Intelligent Service Robotics
https://doi.org/10.1007/s11370-021-00397-0

ORIGINAL RESEARCH PAPER

A novel object slicing-based grasp planner for unknown 3D objects

Sainul Islam Ansary[1] · **Sankha Deb**[2] · **Alok Kanti Deb**[3]

Received: 28 May 2021 / Accepted: 20 October 2021

Abstract
Robotic grasp planning has been one of the active areas of research in robotics but still remains a challenging problem for arbitrary objects even in completely known environments. Most previously developed algorithms had focused on precision/fingertip type of grasps, failing to solve the problem even for fully actuated hands/grippers during enveloping/adaptive/wrapping/power type of grasps, where each finger makes contact with an object at several points. Kinematic closed-form solutions are not possible for such an articulated finger, which simultaneously reaches several given goal points. This paper presents a framework for computing the best grasp for robotic hands/grippers, based on a novel object slicing method. The proposed method quickly finds contacts using an object slicing technique and uses a grasp quality measure to find the best grasp from a pool of pre-grasps. The pool of pre-grasps is generated by dividing the objects into parts and organizing them in a decomposition tree structure, where the parts are approximated by simple box primitives. To validate the proposed method, the developed grasp planner has been implemented on an industrial Motoman robot and a two-finger gripper. Further, the results have been compared with the state-of-the-art in grasp planning to evaluate the performance of the proposed grasp planner. As compared to other existing approaches, the proposed approach has several advantages to offer. It can handle objects with complex shapes and sizes. Most importantly, it works on both point clouds taken using a depth sensor and polygonal mesh models. It takes into account hand constraints and generates feasible grasps for both adaptive/enveloping and fingertip type of grasps.

Keywords Grasp planner · Spatial search trees · Grasp quality metric · Underactuated robotic hand/gripper

1 Introduction

Grasping is a fundamental requirement for object handling tasks that enables autonomous service robots to interact with the real world environment. Over the past decades, substantial efforts have been devoted in attempts to automate grasping using dexterous multi-fingered robot hands with a large number of degrees of freedom [1–3]. However, grasping arbitrary objects even in a completely known environment still remains a very challenging problem in robotics. Intuitive and previously learned knowledge play an important role in the case of human grasping. Humans iteratively learn to grasp and use their cognitive cues whenever encountered with a novel object. The application of these concepts to compute reliable grasps for novel objects in robotic applications is very challenging due to several reasons.

Firstly, the mathematical complexity of the problem is very high due to the complex geometry of the objects. Humans tend to use different prehensile hand postures for grasping different object geometries [4]. The selection of such object shape specific prehensile posture comes from learning and experience. Everyday objects are made of component parts e.g., a cup has a handle and a cylindrical part. For such objects, humans use grasping by parts where it searches for the graspable parts. The concept of prehensile postures in robotics is referred to as pre-grasp/pre-shaping, where the hand/gripper configures itself according to the object geometry. The pre-grasp consists of an initial gripper/hand position,

✉ Sainul Islam Ansary
sainul@iitkgp.ac.in

Sankha Deb
sankha.deb@mech.iitkgp.ac.in

Alok Kanti Deb
alokkanti@ee.iitkgp.ac.in

1 Advanced Technology Development Centre, IIT Kharagpur, Kharagpur 721302, India

2 Department of Mechanical Engineering, IIT Kharagpur, Kharagpur 721302, India

3 Department of Electrical Engineering, IIT Kharagpur, Kharagpur 721302, India

Published online: 19 November 2021

approach direction and its finger configuration depending upon the object geometry. The complex objects are decomposed into component parts, and appropriate strategies are applied to the object parts. Apart from the problem of decomposing the object in a more meaningful way, one problem associated with the grasp selection for an individual part is that the part may not be fully accessible to the gripper/hand due to occlusion by the neighboring parts. Another problem of applying the part-based method only on the individual object parts is that it may miss a good grasp which involves a combination of parts. While the first problem has not been covered in the literature, some attempts were made to address the second problem [5].

Secondly, a large number of degrees of freedom, kinematic constraints and actuation mechanism (e.g., an underactuated mechanism) in the gripper design make grasp planning even harder. The main objective in grasp planning is to find the contact points between the fingers and the object [6]. The approach taken to find the contact point decides the difficulty level. For example, most previously developed algorithms had focused on precision or fingertip type of grasps, where only the fingertips make contact with the object. In such methods, first, the contact points for each fingertip on the object surface are calculated by evaluating some grasp quality measure criteria [7]. And then inverse kinematics is used to find the joint displacements of the fingers. In some cases, kinematic constraints are also included during the contact computation so that infeasible solutions can be avoided. However, such approaches fail to solve the problem even for fully actuated hands/grippers during enveloping/adaptive/wrapping/power type of grasps, where each finger makes contact with an object at several points. It is not possible for an articulated finger to simultaneously reach several given goal points on the object generated by the above grasp synthesis techniques. Even if there is no reachability problem, the contact positions on the object surface alone are not sufficient to compute the finger joint positions for an under-actuated hand/gripper. Thus, it is necessary to compute all the contact points and finger joint positions simultaneously to successfully perform this type of grasp. This prompted the need for novel methods to handle such type of grasp.

In this paper, the problem of finding best grasp for robotic hands/grippers based on a novel object slicing method will be discussed. The objective of this work is focused on addressing the following research gaps: 1) automating the pre-grasps generation for arbitrary unknown 3D objects; 2) handling both enveloping/adaptive/power and precision/fingertip types of grasps; 3) making the developed grasp planner work on point cloud data without any expensive processing such as surface reconstruction. Further, a comprehensive comparison with the existing state-of-the-art in grasp planning has been given to justify its relevance as well as evaluate the performance of the proposed method in terms of computation time and robustness. Finally, the proposed method has been implemented to support three-different gripper/hand designs, and physical experiments have been performed using a two-finger gripper mounted on a Motoman robot.

This paper is an extension of our previous work [8]. The following are the major differences, improvements and extensions from the previous version. An automated object part-based pre-grasp generation has been introduced, where an object is decomposed into parts, and an appropriate type of grasp is applied on each part rather than on the object itself as in [8]. Unlike the previous version, this version can handle objects represented as point clouds as well as polygonal meshes. Furthermore, a new proximity query algorithm has been proposed, which computes contacts between an object and the robot hand modeled as a point cloud as well as triangulated mesh. A more comprehensive comparison with the existing state-of-the-art including some of the recently reported work has been presented, whereas the previous version was compared with only one grasp planner. Implementation has been done on a bigger set of objects comprising common household objects from two different object datasets found in the literature. Most importantly, physical results have been presented to demonstrate the practical relevance of the grasp planner.

1.1 Contributions

The important contributions reported in this paper are as follows. A novel grasp planner has been introduced into the robotic grasping community. Firstly, the proposed method can handle arbitrary 3D objects with complex shapes and sizes. Secondly, the grasp planner is equally effective on objects represented as polygonal meshes and point clouds without the need for computationally expensive processes such as surface reconstruction, as compared to the existing ones, which mostly relies on polygonal mesh model; Thirdly, an automated framework for encoding blocked areas (not reachable to the fingers) and associated rules have been introduced in the pre-grasp generation stage, which helps the planner finding natural like grasps (inspired by the way the humans grasp objects). Fourthly, the planner can generate feasible grasps for both the adaptive/enveloping (which is more challenging and comparatively less work has been done in the literature) and the fingertip types of grasps by taking into account the hand constraints (e.g., underactuation, joint limits, etc.). Most importantly, the paper also presents how readily the grasp planner can be implemented on a real robot platform.

The rest of the paper is organized as follows. Section 2 discusses the related work in the area of grasp planning and different grasp quality metrics. Section 3 presents the pro-

posed grasp planning framework for unknown 3D objects. Section 4 illustrates implementation results for different types of grasps and compares them with the state-of-the-art. Finally, Sect. 5 gives conclusions and scope for future work.

2 Related work

In the following, some of the previous works related to the proposed approach are discussed. An extensive review of recent developments in the area of grasp planning for 3D objects using analytical as well as empirical approaches can be found in [2]. Although, grasp planning has been one of the active areas of research in robotics, much less work has been found in areas where enveloping/wrapping/power type of grasps is considered. Miller et al. [9] addressed the problem by approximating complex unknown objects into primitives shapes e.g., boxes, spheres, cylinders and cones. Then, an appropriate type of grasps was selected for each of the primitive shapes. Li et al. [10] computed enveloping grasps for multi-fingered robotic hands based on the idea of maximizing the contact surface between the object and the hand surface. Another grasp planner for power grasp (i.e., enveloping grasp) based on object surface matching was presented in [11], where the best points for opposing grasp using opposing fingers were found by matching local curvature of the object surface. Li et al. [12] proposed a novel geometric algorithm to find enveloping grasp configurations for a multi-fingered hand. The proposed method performs a low-level shape matching by tightly wrapping multiple cords around an object to quickly isolate potential grasping regions. From these grasping regions, hand poses and contact points can be computed. Xue et al. [13] proposed an efficient method to find all the contact points between a robotic hand and an object. They used the swept volume method to generate a volume by moving each finger along an arbitrary trajectory. Then, a continuous collision detection (CCD) algorithm was applied to find all the possible contact points between the swept volume and the object through the entire configuration space of each finger. Saut and Sidobre [14] computed grasp configurations based on an approximation of the intersection between the object surface and the finger workspace and then, selected the best grasp according to a quality score. Shi and Koonjul [15] presented a grasp planning algorithm based on a simple strategy called finger curling planes (FCP) for real-time bin-picking applications, where multiple 2-D polygon profiles of the object are generated for contacts computation with the finger by parallel planes cutting through the 3D object.

In general, most of the previous work focused on the precision/fingertips type of grasps. Roa and Suarez [16] worked toward ensuring precision grasp in robotic hands for 3D objects. It was found that mechanical hands hardly touch an object precisely at the computed contact points using previously developed algorithms. Thus, the concept of independent contact regions was introduced where a finger can make contact anywhere inside each of these regions, despite the exact contact position. The independent regions (ICRs) were generated by growing them around the contact points of a given starting grasp. Rosales et al. [17] determined hand configuration relative to the object so that a given region on the surface of the hand establishes contact on its corresponding specified object region. They formulated the problem as a system of polynomial equations of a special form and then exploited this form to isolate the solutions using the linear relaxations technique. Unlike the previous algorithms, free-form regions on the hand and object surfaces can be specified and always return a solution whenever one exists. Song et al. [18] voxelized 3D objects and built a contact score map on the object, which shows whether a particular voxel can be touched or not by the fingers/palm. Then they found a set of target contacts on the object surface based on the score map. In [19], a framework for the fingertip grasp synthesis and in-hand grasp adaptation was presented. Hang et al. [20] formulated optimal grasping as a path finding problem and introduced the concept of super-contact. Zheng [21] computed the best grasp by finding contact location in a discrete point set based on combinatorial search.

Besides the complexity in the object geometry, the hand internal degrees of freedom and those in the wrist of the hand create a huge grasp search space. The internal degrees of freedom (DOFs) in the hand along with 6 DOFs in the robot arm make the grasp space too large to be exhaustively searched. Many approaches are there in the literature to find good wrist position and orientation in this huge search space. Cutkosky and Wright [22] attempted to classify hand postures and find grasp taxonomies needed for robot gripper in a manufacturing cell. Predefined prototypes of grasp have been used to reduce the search space [23]. Object part-based grasp selection had been used in the past, where the object was decomposed manually, and the parts were approximated using primitive shapes e.g., planes, boxes, cones, spheres or cylinders [9]. Goldfeder et al. [24] used Super Quadrics (SQs) as generic shape primitives to automate the process and organized the object parts in a decomposition tree. More, recent work [25] had argued that successful grasp selection depends upon the geometry rather than the object identification and put more emphasis on efficiency over accuracy in the shape approximation. A single view 3D point cloud data taken using a depth camera was approximated by simple box primitives. Then heuristics were used to select graspable box faces and finally, an off-line trained neural network gives the best grasp hypothesis. Li et al. [12] employed deterministic sampling on the spherical surface constructed around the object to find the initial hand position and approach direction. Shape diameter function (SDF) was used by Vahrenkamp et al. [26] to segment objects into parts and then principal component analysis

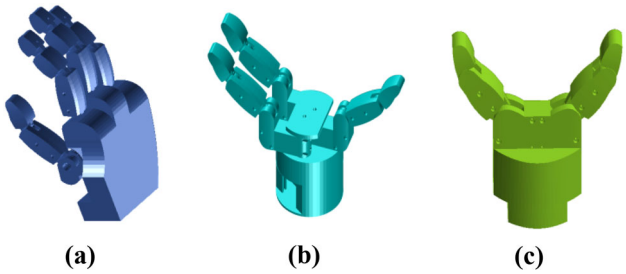

Fig. 1 **a** Four-finger anthropomorphic hand, **b** Three-finger underactuated hand, **c** Two-finger adaptive gripper

was applied on the parts to align the hand with the corresponding object part. The main drawback of the method is that it can only be applied on a polygonal mesh of the object.

3 Grasp planner

This work is mainly focused on automating robotic grasping of 3D objects having arbitrary shapes and sizes. The proposed grasp planner is not limited to a hand/gripper having a specific number of fingers or actuation mechanisms. The core idea of the planner has been described in the subsequent sub-sections and then its implementation has been shown on three-different hands/grippers as illustrated in Fig. 1 to demonstrate the generic nature of the planner. The proposed grasp planner assumes that complete information about the object geometry is available. It works on both 3D point clouds and the object represented as polygonal mesh. A complete framework of the proposed grasp planner is shown in Fig. 2. The workflow can be divided into three phases. In the first phase, the object is decomposed using simple box primitives, and a pool of pre-grasps is generated for the object parts where a pre-grasp consists of an initial hand position, approach direction and the finger configuration of the hand. Then, an object slicing-based method is applied in the second phase to quickly find the contact points on the object for evaluating the quality of the grasps. In the third phase, using the mesh model of the hand, contacts and finger joint displacements are computed by closing the fingers until contacts are found.

3.1 Robotic hands/grippers

For showing the implementation of the grasp planner, three hands/grippers have been chosen from the wide range of robotic hands/grippers, covering different aspects ranging from the number of fingers to actuation mechanisms. The first one is a four-finger fully actuated anthropomorphic hand, which covers the humanlike hands as shown in Fig. 1a. This hand has one thumb and three identical opposing fingers,

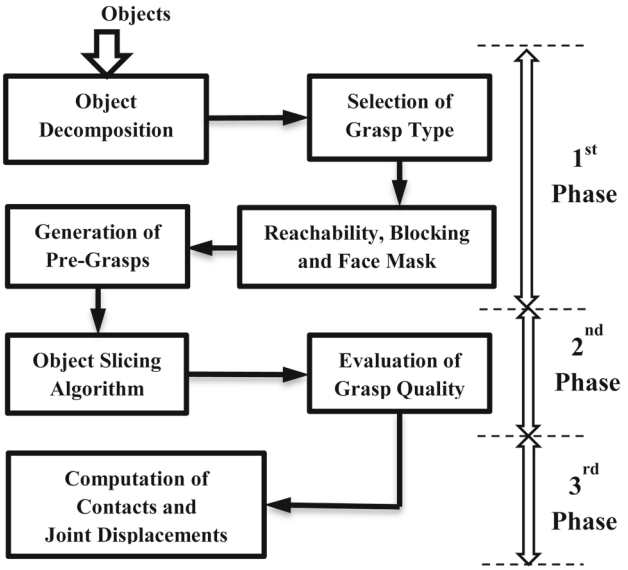

Fig. 2 The framework of the proposed grasp planner

each having three links. The second one is a three-finger underactuated hand [27], and it consists of three identical fingers, each having three links namely knuckle, middle and distal as shown in Fig. 1b. The thumb finger has two joints, and its base is fixed on the palm. The other two fingers have three joints each, and the base joints help the fingers to spread sideways synchronously around an axis perpendicular to the palm surface. The underactuation mechanism [28] works in such a way that a link in an articulated finger only moves after its previous link touches an object or maximum joint limit is reached. Finally, the third one is a simple two-finger adaptive gripper having two links in each finger as shown in Fig. 1c. The links of a finger move in such a way that once a link stops moving, the other link continues to move and wrap around the object. Most of the parts of the grasp planner are not limited to a specific type of hand/gripper, however, some minor adjustments may be needed to set up the planner for different hand/gripper configurations as mentioned in the following sub-sections.

3.2 Object decomposition

The data captured using a dense stereo camera or a depth sensor is a cloud of 3D points, which are further processed to reconstruct the object surface and stored as a polygonal mesh. It is very difficult to come up with a grasp strategy from this kind of low-level representation of the object. A 3D object can be represented using a set of shape primitives (e.g., planes, boxes, spheres or cylinders). A more generic approach is to use Super Quadrics (SQs) as shape primitives, which offer a large variety of different shapes. Shape primitives such as super quadrics give a good approximation

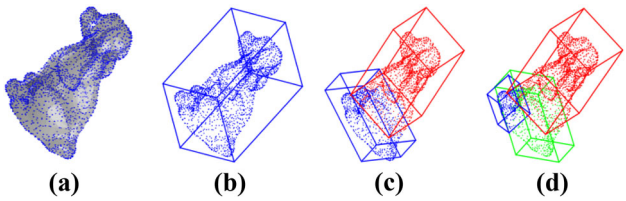

Fig. 3 **a** A complete point cloud of a dog model, **b** bounding box, **c** two child boxes resulted from the root box in the first iteration, **d** final decomposed object parts approximated by three boxes

of the object but are difficult to process. Here more focus is given on the efficiency (in terms of computation) of the grasp planner by using simple boxes as shape primitive for approximating the objects and then the types of grasps are selected per the criterion described in the following Sect. 3.3. The object decomposition is done by using a fit and split algorithm based on Minimum Volume Bounding Box (MVBB) [29]. The output of the object decomposition algorithm is a set of Oriented Bounding Boxes (OBBs) organized in a tree structure. The algorithm starts with tightly fitting the data points by a bounding box having minimum volume. Then, it iteratively splits the box and its data points such that the new point sets yield better box approximations of the shape. The splitting step is about finding a plane that results in a good split of the parent box. A good split is the one which minimizes the summed volume of the two resulting child boxes. Such a method demands extensive search and comparison of a lot of planes with different positions and orientations to find a good split plane. Therefore, only the planes parallel to the parent box are considered for the potential partitioning. The splitting is carried out iteratively until a box is not dividable or reaches a minimum limit of data points as shown in Fig. 3.

3.3 Grasp type selection

The idea of dividing an object into a decomposition tree is to capture the local shape descriptions and apply an appropriate type of grasp to respective parts of the object. Moreover, it enables the use of efficient global shape descriptor like Principal Component Analysis (PCA), which is not possible if the object is considered as a whole. In this sub-section, the object parts are classified into number object categories such as one-dimensional, two-dimensional and three-dimensional objects according to their shape distribution in 3D space as shown in Fig. 4. Further, the three-dimensional object is divided into smaller and larger objects based on its volume while a larger object is divided into longer and symmetrical as per shape distribution. Then appropriate grasp strategies such as cylindrical, spherical and fingertip grasps are assigned to each part of the object. The application of PCA [30] on the data points of an object gives information about the object

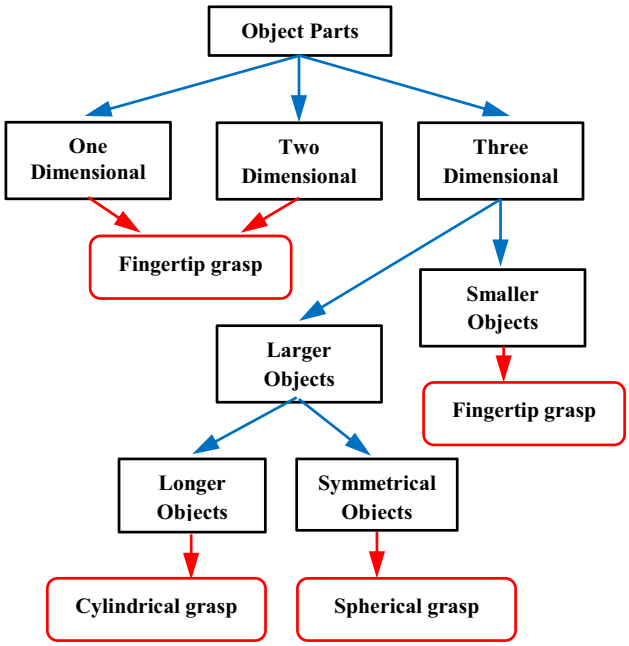

Fig. 4 Object Classification using PCA on object shape information and respective grasp type

distribution in 3D space, for example, eigenvalues denote the extent of object distribution along with the principal directions while eigenvectors give the principal directions.

Let us suppose that an object is a set of 3D points $P \subset \mathbb{R}^3$, and the eigenvalues of the PCA applied on the points set $P \subset \mathbb{R}^3$ be denoted as λ_1, λ_2 and λ_3 where, $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Then, the object is classified into a number of categories using these three components along with volume, for example, the principal component of a one-dimensional object is significantly larger than the other two components i.e., $\lambda_1 \gg \lambda_2 \geq \lambda_3$ (the ratio of 5 has been found for the object dataset used in this paper), for two-dimensional objects $\lambda_1 \cong \lambda_2 \gg \lambda_3$ (the same ratio of 5), and for three-dimensional objects $\lambda_1 \cong \lambda_2 \cong \lambda_3$. In the case of the one or two-dimensional objects, at least one component is significantly larger (at least 5 times) than the others. But for the three-dimensional object, each component does not differ significantly from the other. Now, three-dimensional objects can be a large one or a small one depending upon the actual volume of the object. So, actual dimensions (absolute values of eigenvalues) along the principal components are used to differentiate larger and smaller three-dimensional objects. Finally, a larger object is further divided into longer (e.g., cylinder) and symmetrical (e.g., sphere, cuboid, etc.) objects using the same principle applied for one-dimensional object but with a lower margin, i.e., for a longer object $\lambda_1 > \lambda_2 \geq \lambda_3$ (the ratio of 2).

Different finger configurations enable the hand/gripper to handle a wide range of possible form/force closure grasps. Depending upon the size and shape of the objects shown in

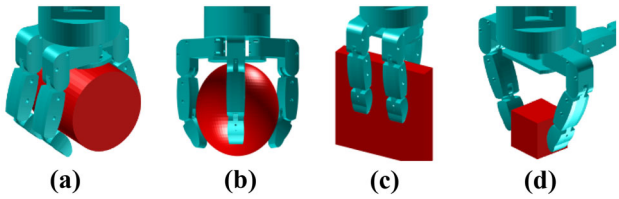

Fig. 5 **a** Cylindrical object grasp, **b** spherical object grasp, **c** two-dimensional flat object grasp, **d** smaller object grasp using fingertips

Fig. 4, a total of three types of grasps has been recognized for this work e.g., cylindrical grasp, spherical grasp and fingertip grasp. The assignment of grasp type to each type of object is inspired by the way the humans grasp an object. For example, a small object cannot be grasped by wrapping the fingers around it but has to use only fingertips, on the contrary use of fingertips for a larger object makes the grasp weaker, aligning the fingers perpendicular to the dominant eigenvector for cylindrical objects (longer) while wrapping the fingers around the spherical objects (symmetrical). The examples of grasping using the three-finger hand shown in Fig. 5 illustrate the concept. Minor changes are required for other hands/grippers as mentioned in the following sections.

3.3.1 Cylindrical grasp

A three-dimensional longer object can be grasped using this configuration. In this configuration, the palm provides support to the object, and all the fingers wrap around the object from two opposite sides as shown in Fig. 5a. Other similar objects can be grasped using this method.

3.3.2 Spherical grasp

Generally, three-dimensional symmetrical objects can be grasped using this configuration. Similar to a cylindrical grasp, the palm provides support to the object. In this grasp, all the fingers place themselves around the object and wrap around the object as shown in Fig. 5b. The spreading of the opposing fingers is similar to the abduction/adduction motion in an anthropomorphic hand. It may be noted that in case of an anthropomorphic hand without abduction/adduction motion or a two-finger gripper, the spherical grasp is replaced by a cylindrical grasp.

3.3.3 Fingertip grasp

Two-dimensional objects can be grasped using the fingertips of all the fingers from two opposite sides of the object as shown in Fig. 5(c). Small three-dimensional objects can be also grasped using the fingertips of the two opposite fingers as shown in Fig. 5(d). The two-finger gripper grasps both the two-dimensional and small three-dimensional objects in similar ways using its two fingertips.

3.4 Reachability, Blocking and Face Mask

Let us suppose that an object is a set of 3D points $P \subset \mathbb{R}^3$, then the decomposition technique partitions the object into a set of object parts $P = \{P_1, ..P_n\}$ which are enclosed by a set of boxes $B = \{B_1, ..., B_n\}$, where, n is the number of partitions. The type of grasps for each of these boxes is selected from the types of grasps available for a particular hand/gripper. Each grasp type puts some constraints on the hand/gripper in terms of finger configuration and hand/gripper alignment with the object. To further reduce the large number of hand/griper approach and alignment possibilities, a reachability analysis is performed. A hand/gripper can reach a box from its six rectangular faces. Now, the two child boxes of a parent box share the common face, and some faces may be occluded by neighboring boxes in the decomposition tree, so all the faces cannot be reached by the hand/gripper. Further, such faces block the fingers when the hand/gripper tries to reach from its adjacent free face. As shown in Fig. 6a, b, each face of a box has four adjacent faces and can be in two states, free or blocked, denoted by white and black colors. The states of the faces are coded in a face-mask matrix (0 for free and 1 for blocked), where each row denotes a face and its four adjacent faces as shown in Fig. 6c. At the time of pre-grasps generation, the hand/gripper is aligned only with the free faces. In addition, a blocked adjacent face eventually reduces the free face-area e.g., a hand/gripper cannot orient its fingers toward the up-face (u) when it is at center-face (c) near to the up-face (u) as shown in Fig. 6b. So, to effectively avoid such situations, the face is divided into sub-faces and associated with the adjacent faces. Only those sub-faces having free adjacent faces are considered for the generation of pre-grasps. The division of the face-area depends upon the types of grasp as shown in Fig. 6d, e.

The encoding scheme can be further extended for obstacle avoidance, where a neighboring object blocks the path of the gripper approaching the graspable object. This can be done by checking reachability of the bounding box of the neighboring objects along with the graspable object parts at the time of face-mask encoding. This serves two purposes; the gripper avoids hitting neighboring objects and reduces the computation time by not generating pre-grasps on the obstructed sides.

3.5 Generation of pre-grasp pool

In the next step, a pool of pre-grasps is generated by sampling enclosing surface areas around the object parts as given in

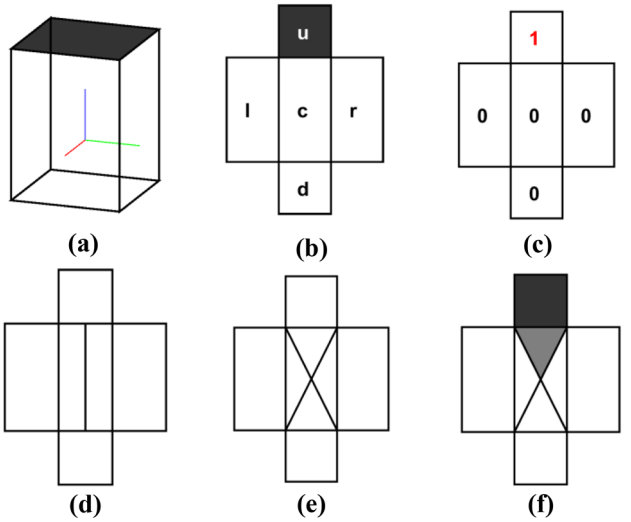

Fig. 6 **a** Bounding box with a blocked face colored black, **b** center face having four adjacent faces left, down, right and up denoted by (l, d, r and u), respectively, **c** face-mask matrix coded with 0 and 1 for free and blocked, respectively, **d**, **e** Sub-face schemes for cylindrical and spherical types of grasps, **f** an example of how a blocked face eventually blocks an adjacent sub-face

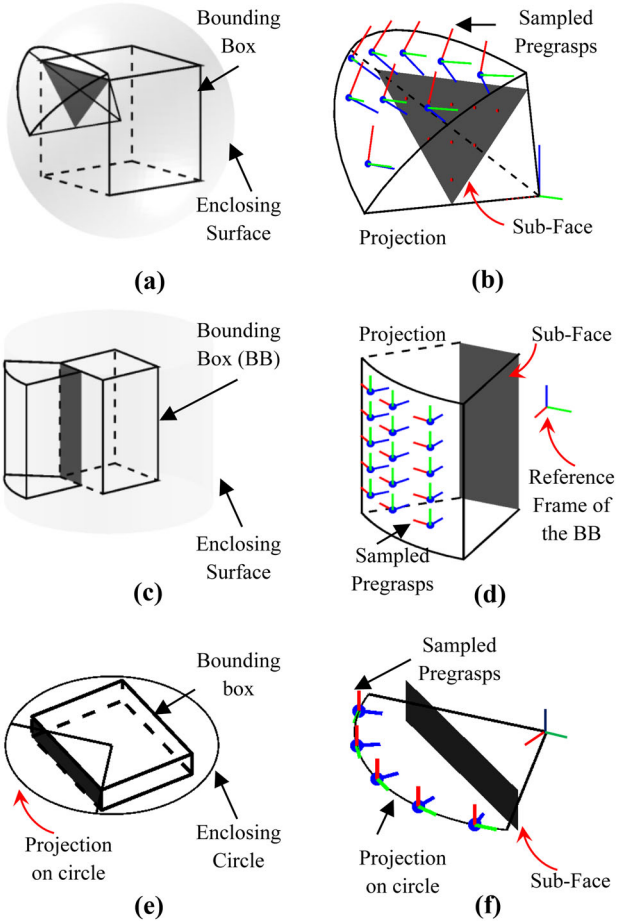

Fig. 7 Different sampling strategies for the generation of pre-grasp pool, **a** the created spherical surface (gray) enclosing the bounding box and sub-face (triangular black) projected on the enclosing surface **b** exploded view of the projection and sampling of the projected surface, **c**, **d** enclosing surface and exploded view of sampling for cylindrical grasp, **e**, **f** projection on an enclosing circle and its sampling for the 2D object using fingertip grasp

algorithm 1. A pre-grasp consists of an initial hand/gripper position, orientation and finger configuration. The orientation gives the approach direction of the hand/gripper toward the object. The enclosing surface is predefined for each grasp type (e.g., sphere for spherical grasp, cylinder for cylindrical grasp, etc.). First, a surface is created which fully encloses the bounding box of the object part (e.g., the gray sphere encloses the box as shown in Fig. 7a). Then all the sub-faces having free adjacent faces are found out by using the face-mask and projected onto the created enclosing surface (e.g., the black triangular sub-face of the bounding box projected on the enclosing sphere as shown in Fig. 7a and its exploded view in Fig. 7b). The projected areas are sampled to get the hand/gripper initial position, where the choice of sampling surface decides the hand/gripper approach direction.

For the spherical grasp, the sub-faces are projected onto the surface of a sphere enclosing the object part, and the projected sub-faces are sampled at a fixed interval, where the hand/gripper approaches along the radial vector as shown in Fig. 7a, b. The cylindrical surface enclosing the object part is used to project the sub-faces for the cylindrical type of grasps as shown in Fig. 7c, d, where the hand/gripper moves along the radial vector for the circular surface and axial-vector for the flat surface of the enclosing cylinder. An enclosing circle is sampled for the fingertip type of grasp (two-dimensional objects), and the hand/gripper is oriented radially as shown in Fig. 7e, f. Similar to the spherical grasp, a spherical surface is used for the fingertip type of grasp (small objects).

Algorithm 1: Generation of Pre-Grasp Pool

```
Procedure PreGraspGeneration(DataPoints, Parent = root)
    DTree = ObjectDecomposition(DataPoints)
    Label: Node = GetCurrentNode(DTree, Parent)
        If Graspable(Node)
            GraspType = GraspSelection(Node)
            NeighNodes = NeighboringNodes(Node)
            BFaces = BlockedFaces(Node, NeighNodes)
            FaceMask = GraspableFaces(NeighNodes, BFaces)
            PreGraspPool =
                SurfaceSampling(Node, FaceMask, GraspType)
        If NotLeafNode(Node)
            Parent = LeftChild(Node)
            GoTo: Label
            Parent = RightChild(Node)
            GoTo: Label
    End
```

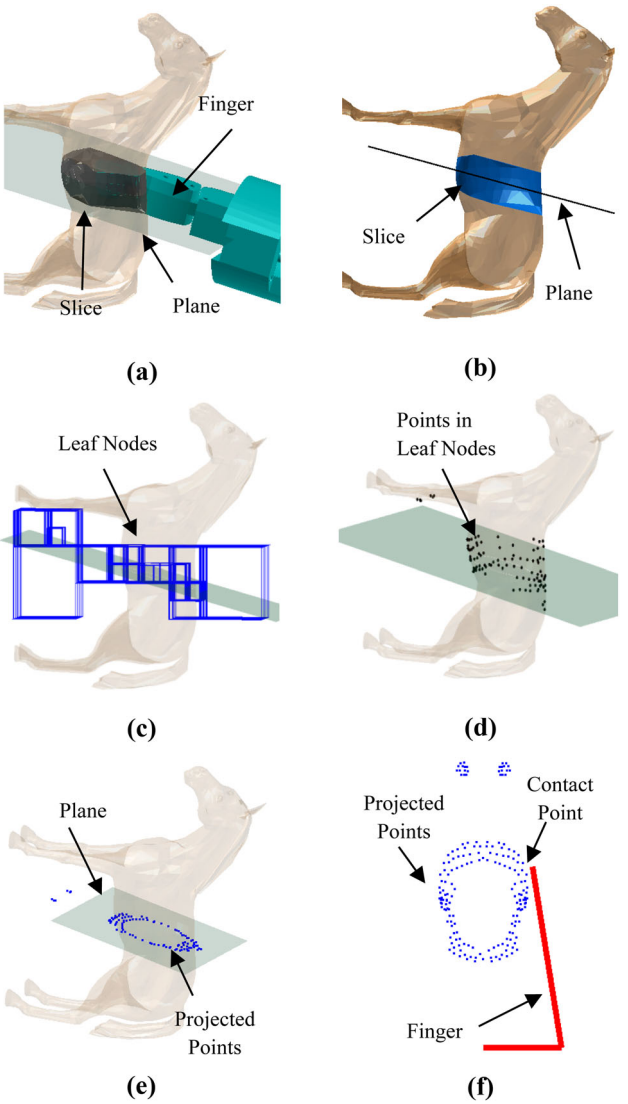

Fig. 8 **a**, **b** Object Slicing along the finger plane, **c** intersection of -planes and leaf nodes of the Octree, **d** data points of the object inside the leaf nodes, (e) Projected points on the plane and **f** contact between a finger and the projected points

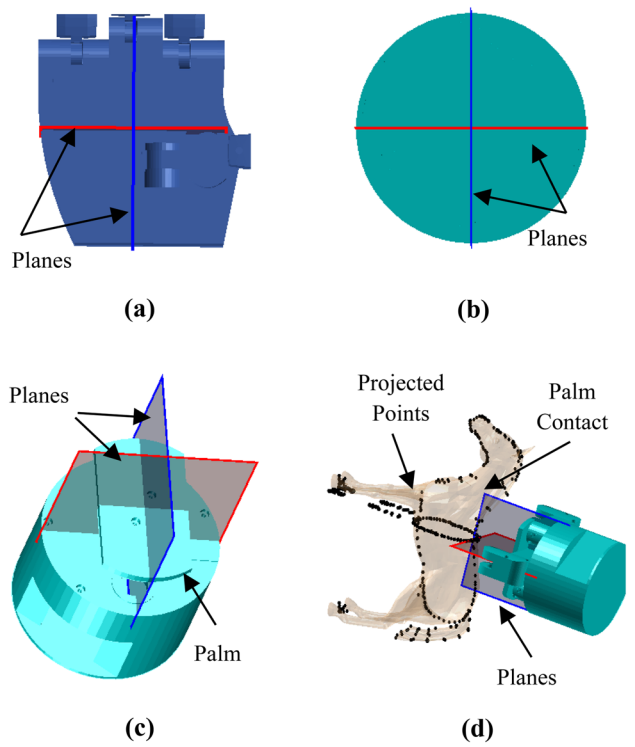

Fig. 9 **a**, **b** The perpendicular planes covering the rectangular and circular palm area, respectively, **c** a close perspective view of two perpendicular planes and the palm, and **d** computing the palm contacts by finding the nearest point from the two sets of projected points

An object consists of a number of parts and different grasp strategies can be applied to the object parts as per the Sect. 3.3. The disadvantage of applying the said method only on the individual object part is that it may miss a good grasp which involves a combination of parts. So to avoid such a scenario, the object parts are stored in a decomposition tree structure (an example shown in Fig. 12) where nodes represent the object parts, and the two parts decomposed from a part are denoted by children (left and right) nodes and parent node, respectively. Then, the process of pre-grasp sampling is iterated over the decomposition tree of the object starting with the part at the root and traversing downwards. At each node, two conditions are checked to decide whether further steps will be performed or not. Firstly, if it has no child node or one of the children is of the type small object, then only further steps are carried out for the current node to generate pre-grasps. Secondly, actual dimensions are checked against a threshold value to decide whether the hand/gripper can hold that particular part or not. For example, there might be a situation where a big part cannot be fitted in a hand/gripper but its child parts can be separately fitted. The second condition successfully prevents a big part but passes the children for the further steps of the pre-grasp generation. All the samples generated from the decomposition tree are stacked to form the pool of pre-grasps for an object.

Algorithm 1 takes the point cloud of an object as input and sets the parent to the root node (i.e., treat the object as a single graspable object for the first iteration). Then the *if* condition checks for whether the part is graspable or not, if graspable then it proceeds further otherwise looks for graspable child nodes. The *GraspSelection()*function selects an appropriate type of grasp for the graspable node as per the method discussed in the Sect. 3.3. Next for finding the graspable faces of the node, first, it finds the neighboring nodes using the *NeighboringNodes()* function and then it looks for the faces blocked by its neighboring nodes using the *BlockedFaces()* function. Then the *GrsapableFaces()* uses the blocked faces information to compute the face-mask as discussed in Sect. 3.4. Finally, the *SurfaceSampling()* generates the pre-grasps

for the graspable node and stores them in *PreGraspPool*.The second *if* condition checks for whether the tree traversal should continue to the next child nodes or stop to its current node. Here, the *NotLeafNode()* function checks the first condition discussed in the previous paragraph. Then, the *LeftNode()* and *RightNode()* functions retrieve the left and right child nodes, respectively, and repeat the method for child nodes by jumping the execution to the *label*.

3.6 Object slicing algorithm

A novel object slicing-based method is proposed to quickly find the grasp points on the object for evaluating the grasp quality. The main idea is that a finger does not make contact with the whole surface of an object. If the object is sliced along the plane of finger flexion, then the finger only makes contact with that slice of the object as shown in Fig. 8a, b. Now, if the data points inside the slice are projected on the plane, then the contact points can easily be computed by considering the finger links as line segments of actual link length and zero thickness. In actual implementation, the data points on the object surface are represented in an Octree data structure for fast spatial search. An Octree [31] is a tree data structure often used to partition a 3D space by recursively subdividing it into eight octants. Then, all the leaf nodes of the Octree that are intersecting with the planes are computed, and all the data points within these leaf nodes are projected onto the plane as shown in Figs. 8(c), 8(d) and 8(e), respectively. Next, the contacts between the projected points and the finger links are computed as shown in Fig. 8(f).

Here, the same idea is extended to find the contact points on the palm by considering two perpendicular planes covering the palm area. The two planes are perpendicular to each other and the palm surface while passing through the center of the palm. The extent of the perpendicular planes to cover the palm area depends on the shape of the palm area. The palm surface is mostly rectangular or circular in shape (e.g., the four-finger and three-finger hands, respectively) and can be covered by appropriately choosing the extents of the planes as shown in Fig. 9a, b.

In this method, for each pre-grasp in the pool, steps in algorithm 2 are applied on the octree structure of the object to compute the possible contacts between the hand/gripper and the object. First, the possible contact points with the palm are computed. Each pre-grasp from the pool gives the initial hand/gripper position and approach direction relative to the object. To find the possible palm contacts, the hand/gripper is placed at the initial position with the palm normal directed along the approach direction. Then, a pair of planes perpendicular to each other and the palm surface while passing through the center of the palm are chosen as shown in Fig. 9c. Next, all the leaf nodes of the octree that are intersecting with the planes are computed and data points within these leaf nodes are projected onto those respective planes as shown in Fig. 9d. Now, the nearest projected points are the first which make contact with the palm as shown in Fig. 9d. For the fingertip grasps, the hand/gripper is kept at a fixed distance from the object, which depends on the maximum reach of the fingers. Once the type of grasp is decided and the hand/gripper is positioned and aligned with the object, each finger flexes on a fixed plane as shown in Fig. 8a. Similarly, all the data points within the leaf nodes intersecting with the plane of finger flexion are projected onto the plane. Next, the flexing joints are closed one by one until the respective link makes contact with the projected points on the plane or maximum joint limits are reached as shown in Fig. 8f. These steps are followed for all the fingers associated with a particular type of grasp.

Algorithm 2: Object slicing based fast contact computation

```
Procedure GraspPointsComputation
    DataPoints = ReadObject()
    OT = OctreeCreation(DataPoints)
    PreGraspPool = PreGraspGeneration(DataPoints)
    Loop: For each PreGrasp in PreGraspPool
       Contacts = ComputeContacts(OT, PalmPlane, Palm)
       Loop: For each Finger
          Loop: For each Link
             Contacts =
                 ComputeContacts(OT, FingerPlane, Link)
          End
       End
    End

Procedure ComputeContacts(OT, Plane, Link)
    Loop: For each Plane
        LeafNodes = Intersection(OT, Plane)
        Points = PointsFrom(OT, LeafNodes)
        ProjectedPoints = Projection(Points, Plane)
        If Link == Palm
           Contact =
               FindNearestPoints(ProjectPoints, Link)
        Else
           Contact =
               FindFirstPoints(ProjectedPoints, Link)
    End
```

3.7 Evaluation of grasp quality

Once, all contacts between the finger links and object are determined, the grasps are ready to be evaluated for stability using grasp quality measure. Two different quality metrics are combined to form a compact grasp quality metric signifying the efficiency of the grasp. The first quality metric determines the maximum external disturbance wrench that can be resisted within a unit grasp wrench space by a grasp [32]. The second quality metric tries to grasp the object closest to the middle.

The steps involved in finding the first quality metric can be best found in Miller and Allen [33]. Let n is the total number

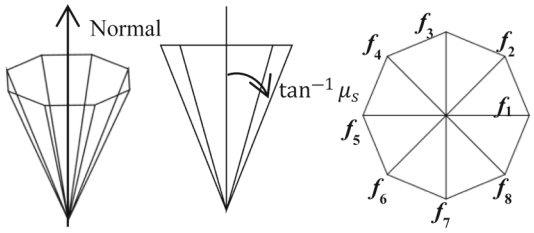

Fig. 10 Approximation of friction cone with an 8 sided pyramid of unit length

of point contacts, and μ_s is the static friction coefficient. Then, the wrench space is the set of forces and torques that can be applied by the fingers on the object through point contacts and is given as follows.

$$\boldsymbol{w}_{i,j} = \begin{pmatrix} \boldsymbol{f}_{i,j} \\ \lambda(\boldsymbol{d}_i \times \boldsymbol{f}_{i,j}) \end{pmatrix} \tag{1}$$

where $\boldsymbol{f}_{i,j}$ is the j-th boundary force vector of the eight-sided pyramid as shown in Fig. 10 at the i-th point of contact. $\boldsymbol{d}_i$ is the distance from torque origin to the i-th point of contact. Multiplier λ is chosen to enforce $\|\boldsymbol{\tau}\| \leq \|\boldsymbol{f}\|$.

The convex hull from the wrench space represents the set of wrenches that can be applied on the object given that the sum total of the contact normal forces is one.

$$W = ConvexHull\left(\bigcup_i^n \{\boldsymbol{w}_{i,1}, \boldsymbol{w}_{i,2}, \ldots.., \boldsymbol{w}_{i,m}\}\right) \tag{2}$$

The grasp is stable only if the origin of the wrench space lies within the convex hull. The one quality measure that is widely accepted is the distance ϵ from the origin to the closest facet of the convex hull. This signifies the smallest maximum wrench that can be exerted by the grasp. The scale of ϵ varies from 0 to 1, closer to 1 signifies a more efficient/stable grasp. However, this measure is not invariant to the choice of torque origin. The volume v of the convex hull can be used as an invariant quality measure.

Intuitively, on a daily basis if the object is grasped at the middle, then it becomes easier to hold the object. Torque induced by the gravity becomes greater as the objects are grasped further from the center of gravity. So, the second quality metric d is defined as the distance between the center of all contact points and the center of gravity. The normalized quality metrics ϵ or v, and d are combined to form a compact grasp quality metric as follows.

$$\mathrm{Q} = \frac{1}{(1+\mathrm{d})} + \mathrm{x} \tag{3}$$

$$\text{Where } x = \begin{cases} \epsilon, distance\, measure\, of\, the\, convex\, hull \\ v, volumn\, measure\, of\, the\, convex\, hull \end{cases}$$

The above quality measure is used to rank all the grasps in the grasp pool, and the best grasp is chosen.

3.8 Computation of contacts and joint displacements

In the previous sub-section, grasps have been computed by object slicing-based method, which quickly finds the hand/gripper position, orientation and contact information. A polygonal mesh model of the hand/gripper can be used to find the contact information for all the grasps in the pool, but the time taken for such procedure is very high compared to the proposed approach. For example, the times required for a single collision query for mesh-mesh and point cloud-mesh of 5 k points are, respectively, 50–60 ms and 500–1000 ms by Proximity Query Package (PQP) library [34] and Flexible Collision Library (FCL) [35], respectively. So, once the best grasp is found, a more accurate method is applied to find the precise contacts and the hand/gripper joint displacements. The fingers are closed until contacts are found or maximum joint limits are reached.

3.8.1 Point cloud-mesh

In order to handle point cloud data, a new proximity query algorithm is proposed, which computes contacts between an object and the robot hand/gripper modeled as a point cloud and triangulated mesh, respectively. Here, it is assumed that a complete point cloud of the object is available. The point cloud of the object and the triangulated mesh of the hand/gripper are represented in the Octree data structure. Oriented Bounding Boxes (OBBs) are used to tightly fit the geometry of the articulated hand/gripper links and the object. Then the Octrees are constructed for each of the OBBs. The proximity query algorithm is made faster by culling the points on the object and triangles on the finger links that are not likely to make contacts. The culling is done in two stages. In the first stage, an intersection test is done between the bounding box of a link (which is the root node of the link Octree) and the object Octree. A temporary bounding box is created around all the points within the leaf nodes that are intersecting with the bounding box of the link. Next, another intersection test is performed between the temporary bounding box and the link Octree, which gives all the possible triangulated facets that can be in contact with the object. Further, culling is done in the second stage before finally pair wise closest distance computation between the points on the object and triangulated facets on the hand is performed. In the second stage, it tries to find a separating plane that separates the points on the object and the triangulated facets on the link found in the first stage. A plane passing through the side of the temporary bounding box and near to the link is chosen for the test. If no triangle intersects

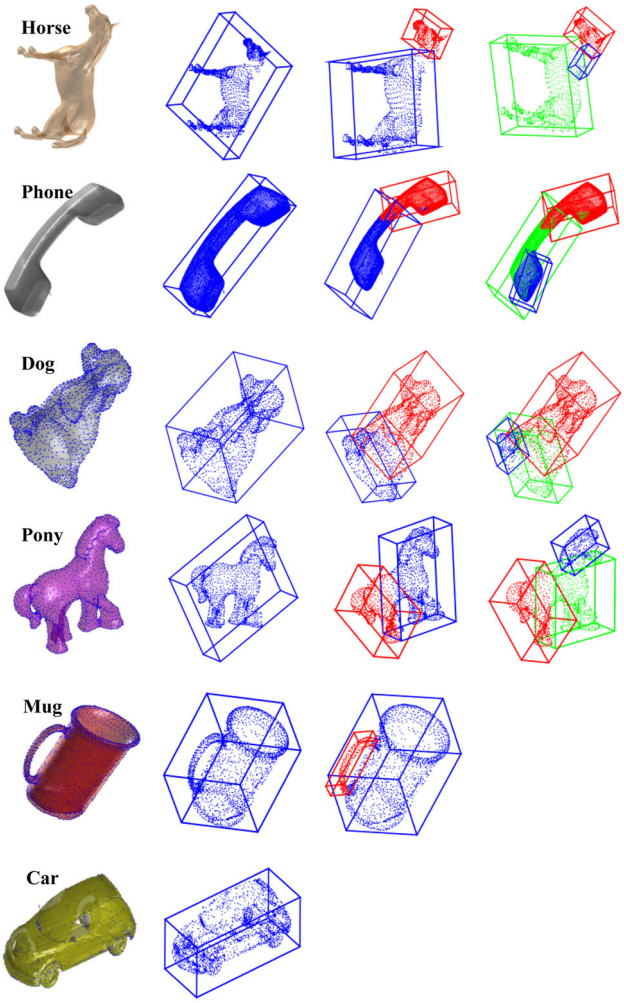

Fig. 11 Object decomposition results at different splitting stages. The first two objects are polygonal mesh models, and the rest are point cloud models

with the plane then the algorithm jumps to the next iteration. Otherwise, further steps are performed only on the triangle faces that are intersecting with the plane. Pairwise minimum distances between the points on the object and the triangle faces are computed, and the pairs having distance less than a threshold are supposed to be in contact.

3.8.2 Mesh-mesh

The robot hand and the objects are modeled as triangulated mesh. First, the hand is placed according to the best grasp found as described in the Sects. 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7. Then, the Proximity Query Package (PQP) library [32] is used to find the contact information, where all the fingers are moved until contacts are found or maximum joint limits are reached.

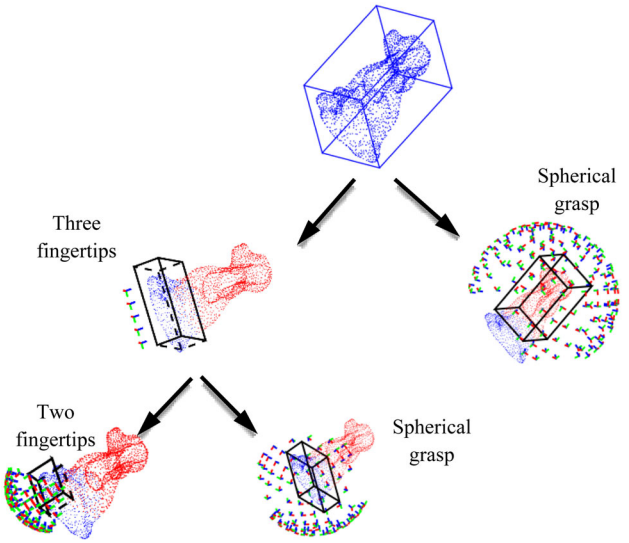

Fig. 12 Type of grasp selection and pre-grasps generation at different stages of the decomposition tree

4 Results and discussions

The implementation has been shown on three-different grippers/hands as illustrated in Fig. 1. A total of forty-eight common household objects and toys have been chosen for performing grasping. Eighteen mesh models are taken from the Princeton Shape Benchmark [36], twenty-four scanned point clouds of real objects are taken from the KIT dataset [37], and six objects are designed. The scanner Konica Minolta Vi-900 had been used to build the KIT dataset, which has a high point accuracy of 50–170 μm at a distance of 0.6 m. All the chosen objects for the evaluation have data points within the range of 2500–3000 points and around 5000 triangle faces. The implementation has been done using Matlab platform running on a PC with configuration Intel Core i5 4570U CPU at 3.2 GHz, 16 GB RAM. Here, only results of a subset of 10 objects with diverse shapes and sizes from the total of 48 objects are given.

4.1 Pre-grasps generation results

Figure 11 shows the object decomposition results at different splitting stages, where the parent box volume to the summed child boxes volume ratio of 0.9 and minimum data of 500 points (experimentally found over the object dataset of 48 objects) in a box are used as termination conditions for the splitting. The decomposition trees have 3, 3, 3, 3, 2 and 1 number of parts for the horse, phone, dog, pony, mug and toy-car, respectively. Figure 12 shows an example of the tree traversal for pre-grasp generation, which starts at the root and goes downwards. The final pre-grasp pool is shown in Fig. 13, where pre-grasps for each part in the decomposition tree are

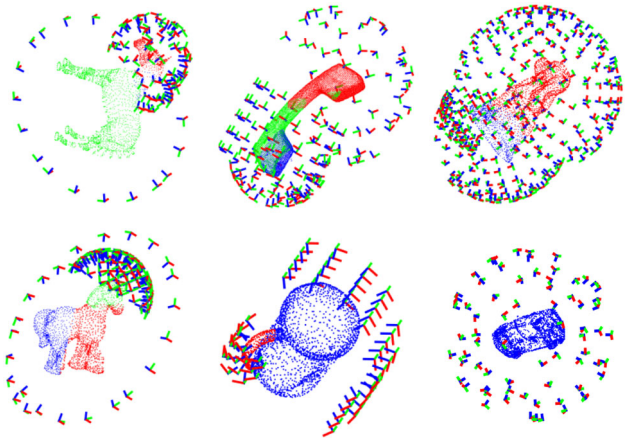

Fig. 13 The final pool of pre-grasps for the six example models

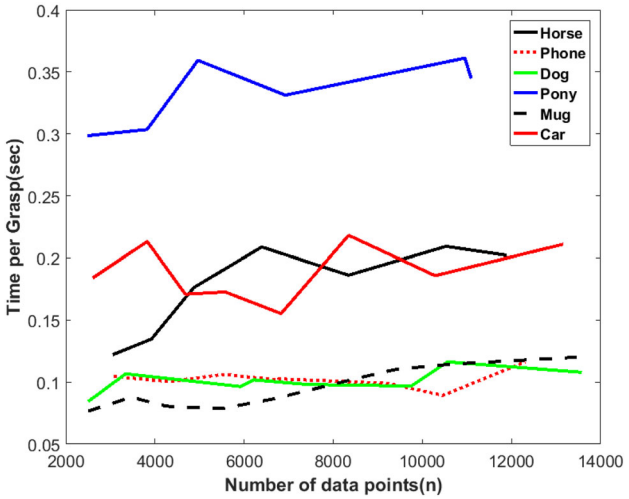

Fig. 15 Analysis of time per grasp for various objects with varying 3D data points

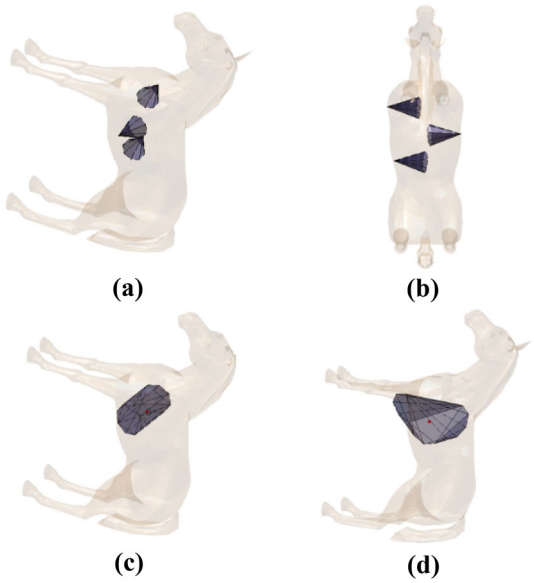

Fig. 14 **a**, **b** Contact friction cones with approximated eight sides, **c**, **d** Convex hull for contact force and torque components, respectively

assembled to form the final pool. In case of a small object (e.g., toy-car as shown in Fig. 11), no object decomposition is needed, and pre-grasp generation is applied directly on the object.

In the example of the dog model, three types of grasps can be applied as shown in Fig. 12, but only the spherical type of grasp has been automatically selected by the grasp planner. It is because of the three fingers involved in the spherical grasp make more number of contacts with the object than the other types of grasps and the more number of contacts make the grasp more stable. Similarly, a three-fingertip grasp is chosen over two-fingertip grasp for the pony, and a cylindrical grasp is chosen over two-fingertip grasp for the mug.

4.2 Object slicing results

Fig. 8 shows an example of object slicing and the contact computation between an object and a finger. All the data points within the slice are projected on the finger plane and then the contacts with the finger are computed. Fig. 14 shows the friction cones and wrench space for force and torque components, respectively. All the generated grasps for each object are tested using the grasp quality measure as discussed in Sect. 3.7. Quantitative results of the proposed object slicing-based grasp planner for point clouds as well as mesh models are given in Table 1 (Here, only results for the three-finger hand is given). The set of found grasps (N_f) is the number of grasps found from the pool of pre-grasps having good grasp quality.

The time per grasp ($\boldsymbol{t}$) is the average computation time (in seconds) to get a good grasp, which is the ratio of the total run time ($\boldsymbol{T}$) to the total number of good grasps $(\boldsymbol{N_f})$. Analysis has been carried out to study the effect of the number of 3D data points on the computation time of the proposed grasp planner. It is expected that the graph to be monotonically increasing with logarithmic complexity for the use of Octree data structure. But the increasing of the data points has a positive effect on the contacts point finding, where some contact points may have been missed due to insufficient data points. Nevertheless, it is found that increasing the number of data points has little effect on the time per grasp (t) as shown in Fig. 15 for various objects. The computation time rises around 2 to 3 times for the range of 2500–14,000 data points.

Table 1 Results of the proposed object slicing-based planner (The three-finger hand)

Type of data representation	Objects	Pre- grasps pool size (N)	Found grasps (N_f)	Pre-Grasps Generation time (T_g)	Total slicing stage time (T_s)	Total time $T = T_g + T_s$	Time per grasp $t = T/N_f$
Mesh models	Bottle	192	125	0.0700 s	6.4285 s	6.4985 s	0.0520 s
	Horse	76	19	0.0981 s	2.2217 s	2.3198 s	0.1221 s
	Phone	122	34	0.1258 s	3.4287 s	3.5545 s	0.1045 s
Point clouds	Dog	168	73	0.1523 s	6.0038 s	6.1561 s	0.0843 s
	Pony	79	8	0.0820 s	2.2675 s	2.3495 s	0.2937 s
	Mug	64	22	0.0389 s	1.6460 s	1.6849 s	0.0766 s
	Car	80	10	0.0305 s	1.7369 s	1.7674 s	0.1767s
	Cat	144	79	0.0573 s	5.2090s	5.2663 s	0.0667 s

4.3 Contact computation results

All the object point clouds are noise free as a number of filtering and smoothing operations had already been performed on the raw scanned data at the time of building the KIT dataset. So, to study the robustness of the proposed planner, Gaussian noise has been added artificially to the object point clouds at varying levels. The noise varies depending upon the sensing devices, e.g., the laser-based Konica Minolta Vi-900 has an accuracy of 50–170 μm, whereas infrared (IR)-based Kinect has an accuracy of 1–1.5 mm at a distance of 0.6 m. Figure 16 shows the effect of different levels of noise on the grasp quality of the proposed planner. The average grasp quality changes between 1%- 5% for the noise level of 0.05–0.5 mm, whereas 17–38% for the noise level of 1–2 mm. The final results of the grasp planner at contact-level for various objects using the three hands/grippers are shown in Fig. 17. The threshold value of 0.3 mm is used for the point cloud-mesh and mesh-mesh proximity query algorithms. The average time per proximity query of the proposed method is 50-80 ms for mesh-point cloud collision.

4.4 Experimental results

The planned grasps are further validated by executing the grasps using a two-finger gripper mounted on an industrial Motoman robot. The object point cloud is captured using two Intel-Realsense depth cameras placed 180 degrees apart and pointing toward the object as shown in Fig. 18. The underlying point cloud data acquisition is out of the scope of this paper, but a brief description is given for better understanding. Firstly, the object is segmented out by removing the background from the raw point clouds taken from the depth cameras. Then, both the point clouds are transformed into the robot base frame and simply merged to form a single point cloud, which constitutes the visible parts of the object surface. A hole filling technique is used to fill the holes, mostly, at the joining areas of the two point clouds. The point cloud captured in such a way missed the bottom side as the object is resting on the supporting plane (the tabletop). On the other hand, information about the bottom surface of the objects is not needed or redundant for grasp planning as the table surface also prevents the gripper to reach there. Further, the tabletop is included as a constraint at the time of computing pre-grasp pool considerably reducing the pool size compared to the one with no such constraints as shown in Fig. 13.

Further, the face-mask encoding scheme has been extended to handle the situation, where a neighboring object blocks the path of the gripper approaching the graspable object. This is done by including the neighboring objects along with the graspable object parts for checking reachability at the time of face-mask encoding. Figure 19 shows two instances of grasping operation, with and without the presence of neighboring objects.

Finally, the examples of successful stable grasps on a few real objects during a simple pick and place operation are shown in Fig. 20. It is to be noted that the finger joint displacements computed using the method given in Sect. 3.8, only, ensure the fingers making contacts with the object. The fingers not only reach the object but also need to apply appropriate gripping force for holding the object. So, an impedance controller is implemented on the gripper to generate gripping force as well as control joint displacements, simultaneously.

4.5 Comparative evaluation with the state-of-the-art

In this sub-section, a comprehensive comparison with the existing state-of-the-art including recent work has been presented. Three grasp planners [9, 11] and [12] have been found to be most relevant and chosen for evaluating the proposed grasp planner quantitatively and qualitatively. The computation time and the grasp quality measure described in Sect. 3.7 are the two main criteria for the quantitative and qualitative comparisons. For a platform-independent fair evaluation, all the grasp planners have been implemented with the author's

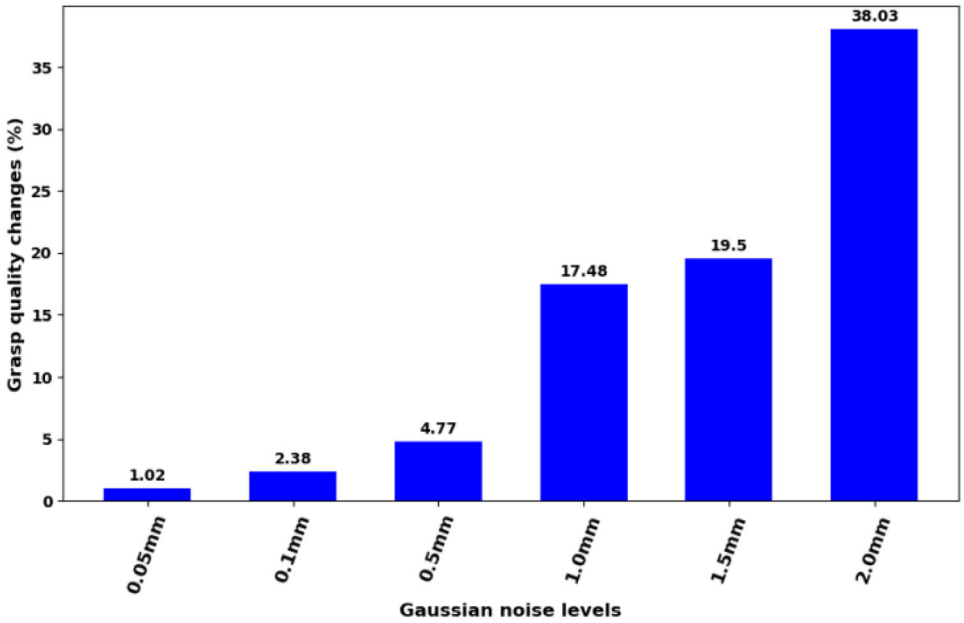

Fig. 16 Effect of different level of noise on the proposed planner

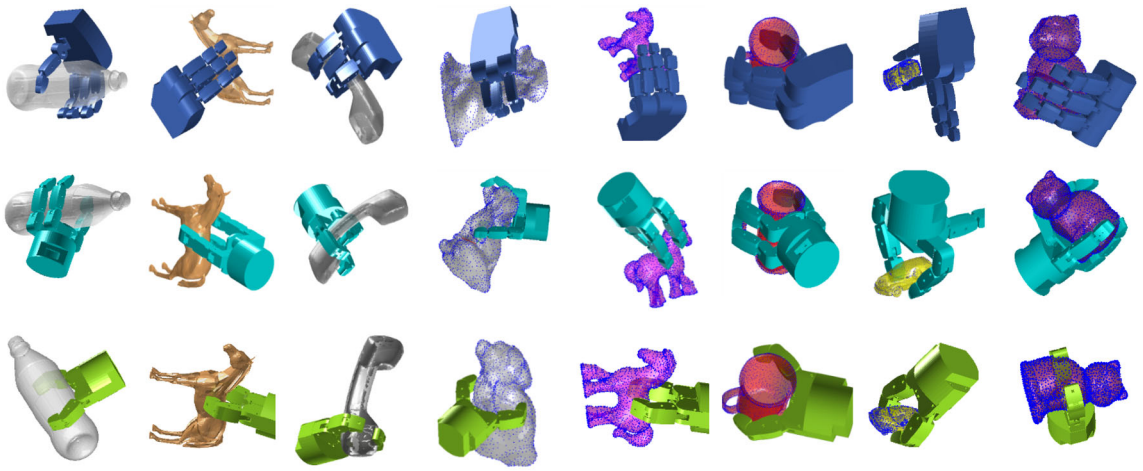

Fig. 17 Final results of the contact-level grasp planner for various objects found using the object slicing method and on three-different grippers/hands

best understanding, and the same grasp quality measure metric as discussed in Sect. 3.7 has been used. Three-dimensional Octrees have been used for implementing ray tracing in the grasp planners [11, 12] to speed up the spatial search. The Proximity Query Package (PQP) library [32] is used for distance queries between two mesh models, which is required for implementing the grasp planners [9, 11]. The implementation results for all the planners presented in this paper are at par with the results given in the respective papers.

The grasp planner presented by Li et al. [12] is the most recent and relevant work found in the literature. The planner has several advantages over its predecessor planners. Although multiple cords (which is computationally more expensive) have been used in the method proposed by Li et al., here for simplicity, only a single cord has been used

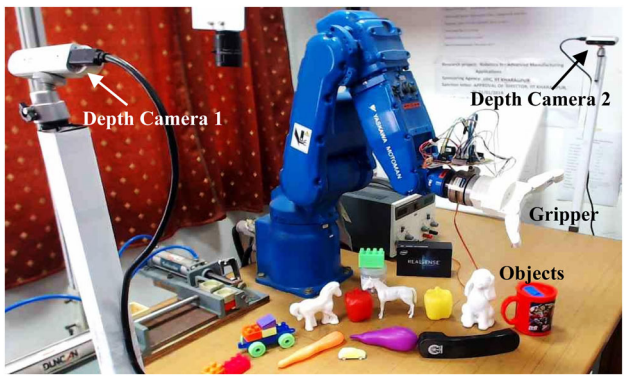

Fig. 18 The experimental setup for performing simple pick and place operation using a gripper mounted on the Motoman robot

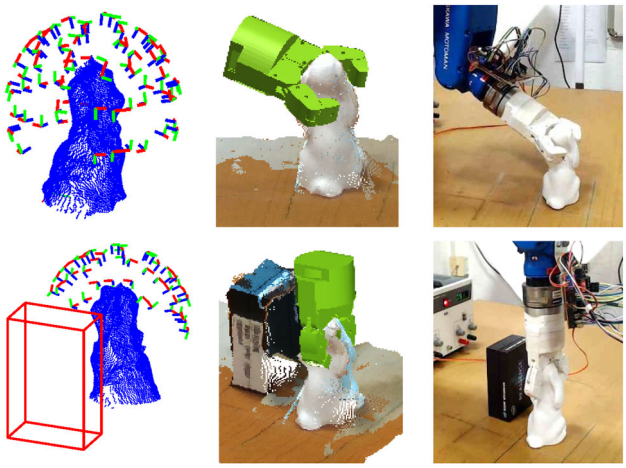

Fig. 19 The computation of the best grasp and its execution on a real robot platform with and without the presence of neighboring objects

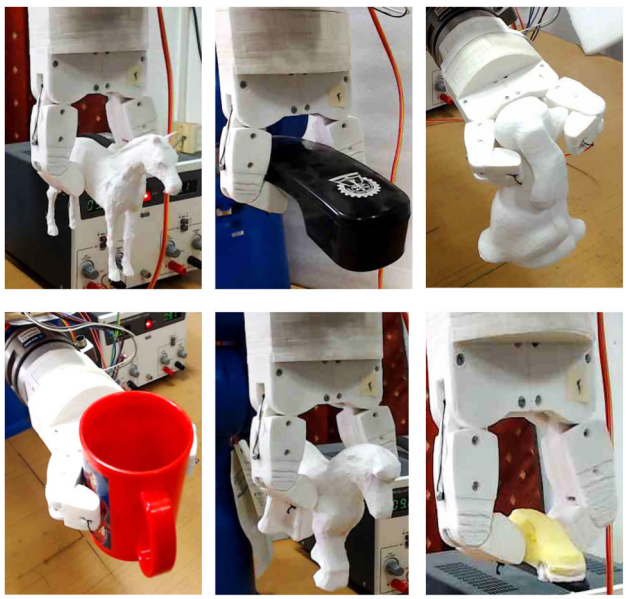

Fig. 20 Examples of gripper successfully performing stable grasping on a few real objects

and the computation time per grasp for the various object is given in Table 2. On comparing the time per grasp, it is found that the proposed method is nearly 5–35 times faster depending upon the types of grasps. Unlike the work presented by Li et al., which is based on ray tracing, the proposed planner works successfully even on objects with small holes on the surface as it does not need facet information. Further, a single cord around the region is not enough to capture whether the region is good for grasping. To overcome this, Li et al. had used multiple cords which, however, give rise to an increase in computation time. But in the method that is presented in this paper, the entire contact region is explored to find possible contacts by projecting on the finger plane in a single step, which is computationally more efficient. Another advantage offered by the proposed planner is that it does not depend on the underlying representation of the object model. It works on both polygon mesh models and point clouds taken using a depth sensor without any expensive processing such as surface reconstruction. This makes the proposed planner robust as it can handle different object representations e.g., polygonal mesh, unstructured point clouds, etc.

The power grasp (i.e., enveloping grasp) planner [11] based on object surface matching is the second grasp planner. The grasp planner is best suited for primitive volumes like cylinders, boxes and cones. The planner uses ray tracing for sampling the object surface and can only be applied on the polygonal mesh model. The computation time is much higher than the proposed grasp planner.

The third grasp planner is the primitive shapes-based planner proposed by Miller et al. [9]. The planner has shown promising results and has been widely used by the grasp planning research community. The main shortcomings of the planner are as follows. The objects were manually approximated into primitive shapes for the pre-grasps generation. Although, they had automated the pre-grasps generation in [24] by using a decomposition tree based on super-quadrics approximation of the object parts, the computation time for the planner is much higher than the proposed planner. Besides, it did not mention any procedure to handle occlusion by the neighboring parts. Unlike the proposed planner, it cannot be applied directly on the point cloud data and can only be applied after converting the point cloud to a mesh model by using expensive surface reconstruction.

On comparing with the state-of-the-art in grasp planning, the results are summarized as follows.

i. Unlike the previous work [9–15] and [26], the proposed planner works on both point clouds taken using depth sensors and objects represented as polygonal mesh, which eliminates the need for expensive processing like surface reconstruction for converting point clouds to mesh models.
ii. The generation of pre-grasps has been automated by using object decomposition and employing part-based grasping, whereas in [9], it was done manually. The use of PCA gives natural-looking hand alignment with the object. Additionally, a face-mask scheme for encoding the blocked faces and associated rules have been introduced to prune the pre-grasps, which noticeably reduce infeasible grasps at the pre-grasp generation stage.
iii. The proposed planner does not use ray tracing as in [11, 12] for mesh models, so it works successfully even on objects with small holes, bumps etc., on the surface. As the planner uses data points on the surface instead of polygonal faces, it can handle non-manifold meshes.

Table 2 Comparison of time per grasp of the proposed planner with the existing grasp planners

Objects	Object slicing-based grasp planner	Cord wrapping-based grasp planner by Li et al. [12]	Surface matching-based grasp planner by Roa et al. [11]	Shape primitives-based Grasp planner by Miller et al. [9]
Bottle	0.0520 s	0.6769 s	14.6798 s	04.8749 s
Horse	0.1221 s	2.2953 s	17.5332 s	11.7704 s
Phone	0.1045 s	1.3806 s	24.4810 s	13.6206 s
Dog	0.0843 s	2.9229 s	15.0040 s	06.6544 s
Pony	0.2937 s	1.9018 s	11.9557 s	05.2607 s
Mug	0.0766 s	1.9485 s	19.1141 s	13.3994 s
Car	0.1767s	0.9571 s	12.3070 s	03.6456 s
Cat	0.0667 s	1.4759 s	09.3642 s	06.0654 s

Table 3 Comparison of the average quality of top five grasps from the grasp pool of the proposed planner with the existing grasp planners

Objects	Object slicing-based grasp planner	Cord wrapping-based grasp planner by Li et al. [12]	Surface matching-based grasp planner by Roa et al. [11]	Shape primitives-based Grasp planner by Miller et al. [9]
Bottle	0.8041	0.5280	0.6526	0.7390
Horse	0.8485	0.4045	0.6189	0.6362
Phone	0.6687	0.6346	0.6928	0.6510
Dog	0.8529	0.5698	0.7516	0.7241
Pony	0.7793	0.7340	0.4809	0.7688
Mug	0.7655	0.5084	0.6304	0.7162
Car	0.5223	0.7032	0.6588	0.7972
Cat	0.8364	0.6655	0.6529	0.7713

iv. The proposed grasp planner is computationally less expensive than the previous grasp planners [9–14] and [26], which makes the planner suitable for online grasp planning. Further, increasing the number of data points has little effect on the computation time.
v. The grasp quality of the proposed grasp planer is comparable with the previous grasp planners. A significant increase in the quality can be seen in Table 3 for most of the objects, while the rest are comparable if not better than the other grasp planners.

4.6 Limitations

The proposed planner assumes that complete geometric information about the objects is available i.e., the point cloud of the whole surface or complete mesh model of the object. So, it will not work on the point cloud taken using a depth sensor from a single view, which gives only partial information about the object. A number of methods are available in the literature [38] to reconstruct the complete geometry from the partial object information, which can be used as a pre-processing stage to the proposed gasp planner.

Another limitation of the proposed method is that it fails on objects having sparse data points especially, for the coarse mesh model. The data points on the surface of such objects are up-sampled to increase the number.

The current implementation of the fit and split algorithm based on MVBB [29] fails to decompose objects having symmetric parts. For example, the decomposition results of the horse model as shown in Fig. 11, where the legs and the main body part are symmetric about all the three axes of the bounding box and the algorithm fails to divide further into legs and body parts. The effect of this limitation on the grasp planner is that some good grasps might be missed.

5 Conclusions

In this paper, the problem of finding grasp plans based on a novel object slicing-based method has been developed and implemented on three-different grippers/hands. Object part-based grasping has been implemented for arbitrary 3D objects. The object decomposition and the generation of pre-grasps for each part have been automated by a fit and split algorithm based on Minimum Volume Bounding Box (MVBB) and PCA-based object shape categorisation. The

average computation time per grasp has been found to remain nearly the same for all the objects. Moreover, the computation time rises by only around 2 to 3 times for the objects having data points within the range of 2500–14,000. Further, a comprehensive comparison with the existing state-of-the-art including recent work has been presented. The proposed planner has been found to be computationally less expensive than the previous grasp planners, which makes the planner suitable for online grasp planning. The planner has been tested on common household objects using a two-finger gripper mounted on an industrial Motoman robot. Further, finding the best feasible grasps has been demonstrated by handling scenarios such as avoiding the tabletop and surrounding objects.

In future, it is planned to address the limitations and extend the work for partial point clouds taken using depth sensors from a single view. The stability of the grasps can be improved by considering the local surface patch for quality measurement at the contacts instead of the point contact friction model.

References

1. Shimoga KB (1996) Robot grasp synthesis algorithms: a survey. Int J Robot Res 15(3):230–266
2. Sahbani A, Khoury S, Bidaud P (2012) An overview of 3D object grasp synthesis algorithms. Robot Autonom Syst 60(3):326–336
3. Bohg J, Morales A, Asfour T, Kragic D (2013) Data-driven grasp synthesis—a survey. IEEE Trans Rob 30(2):289–309
4. Napier J (1956) The prehensile movements of the human hand. J Bone Joint Surg 38B(4):902–913
5. Goldfeder C, Allen PK, Lackner C, Pelossof R (2007) Grasp planning via decomposition trees. In: International conference on robotics and automation, pp 4679–4684
6. Prattichizzo D, Trinkle JC (2008) Grasping. In: Siciliano B, Khatib O (eds) Springer handbook of robotics. Springer, Heidelberg, pp 631–700
7. Suarez R, Roa M, Cornella J (2006) Grasp quality measures. Institute of Industrial and Control Engineering, Technical University of Catalonia. Barcelona, Spain, Tech. Rep
8. Sainul IA, Deb S, Deb AK (2019) A novel object slicing based grasp planner for 3D object grasping using underactuated robot gripper. In: IECON 2019–45th annual conference of the IEEE industrial electronics society, vol 1, pp 585–590
9. Miller A, Knoop S, Christensen H, Allen P (2003) Automatic grasp planning using shape primitives. In: IEEE international conference on robotics and automation, ICRA, pp 1824–1829.
10. Li Y, Saut JP, Cortes J, Simeon T, Sidobre D (2011) Finding enveloping grasps by matching continuous surfaces. In: IEEE international conference on robotics and automation, pp 2825–2830
11. Roa M, Argus M, Leidner D, Borst C, Hirzinger G (2012) Power grasp planning for anthropomorphic robot hands. In: IEEE international conference on robotics and automation, pp 563–569
12. Li Y, Saut JP, Pettre J, Sahbani A, Multon F (2015) Fast grasp planning using cord geometry. IEEE Trans Robot 31(6):1393–1403
13. Xue ZJ, Zoellner M, Dillmann R (2007) Grasp planning: find the contact points. In: IEEE international conference on robotics and biomimetics, pp 835–840
14. Saut JP, Sidobre D (2012) Efficient models for grasp planning with a multi-fingered hand. Robot Auton Syst 60:347–357
15. Shi J, Koonjul GS (2017) Real-time grasping planning for robotic bin-picking and kitting applications. IEEE Trans Autom Sci Eng 14(2):809–819
16. Roa MA, Suarez R (2009) Computation of independent contact regions for grasping 3-D objects. IEEE Trans Robot 25(4):839–850
17. Rosales C, Ros L, Porta JM, Suarez R (2011) Synthesizing grasp configurations with specified contact regions. Int J Robot Res 30(4):431–443
18. Song P, Fu Z, Liu L (2018) Grasp planning via hand-object geometric fitting. Vis Comput 34:257–270
19. Hang K, Li M, Stork JA, Bekiroglu Y, Pokorny FT, Billard A, Kragic D (2016) Hierarchical fingertip space: a unified framework for grasp planning and in-hand grasp adaptation. IEEE Trans Robot 32(4):960–972
20. Hang K, Stork JA, Pollard NS, Kragic D (2017) A framework for optimal grasp contact planning. IEEE Robot Autom Lett 2(2):704–711
21. Zheng Y (2018) Computing the best grasp in a discrete point set with wrench-oriented grasp quality measures. Auton Robot 43(4):1041–1062
22. Cutkosky MR, Wright PK (1986) Modeling manufacturing grips and correlation with the design of robotic hands. In: Proceedings of the 1986 IEEE international conference on robotics and automation, pp 1533–1539
23. Wren D, Fisher R (1995) Dexterous hand grasping strategies using preshapes and digit trajectories. In: IEEE international conference on systems, man and cybernetics, pp 910–915
24. Goldfeder C, Allen PK, Lackner C, Pelossof R (2007) Grasp planning via decomposition trees. In: International conference on robotics and automation, pp 1050–4729
25. Huebner K, Kragic D (2008) Selection of robot pre-grasps using box-based shape approximation. In: IEEE/RSJ international conference on intelligent robots and systems, pp 1765–1770
26. Vahrenkamp N, Westkamp L, Yamanobe N, Aksoy EE, Asfour T (2016) Part-based grasp planning for familiar objects. In: IEEE-RAS 16th international conference on humanoid robots (humanoids), Cancun, pp 919–925
27. Ulrich N, Paul R, Bajcsy R (1988) A medium-complexity complaint end effector. In: IEEE international conference on robotics and automation, pp 434–436
28. Sainul IA, Deb S, Deb AK (2016) A three finger tendon driven robotic hand design and its kinematics model. In: International conference on CADCAM, robotics and factories of the future, pp 313–321
29. Huebner K, Ruthotto S, Kragic D (2008) Minimum volume bounding box decomposition for shape approximation in robot grasping. In: IEEE international conference on robotics and automation, pp 1628–1633
30. Jolliffe IT (2002) Principal component analysis. 2nd ed. Springer
31. Meagher DJR (1980) Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3-D objects by computer. Rensselaer Polytechnic Institute, New York, Tech. Rep. IPL-TR-80-1111980
32. Ferrari C, Canny J (1992) Planning optimal grasps. In: International conference on robotics and automation, pp 2290–2295
33. Miller AT, Allen PK (1999) Examples of 3D grasp quality computations. In: IEEE international conference on robotics and automation, pp 1240–1246
34. Eric L, Stefan G, Ming CL, Manocha D (2000) Fast proximity queries with swept sphere volumes. In: International conference on robotics and automation, pp 3719–3726
35. Pan J, Chitta S, Manocha D (2012) FCL: a general purpose library for collision and proximity queries. In: International conference on robotics and automation, pp 3859–3866

36. Shilane P, Min P, Kazhdan M, Funkhouser T (2004) The princeton shape benchmark. In: Proceedings shape modeling applications, pp 167–178
37. Kasper A, Xue Z, Dillmann R (2012) The KIT object models database: an object model database for object recognition, localization and manipulation in service robotics. Int J Robot Res 31(8):927–934
38. Varley J, De Chant C, Richardson A, Ruales J, Allen P (2017) Shape completion enabled robotic grasping. In: IEEE/RSJ international conference on intelligent robots and systems, pp 2442–2447