 4-fork structure at the node.

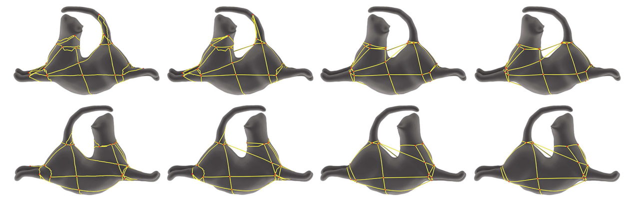


Fig. 16. We only need a few iterations to shrink the initial rope net to a tight one. Here we show the results of the front (top row) and back (bottom row) of an object respectively in the 0^{th}, 1^{th}, 2^{th}, 3^{th} iteration.

5 ROPE NET CONSOLIDATION

Given the rope net generated in the previous section, the rope net needs to wrap tightly around the object so that it can confine the object movement within it as much as possible. We also need to find a suitable hoisting plan for the rope net that prevents overloading of the rope net and satisfies safety under the hoisting scheme [Johnson 2016]. Moreover, for rope tying in practical usage, we need to assemble the rope net and tie the rope into a knot at each node as well so that it can provide a strong force while being easy to tie and material saving [Patil et al. 2020].

Thus, in this section, we propose a rope net consolidation method to achieve the preceding requirements. First, to tighten the rope net, we minimize the length of the rope net while avoiding any penetrations. Second, to find a suitable hoisting plan, we look for anchor points that satisfy safety in hoisting operation. From them, we find a suitable hoisting plan by minimizing the stresses on the rope net while considering the physical properties of the rope. If no suitable plan is available, we locate the weak curve edges that violate the stretch stress limit, add *reinforcement loops* through them as key loops, and recompute the rope net unit to find a suitable hoisting plan. Since our rope net can be viewed as a graph where each node has exactly a degree of 4, we can use a single rope to construct the rope net according to the theorem of the Eulerian circuit [Bondy and Murty 1976]. For rope tying, we adopt twisting knot, which is a simple and effective knot type for rope net composition. To assemble the rope net for practical usage, we present a rope-able Euler cycle to guide the assembly process of the resulting rope net in practice, which guarantees that the rope net can be constructed by a single rope and each node is tied into a twisting knot.

5.1 Rope Net Tightening

To tighten the rope net, we minimize the total length of the rope net:

$$\min_{\mathcal{V}} \sum_{e_{ij} \in \mathcal{E}} \|e_{ij}\|, \tag{10}$$

where e_{ij} denotes the curve edge between adjacent nodes v_i and v_j. By taking the node set $\mathcal{V}$ as the variables, our optimization alternatively moves the nodes with the L-BFGS solver and shrinks the curve edges $e_{ij} \in \mathcal{E}$ between adjacent nodes. Every curve edge e_{ij} is updated in each iteration as the shortest collision-free path between adjacent nodes. The pseudo-code is available in Algorithm 1. Note that during the optimization, both the nodes and the curve edges are allowed to leave the surface model rather than strictly constrained on the surface but are prohibited to penetrate into the surface model (collision between the rope net and surface model).

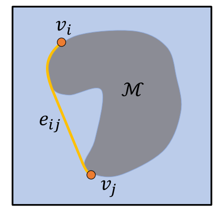


Fig. 17. Illustration of collision-free path.

ALGORITHM 1: The algorithm for tightening rope net.

Input: an initial node set $\mathcal{V}$ and an initial rope net configuration $\mathcal{E}$

Output: a tight rope net $\mathcal{R}$

Compute the total length of the rope net $\sum_{i,j} \|e_{ij}\|$;

Compute the gradients of $\sum_{i,j} \|e_{ij}\|$ w.r.t. each node v_i;

while *the norm of the gradient vector is larger than the specified tolerance* **do**

  Move every node $v \in \mathcal{V}$ by Equation (11);

  Update the collision-free path e_{ij} between v_i and v_j by [Crane et al. 2013];

end

Node movement. For a node, the L-BFGS solver moves it from its initial position to a locally stable position by the gradient vectors of the object function (Equation (10)). Suppose every node v is adjacent to four neighboring points p_1, p_2, p_3, p_4. We use $\overrightarrow{\text{dir}}_i = \frac{v-p_i}{\|v-p_i\|}$, $i = 1, 2, 3, 4$, to represent four unit vectors at the node v of the rope net (Figure 15). Thus, the gradient of each node v is computed as follows:

$$\begin{aligned} \frac{\partial \sum_{e_{ij}\in\mathcal{E}} \|e_{ij}\|}{\partial v} &= \sum_{e_{ij}\in\mathcal{E}} \frac{\partial \parallel e_{ij} \parallel}{\partial v} \\ &= \frac{v-p_1}{\|v-p_1\|} + \frac{v-p_2}{\|v-p_2\|} + \frac{v-p_3}{\|v-p_3\|} + \frac{v-p_4}{\|v-p_4\|} \\ &= \sum_{i=1}^{4} \frac{v-p_i}{\parallel v-p_i \parallel} = \sum_{i=1}^{4} \overrightarrow{\text{dir}}_i, \end{aligned} \tag{11}$$

where e_{ij} is the curve edge (shortest collision-free path) between the nodes v_i and v_j. To ensure that the rope net does not penetrate the surface model, all nodes should be on or outside the surface. Therefore, during the optimization process, we check the position of each node and pull it onto the surface using [Larsen et al. 1999] if lying inside (leave it alone if it is located on the surface or in the exterior).

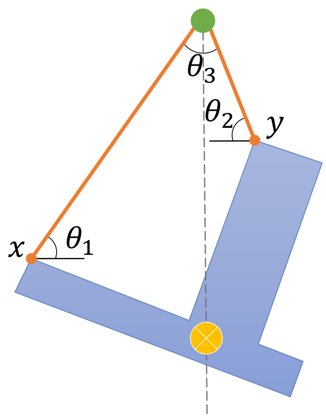


Fig. 18. 2D illustration of our hoist configuration for hoisting an object (blue). Given the object's center of gravity (yellow dot) and one lifting point (green dot) directly above it, we compute a pair of anchor points (orange dots) x and y. The sling angles θ_1 and θ_2 are formed by their slings (orange lines) with the horizontal axis.

Figure 16 shows an example of the optimization process. For this example, it has 51 nodes and each node takes 0.65 seconds on average to run the L-BFGS optimization. Moreover, it only needs 3 iterations (1 iteration refers to all nodes moving once), which is due to the fact that the initial rope net is good enough. In our experiments, the number of iterations required for all models ranges from 3 to 68, with 5 as the average. The cactus example (see Figure 33(c)) requires the most iterations.

Collision-free path. For any two adjacent nodes, we compute the collision-free path using the heat-based method proposed in the work of Cranet et al. [2013]. This method computes the shortest path going through the specified domain (i.e., the surface and the exterior in our problem).

Specifically, in the implementation, we first discretize the space (colored blue) bounded by the surface $\mathcal{M}$ and a large box covering the model (drawn by the black rectangle) into a tetrahedral mesh. Figure 17 shows a 2D example. Limiting the path inside the blue space can naturally prevent the penetration of the rope net. We then compute the discrete gradient, divergence, and Laplace operator for the tet-mesh, which are well-established in the work of Desbrun et al. [2008]. Finally, we employ the heat-based method [Crane et al. 2013] to compute the shortest distance field, from which the shortest distance $\|e_{ij}\|$ and the collision-free path e_{ij} can be quickly found. In particular, by running the heat-based method in the tetrahedral mesh of the exterior space, the collision-free path between the node v_i and the node v can be traced along the negative gradient direction, assuming that the distance field is linear in each tetrahedral element. Therefore, the collision-free path consists of a sequence of corner points, each being the intersection between the path and a triangle face of the tetrahedral mesh. In Equation (11), p_1, p_2, p_3, p_4 are four corner points incident to the node v.

5.2 Hoist Planning

So far, we have a rope net that secures the object tightly. Next, we introduce how to use our rope net in lifting practice. Specifically, we consider the most used sling configuration of bridle hitch, where two anchor points are used together to lift an object with one lifting point (Figure 18). The goal is to distribute stresses evenly across the entire rope net to avoid overloading and ensure safety in lifting operation [Johnson 2016].

Our solution is to first search for a set of candidate anchor pairs and then perform mechanics analysis to find the best one, as is done in the industry practice [Johnson 2016]. Note that the lifting point is always located on the object's plumb line, so we do not optimize the lifting point in our algorithm. In our implementation, the lifting point is placed above the object's center of gravity (0.3 as the default height).

Candidate pairs of anchor points. Our guidelines to select the candidate anchors come from the standard constraints [Johnson 2016] in the industry practice:

- The anchor points should be always visible from the lifting point since we never drag the slings over the object surface.
- The object's center of gravity must be not only directly under the lifting point but also below the lowest anchor point before the object being lifted, to reduce the forces on the slings and the anchor points.
- The sling angles of anchors θ_1 and θ_2 (formed by their slings with the horizontal direction) need to be greater than $pi/4$ and smaller than $pi/3$.

Given the object's center of gravity (yellow dot) and a lifting point (green dot) directly above it (see Figure 18), we assume that the orientation of the input object conforms to the guidelines as described earlier. The first step is to uniformly sample points on the entire rope net. The unit length of rope (0.001 in our implementation) is taken as the step size for the dense sampling. Next, following the guidelines, we remove the sampling points if they lie beneath the horizontal plane through the center of gravity or the connecting line to the lifting point penetrates the surface model.

For the remaining sampling points, we generate a set of point pairs. Since accurately measuring the sling angle for any free-shape object is difficult, we instead use the angle θ_3 formed at the lifting point [Johnson 2016] (the angle between the two orange lines). For a pair of sampling points, its two points and the lifting point form a triangle. We refer to this pair as a candidate if the object's plumb line passes through the triangle and the angle θ_3 formed at the lifting point is greater than $pi/3$ and less than $pi/2$. We denote the set containing all pairs of points as $\mathcal{A}$.

Mechanics analysis of the rope net. We search for a suitable point pair $(x, y) \in \mathcal{A}$ via mechanic analysis. To ensure safety, the stresses acting on the rope net during lifting should be as small as possible, whereas the stress of each curve edge should not exceed the stress limit of the rope. Therefore, the problem can be formulated as

$$\begin{aligned} &\min_{(x,y)\in\mathcal{A}} \; E(x, y, \mathcal{R}, F) \\ &\text{s.t. } I(x, y, e, F) \; < \lambda, \; \forall\, e \in \mathcal{E}, \end{aligned} \tag{12}$$

where λ is the yielding point of a specific material (by default we use $\lambda = 5.48e^7 N/m^2$ for elastic). $I(x, y, e, F)$ is the stress of a curve edge $e \in \mathcal{E}$ when taking points x and y as the anchors and applying

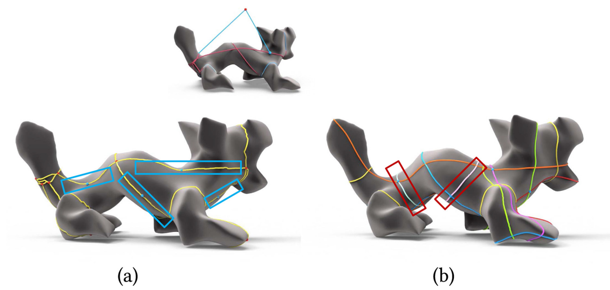


Fig. 19. An example of the loop-reinforcement processing. (a) The weak curve edges (highlighted by the blue rectangles) whose stresses are greater than λ. (b) The non-trivial loops (highlighted by the red rectangles) searched from the traced closed curves (with various colors).

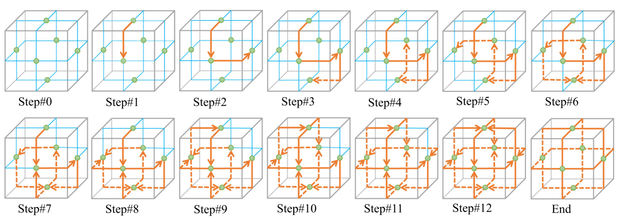


Fig. 20. 2D illustration of the assembling process of our rope-able circuit net. The green dots denote the nodes of rope net and the rope-able circuit net consists of the order lines in orange. The circuit net starts with a first direction for the starting node. Then we turn left or turn right, rather than straight ahead to the next exit direction.

the lifting force F. The objective function is then formulated by

$$E(x, y, \mathcal{R}, F) = \int_{e \in \mathcal{E}} I(x, y, e, F)\, de. \tag{13}$$

Our goal is to compute a pair of anchor points so that the rope net is not overloaded and the sum of the stresses on the rope net is minimal.

In the implementation, since the mechanical analysis takes on average 3 minutes for each candidate pair, to reduce the time consumption, we sort the pairs in set $\mathcal{A}$ in descending order by their angles θ_3 formed at the lifting point. We then select the first 10 sampling point pairs in $\mathcal{A}$ as the candidate. Our input consists of a 3D object positioned in a physically simulated environment, a rope net whose material is specified as nylon, and two sampling points called *anchor points*. The lifting forces at the anchor points are determined by the gravity of the object. We compute the stress field of the rope net based on the finite element analyses in a popular FEM software [Abaqus 2018]. Let V_e denote all the elements in the curve edge e. The stress value of the curve edge e is computed as: $I(x, y, e, F) = \max_{t \in V_e} \sigma(t)$, where $\sigma(t)$ is the stress value of the element t in the curve edge e.

Loop-reinforcement processing. In case no anchor point pair is found that is capable of lifting the object under the safety constraint, we perform the following reinforcement of the rope net. As shown in Figure 19(a), we find the weak curve edges (highlighted by the blue rectangles) whose stresses are greater than λ. For every weak curve edge, we first project it onto the surface, and start tracing closed curves passing through the midpoint of

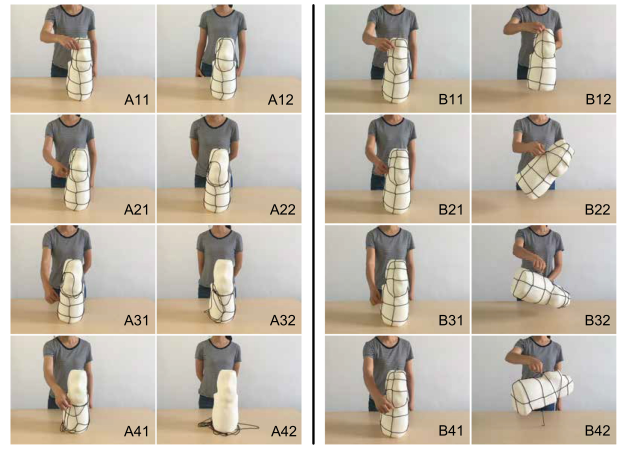
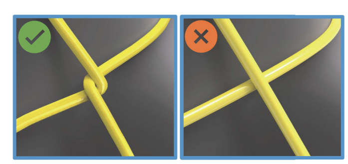


Fig. 21. Comparison to the rope net with various assembling way in physical reliability. The left two columns show the general Euler cycle based rope net before lifting and after shaking at the four different hoisting points in order. The right two columns are the results of rope net with our rope-able Euler cycle.

the projection, guided by the directional field. From the newly traced closed curves (with different colors in Figure 19(b)), we search for non-trivial loops (highlighted by the red rectangles in Figure 19(b)) that are short and field aligned by minimizing the measure described in the work of Campen et al. [2012]: $c_\alpha(l) = \sum_{p \in l} \sqrt{cos^2\theta(p) + \alpha^2 sin^2\theta(p)}$, where $\theta(p)$ is the angle between the loop's tangent and the field direction at the point p on the loop l and $\alpha = 30$ is the balanced parameter. After that, these non-trivial loops, which we call *reinforcement loops*, are added to $\mathcal{L}$ and recompute the tightened rope net through the approaches described in Section 4 and Section 5.1. Note that we do not add such a non-trivial loop to $\mathcal{L}$ if it intersects with a key loop in $\mathcal{L}$. Moreover, if two non-trivial loops intersect, we add the shorter one to $\mathcal{L}$.

5.3 Assembly



Our rope net can be seen as a graph with each node has exact a 4 degree. According to the theorem of the Eulerian circuit [Bondy and Murty 1976], every piece of the graph can be visited exactly once, the starting and ending nodes of the traversal is the same, and the starting node can be chosen arbitrarily. We employ the Fleury algorithm [Skiena 1990] that fully exploits these properties to assemble the rope net. However, the constructed rope net using the original Fleury algorithm cannot guarantee to hold the object tight, since the rope is not knotted at the nodes (see the inset, right). We propose to practically construct a *rope-able* circuit net by physically pinning each node to enhance the stability (see the inset, left).

Based on the Fleury algorithm, we modify the rope tracing on how to select the in-path and out-path at every node. Each 4-degree node forms a local 4-fork structure located on a plane that allows a counter-clockwise order for the four directions, as in Figure 15. For a node v, we assume the $\overrightarrow{\text{dir}}_1$ is the first entry direction

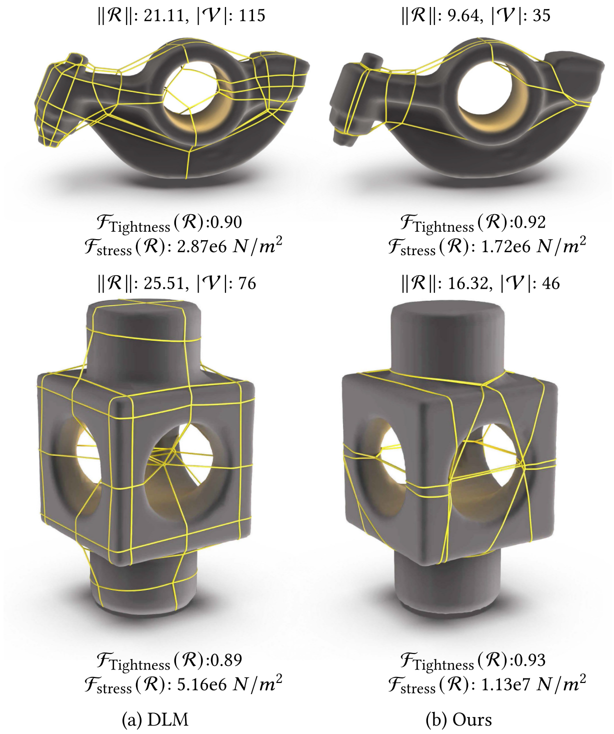


Fig. 22. We compared the final rope net initialized with the dual loops (field-aware geodesic loops) computed in DLM method [Campen et al. 2012] and optimization.

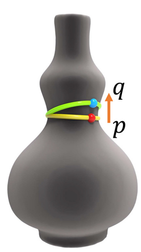


Fig. 23. An example of a perturbation, such as pulling p to q.

for v, and then we choose $-\overrightarrow{\text{dir}}_2$ (right turn) or $-\overrightarrow{\text{dir}}_4$ (left turn), rather than $-\overrightarrow{\text{dir}}_3$ (straight ahead), as the next exit direction. In this way, the algorithm starts from an arbitrary node and chooses the next curve edge at each step as described earlier. It then moves to the other endpoint of the curve edge and deletes the current curve edge. At the end of the algorithm, there are no curve edges left, and the tracing path forms a single-rope based Eulerian circuit. An example of the rope net assembly process can be found in Figure 20.

We conduct a real shaking experiment to compare our rope-able circuit net with the Eulerian circuit generated from the original Fleury algorithm. As shown in Figure 21, our rope net exhibits stable resistance to forces with varying strength and directions, which also demonstrates the physical reliability of our method.

6 RESULTS AND EVALUATION

In this section, we propose a set of metrics (Section 6.2) to quantitatively measure the effectiveness of the computed rope nets for 3D objects with various complexities. We also perform ablation studies (Section 6.3) and compare to alternative approaches to demonstrate the advantages of our method. Last, we give additional experimental results (Section 6.4) to analyze the performance of the algorithm under the conditions such as different parameters and high genus, and show physical results.

6.1 Implementation

We implement the rope net computation on a 64-bit version of the Win10 system with an Intel CoreTM i7-7700 CPU at 4.2 GHz and 8 GB of memory. We test our algorithm on 37 meshes from Thingi10K [Zhou and Jacobson 2016], the McGill 3D Shape Benchmark [Siddiqi et al. 2007], and the AIM@SHAPE Shape Repository. The computation of our workflow from rope net generation to rope net tightening has an average of 5 to 10 minutes per model.

6.2 Evaluation Metrics

We design a series of evaluation metrics to quantitatively evaluate various aspects of our rope net (i.e., its tightness, stress distribution, and simplicity).

Tightness. Recall that the resulting optimized rope net $\mathcal{R}$ may contain some parts lying in the exterior space but must have at least a point on the surface $\mathcal{M}$. Let $p \in \mathcal{R}$ be an arbitrary point exactly lying on the surface $\mathcal{M}$, and let $q \in \mathcal{M}$ be a point in a small neighborhood of p. Generally speaking, if we slightly perturb $\mathcal{R}$ at p (keeping the rope net structure unchanged) such that the new rope net $\mathcal{R}_q$ (we call it *q-based rope net*) passes through q, the length of the q-based rope net must be greater than or at least equal to that of the p-based rope net since $\mathcal{R}$ is stable (length-minimized), as illustrated in Figure 23. We can compute the length change rate of $\|\mathcal{R}\|$ w.r.t. p by

$$\max_{q \in \text{Neigh}(p)} \frac{\|\mathcal{R}_q\| - \|\mathcal{R}\|}{\|p - q\|},$$

where Neigh(p) denotes the neighborhood of p on the surface $\mathcal{M}$. Note that since the rope net is allowed to leave the surface, we use geodesic distance to measure the lengths of the parts of the rope net lying on the surface and use Euclidean distance to compute the lengths of the other parts not on the surface. The tightness of $\mathcal{R}$ can be thus defined by

$$\mathcal{F}_{\text{Tightness}}(\mathcal{R}) = \max_{p \in \mathcal{R} \cap \mathcal{M}} \max_{q \in \text{Neigh}(p)} \frac{\|\mathcal{R}_q\| - \|\mathcal{R}\|}{\|p - q\|}. \tag{14}$$

In implementation, we sample p to be the middle point of each curve edge. For a fixed p, we select the point q along the direction that is orthogonal to the curve edge of the rope net and tangent to the surface. The bigger $\mathcal{F}_{\text{Tightness}}(\mathcal{R})$ is, the tighter the rope net is, indicating that the rope net can tightly secure the target without slipping.

Stress distribution. The stress distribution of a rope net depends on where to lift the rope net and where to wrap the target object. Hence, we compute the stress distribution $\mathcal{F}_{\text{stress}}(\mathcal{R})$ as the sum of the stresses on each curve edge based on formula $E(x, y, \mathcal{R}, F)$ (Equation (13)). The lower value of $\mathcal{F}_{\text{stress}}(\mathcal{R})$, the smaller of the stresses on the rope net, which implies that the rope net can perform the load-balance lifting task better.

Fig. 24. Comparing with the dual layout-based rope net. The final rope nets (a–c) are initialized by IM [Jakob et al. 2015], QF [Huang et al. 2018], and SQD [Tarini et al. 2011a] and tightened using our tightening algorithm.

Simplicity. We use the number of the rope nodes $|\mathcal{V}|$ and the length of the rope net $\|\mathcal{R}\|$ to measure the simplicity of the rope net. It is apparent that the fewer nodes the rope net has, the simpler the rope net is. For fair comparisons, we uniformly scale the input model to a $1 \times 1 \times 1$ box.

6.3 Ablation Studies

Although we cannot find prior work solving the same problem, we demonstrate the rationality of our designed pipeline by com-

Fig. 25. Comparison on the models with sharp feature and high genus. The rope nets (le column) are the results of the Skeleton-driven method [Usai et al. 2015] and IGM method [Bommes et al. 2013a] taken as initializers and tightened by our tightening algorithm. The rope nets (right column) are our results.

paring our method against a set of possible alternatives using the proposed metrics.

Various key loop strategies. We look into an ablation experiment that replaces our SDF-based key loop with dual loops (field-aware geodesic loops) computed in the DLM method [Campen et al. 2012]. We extract dual loops from the quad layout of its results. These loops automatically construct a rope net that guarantees the 4-degree property but are not necessarily satisfy the specific rope net requirements. As shown in Figure 22, our method yields better performance in covering the critical wrapping regions of the object and is more suitable to generate a simple and cost-effective rope net.

Various initial rope nets. We note that the dual of a quad layout is naturally a curve network with every node having a valence of 4, which can be directly used for the initialization of our rope net. However, as shown in Figure 24, we compare our generated rope nets with the ones from instant meshing [Jakob et al.

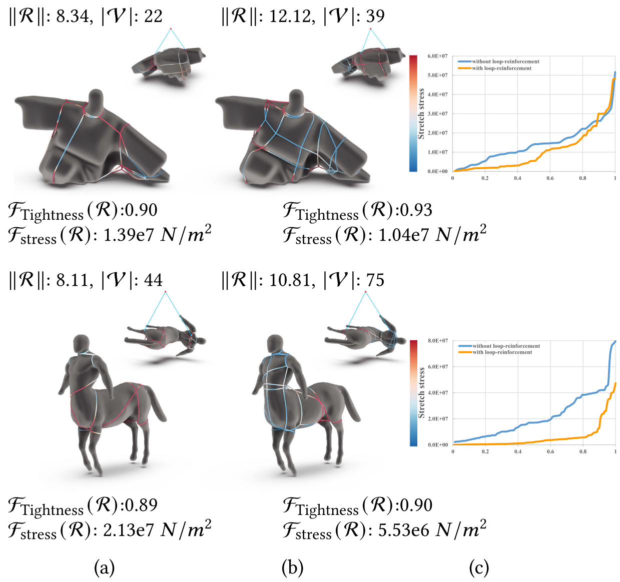


Fig. 26. Comparing the stress distribution of the rope net with (a) and without (b) the loop-reinforcement step under the similar hoisting plan. (c) The distribution of the stress on each edge with and without loop reinforcement. The reinforcing rope nets have smaller maximum stress and perform better in load balancing than those without the reinforcements.

2015], feature-aligned meshing [Huang et al. 2018], and simple quad domain [Tarini et al. 2011a], respectively. Our results are considerably simpler and cost less materials for all the models. Moreover, our rope nets capture more easily of the crease regions than the other approaches. The main reason is that our initial rope net generation is directly guided by a shape descriptor that segments parts of an object effectively. The misalignment of the separatrices traced out from irregular vertices is a long-standing problem in quad meshing [Tarini et al. 2011b], and it can lead to arbitrarily long separatrices and a complex quad layout. Even there is no singularity misalignment problem present in the quad layouts, as shown in Figure 25, our results are still simpler since the quad-meshing methods usually need to take into consideration the mesh quality that is irrelevant to our rope net generation.

With vs. without the loop-reinforcement processing. To study the effectiveness of the reinforced rope net with more key loops, we compare our method to itself without any reinforcement. The results of the stress distribution are shown in Figure 26. Note the additional key loops of weak curve edges after applying the reinforcement processing step. Naturally, the rope net becomes denser as demonstrated in Figure 26(b). With the reinforcement step, the stresses distributed over the rope net are much smaller than the one without, as shown in Figure 26.

6.4 Additional Results

Gallery. Our method can compute the rope nets and hoisting plans over 3D models with various complexities. In Figure 27, we show a gallery of examples generated by our method. For each model, its hoisting plan consists of one lifting point (dotted by a red sphere) and a pair of anchor points (dotted by blue spheres) on the rope net.

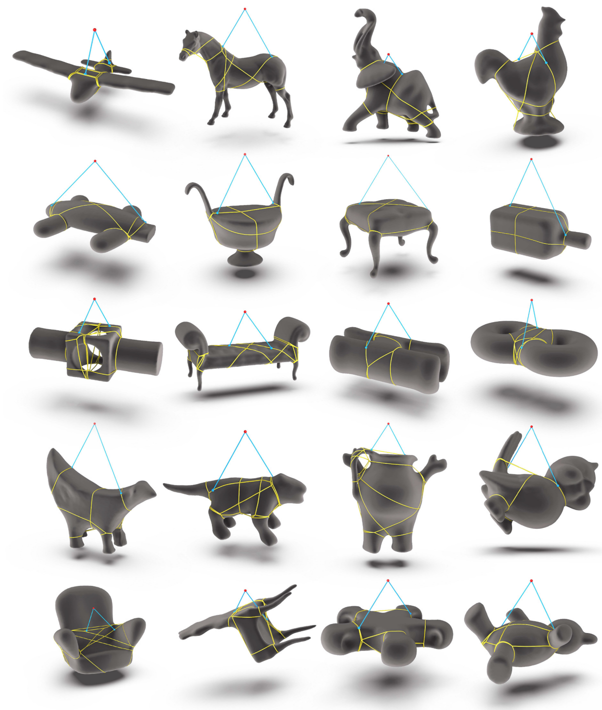


Fig. 27. Gallery of examples generated by our method. We compute the rope nets and hoisting plans over 3D models with various complexities. For each model, its hoisting plan consists of one lifting point (dotted by a red sphere) and a pair of anchor points (dotted by blue spheres) on the rope net.

Key parameters. Our approach allows the easy change of parameters k_1 and k_2 in Equation (7) to adjust the simplicity of the generated rope net. In Figure 28, we use different k_1 and k_2 to generate rope nets with varying simplicity. We also expose the variant λ of our method to users for controlling safety aspect during hoisting. As shown in Figure 29, we compare the performances of the rope nets made of carbon fiber (Figure 29(a)) with that made of nylon (Figure 29(b)). Note that when using a strong rope (carbon fiber rope), we can provide a simpler rope net while satisfying the stress constraint.

Robustness to high genus. As shown in Figures 22, 25, and 27, our approach can handle models with high genus. The complexity of the rope net depends on the cross field. High genus shapes often have complicated cross field. We can easily simplify the complexity of such kind of models by wrapping the rope nets around their enveloping surfaces presented by the nested cage [Sacht et al. 2015], as shown in Figure 30. However, this method only can work for high genus shapes with tiny holes since the enveloping surface could cover the small holes. The rope net is still complex if the shape models with many large holes in geometry (see Figure 22 and Figure 25).

Physical results. As shown in Figure 31, we printed 12 models and realized our computed rope nets on the corresponding

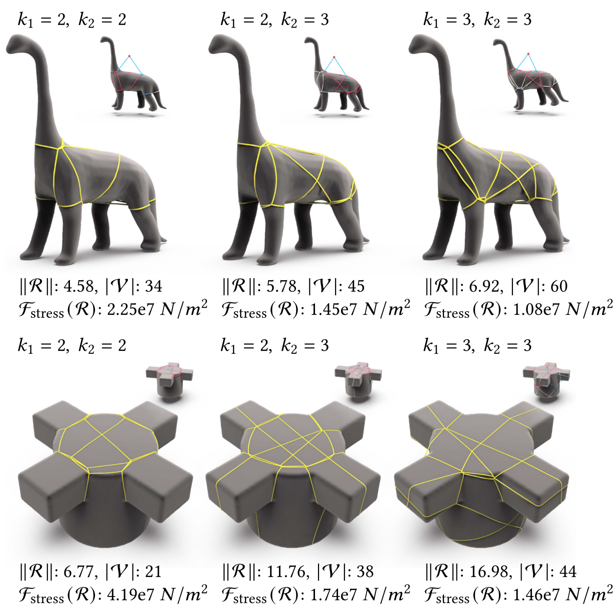


Fig. 28. We can easily vary k_1 and k_2 for different simplicity to generate the rope nets.

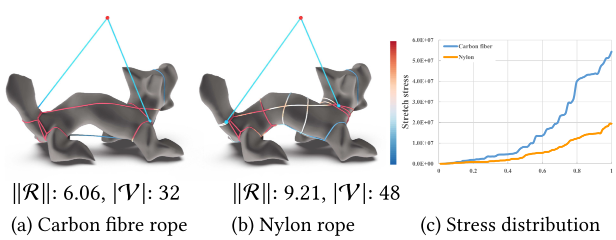


Fig. 29. Our method can incorporate rope usages to generate the rope nets formed by different physical materials.

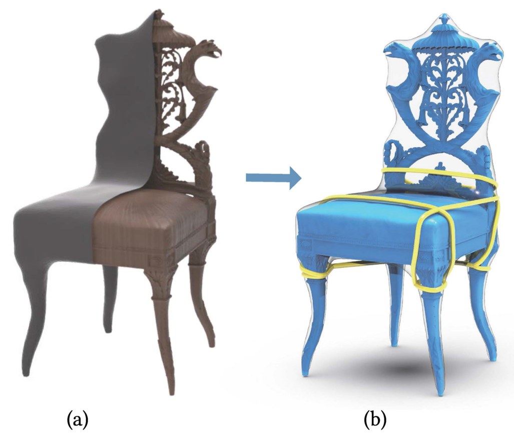


Fig. 30. An example of the simplification process for high genus shapes with many small holes. To simplify the complex of rope net, we compute an enveloping surface (plotted by the gray color) of the original chair model (a) by the nested cage [Sacht et al. 2015] and compute the rope net wrapping around the enveloping surface based on our method (b).

physical objects. We employ four people and provide them with our assembly GUI for constructing the rope net. As expected, every rope net can be assembled using a single rope. The assembly time ranges from 30 minutes to 180 minutes for one model with an average of 60 minutes over all 12 models.

7 DISCUSSION AND CONCLUSION

We introduce an interesting problem of a computational object-wrapping rope net, which not only tightly secures the object in practice but can also be composed with a single rope. We present a shape-aware curve network to effectively solve the problem. Both topology and geometry of the curve network are optimized via a discrete-continuous optimization to satisfy the requirements of the rope net. Through extensive experiments, we demonstrate that our approach is noticeably effective in terms of robustness and generally applicable for 3D models with different shape complexities. Using the visual guidance tool that we provide for users, the assemble property of our rope net is also demonstrated through physical experiments. Moreover, our method produces high-quality rope nets for a wide variety of shapes and proposes extensive metrics for the rope net evaluation that can be well generalized to new problem instances.

As a first attempt to solve the new problem, our current solution still has some limitations. Our method starts with the key loop generation, which is such an important step of our pipeline since every other step (e.g., assistant curves, initial rope net, rope net consolidation) is computed from this initial set of loops. After that, the rope net consolidation step moves the curve edges by minimizing the total length of the rope. However, in some rare cases, the two steps do not connect well, such as in the example of the cactus model (Figure 33(c), which is due to the requirements of the rope net are not directly embedded in generating the initial key loops. In Figure 33(c), the optimized rope net is quite different from the initial result, indicating that our initial key loops are not helpful for this case. Directly incorporating mechanical aspects into our formulation to find stable key loops would be a better choice. If considering the frictional property of a surface, the rope net, after being shortened, may not be exactly a geodesic net, as shown in Figure 33(d). Then the interesting observation is that the lateral frictional force is proportional to the geodesic curvature. Therefore, in real-life scenarios, the rope net is not a geodesic net even if in the stable state. A promising direction is to solve a coupled problem by jointly learning or optimizing the key loop generation and the rope net consolidation together.

However, the heuristic key loop strategy based on the SDF approach is motivated by real-world lifting experience and works well for most shapes, although it still has some space to be improved. For example, it may extract no key loops for some extreme cases, such as primitive shapes and very thin parts (see Figure 33(a) and (b)). In addition, it generates inconsistent key loops across various mesh resolutions, resulting in different rope nets of the same object (see Figure 32). Nevertheless, how to obtain the stable loops of 3D objects is an exciting research problem. Our algorithm, in its current form, still lacks enough physics considerations, which needs to be further improved in the future.

Fig. 31. Some rope nets physically realized using our method.

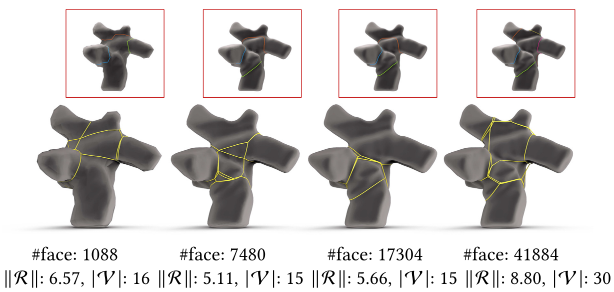


Fig. 32. The number of faces used in the input model can effect its critical wrapping regions extracted by the SDF-based key loops (plotted by various colors in the top row). Although exhibiting similar shape, our rope net is still not invariant to different resolutions.

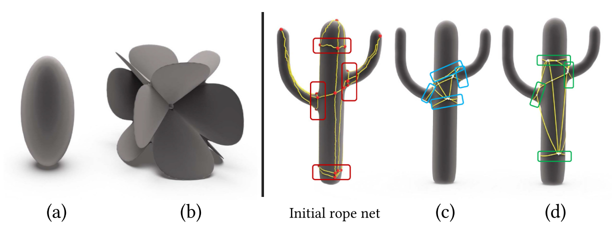


Fig. 33. Limitations of our rope net. Our method may fail for objects where no key loops are extracted such as primitive shape objects (a), and shapes composed with extremely thin features that may break ropes (b). (c) Without considering the frictional coefficient of the surface, the rope net after optimization is far from the initial result. (d) The stabilized rope net when fractional force exists.

Interesting results are observed when the input models are symmetric. The rope nets tend to be also symmetric. However, our current approach does not explicitly guarantee this property. We also believe that this would be a future work of our algorithm.

ACKNOWLEDGMENTS

We thank all the reviewers for their valuable comments and suggestions. Thanks for the models from the Thingi10K, the McGill 3D Shape Benchmark, and the AIM@SHAPE Shape Repository.

REFERENCES

Abaqus. 2018. Abaqus. Retrieved July 31, 2021 from http://www.feasol.com.

Tobias Achterberg. 2009. SCIP: Solving constraint integer programs. *Math. Program. Comput.* 1 (2009), 1–41.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.* 32 (2013), Article 98, 12 pages.

David Bommes, Timm Lempfer, and Leif Kobbelt. 2011. Global structure optimization of quadrilateral meshes. *Comput. Graph. Forum* 30 (2011), 375–384.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Cláudio T. Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-Mesh generation and processing: A survey. *Comput. Graph. Forum* 32 (2013), 51–76.

David Bommes, Tobias Vossemer, and Leif Kobbelt. 2008. Quadrangular parameterization for reverse engineering. In *Proceedings of the 7th International Conference on Mathematical Methods for Curves and Surfaces*. 55–69.

J. A. Bondy and U. S. R. Murty. 1976. *Graph Theory with Applications*. Elsevier. https://doi.org/10.1007/978-1-349-03521-2_8

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 11 (2001), 1222–1239.

Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph.* 31 (2012), Article 110, 11 pages.

Marcel Campen and Leif Kobbelt. 2014. Dual strip weaving: Interactive design of quad layouts using elastica strips. *ACM Trans. Graph.* 33 (2014), Article 183, 10 pages.

Keenan Crane, C. Weischedel, and M. Wardetzky. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.* 32 (2013), Article 152, 11 pages.

Mathieu Desbrun, Eva Kanso, and Yiying Tong. 2008. *Discrete Differential Forms for Computational Modeling*. Vol. 38. Birkhauser, 287–324.

Tamal K. Dey, Jian Sun, and Yusu Wang. 2010. Approximating loops in a shortest homology basis from point data. In *Proceedings of the 26th Annual Symposium on Computational Geometry*. 166–175.

Rosen Diankov, Siddhartha S. Srinivasa, Dave Ferguson, and James J. Kuffner. 2008. Manipulation planning with caging grasps. In *Proceedings of the 8th IEEE-RAS International Conference on Humanoid Robots (Humanoids'08)*. 258–292.

Thomas Eiter and Heikki Mannila. 1994. Computing discrete Fréchet distance. arXiv:1204.5333. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.90.937.

Jianhui Fu, Jaedeuk Yun, Yoongho Jung, and Deugwoo Lee. 2017. Generation of filament-winding paths for complex axisymmetric shapes based on the principal stress field. *Comp. Struct.* 161 (2017), 330–339.

Jingwei Huang, Yichao Zhou, Matthias Nießner, Jonathan Richard Shewchuk, and Leonidas J. Guibas. 2018. QuadriFlow: A scalable and robust method for quadrangulation. *Comput. Graph. Forum* 37 (2018), 147–160.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Trans. Graph.* 34 (2015), Article 189, 15 pages.

Zhongping Ji, Ligang Liu, and Yigang Wang. 2010. B-Mesh: A fast modeling system for base meshes of 3D articulated shapes. *Comput. Graph. Forum* 29, 7, 2169–2177.

Dave Johnson. 2016. *Hoisting and Rigging Safety Manual*.

David J. Ketchen and Christopher L. Shook. 1996. The application of cluster analysis in strategic management research: An analysis and critique. *Strat. Manage. J.* 17, 6, 441–458.

Lingchen Kong, Chuanqi Qi, and Hou-Duo Qi. 2019. Classical multidimensional scaling: A subspace perspective, over-denoising, and outlier detection. *IEEE Trans. Signal Process.* 67 (2019), 3842–3857.

Tsz-Ho Kwok, Weiwei Wan, Jia Pan, Charlie C. L. Wang, Jianjun Yuan, Kensuke Harada, and Yong Chen. 2016. Rope caging and grasping. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA'16).* 1980–1986.

Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. 1999. *Fast Proximity Queries with Swept Sphere Volumes.* Technical Report TR99-018. Department of Computer Science, UNC Chapel Hill.

Jian Liu, Shiqing Xin, Zengfu Gao, Kai Xu, Changhe Tu, and Baoquan Chen. 2018. Caging loops in shape embedding space: Theory and computation. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA'18).* 1–5.

Satoshi Makita and Weiwei Wan. 2017. A survey of robotic caging and its applications. *Adv. Robotics* 31 (2017), 1071–1085.

Giorgio Marcias, Kenshi Takayama, Nico Pietroni, Daniele Panozzo, Olga Sorkine-Hornung, Enrico Puppo, and Paolo Cignoni. 2015. Data-driven interactive quadrangulation. *ACM Trans. Graph.* 34 (2015), Article 65, 10 pages.

Nets4You. 2019. Hoisting and Lifting Nets. Retrieved July 31, 2021 from https://www.nets4you.com/hoist-and-lifting-nets/.

Vishal P. Patil, Joseph D. Sandt, M. Kolle, and J. Dunkel. 2020. Topological mechanics of knots and tangles. *Science* 367 (2020), 71–75.

Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Roberto, and Paolo Cignoni. 2016. Tracing field-coherent quad layouts. *Comput. Graph. Forum* 35 (2016), 485–496.

Florian T. Pokorny, Johannes A. Stork, and Danica Kragic. 2013. Grasping objects with holes: A topological approach. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation.* 1100–1107.

Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. 2015. Perfect matching quad layouts for manifold meshes. *Comput. Graph. Forum* 34 (2015), 219–228.

Alberto Rodriguez, Matthew T. Mason, and Steve Ferry. 2011. From caging to grasping. *I. J. Robotics Res.* 31 (2011), 886–900.

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Trans. Graph.* 34 (2015), Article 170, 14 pages.

Andrew O. Sageman-Furnas, Albert Chern, Mirela Ben-Chen, and Amir Vaxman. 2019. Chebyshev nets from commuting PolyVector fields. *ACM Trans. Graph.* 38 (2019), Article 172, 16 pages.

Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. 2007. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Visual Comput.* 24 (2007), 249–259.

Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven J. Dickinson. 2007. Retrieving articulated 3-D models using medial surfaces. *Mach. Vision Appl.* 19 (2007), 261–275.

S. Skiena. 1990. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.* Basic Books.

Johannes A. Stork, Florian T. Pokorny, and Danica Kragic. 2013. Integrated motion and clasp planning with virtual linking. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 3007–3014.

Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph.* 32 (2013), Article 97, 8 pages.

Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011a. Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph.* 30 (2011), 142.

Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011b. Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–12.

Julien Tierny, Joel Daniels, Luis Gustavo Nonato, Valerio Pascucci, and Cláudio T. Silva. 2012. Interactive quadrangulation with Reeb atlases and connectivity textures. *IEEE Trans. Visual. Comput. Graph.* 18 (2012), 1650–1663.

Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. 2006. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing.* 201–210.

Francesco Usai, Marco Livesu, Enrico Puppo, Marco Tarini, and Riccardo Scateni. 2015. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Trans. Graph.* 35 (2015), Article 6, 13 pages.

US Netting. 2019. Cargo Lifting. Retrieved July 31, 2021 from https://www.usnetting.com/cargo-netting/cargo-lifting-nets/.

Weiwei Wan, Rui Fukui, Masamichi Shimosaka, Tomomasa Sato, and Yasuo Kuniyoshi. 2012. Grasping by caging: A promising tool to deal with uncertainty. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation.* 5142–5149.

Weiwei Wan, Rui Fukui, Masamichi Shimosaka, Tomomasa Sato, and Yasuo Kuniyoshi. 2013. A new 'grasping by caging' solution by using eigen-shapes and space mapping. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation.* 1566–1573.

Weiwei Wan, Boxin Shi, Zijian Wang, and Rui Fukui. 2020. Multirobot object transport via robust caging. *IEEE Trans. Syst. Man. Cyber.: Syst.* 50, 1 (2020), 270–280.

Dmitry Zarubin, Florian T. Pokorny, Marc Toussaint, and Danica Kragic. 2013. Caging complex objects with geodesic balls. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 2999–3006.

Sen Zhang, Hui Zhang, and Jun-Hai Yong. 2015. Automatic quad patch layout extraction for quadrilateral meshes. *Comput.-Aided Design Appl.* 13 (2015), 1–8.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A dataset of 10,000 3D-printing models. arXiv:abs/1605.04797.

Received November 2020; revised June 2021; accepted July 2021