Recovering Functional Mechanical Assemblies from Raw Scans

Minmin Lin, Tianjia Shao, Youyi Zheng, Niloy J. Mitra, Kun Zhou

Abstract—This paper presents a method to reconstruct a functional mechanical assembly from raw scans. Given multiple input scans of a mechanical assembly, our method first extracts the functional mechanical parts using a motion-guided, patch-based hierarchical registration and labeling algorithm. The extracted functional parts are then parameterized from the segments and their internal mechanical relations are encoded by a graph. We use a joint optimization to solve for the best geometry, placement, and orientation of each part, to obtain a final workable mechanical assembly. We demonstrated our algorithm on various types of mechanical assemblies with diverse settings and validated our output using physical fabrication.

Index Terms—3D scanning, mechanical assembly, functionality, mechanical constraints, motion

1 Introduction

Nowadays, obtaining high-quality 3D geometry of a target object has become an easier and easier task with the fast development of acquisition devices. For objects with complex geometry (e.g., mechanical assemblies), however, the scan data remain difficult to use. A fundamental problem is the *loss of functionality* during the scanning, which limits the usability and further applications of the scans, such as fabrication and redesign.

Individual components of a mechanical assembly are typically manufactured with precise parameters, and are intentionally designed for particular functions, such as gears and drivers for mechanical motion generation. Thus, geometry and functionality are essentially coupled in such forms of mechanical objects. That is, the functionality affects the explicit shape and the inter-relations among geometric parts which on the contrary are very hard to recover with the geometry alone. This leads to the reconstruction of a *functional mechanical object* essentially an ill-posed problem. What's worse, inevitable noise and region missing due to occlusion would easily damage the function. For example, data are usually bad near boundaries, and regions are occluded where gears interact (e.g., the worm gear drive arrangement in Figure 12), which indicate that scans are often bad where they matter most.

Traditionally, in order to recover functional mechanical tools from the scan data, popular reverse engineering softwares such as SolidWorks provide interactive tools to convert low-level geometry (e.g., a point cloud) to a parameterized CAD model, requiring users to manually segment the scan data into different parts and locally adjust the part parameters. Such local methods can be unreliable for mechanical tools, which have stringent restrictions (e.g., meshing, coaxial, orthogonal) among parts to ensure workability. Even small inevitable geometric errors due to noise and occlusion would easily break the global restrictions. For example, as shown in Figure 1(a), a small position offset breaking the placement constraint would make a gear disconnect with another in a gear train after fabrication. Another example in Figure 1(b) shows that the locally reconstructed worm gear fails to mesh with the worm because the mechanical parameters do not satisfy the meshing constraint, leading to an interruption in the mechanical motion transmission.

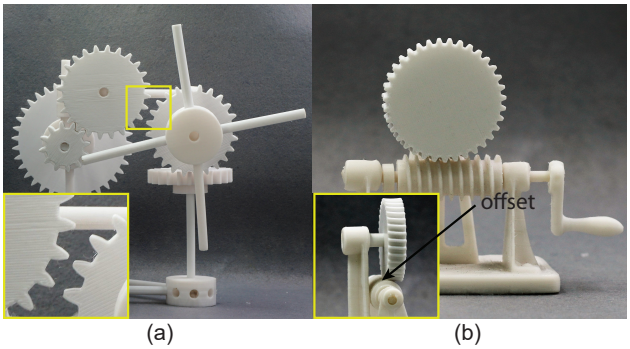


Fig. 1. (a) shows that a small position offset breaking the placement constraint would make a gear disconnect with another in a gear train after fabrication; (b) shows that the locally reconstructed worm gear fails to mesh with the worm because the mechanical parameters do not satisfy the meshing constraint.

In this paper, we present a framework to automatically recover functional mechanical assemblies from the scan data using a *coupled analyze-and-reconstruct* algorithm. We leverage geometry to infer the internal functionality of individual shape parts, and subsequently use the inferred functionality to reconstruct functional parts, whose assembly enables a global functional mechanical object. We make the key observation that, for functional objects such as mechanical assemblies, although the general functionality is lost, the functionality of an individual part can be implicitly inferred from its geometry if we know some priors such as the part type and its interaction relations with other parts; then the global functionality can be inferred from these individual

- *M. Lin, T. Shao, and K. Zhou are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: tianjiashao@gmail.com, kunzhou@acm.org*
- *Youyi Zheng is with ShanghaiTech University. E-mail: zhengyy@shanghaitech.edu.cn*
- *Niloy J. Mitra is with University College London. E-mail: n.mitra@cs.ucl.ac.uk*

functional parts and the interaction relations. This is the key to our method. The goal is to convert the raw imperfect acquisition, possibly coupled with geometric noise and missing data, to a parameterized functional mechanical model with semantic parts (e.g., gears, cranks, cams). These parameterized parts must geometrically conform to the input data, while globally satisfy the mechanical restrictions to ensure workability.

The challenges, as we discussed, lie in three folds. Since the input is raw point clouds, we do not have any prior knowledge about the underlying geometry: What are the parts? How are they inter-related and what are the mechanical restrictions? What parameters to fit to the parts so that the parameterized model is mechanically functional as the physical model? It makes the problem more troublesome when the raw scanned input data contain erroneous sampling or large missing regions. Thus our task is to solve, essentially, the problems of (i) extracting mechanical parts from the raw data; (ii) inferring the initial part parameters; (iii) extracting global mechanical restrictions among parts; and (iv) optimizing the part parameters under mechanical constraints.

Our algorithm starts with raw scans under different motion configurations (Figure 2 (a)). We introduce a novel part segmentation algorithm based on motion variations. That is, we take multiple scans of a mechanical assembly with different motion configurations, where each configuration represents one state of the mechanical parts moving from the rest pose. We exploit a hierarchical registration strategy to robustly find consistent motions among configurations. Then a Markov Random Field (MRF) model is used to efficiently solve the segmentation problem.

After segmentation, our algorithm estimates initial mechanical parameters of the segmented parts. The initial parameters are unstable, as the segmented mechanical parts are often noisy and even incomplete (c.f. [1]). We take a global approach to constrain and optimize the local fitted parameters. In a key step, we infer valid mechanical constraints from the input data by verifying with a graph-based forward motion propagation to check the motion correctness, followed by a nonlinear data optimizer to minimize the fitting error between the parameterized shape and the input scans, subject to the inferred mechanical constraints. Finally, a functional mechanical model is recovered with each part moving correctly as the originally scanned object.

We tested our algorithm on a variety of scanned mechanical assemblies, which are typical mechanical tools like the gear train, the worm gear drive arrangement, the piston mechanism and so on. Experimental results show that the proposed approach can robustly recover functionality mechanical assemblies from raw scan data with geometric errors and missing data. The reconstructed functional model largely eases the reproduction of the original tool and can be used for further illustration or fabrication, as shown in Figure 12. In summary, our contributions are:

- We propose a novel motion-guided, patch-based hierarchical segmentation algorithm for extracting functional parts from raw scans;
- We formulate the mechanical constraints as parametric constraints and solve the part parameters using a joint optimization;
- We develop a unified framework that simultaneously couples the part parameter localization and the functionality recovery.

2 Related work

Our work is closely related to the problem of surface reconstruction, which has achieved substantial progress in the past two decades. Although a variety of methods have been proposed [2], surface reconstruction from a scanned, unstructured point cloud is still difficult and ill-posed, especially when dealing with structural recovering [3]. A full review of previous surface reconstruction methods is beyond the scope of our paper. We refer interested readers to the excellent survey of [4] and [2]. Most prevailing surface reconstruction methods rely on shape continuity to reconstruct or complete the underlying surface using smooth interpolation or extrapolation [5] [6]. In contrast to these works, our method targets one step further: we aim at recovering the underlying geometry along with the high-level functionality of the input scans of mechanical assemblies. Our input is point clouds taken under varying motion configurations, which typically exhibit complex geometry and missing regions.

Mechanical assemblies often contain certain canonical geometric properties such as coplanar faces, orthogonal faces and so on due to aesthetic considerations and a variety of practical constraints. The detection of simple geometric structures has shown to be particularly helpful for de-noising and filling in missing data [2]. 3D fitting of primitive shapes have been proposed in reverse engineering [7] [8] [1] [9] [10] and references therein. These methods support reconstruction of simple or regular shapes consisting of sharp features through local fitting of surface geometry. The primitive-based methods often require a reliable initial estimate to ensure the method not to degrade [2] (e.g., when certain portions of the shape are poorly explained by a primitive). Our work is largely inspired by the work of Li et al. [1] which takes a global approach, accounting for both local and global inter-primitive relations, to constrain and optimize the local RANSAC based primitives. The difference is that we do not use simple primitives but parametric models instead, and leverage motion analysis to extend our algorithm to the functionality recovery of complex mechanical tools.

Our work is also inspired by recent works on functionality recognition and analysis [11], [12], [13], [14], [15]. Xu et al. [11] performed slippage analysis over contact surfaces to segment and categorize joints in man-made objects, and used the information for interactive volumetric-based space deformation. Guo et al. [12] analyzed the individual parts using sharp edge loops and extracted the contact faces between each pair of neighboring parts and then clustered the sets of the individual parts into meaningful sub-assemblies used for a hierarchical decomposition. Hu et al. [13] introduced the *contextual descriptor* and showed that the contextual object-to-object interactions are effective to exploit object functionality. Mitra et al. [14] analyzed the interactions and motions of mechanical parts from contact detection and relations between part axes. The input of this approach is required to be clean and with no data loss.

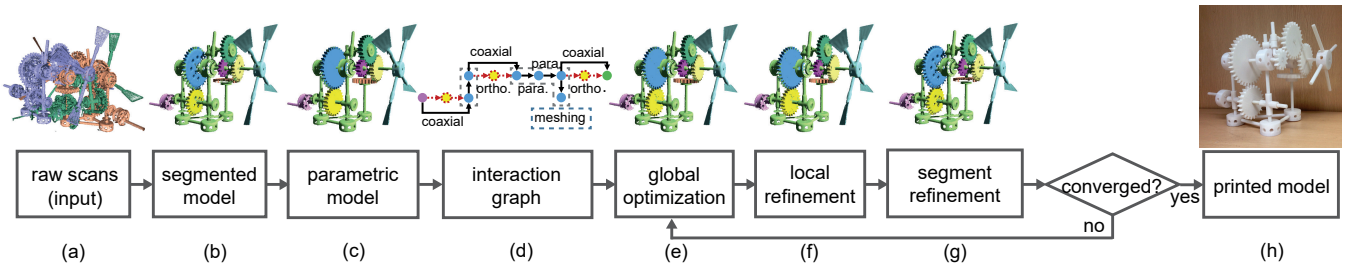


Fig. 2. The pipeline of our algorithm. Our algorithm starts with raw scans under different motion configurations (a). We first extract functional mechanical parts using a hierarchical motion-guided, patch-based registration and segmentation algorithm (b) and then estimate the parametric model based on the segments (c). We take a global approach (e) to constrain and optimize the local fitted parameters based on relations encoded in an interaction graph (d). After that, parameters of the parametric model are locally refined to approximate the geometry of scans without breaking the previous constraints (f). The segments are then refined based on the parametric model (g). We iterate the process until convergence. At last, the parametric model is validated using physical fabrication (h).

However, in real scan data, many critical contact regions are not able to be captured due to occlusion. We have to automatically complete the missing regions to make the mechanical tool work as expected.

It is noteworthy that in the work of assembly-based mechanical modeling [15] [16], the position and orientation of parts are determined according to the constraints governing their mating or alignment. Symbolic, rule, and graph-based approaches construct the geometry constraint equations which are then solved via numerical optimization. Zhu et al. [15] generated a physically-realizable assembly from a description of the target motion. They used a higher-level abstraction by automatically selecting the shape from a small set of simplified parametric models according to a priori knowledge of motion transmission in mechanisms, and refined the initial selection by optimizing over both discrete and continuous variables. Unlike their work, our aim is to recover the geometry of the scans and in the meanwhile estimate its target motion. Our approach does not require a large series of time-varying motion configurations and does not use pre-parameterized models to represent the scanned parts.

Our work also shares resemblance to the large body of methods on structure-based shape analysis and modeling [17], [18], [19], [20], [21], [22], [23], wherein high-level shape structures are identified with low-dimensional parts of different sizes, positions, and interactions. Gal et al. [17] demonstrated that working with a set of 1D feature curves extracted from engineered objects, and preserving their intra- and inter-relations while deforming the shapes lead to an intuitive manipulation framework. Shen et al. [19] recovered structures by combining existing labeled parts from different objects, considering both partial matching with the acquired data and the interactions between parts. Laga et al. [21] exploited the context of structural relationships between shape parts and use them, in conjunction to their geometry, as cues for functionality recognition. Alhashim et al. [23] presented a system to interpolate new shapes from two database shapes by decomposing input shapes and recombining individual parts according to constraints deduced through the structure analysis. Our method also relates shape parts with their interactions. However, as a key difference, our parts and their interactions are not known as priori and are hierarchically recovered under a top-down motion analysis coupled with geometry optimization.

3 Overview

Our goal is to reconstruct functional mechanical assemblies from raw scans. Thus workability is the main objective, for which the key is to fulfill a set of mechanical constraints. Such mechanical constraints (e.g., gear meshing) are the presupposition to our algorithm, with which the parts and the relations can then be progressively recovered, parameterized and optimized.

Figure 2 shows the pipeline of our method. Our algorithm has 4 main steps. In the first step, we identify the mechanical parts by analyzing the motion of multiple scans (Section 4.1). This is a non-trivial task, as the motion of the mechanical assembly might exhibit large variations from one configuration to another. For example, multiple parts may move simultaneously (see in Figure 2 and 3). Unlike previous segmentation methods [24] which used motions of articulated shapes where the correspondences among different shapes are known, we do not have the correspondence information between different scans. We propose a simultaneous registration and segmentation algorithm to decouple functional parts one by one. We employ a patch-based registration algorithm to align the scans to distinguish moving parts. Since the scans have parts with different motions, considering all patch correspondences as traditional ways will easily produce wrong registration (Figure 5(a)). We register the scans hierarchically, ignoring less-confident correspondences first and reconsidering them in the next level (Figure 5(b)).

Second, in the parametrization stage (see Section 4.2), we infer the individual part type by fitting the part with a set of typical 3D template part models in our database (e.g., gears, cranks, etc.). We then parameterize the segmented parts using corresponding parameters for different mechanical types, as described in the appendix. Third, after the parametrization, in a key stage, our system infers the potential inter-relations among individual parts (see Section 4.3). Pairs of parts that are in contact or have orientation/placement relations with each other are considered and their interaction relations are analyzed and encoded into an interaction graph. Fourth, based on the interaction graph, a global optimization algorithm is conducted to ensure proper motion of parametric models to enable the global functionality of the mechanical assembly. The global optimization takes account of both the individual shape geometry and the mechanical

constraints among different parts that have interactions (see details in Section 4.4). In Section 4.5, the parametric model is further locally optimized without breaking the previous constraints. Finally, based on the parametric models, the initial segments are locally refined (Section 4.6) and the entire process is iterated until the local refinement does not affect much on the global optimization.

4 Algorithm

4.1 Motion-based hierarchical part segmentation

The input of our algorithm is a set of raw scans $S = \{S_1, S_2, ..., S_k\}$ of a mechanical tool, which are expected to be under different motion configurations. The scans are initially in the form of point clouds, and then we convert them to polygon meshes using the method of [25]. To recover the functional mechanical assembly from several raw scans, it is necessary to identify all the functional parts that are involved in the motion of the mechanical tool. As far as we know, purely geometric-based segmentation or co-segmentation methods [26], [27] are not sufficient for our purpose, which often involve training processes in order to account for semantics. On the other hand, as our input is raw scans under different motion configurations, which typically inhabit no correspondence information, thus deformation-based segmentation methods [24], [28] also do not work for our case.

Our key observation is that the motion of each functional part in one mechanical tool usually differs from each other, especially between two adjacent parts. This inspires us to devise a motion-based hierarchical segmentation algorithm, which extracts moving parts in a coarse-to-fine manner. To be specific, to correctly analyze the motion difference, we need to first align the scans so that their static parts stay in correspondence (Figure 4), and parts under motion are then identified iteratively. To identify common static parts, we employ a patch-based registration method and separate the models into two parts: moving part A and static part B. The moving part A is further taken into the above process to identify sub- "moving" and "static" parts and the process stops until no moving parts can be further identified. Since our input data contain part movements, traditional point-based registration methods [29] might be unreliable in our case (see the comparison result in Figure 4). Our observation is that manmade objects are often composed of many planar regions for the sake of aesthetics and cost, and large planes are more stable for registration. Hence we adopt a patch-level approach for registration. After registration, the patches are labeled with an MRF formulation to solve the segmentation problem. Figure 3 shows the pipeline of the hierarchical patch registration and segmentation.

4.1.1 Patch generation

For each raw scan S_i, we first over-segment it into patches $P_i = \{p_i^1, p_i^2, ...\}$ using region growing. Specifically, each time we start with a seed face which is randomly selected. Then we grow the region by adding its neighboring faces. A neighboring face is added to the current region if the angle between the face normal and the seed face normal is below a threshold β. We stop growing until no more faces can be added and the process is repeated until all faces are grouped. For each patch p, we calculate its total area of faces a_p, and compute patch center $\mathbf{c}_p$ and normal $\mathbf{n}_p$ using principal component analysis (PCA). Note that there may exist many tiny patches, which will be unstable for registration and costly for labeling. We adopt a post-process strategy by merging such tiny patches to their neighboring patch if the angle between their normals is below a threshold γ (γ is usually set to be equal to β in our experiments). An example of generated patches is shown in the insets of Figure 4.

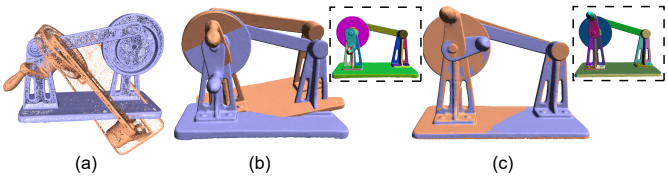


Fig. 4. Patch-based registration. (a) shows the initial position of scan 1 and 2 of the cam mechanism; (b) shows the registration result with the state-of-the-art method Super 4pcs [29]; (c) is our registration result based on patches. The generated patches are shown in the insets.

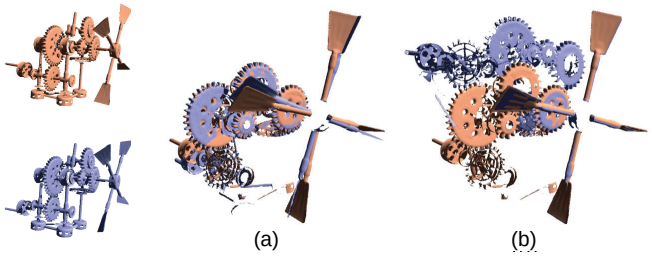


Fig. 5. For the registration of mechanical tools with moving parts, using the overall pairwise error between patches as the matching score will produce an undesired result as shown in (a). Instead, we get the desired result as shown in (b) by filtering outlier patches that are far away from the target scan.

4.1.2 Patch-based registration

Given two scans S_i and S_j to be registered, a patch pair $\langle p, q \rangle$ is called a potential patch correspondence $p \sim q$ if they have similar patch areas, where patch p and patch q belong to S_i and S_j respectively. We consider $p \sim q$ if $|a_p - a_q|/|a_p + a_q| < \alpha$ ($\alpha = 0.2$ in our experiments). A pair of patch pairs $\langle (p_i, p_j), (q_s, q_t) \rangle$ is called a match if both $\langle p_i, q_s \rangle$ and $\langle p_j, q_t \rangle$ are potential patch correspondences. For each match $\langle (p_i, p_j), (q_s, q_t) \rangle$, we estimate the rigid transformation that maps the positions and normals of (p_i, p_j) to the corresponding positions and normals of (q_s, q_t).

All these pair-wise candidate transformation matrices form a large transformation space. Our task is to search for one best transform matrix T_{ij} such that S_i and S_j are registered in terms of minimizing their patch-to-patch registration error. We first employ a strategy which is similar to the voting procedure presented in [30] that takes all patch correspondences into consideration to efficiently find approximate best matches. The difference is that they use corresponding point pairs to estimate the rigid transformation while we use patch pairs instead. Besides, they directly select the match with the highest voting score as the best match, but in our case, approximate voting scores may be inaccurate for input data with moving parts. Thus we keep

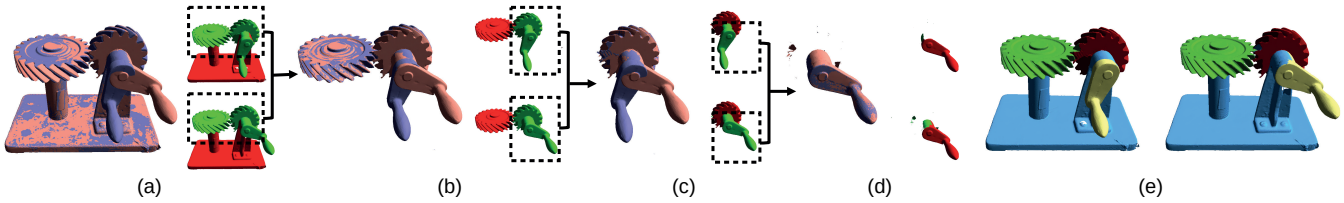


Fig. 3. Hierarchical segmentation pipeline, from the first level (a) to the fourth level (d). The left images in (a) to (d) show the patch-based registration results, and the right images in (a) to (d) show the labeling results, where the red color suggests the same motion at current level while green color suggests different motions. (e) shows the final segmented model.

top-K ($K = 5$) transformations rather than the one with the largest vote, and then refine them using the iterative closest point (ICP) algorithm [31]. Finally, to select the best match, simply using the overall pairwise registration error as the match score may not be a wise choice, because the error in the least square manner is often undesired, as shown in Figure 4. We define the match score as follows: we first disregard those matched patches whose matching distance is larger than a given threshold d_ε, and then compute the total matched area for the patches. The matched area for a pair of patches (p, q) is defined as $area(p \cap q)$, here $\cap$ denotes the overlap region. The transformation which leads to the largest matched area is selected as the best match.

4.1.3 Patch labeling

Next, we device a Markov Random Field (MRF) algorithm to separate the moving parts from the static parts. Essentially, our task is to assign each patch p_i^s a label $l_s \in \{0, 1\}$, where 0 indicates the patch belonging to a static part while 1 indicates the patch belonging to a moving part. Please note that static has its meaning of being "unmoving" only in the context of the current level of hierarchy (c.f. Section 4.1).

We measure the confidence of a patch being static by considering its overlapping ratio to its corresponding patch. The more overlapping region it has, the more likely it belongs to a static part. Hence, the probability of a patch p_i^s in scan i to be static (i.e., labeled 0) is defined as $prob(p_i^s, S_j)$. More specifically,

$$prob(p_i^s, S_j) = \frac{1}{|f \in p_i^s|} \sum_{f \in p_i^s} (1 - e(f, S_j)), \tag{1}$$

where

$$e(f, S_j) = \begin{cases} d(f, S_j)/d_\varepsilon & if\ d(f, S_j) \le d_\varepsilon, \\ 1 & otherwise. \end{cases}$$

Here f is a face belonging to patch p_i^s and $d(f, S_j)$ is the distance from the barycenter of face f to the nearest face in S_j. d_ε is the aforementioned distance threshold.

The MRF energy function for a possible labeling L is defined as the following term:

$$\underset{L(p_i^s)}{\operatorname{argmin}} \sum_{p_i^s} D(L(p_i^s)) + \sum_{\{p_i^s, p_i^r\}} \lambda W(L(p_i^s), L(p_i^r)). \tag{2}$$

Here $L(p_i^s)$ is the label of p_i^s, $D(L(p_i^s))$ is a data term which is defined as:

$$\begin{aligned} D(L(p_i^s) \to 0) &= 1 - prob(p_i^s, S_j^t), \\ D(L(p_i^s) \to 1) &= prob(p_i^s, S_j^t). \end{aligned} \tag{3}$$

$W(L(p_i^s), L(p_i^r))$ is a smooth term to penalize neighboring patches being assigned different labels:

$$W = \begin{cases} \exp\left(-\frac{(a_s - a_r)^2}{2\sigma_a^2} - \frac{(\arccos(\mathbf{n}_s \cdot \mathbf{n}_r))^2}{2\sigma_\theta^2}\right), & \text{if } L(p_i^s) \neq L(p_i^r). \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where a_s and $\mathbf{n}_s$ are the area and normal of p_i^s. σ_a is the difference between the largest patch area and the average patch area in current level, and σ_θ is $\pi/2$ in our implementation. Here the smooth term is crucial for the labeling of planes which share similar motion status. For example, as shown in Figure 6, the center planar region of a gear would be labeled as "static" if no smooth term is added. However, because a large number of its neighboring gear teeth are labeled as moving, the center plane is labeled as "moving" with our MRF formulation.

The labeling process divides each scan S_i into two parts: a part P_s^i which is static at current level(i.e., labeled as 0) and a part P_m^i that moves at current level (i.e., labeled as 1). Please note that P_m might still contains multiple moving parts (Figure 3). To further separate them, we continue the aforementioned patch-based registration and segmentation steps using the part P_m as input and subsequently obtain another binary separation of P_m. This process stops when such a P_m is not found. In our experiments, we stop when the matched triangle number exceeds 90% of the total mesh triangle number in the current level.

Fig. 6. Labeling results with/without smooth term: given two scans after registration (a), the center large plane of the gear is labeled as "static" (shown in red) if no smooth term is added (b). With smooth term, the neighbors (gear teeth) propagate the "moving" label (shown in green) to the center plane.

4.2 Parametric model estimation

The segmented parts from raw data are noisy, inaccurate and usually incomplete due to occlusion (as shown in Figure 7), so they cannot work properly if they are directly fabricated. In this step, our goal is to convert the raw segments to parameterized parts, which will be optimized later to produce the same functionality as the input object.

We first identify the semantic type for each part by matching the part with a set of typical 3D template part models in our database, including crank, cam, worm, spur gear, driving part, etc. We adopt a similar strategy as in [19] to perform shape matching between the template and the part. The normals of their largest planes are firstly aligned. Then the template is translated and scaled to fit into the bounding box of the part. We exhaustively search for the orientation of the template around the aligned normal, to estimate the best match that minimizes the sum of distances between corresponding points on the template and the part, which is further refined by an ICP method. The part type is determined as the type of the template with the best matching score.

Given the part type, our next task is to estimate the specific mechanical parameters for the part. As the parametric models for different mechanical parts vary from type to type, for the clarity of exposition, we focus on spur gears as the illustrating example throughout the following sections. The parametrizations of other types of mechanical assembly are presented in the appendix.

For a spur gear, to estimate the part parameters, we first project the part onto the 2D plane (see Figure 7) which is perpendicular to the gear axis. We extract the outside contour of the gear and then detect points on both the root circle and the outside circle. Then a rough tooth number is estimated with the root diameter and outside diameter, which is further refined by matching one tooth to all other teeth. Please refer to the appendix for more details.

With the tooth number and rough outside diameter and root diameter, we estimate initial gear parameters with the standard formula used in gear manufacture [32]. Specifically, we wish to optimize the gear center, its rotation angle, together with the tooth thickness, outside diameter, root diameter and gear thickness so that the optimized parametric model geometrically fits the input scan. To this end, we use a gradient decent method to optimize the parameters first in 2D space (gear center, rotation angle, tooth thickness, outside diameter and root diameter, see Figure 7) and then optimize the gear thickness in 3D. Figure 7 illustrates the algorithm in 2D. Our method is able to extract the correct parameters under noise and data loss.

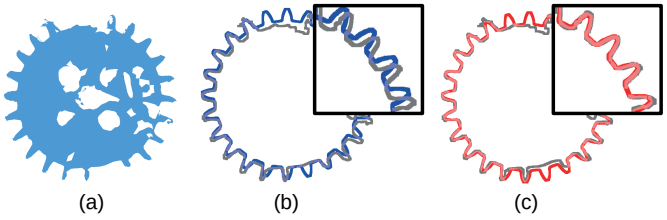


Fig. 7. Contour comparison between the scan data and the parametric model of a spur gear before and after local optimization. The segmented gear from raw scan (a), which is usually incomplete, is projected to the plane perpendicular to gear axis. The extracted contour of the gear is displayed with gray color in (b) and (c). The blue curve in (b) is the contour of initial parametric model while the red curve in (c) is the contour of parametric model after optimization in 2D plane.

4.3 Interaction graph

The estimated local parametric parts need to be globally optimized such that their arrangements form a physically valid configuration to enable mechanical function. To this end, we first propose a graph-based forward motion propagation algorithm to identify parts interactions as well as wrongly estimated or missing parts and then jointly optimize the part parameters to obtain a physically workable mechanical assembly.

To infer the part interactions, we build an interaction graph $G := \{V, E\}$ where V denotes functional parts and E denotes a set of edges that link pairs of functional parts with interaction. The nodes involve common types of the mechanical parts including spur gear, helical gear, worm, worm gear, crank, cam, slider, driver, end effector and rod. The edges are attributed with four interaction types: meshing, parallel, coaxial and orthogonal.

We first detect contacts between parts as in [14] with the distance threshold set to $5mm$. Then meshing relations are built between contact parts whose types are among the following combinations: spur gear and spur gear, helical gear and helical gear, worm and worm gear as well as cam and its follower. And parallel, coaxial and orthogonal relations among two parts are established by evaluating the relations between the part axes $\mathbf{n}_1$ and $\mathbf{n}_2$ and part centers $\mathbf{c}_1$ and $\mathbf{c}_2$:

- *Parallel:* $\langle \mathbf{n}_1, \mathbf{n}_2 \rangle \leq \frac{\pi}{12}$.
- *Coaxial:* $\langle \mathbf{n}_1, \mathbf{n}_2 \rangle \leq \frac{\pi}{12}$, $\langle \mathbf{c}_1 - \mathbf{c}_2, \mathbf{n}_1 \rangle \leq \frac{\pi}{12}$, $\langle \mathbf{c}_1 - \mathbf{c}_2, \mathbf{n}_2 \rangle \leq \frac{\pi}{12}$.
- *Orthogonal:* $\frac{\pi}{2} - \langle \mathbf{n}_1, \mathbf{n}_2 \rangle \leq \frac{\pi}{12}$.

The motion parameters of each functional part, i.e., the rotation center, rotation axis and translation direction, are calculated from the transformation matrix we get in the registration stage.

In many cases, the inferred interaction graph is disconnected. This is because the scanned data often contain missing parts due to occlusion (see Figure 13), or the rotation rods are visually "static" during motion and cannot be segmented with the algorithm (e.g., in Figure 8 the "static" rods are not segmented). To infer such missing rods and correct the interaction graph, we perform a forward motion propagation which operates in the following manner: starting from a driving part, we trigger a motion on it and propagate the motion to the rest of the parts by traversing the interaction graph along the edges representing contacts. By examining the existence of loops and disconnected components of the graph during the traversal, we identify wrongly estimated motion interactions (e.g., a loop in the graph) or missing parts (e.g., where the motion fails to propagate). For missing parts, we fill in parametric rods. Rods may be filled between two coaxial parts or between two mechanical parts based on their types and relations. For example, the parallel relation will infer the existence of a missing rod between a cam and a slider or a crank and a slider if they are close to each other. For loops, we try breaking possible loop edges and restart the propagation to see if the motion can be propagated properly (with the missing parts filled in). Figure 8 shows an interaction graph for the windmill model. For the purpose of clarity, we only show relations built between two meshing parts and between two parts with missing joints.

4.4 Global optimization

Though the parametric parts approximate the geometry

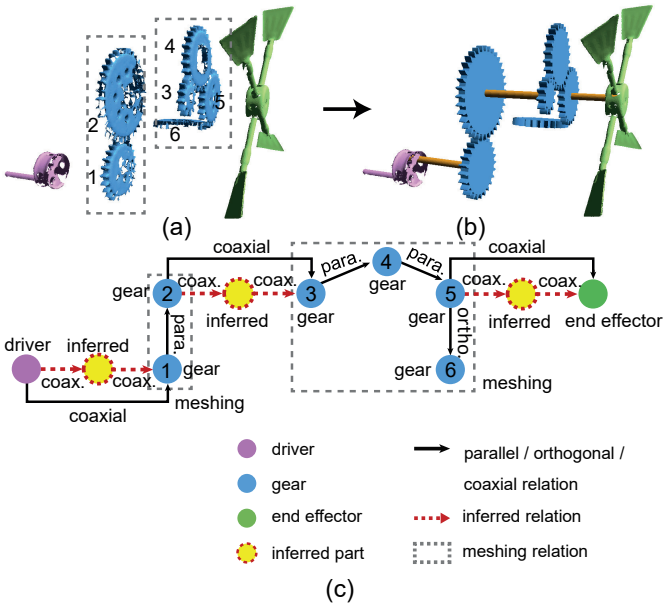


Fig. 8. Our algorithm automatically infers missing rods between coaxial parts and validates the inference with the help of interaction graph. (a) The segmented model which is disconnected because "static" rods are not detected with our algorithm. (b) The parametric model with missing parts (orange rods) filled. (c) The interaction graph of the windmill model.

of the scanned mechanical parts, they typically can not ensure a global workability of the entire model. With the interaction graph, there are various global relations that individual parts must satisfy, to ensure the workability of the mechanical tool.

We employ a global joint optimization to integrate the global relations with geometric optimization. Similar to [1], we first decouple the global optimization process into three independent stages to avoid the conflict of various nonlinear constraints. In particular, we enforce three classes of commonly encountered relations in mechanical tools that will affect the assembly: (i) mechanical constraints between specific mechanical parts; (ii) orientation relations like parallelism and orthogonality; (iii) placement relations like coplanarity and coaxiality. During the optimization, we first enforce the orientation relations, followed by the placement relations, and finally the mechanical constraints. This is because the mechanical adjustment does not affect the adjustment of orientation and placement, and thus will not break the already satisfied relations as in [1].

Before we go into details of the global optimization, we first define a distance metric used for the global optimization which measures the distance between the parameterized part and the scan. In particular, given a scanned part S_i and its parameterized part M_i, we measure the data fitting error as:

$$E_d(S_i, M_i) = \frac{1}{|s \in S_i|} \sum_{s \in S_i} d(s, M_i)^2 + \frac{1}{|m \in M_i|} \sum_{m \in M_i} d(m, S_i)^2, \tag{5}$$

where $d(s, M_i)$ measures the point-to-surface distance from the raw part to the parameterized part, while $d(m, S_i)$ measures the point-to-surface distance from the parameterized part to the raw part. As the computation of point-to-surface distances is slow, to speed up the optimization process we cluster the parameterized part into planar regions using region growing, and the distance computing is approximated using these planar regions directly.

4.4.1 Mechanical constraints

We now detail our global optimization algorithm. Again, as different types of mechanical meshing impose different sets of constraints, we here focus on one specific type of meshing, i.e., gears, for the clarity of exposition.

Two meshing spur gears are required to have the same module and pressure angle, while two meshing helical gears are additionally required to have the same helix angle besides the constraints for spur gears. To simplify the problem, we use a constant pressure angle of $20°$. We use G to denote a gear. For two meshing gears G_1 and G_2, the center distance d_c between the gears has $d_c(G_1, G_2) = ||\mathbf{o}_1 - \mathbf{o}_2||$, where $\mathbf{o}_1$ and $\mathbf{o}_2$ are centers of gear G_1 and gear G_2 respectively. In the standard formula, $d_c(G_1, G_2)$ satisfies $d_c(G_1, G_2) = (d_{p1} + d_{p2})/2$, where d_p is the pitch diameter (see Figure 16), and the module m of a gear can be calculated with $m = d_p/Z$, where Z is the gear tooth number. Please refer to the appendix for more details. In this way we can derive the module by

$$m = 2 * d_c(G_1, G_2)/(Z_1 + Z_2). \tag{6}$$

In the case of a gear chain G^*, where more than two gears are involved and some gears mesh with more than one gear at the same time, the situation becomes more complex. Take a simple gear train with three gears for example (shown in Figure 9(a)), the input gear G_a meshes with an intermediate gear G_i which in turn meshes with the output gear G_b. In the ideal condition, we will have

$$\frac{d_c(G_a, G_i)}{d_c(G_i, G_b)} = \frac{Z_a + Z_i}{Z_i + Z_b}.$$

Since the variation of center distance may exist, this constraint is not always satisfied. Therefore, we use a soft constraint to penalize the module difference caused by the center distance variation. Then the objective function for the mechanical optimization becomes

$$\sum_i E_d(M_i, S_i) + \lambda \sum_{\substack{i \in G^* \\ |N(i)| \geq 2}} \sum_{\substack{x,y \in N(i) \\ x \neq y}} \left(\frac{d_c(G_x, G_i)}{Z_x + Z_i} - \frac{d_c(G_i, G_y)}{Z_i + Z_y}\right)^2, \tag{7}$$

where the variables to be optimized are the gear centers $\{\mathbf{o}_i\}$, and λ is 10 in our implementation. We use a trust region method based on interior point nonlinear programming to solve this optimization as in [1].

After the gear centers are determined, the gear modules are calculated with Equation 6 for all gears. Then they will be jointly optimized to make sure that the gear modules in the same gear train are all the same. Note that the parametric gear will be updated when its module changes. The optimization objective here is to minimize the total fitting error between the parametric data and scan data

$\sum_i E_d(M_i, S_i)$ for each gear train, and now there is only one variable (module) to be optimized for each train. The initial module value for a gear train is set as the mean value of the gear modules in this gear train. We use a gradient decent algorithm since the initial value is actually pretty good. For two meshing helical gears, we will also optimize the helix angle besides the gear module.

4.4.2 *Orientation alignment*

We constrain the orientations of mechanical parts to be exactly parallel or orthogonal. We solve a nonlinear optimization over the parameters of the parametric parts $\{M_i\}$ to minimize the data fitting error while exactly satisfying the constraints, using the method presented in [1]. The difference is that our optimization is solved using an iterative method because the point-to-surface correspondences between S_i and M_i will change when either the axis or the center of M_i changes, and the objective function is difficult to be expressed in closed form.

In each iteration, to ensure the point-to-plane correspondences stay unchanged, we bound the angle change of part orientation before and after optimization to be no more than $0.5°$. We use a trust region method based on interior point nonlinear programming to solve this optimization as in [1]. After each iteration, we update the corresponding plane for each point and reevaluate the error. The algorithm stops when it converges (the algorithm usually converges in 10 iterations).

4.4.3 *Placement alignment*

Since the coplanarity and coaxial relations carry important cues about the object parts in manmade shapes (see also [1]), we also conform to such placement relations after orientation alignment, while preserving the already established orientation relations. For the detection and enforcement of coaxial relation and coplanar relation, we refer to [1].

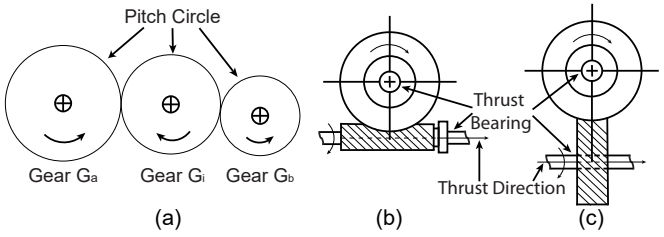


Fig. 9. (a) A simple gear train with three gears. The input gear G_a meshes with an intermediate gear G_i which in turn meshes with the output gear G_b. (b)(c) Gearing examples with vertical shafts, where (b) is a worm-gearing and (c) is a helical-gearing.

In vertical transmission relations as shown in Figure 9, suppose the center and axis of gear a and gear b are $\mathbf{o}_a$, $\mathbf{n}_a$ and $\mathbf{o}_b$, $\mathbf{n}_b$, respectively. Then we force

$$(\mathbf{o}_a - \mathbf{o}_b) \cdot \mathbf{n}_a = 0,$$
$$(\mathbf{o}_a - \mathbf{o}_b) \cdot \mathbf{n}_b = 0.$$

Remark. In our implementation, the center constraint mentioned in the mechanical constraint is enforced in the placement alignment stage as it is a special type of placement constraint.

4.5 Local parameter refinement

After placing the functional parts in correct positions and orientations, we locally refine their geometry so that the parametric model fits better to the scan, not breaking the already learned and satisfied relations (as shown in Figure 10). All the parameters that do not affect the already optimized parameters could be optimized in this step. Take a spur gear for example, the gear thickness, tooth thickness, rotation angle have no influence on its relations with other gears and only affect its geometry. For a cam, the face width and its contour will not affect its relations with others thus can be optimized to better fit the scan. We use a gradient decent algorithm to minimize the data fitting error as the parametric model estimation step in 4.2.

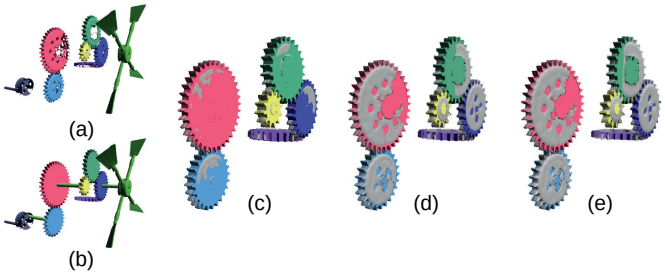


Fig. 10. (a) The scan of mechanical parts. (b) The final parametric model for these mechanical parts. (c-e) The comparison between scan and parametric parts obtained by local estimation, global optimization and local refinement respectively.

4.6 Segmentation refinement

Once the parametric parts are aligned and optimized using the detected global relations, we refine the initial segmentation results. Specifically, we apply the MRF labeling algorithm again on the initially generated over-segmented patches (Section 4.1.3) using the updated parametric model as reference, where the data cost now becomes:

$$\begin{aligned} D(L(p_i^k) \to 0) &= 1 - prob(p_i^k, M_i), \\ D(L(p_i^k) \to 1) &= prob(p_i^k, M_i), \end{aligned} \tag{8}$$

where M_i denotes a parametric part.

Patches with small data error to the parametric part M_i are thought to belong to the functional part S_i. Patches with large data error to all the parametric parts will be collected as unclaimed/static. The data error between a patch and a parametric part is the distance from the patch center to the parametric part. A result of segmentation refinement is shown in Figure 11.

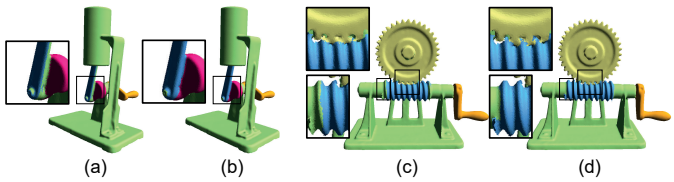


Fig. 11. Comparison between the initial segmentation and the refined segmentation. (a)(c) The segmentation results using our patch-based hierarchical segmentation algorithm. (b)(d) The re-segmentation results which are refined with the help of parametric model.

5 Results and Discussion

In this section, we show the functional recovery results of mechanical assemblies generated by our approach. We tested our method on real scanned mechanical assemblies of varying types and complexity[1]. The input objects were all captured using a handheld 3D scanner called $HSCAN-330$. All experiments are conducted on a desktop with an Intel(R) Core(TM) i7-4770 3.4GHz CPU and 16GB memory. The performance of the optimization process on various scanned models is presented in Table 1. The running time depends on the number of functional parts, the number of parameters as well as the complexity of relationship. Since only parallel constraints are forced on the cam and crank models, their global optimization processes are much faster than other models.

model	point #	functional parts #	added joints #	time (in sec)		
				local estimation	global optimization	local refinement
helical gear	40k	3	1	200	155	828
worm & worm gear	114k	3	1	541	285	1237
piston	50k	4	3	225	141	305
cam	149k	3	3	678	131	1561
windmill	102k	8	3	1697	3452	2360

TABLE 1
Performance statistics on the testing examples.

Our test models are listed in Figure 12: a helical gear arrangement, a worm gear drive arrangement, a cam mechanism, a piston mechanism and a windmill model. Please also refer to our supplementary video to see the recovered models which can move correctly after fabrication. There are eight typical mechanical parts that are involved in these devices/models, including spur gear, helical gear, worm gear, worm, crank, slider, cam and rod. In the worm gear drive arrangement (see Figure 12), the segmented worm from scan is seriously broken which poses large difficulty to the parametrization process. In our system, the initial worm lead is determined by the module of worm gear rather than the segmented geometry before its local optimization. This turns out to be efficient and robust based on the fact that a worm always works with a worm gear and our parametric gear initialization shows robustness to noise. Note that the windmill model (see Figure 12) is a toy model which does not strictly follow the rule of mechanism manufacture. The coplanar constraint can be optional in this case as the meshing gears in this real model are not coplanar.

We observe that many crucial joints like the rods between drivers and functional parts are usually hidden inside the model which are typically missing after scanning. Such situations are very common in mechanical devices. Almost all our test models have such problems (except for the windmill model (see Figure 12)). Our system automatically infers a rod between two coaxial functional parts. We also handle other cases such as the missing followers in piston mechanism and cam mechanism. In such cases, the joints are usually inferred based on specific mechanical type, i.e., there always exists a rod between a crank and a slider (see Figure 18) and a roller between a cam and slider (see Figure 13, Figure 18 and appendix for more details).

1. The code, data and demo software are available at http://tianjiashao.com/Codes/2016/MechAssemRecovery.rar.

Non-parametric parts, such as static base, end effector and driver, are reconstructed using poisson surface reconstruction [25] making use of all scans data. The piston in the piston mechanism is first fitted with a cylinder and then hollowed out with another smaller cylinder. All the non-parametric parts of the windmill model are modeled, at one-to-one scale, according to the real model.

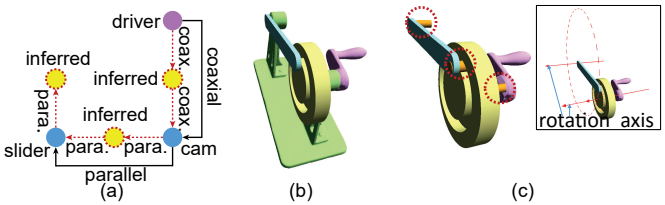


Fig. 13. Missing parts inference for the cam mechanism. (a) The interaction graph of the cam model. (b) The parametric model. (c) Three rods are automatically added for this mechanism. The inset in (c) shows the constraints between functional parts.

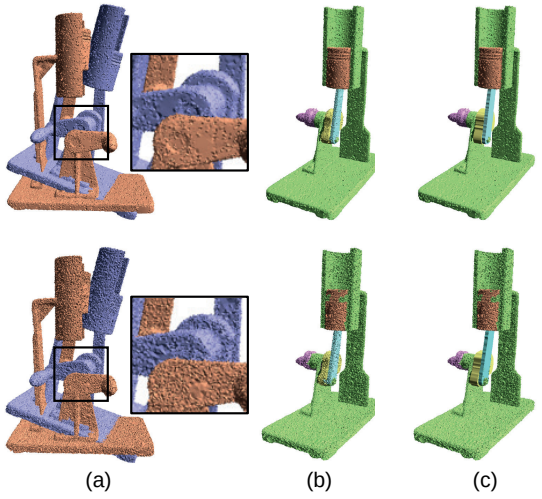


Fig. 14. The piston mechanisms with 20% and 50% noise respectively. (a) Input scans. (b) The segmented model. (c) The parametric model.

We also evaluate the robustness of our algorithm. We test on crank model and windmill model with 20% and 50% gaussian noise added. Our algorithm successfully recover the functional assemblies for the crank model in both cases. In general, our patched based registration algorithm works even under large noise (see Figure 14). However, for mechanical parts where fine geometric details are critical (e.g., the windmill gear in Figure 12), when large noise arises, the hierarchical segmentation would fail since small patches on gear teeth that are important in our patch-based registration become unreliable under noise. This limitation is not hard to perceive as once the noise level breaks the fine details, it would be hard to recover the correct parameters.

Parameters. Our algorithm has a set of parameters. Here we briefly discuss the important parameters that require manual tuning in our pipeline. Please also refer to Table 3 for the parameters for each example. These parameters are inherently dependent on the quality of scans as well as types of assemblies to be reconstructed. Firstly, in the patch generation process, the normal angle threshold β to merge

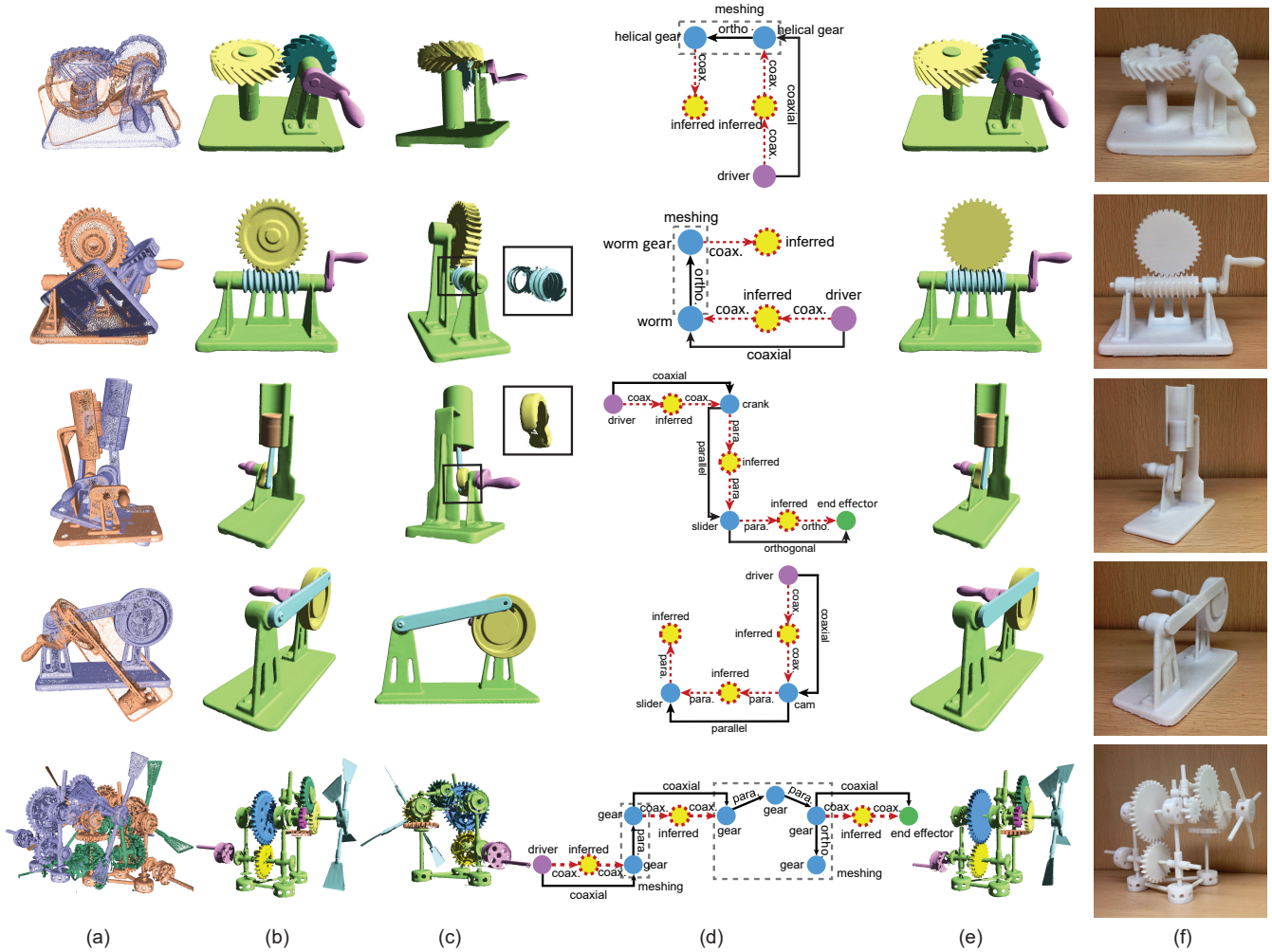


Fig. 12. The test models are listed in order as follows: a helical gear arrangement, a worm gear drive arrangement, a piston mechanism, a cam mechanism and a windmill model. (a) Input raw scans. (b) and (c) are the final segmented models after segmentation refinement under different views. (d) Interaction graphs. (e) Parametric models. (f) Printed models.

neighboring faces is essential to the success of registration, especially for gears and worms whose key features are seriously contaminated by noise and self-occlusion. For example, too many small patches on the gear teeth will be generated with a small β for noisy scan data, which is unstable and unreliable, while the large surfaces of coplanar gears may be merged together with a large β, which will lead to incorrect labeling for such surfaces. As a result, β needs a manual tuning depending on the mechanical type. In our implementation, for the assemblies with gears, we set $\beta = 40^\circ$ in the beginning of hierarchy to identify large parts (e.g. bases) and set $\beta = 30^\circ$ for subsequent small parts. When coplanar cases occur (see the windmill model), we switch β to 20°. Secondly, the parameter λ used in the MRF formulation which enforces the pairwise data smoothness, is set to be in range of $[1.0, 2.0]$ in the first level of hierarchy and 0.5 in the subsequent levels. This is because the fidelity of data matching will increase as the hierarchy increases. Finally, the matching distance threshold d_ε which determines whether two patches are considered to be matched, is dependent on the noise level and sample rate, and takes range from $2mm$ to $6mm$ in our examples.

User interaction and limitations. Our automatic motion-based segmentation algorithm could fail in two cases. The first case is that the motion of some parts cannot be detected with geometry if the motion is very slight (e.g. the end effector of the planetary gear model in Figure 15) or the shape always looks "static" under motion (e.g., the "static" rods of the windmill model in Figure 12, and the outer contour of the cam in Figure 4). Though our automatic inference with the interaction graph could recover those "static" rods that are ignored in the segmentation stage, it could fail if the ignored parts are the end nodes of the graph. The second

Case	Helical Gear Arrangement	Worm Gear Drive Arrangement	Piston Mechanism	Cam Mechanism	Windmill Model
Segmentation	0/3	0/3	1/4	1/3	3/8
Type Designation	Helical Gear	Worm Gear	-	-	End Effector

TABLE 2
User interaction involved in each test case. The first row shows how many parts out of all the parts need the manual correction of segmentation. The second row shows which mechanical part type in the assembly needs manual designation.

Case	Helical Gear Arrangement	Worm Gear Drive Arrangement	Piston Mechanism	Cam Mechanism	Windmill Model
β	$40^\circ/30^\circ/30^\circ/30^\circ$	$40^\circ/30^\circ/30^\circ/30^\circ$	20°	30°	$40^\circ/30^\circ/30^\circ/30^\circ/20^\circ/30^\circ/30^\circ$
$d_\varepsilon(mm)$	3	4	6	2	4
λ (Level 1)	2.0	2.0	1.0	2.0	1.1

TABLE 3
Parameters used for each test case.

case is that some parts are severely incomplete, causing a large number of patches cannot find correspondences. The MRF-based labeling could fail. In these cases, users are allowed to simply drag a rectangle to alter the labels of patches inside the rectangle. Another manual intervention happens in the step of determining part types. In our automatic pipeline, the part type is determined as the type of the best matched template. Such strategy works well for distinguishing parts of different main types, like gears, cranks, cams and driving parts, but might fail to distinguish sub-types, like a helical gear or a worm gear. Hence we allow users to manually designate the specific semantic type if needed. The manual intervention for each example is listed in Table 2. Finally, our system currently only supports limited kinds of popular functional mechanical assemblies. More functional mechanic assemblies such as levers are to be added in the future.

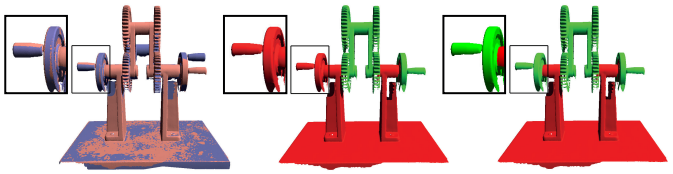


Fig. 15. Automatical segmentation fails for mechanical part whose amplitude of movement is very small. (a) Two different motion states of the end effector in a 2K-H planetary gear train model, whose variation can be hardly observed even by human eyes. (b) The original incorrect labeling result by our algorithm. (c) The labeling result with the help of user interaction.

6 Conclusion

In this paper, we introduced an algorithm for recovering functional mechanical assemblies from multiple raw scans. Our algorithm builds upon a simultaneous hierarchical registration and segmentation paradigm to separate functional mechanical parts under motions. To estimate the functionality, our system leverages an estimated interaction graph followed by a multi-stage parametric optimization process to faithfully reconstruct the underlying functional geometry.

To the best of our knowledge, leveraging functionality as the key ingredient for 3D reconstruction from raw scans has never been exploited in the literature. Our approach takes the first step towards the recovery of 3D functional and printable mechanical objects. A natural extension of this work is to handle other types of inputs such as depth images captured from less-accurate scanners (e.g., Microsoft Kinect) or scans of other types of objects or even from photographs. The next question to ask is whether we are able to alleviate the dependency on the parametric models used in our system in order to faithfully recover the original geometry while guaranteeing its functionality. Finally, the reconstructed models, being partially parameterized, are further adaptable to applications such as manipulation or redesign.

Acknowledgments

The authors would like to thank reviewers for their insightful comments. This work was supported in part by Microsoft Research Asia, the NSF of China (No. 61402402, No. 61572429 and No. U1609215), the Fundamental Research Funds for the Central Universities, the China Young 1000 Talents Program, and the ERC Starting Grant SmartGeometry (StG-2013335373). Tianjia Shao is the corresponding author of the work.

References

[1] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra, "Globfit: Consistently fitting primitives by discovering global relations," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4. ACM, 2011, p. 52.

[2] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *EUROGRAPHICS star reports*, vol. 1, no. 1, 2014, pp. 161–185.

[3] T. K. Dey, *Curve and surface reconstruction: algorithms with mathematical analysis*. Cambridge University Press, 2006, vol. 23.

[4] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, "A benchmark for surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, p. 20, 2013.

[5] S. Shalom, A. Shamir, H. Zhang, and D. Cohen-Or, "Cone carving for surface reconstruction," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6. ACM, 2010, p. 150.

[6] M. Attene, "A lightweight approach to repairing digitized polygon meshes," *The Visual Computer*, vol. 26, no. 11, pp. 1393–1406, 2010.

[7] P. Benkő, R. R. Martin, and T. Várady, "Algorithms for reverse engineering boundary representation models," *Computer-Aided Design*, vol. 33, no. 11, pp. 839–851, 2001.

[8] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, Jun. 2007.

[9] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, and W. Puech, "A comprehensive process of reverse engineering from 3d meshes to cad models," *Computer-Aided Design*, vol. 45, no. 11, pp. 1382–1393, 2013.

[10] A. Monszpart, N. Mellado, G. Brostow, and N. Mitra, "RAPter: Rebuilding man-made scenes with regular arrangements of planes," *ACM SIGGRAPH 2015*, 2015.

[11] W. Xu, J. Wang, K. Yin, K. Zhou, M. Van De Panne, F. Chen, and B. Guo, "Joint-aware manipulation of deformable models," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 35.

[12] J. Guo, D.-M. Yan, E. Li, W. Dong, P. Wonka, and X. Zhang, "Illustrating the disassembly of 3d models," *Computers & Graphics*, vol. 37, no. 6, pp. 574–581, 2013.

[13] R. Hu, C. Zhu, O. van Kaick, L. Liu, A. Shamir, and H. Zhang, "Interaction context (icon): Towards a geometric functionality descriptor," *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, vol. 34, no. 4, p. Article 83, 2015.

[14] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala, "Illustrating how mechanical assemblies work," *ACM Transactions on Graphics-TOG*, vol. 29, no. 4, p. 58, 2010.

[15] L. Zhu, W. Xu, J. Snyder, Y. Liu, G. Wang, and B. Guo, "Motion-guided mechanical toy modeling," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 127:1–127:10, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366146

[16] X. Peng, K. Lee, and L. Chen, "A geometric constraint solver for 3-d assembly modeling," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 5-6, pp. 561–570, 2006.

[17] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: an analyze-and-edit approach to shape manipulation," vol. 28, no. 3, pp. 33:1–33:10, 2009.

[18] Y. Zheng, H. Fu, D. Cohen-Or, O. K.-C. Au, and C.-L. Tai, "Component-wise controllers for structure-preserving shape manipulation," vol. 30, no. 2, pp. 563–572, 2011.

[19] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu, "Structure recovery by part assembly," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 180, 2012.

[20] Y. Zheng, D. Cohen-Or, and N. J. Mitra, "Smart Variations: Functional Substructures for Part Compatibility," vol. 32, no. 2pt2, pp. 195–204, 2013.

[21] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 5, p. 150, 2013.
[22] Y. Zheng, D. Cohen-Or, M. Averkiou, and N. J. Mitra, "Recurring part arrangements in shape collections," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 115–124, 2014.
[23] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, and H. Zhang, "Topology-varying 3d shape creation via structural blending," *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, vol. 33, no. 4, p. Article 158, 2014.
[24] S. Ghosh, M. Loper, E. B. Sudderth, and M. J. Black, "From deformations to parts: Motion-based segmentation of 3d objects," in *NIPS 25*, 2012, pp. 1997–2005.
[25] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 29, 2013.
[26] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D Mesh Segmentation and Labeling," *ACM Transactions on Graphics*, vol. 29, no. 3, 2010.
[27] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, "Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 30, no. 6, pp. 126:1–126:9, 2011.
[28] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen, "Segmenting a deforming mesh into near-rigid components," *The Visual Computer*, vol. 22, no. 9-11, pp. 729–739, 2006.
[29] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," in *Computer Graphics Forum*, vol. 33, no. 5. Wiley Online Library, 2014, pp. 205–215.
[30] X. Li and I. Guskov, "Multiscale features for approximate alignment of point-based surfaces." in *Symposium on geometry processing*, vol. 255. Citeseer, 2005, pp. 217–226.
[31] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
[32] F. L. Litvin and A. Fuentes, *Gear geometry and applied theory*. Cambridge University Press, 2004.
[33] D. Cheng, "Mechanical design handbook," 2004.
[34] D. B. Marghitu, *Mechanical engineer's handbook*. academic press, 2001.
[35] W. Ma and J.-P. Kruth, "Nurbs curve and surface fitting for reverse engineering," *The International Journal of Advanced Manufacturing Technology*, vol. 14, no. 12, pp. 918–927, 1998.
[36] J. R. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," in *Applied computational geometry towards geometric engineering*. Springer, 1996, pp. 203–222.

APPENDIX

There are seven typical mechanical parts involved in our system now. They are spur gear, helical gear, worm gear, worm, crank, slider and cam. Since each mechanical part has its own feature and parameters, their parametric model initialization differs with each other. It's too long to cover all of this process in the paper, so we explain the details in this document instead. For each mechanical part, its parameters are first estimated from the geometry and then locally optimized to approximate the geometry as much as possible.

We first introduce important parameters that are related to the geometry for each mechanical part and then explain how to obtain these parameters from geometry. Note that all the gears in our system are parameterized as involute gears for simplicity.

Spur Gear

The spur gear is the simplest type of gear. Parameters of spur gear that matter are module m, tooth number Z, addendum ha, dedendum hb, circular tooth thickness t, face width w, fillet radius fr and pitch diameter d_p (diameter of pitch circle). The illustration of these parameters and some other terminologies that mentioned in the paper such as whole depth h, outside diameter d_o (diameter of addendum circle), root diameter d_r (diameter of dedendum circle), and center distance d_c can be found in Figure 16(a), Figure 17. Please refer to [33] [34] for more specific explanation of these terminologies and related calculations. The calculations for spur gears are used as the basis for the calculations for other types of gears:

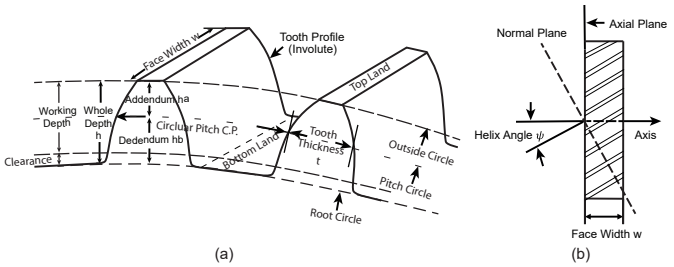


Fig. 16. Nomenclature illustration for spur gear and helical gear. (a) A spur gear. (b) The side view of a helical gear.

$$\begin{aligned} &d_p = d_o - 2ha = d_r + 2hb = m * Z, \\ &h = ha + hb, \\ &fr = 0.3m, \\ &t_{max} = \frac{m}{2\pi}. \end{aligned} \tag{9}$$

To generate the parametric model of a gear, the necessary parameters are pitch diameter, tooth number, addendum, dedendum, tooth thickness, face width and pressure angle. To simplify the problem, we use a constant pressure angle $20°$. The initial tooth thickness t is set as $t := (t_{max} + t_{min})/2$, where t_{max} and t_{min} are the maximum and minimum valid tooth thickness for a gear of module m. t_{max} is calculated with the formula in Equation 9. t_{min} is the tooth thickness when top land of the gear tooth becomes a line.

Observing that the addendum, dedendum, fillet radius and tooth thickness can be calculated from the module, and the pitch diameter can be calculated from the module and tooth number, the only required parameters are the module and tooth number. Since we are not able to estimate the module directly from geometry, we use the outside diameter, root diameter and tooth number to obtain an approximate module using the following equations:

$$\begin{aligned} &h = (d_o - d_r) * 0.5, \\ &ha = h/2.25, \\ &hb = h - ha, \\ &m = (d_r + 2hb)/Z, \end{aligned} \tag{10}$$

where the outside diameter, root diameter and tooth number are estimated from geometry with the following method.

We project the gear onto its axial plane (a plane that is perpendicular to the axis). Note that the initial gear axis is calculated from the transformation matrix obtained in registration process. The contour gear is first extracted and then denoised with dilation and erosion operations (see Figure 7). To estimate the outside diameter and root diameter from the denoised gear contour, we detect points

on the top land and bottom land separately. The outside diameter is twice of the average distance from points on top land to the gear center and the root diameter is twice of the average distance from points on the bottom land to the gear center.

To get an accurate tooth number, we first guess several approximate tooth numbers:

$$Z_1 = d_o/(d_o - d_r) * 4.5 - 2,$$
$$Z_2 = d_o/(d_o - d_r) * 3.6 - 1.6,$$
$$Z_3 = \pi/\arcsin(d/d_o),$$

where Z_1 and Z_2 are the potential tooth numbers for the gear of standard teeth and the gear of stub teeth respectively, while Z_3 is calculated with the estimated distance d between adjacent teeth. d is obtained by first clustering the points on the top land and bottom land separately and then calculating the mean value of the average distance between the point clusters.

Then for each Z_i, we estimate the angle θ_i between two teeth as $360/Z_i$. We further refine θ_i so as to obtain the refined tooth number Z_i^*. For the refinement of θ_i, we rotate the gear with θ_i about its rotation axis, and then perform an ICP registration to get the actual rotation angle θ_a. For the sake of robustness, we perform the above process multiple times until the total rotation angle θ_t is around $360°$. In this way, Z_i^* is computed as θ_t/θ_a. At last we choose the Z_i^* which gives the best matching during the rotation as the final tooth number. With this strategy, we get correct tooth numbers in all our experimental cases.

After the initial parametric model is guessed, we then optimize the outside diameter, root diameter, tooth thickness, face width, rotation angle and center of the parametric gear to approximate the geometry of gear segment as much as possible, with a gradient decent algorithm. To accelerate the convergence, the optimization is first done in 2D space and then in 3D space.

Helical Gear

Helical gear is a cylindrical gear whose tooth flanks are helicoid (see Figure 16(b)). The tooth profile of the helical gear is an involute curve in the plane perpendicular to the axis which is based on a normal system. Helix angle is what differs helical gears from spur gears.

To extract the gear parameters, we can not project the entire helical gear directly onto a 2D plane due to the existence of the helix angle. Instead, we only make use of one face (front or back) of the helical gear. The parameters are estimated in the following manner: we first divide all patches into two sets S_p and S_q (patches whose normals are almost parallel to the axis and whose are not, the angle threshold is $\pi/12$ in our implementation). Patches in S_p can be further clustered into two sets S_{p1} and S_{p2} on different sides. The patch set S_{pi} ($i = 1$ or $i = 2$) with larger total area is selected to estimate the parameters. The parameter estimation process is the same as that for spur gear. Then we cluster the angles between patches in S_q and the gear axis using a histogram whose bin width is $5°$. The initial helix angle is $\psi := \pi/2 - \alpha$, where α is the average angle in the largest cluster.

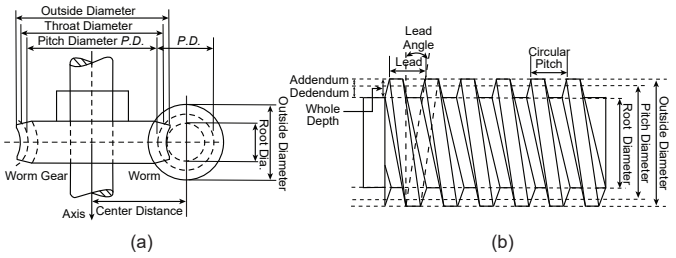


Fig. 17. Illustration of the nomenclature for worm and worm gear. (a) Two mating worm gear and worm. (b) Worm.

Worm Gear

Worm gear is also a variation of spur gear but is more complex than helical gear (see Figure 17(a)). The estimation of the outside diameter, root diameter and tooth number is similar to helical gear. Note that the actual working depth diameter is measured in the middle of the gear rather than the front side or back side, which is different from the helical gear or normal spur gear. Again the rough guess will then be optimized based on the geometry.

Worm

An worm example is shown in Figure 17(b). The estimation of the outside diameter and the root diameter is similar to that of gear. All points of the worm segment are clustered by their distance to the worm axis which is similar to the gears. The outside diameter is twice of the average distance from the furthest cluster to the axis and the root diameter is twice of the average distance from the closest cluster to the axis. Then the addendum and dedendum are calculated with Equation 10. The pitch diameter is calculated with Equation 9.

Slider/Connecting Rod

Parameters for a slider include an anchor point, an axis, a NURBS curve which figures the outer contour, and the face width w.

The thickness t is initially set as the shortest edge length of the oriented bounding box (OBB) of the slider/crank segment, which will be optimized locally later. The NURBS curve is obtained as following: we first project the segment onto a 2D plane perpendicular to the shortest axis and extract the contour for slider/crank, and then fit a closed NURBS curve for the contour [35]. After that, a 2D mesh is generated with the method presented by [36] and then we extrude the surface to a volume with thickness t.

Crank

Parameters for a crank include a rotation center, a rotation axis, a radius, a NURBS curve which figures the outer contour and the thickness t (see Figure 18(a)). Shape parametrization for the crank is the same as the slider.

Figure 18(a) shows two configurations of the piston mechanism. The xy plane is perpendicular to the crank axis. Axis y is parallel to the moving direction of piston. $\mathbf{O}_1$ and $\Delta\theta$ are computed from the transformation matrix of crank while Δh is computed from the transformation matrix of

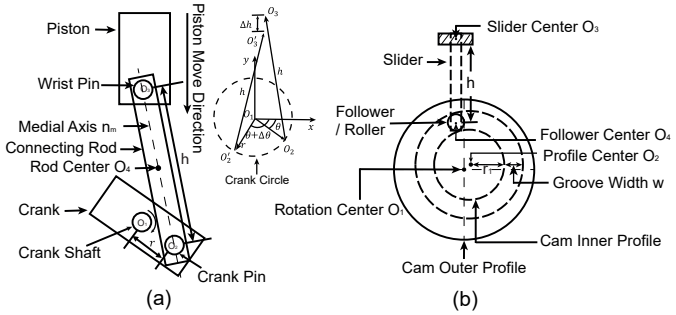


Fig. 18. Nomenclature illustration for piston mechanism and cam mechanism. (a) A piston mechanism (left) and two configurations of the piston mechanism (right). (b) A grooved cam mechanism.

piston. We assume that $\mathbf{O}_2$ and $\mathbf{O}_3$ are located in the medial axis $\mathbf{n}_m$ of the connecting rod. Then $\mathbf{O}_2$ can be decided by θ and $\mathbf{O}_3$ can be decided by h. If θ is decided, r can be computed and h can be solved according to the following equations:

$$\begin{aligned}
&\mathbf{O}_2 = (r\cos\theta, r\sin\theta),\\
&\mathbf{O}_2' = (r\cos(\theta+\Delta\theta), r\sin(\theta+\Delta\theta),\\
&\mathbf{O}_3 = \mathbf{O}_2 + h * \mathbf{n}_m,\\
&\mathbf{O}_3' = \mathbf{O}_3 + (0, \Delta h),\\
&||\mathbf{O}_3' - \mathbf{O}_2'|| = h.
\end{aligned}$$

The initial θ is set as the angle between the medial axis of crank and axis x. We will further optimize θ with a gradient decent method.

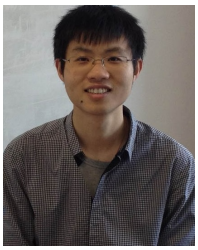
Cam

A cam mechanism usually consists of four moving elements, a driver, a cam, a follower (a roller in our system) and a follower system (a slider in our system) (see Figure 18(b)). There are many different types of cam shape. The cam in our system is a grooved cam with an inner profile and an outer profile, which is more complex than plate cams or disk cams.

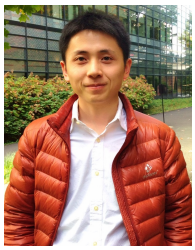
The cam thickness t is initially set as the shortest edge length of the OBB of the cam segment. The depth of the groove is initially set as half of the cam thickness, which will be locally optimized later. Then we detect front face and back face of the cam by first collecting patches that are almost perpendicular to the cam axis whose distance to the cam center are larger than a threshold ϵ ($0.4t$ in our implementation) and then dividing them into two sets with opposite normal directions. Patches on these two sets are projected onto the plane perpendicular to the cam axis separately and we will get three contours for the front face and two for the back. Since the inner profile of the cam is a circle, center $\mathbf{O}_2$, radius r and groove width w can be decided by fitting two concentric circles for the two small contours of the front face. We get the outer profile by fitting a circle for the largest contour of the front face.

Because of self-occlusion, the roller is missing, which results in the failure in motion propagation. The roller is automatically inferred in our system (see Figure 13). Roller radius is set as $r_2 := w/2$ where w is the groove width of the cam. Slider center $\mathbf{O}_3$ is computed from the transformation matrix of the slider. Then the position of the roller can be decided by h.

Minmin Lin received the bachelor's degree in digital media technology from Zhejiang Sci-Tech University, Hangzhou, China, in 2012. Currently, she is working toward the PhD degree at Graphics and Parallel Systems Lab, Zhejiang University. Her research interests include modeling, object recognition and structure analysis.



Tianjia Shao is currently an Assistant Researcher in the State Key Lab of CAD&CG, Zhejiang University. He received his PhD in Computer Science from Institute for Advanced Study, and his B.S. from the Department of Automation, both in Tsinghua University. His research interests include RGBD image processing, indoor scene modeling, structure analysis and 3D model retrieval.



Youyi Zheng is currently an Assistant Professor at the School of Information Science and Technology, ShanghaiTech University. He obtained his PhD from the Department of Computer Science and Engineering at Hong Kong University of Science & Technology, and his M.Sc. and B.Sc. degrees from the Department of Mathematics, Zhejiang University. His research interests include geometric modeling, imaging, and human-computer interaction.

Niloy J. Mitra is a Professor of Computer Science at University College London (UCL). His research focuses on algorithmic issues in shape analysis and geometry processing. He received the ACM SIGGRAPH Significant New Researcher award in 2013 and the BCS Roger Needham award in 2015.

Kun Zhou is a Cheung Kong Professor in the Computer Science Department of Zhejiang University, and the Director of the State Key Lab of CAD&CG. Prior to joining Zhejiang University in 2008, Dr. Zhou was a Leader Researcher of the Internet Graphics Group at Microsoft Research Asia. He received his B.S. degree and Ph.D. degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests are in visual computing, parallel computing, human computer interaction, and virtual reality. He currently serves on the editorial/advisory boards of ACM Transactions on Graphics and IEEE Spectrum. He is a Fellow of IEEE.