Three-Finger Precision Grasp on Incomplete 3D Point Clouds

Ilaria Gori\textsuperscript{1}, Ugo Pattacini\textsuperscript{1}, Vadim Tikhanoff\textsuperscript{1} and Giorgio Metta\textsuperscript{1}

\textbf{Abstract}—We present a new method for three-finger precision grasp and its implementation in a complete grasping toolchain. We start from binocular vision to recover the partial 3D structure of unknown objects. We then process the incomplete 3D point clouds searching for good triplets according to a function that weighs both the feasibility and the stability of the solution. In particular, while stability is determined in a classical way (i.e. via force-closure), feasibility is evaluated according to a new measure that includes information about the possible configuration shapes of the hand as well as the hand’s inverse kinematics. We finally extensively assess the proposed method using the stereo vision and the kinematics of the iCub robot \cite{1}.

\section{I. INTRODUCTION}

Object manipulation is a crucial robotic skill. It is certainly one of the key enabling technologies in a variety of different robotic domains. Manipulation is also a fertile research topic due to its intrinsic complexity. The dimensionality of the hand & arm’s configuration space and the difficulty of retrieving accurate visual priors are perhaps the two major factors to its complexity. In Napier’s taxonomy \cite{2} grasp actions are classified in power grasps and precision grasps. In power grasp \cite{3, 4} and \cite{5} the object and the hand share large contact areas, preventing further movements of the fingers. On the other hand, in precision grasp, the object is contacted with the tips of the fingers. Power grasps are particularly suited when the object does not need to be handled precisely, whereas precision grasps are useful when specific tasks have to be fulfilled as e.g. in tool use. This work focuses on precision grasp.

Generating a precision grasp requires finding a set of contact points on the object that are stable and feasible given the hand’s size and kinematics, its material and contact properties as well as its overall joints stiffness. A set of contact points is said to be stable if it can resist external forces and moments without dropping the object. Instead, a grasp is feasible when its configuration, defined as the finger joint angles along with the position and the orientation of the end-effector, allows appropriate contacts at the desired locations on the object. As finding complete solutions is difficult, the literature abounds of partial solutions that focus on isolated aspects of the problem. For example, the earlier proposals only formalize the question of stability, assuming that the contact points were given \cite{6, 7, 8}. Such methods are called “analytical” \cite{9}. More recently, the work of Erkan et al. \cite{10} and Rao and colleagues \cite{11} attack the problem by trying to learn good grasps on the basis of empirical experience using machine learning techniques. However they neither account for stability nor feasibility of the grasp. These methods belong to the class of the “empirical” approaches \cite{9}. Yet other methods start from computing a feasible hand configuration for a set of given stable points \cite{12, 13}. There are also “hybrid” approaches that try to provide contact points that are stable, feasible or both. Our work falls squarely into the class of hybrid approaches, solving the full grasping problem by computing stable contact points, and determining a hand configuration that guarantees feasibility.

To the best of our knowledge, all the precision grasp hybrid approaches use either synthetic objects or complete point clouds of known objects to compute reliable grasps \cite{14, 15, 16}. We can certainly store and manage information on 3D models of a very large number of objects and this opportunity should be exploited to the greatest possible advantage. Nevertheless, because of the variety of objects’ sizes and shapes, a robotic grasping strategy should be also robust to unknown objects. This implicitly extends the robot’s skills to situations where recognition fails to classify the object correctly. Being able to deal with unknown objects contributes to the overall quality of the robot’s grasping skills. This line of reasoning leads us to the consideration that the robot can only use incomplete 3D shape information as acquired from a single viewpoint unless, in the presence of an unknown object, it starts a complicated exploration and data collection procedure. Assuming that this is not desirable, we can then reason on the fact that the robot cannot use points in the occluded parts of the object. Our method is designed to work with incomplete 3D point clouds resulting from a single view of the object albeit nothing prevents us from exploiting the additional information were better models available.

Another popular procedure that often characterizes hybrid approaches is the approximation of the 3D data with specific elementary shapes, as for example, boxes \cite{17}, shape primitives \cite{18} or superquadrics \cite{14, 19}. This approach allows handling regular objects quite precisely although, unfortunately, in many cases it may introduce an additional source of uncertainty on the contact point determination. On highly irregular object, the error of the approximation procedure coupled with the intrinsic noise of the sensors (vision) and the mechanical error of the hand becomes the recipe for disaster. For all these reasons we do not employ any object modeling besides the raw collection of the 3D point clouds from stereo vision. We present a method that builds on this “raw data” to plan a three-finger precision grasp directly.

---

\textsuperscript{*}This work was supported by the European FP7 ICT project No. 270490 (EFAA), project No. 270273 (Xperience) and project No. 288382 (Poeticon++)

\textsuperscript{1}iCub Facility, Istituto Italiano di Tecnologia, Via Morego 30, Genova \{ilaria.gori, ugo.pattacini, vadim.tikhanoff, giorgio.metta\}@iit.it

Most of the hybrid approaches focus on finding contact points on the object surface, and assure their stability using standard force-closure criteria [16], [20], [14]. Nevertheless, they do not take into account the physical characteristics of the robotic hand; importantly, this limitation may result in unfeasible configurations. In this sense our second contribution is a measure of the robot’s hand physical properties in evaluating the stability of the contact points. Privileging fingers configurations that lead more often to feasible grasps can upfront increase the probability of success. Subsequently the inverse kinematics of the hand helps in pruning unreachable configurations altogether.

There are a few hybrid methods that search for stable and feasible contact points simultaneously [19], [15]. However they require complete 3D models of the objects. They limit their analysis to the hand, assuming that it can move freely around the object: this is not sufficient to assure that the robot can reach the desired position, as the hand is connected to robot’s body, which lies in a specific position with respect to the object. Our last contribution is a preliminary attempt to account for the robot’s position when selecting the best grasp configuration, in order to use the proposed algorithm on real robots and specifically on humanoids.

We identified a roadmap that progressively evolve from grasping in purely simulated environments to complete real world case-studies with a taxonomy determined by the accuracy of the information that is available to the robot (i.e. complete 3D point cloud vs. single view): (A) Complete 3D point clouds with simulated grasps; (B) partial 3D point clouds with simulated grasps; (C) partial 3D point clouds with the grasp actually executed by the real robot. As far as we know, all the precision grasp hybrid approaches presented in the literature were developed either on synthetic objects or on complete point clouds, exploiting the advantage afforded by the perfect knowledge of the object. In this paper we focus on the second scenario, developing a method that is closer to be utilized on a real robot on realistic everyday’s conditions.

II. METHOD

El-Khoury [19] supports the view that finding a stable and feasible set of contact points on an object is the primary goal when dealing with precision grasp. He further claims that the simultaneous fulfillment of these two conditions brings robustness to the grasp retrieval procedure. This implies that both stability and feasibility have to be calculated on the same type of variables, either discrete or continuous. In El-Khoury’s work, the optimization is continuous since the object is approximated with a superquadric and the hand configuration is also continuous (e.g. the kinematics). In our case, on the contrary, as we do not make use of any surface fitting, our object models belong to the discrete domain and cannot be explored jointly with the space of the hand configurations. Grasping has thus to be made of two distinct routines that find a sufficient number of candidate points (triplets) and solve the inverse kinematic problems for the hand (see Fig. 1).

In the first phase, we seek for a number of candidate triplets – sets of three contact points with corresponding normals – under the condition that they allow the generation of a good grasp (according to the stability criterion). Since running the inverse kinematics on every single triplet (and contact point) is computationally expensive, we need to guarantee that the initial selection is selective. As a first approximation we can therefore privilege triplets that are certainly within the grasping envelope of the iCub’s hand. That is, it is pointless to ask the iCub to grasp a large object (give its child size). Similarly the iCub cannot position its first three fingers at an equal angle with respect to the center of the object.

Once a certain number of candidate sets of contact points has been found, we select the best one on the basis of the feasibility of the grasp configuration. We compute the inverse kinematics of the hand for each candidate triplet and evaluate the physical feasibility of the solution. At this point we further consider the position of the robot with respect to the object, and we select the best grasp among those that do not imply critical or singular configurations of robot’s arm and torso.

In short, in the following, we will describe a complete grasping tool-chain – specifically for three-fingers grasp – which include the acquisition of visual data, the selection of feasible hand configurations and their optimization on the basis of a feasibility measure resulting in grasps that can be effectively executed by the robot. The remainder of the paper is structured as follows: Sec. III illustrates the selection of the candidate triplets, Sec. IV describes the determination of the best grasp using inverse kinematics, Sec. V presents the experimental results and Sec. VI draws the conclusion and illustrate future improvements.
Fig. 2. Cost function accounting for the robot’s hand dimension. From the picture it is possible to infer that bigger areas are privileged. The green line represents the cost function for acceptable areas. Nevertheless, if the area is bigger than the maximum area that can be covered by the hand – which is represented by the big black dot – the measurement grows up dramatically (red line), indicating a bad triplet.

III. TRIPLETS EXTRACTION

We start from an object lying in front of our robot, iCub [1]. The first step of our algorithm consists in exploring the visible portion of the object and select a number of best candidate triplets. We first reconstruct the object in 3D, from a single viewpoint, obtaining an incomplete 3D point cloud. We then estimate surface normals on each point, and we sample the cloud assuming that close points have similar normals. On the sampled cloud, we apply a variant of the Discrete Particle Swarm Optimization algorithm [21] finding a number of triplet points that satisfy several conditions. A first condition is related to the triplet stability. The second and the third conditions instead are associated, respectively, to the hand size and shape.

A. 3D Reconstruction and Sampling

We rely on stereo vision algorithms in order to retrieve 3D information. The Hirschmuller algorithm [22] is applied to the images recorded by our robot, to estimate the depth map, and project each visible pixel of the object in the 3D space. We then estimate the normal to the surface at each point of the cloud by running a least-squares fitting over the point neighborhood to determine the corresponding tangent plane. We then estimate surface normals on each point, and we project each visible pixel of the object in the 3D space. To ensure that, once in contact, the finger will not slip along the surface of the object, the force applied by the fingertip on the point \( p_i \) must lie within the friction cone \( F_i \).

B. Triplet Desired Properties

At this stage a downsampled incomplete 3D point cloud of the object is available to be explored. We now need to retrieve a certain number of good triplets. A triplet is defined “good” on the basis of three properties. A first, necessary property is the triplet stability. If the triplet is not stable, it is not taken into account. The other two properties are related, respectively, to the dimension and the shape of the robotic hand that is performing the grasp.

1) Force-closure Grasp: We will exploit the stability analysis to define if a triplet is acceptable or not acceptable. In case the set of contact points is not stable, it cannot be considered as a candidate triplet.

We adopt the hard finger contact model [6], which is most commonly used. It is characterized by a significant friction acting on small contact patches, and by no transmission of the angular velocity components or moment components to the object. Assuming that only static friction holds, we can build a friction cone \( F_i \) with the vertex in contact point \( p_i \) and aligned along the normal \( n_i \) on the point. The friction cone at point \( p_i \) represents the set of forces that can be exerted on the point without producing a slippage of the finger along the surface of the object. It is determined by the friction coefficient \( \mu_i \) on point \( p_i \) and it is defined as

\[
F_i = \{(f_{in}, f_{it}, f_{io})|\sqrt{f_{it}^2 + f_{io}^2} < \mu_i f_{in}\}.
\]  

To ensure that, once in contact, the finger will not slip along the surface of the object, the force applied by the fingertip on the point \( p_i \) must lie within the friction cone \( F_i \).

A large number of algorithms computing stability have been proposed in the literature. The most used criteria to evaluate a grasp are form-closure and force-closure. In-tuitively speaking, a grasp is defined form-closure if the grasped object is totally constrained by the set of contacts, irrespective of the magnitude of the contact forces [24]. Differently, a grasp is considered force-closure, if and only if we can exert arbitrary force and moment on the grasped object by pressing the fingertips against the object [24]. Form-closure grasp is often considered as being too much restricting and much more difficult to obtain than force-closure, thus most of the grasp synthesis algorithms analyze the force-closure property of a given set of contact points.

Most of the force-closure evaluation techniques rely on the examination of the contact wrenches [25] [26] [14], which consists in the computation of a 6D convex hull and implies the approximation of the friction cone to a pyramid. In this respect, approximating the cone with a pyramid having a large number of sides entails an increasing computational complexity. On the other hand, using a small number of sides makes the friction cone approximation coarser and the force-closure analysis less reliable. Differently, we based our
In row A the four classes of force-closure grasps are depicted. Each of them is characterized by a different number of pairwise counter-overlapping cones. In row B the four classes are illustrated using a human hand that is grasping a ball. Yellow arrows indicate which pairs of friction cones are counter-overlapping. In row C, a 3D model of the iCub’s hand is depicted. The red arrow represents the biggest distance between the index and the middle that can be realized. In the center, an image of the iCub’s hand while closing is shown. This picture is meant to illustrate in a pictorial way that iCub’s hand is not suitable for shape3 grasps. Differently, the Barrett hand, showed on the right, can move the fingers independently and therefore perform grasps belonging to shape3 easily.

evaluation on geometrical considerations. Specifically, we use the necessary and sufficient condition proposed by [8]:

- the three points are not collinear and meet on a plane S;
- each friction cone \( F_i \) intersects the contact plane \( S \) on a plane, generating two unit vectors \( n_{i1} \) and \( n_{i2} \) that bound the projection of the cone on the plane;
- the contact unit vectors construct a 2D force-closure grasp in \( S \).

This condition is fast to compute and it does not require any approximation of the friction cones.

2) Hand Dimension: In order to select a candidate triplet among the force-closure grasps, we need some heuristics that assure the grasp to be suitable for the robotic hand. This will also increase the possibility of obtaining a feasible grasp later on. A first condition is related to the dimension of the hand. A well-known quality measure to evaluate how good a triplet is, is the area covered by the grasp polygon [27]. The bigger the area is, the better the grasp would be. We modify this quality measure in order to account for the robotic hand size (see Fig. 2). In particular, we privilege triplets that cover large areas, and at the same time do not exceed the physical limit imposed by the size of the robot’s hand. Given a triplet \( T \), the area of the grasp polygon \( a \), and the maximum area \( a_{\text{max}} \) that the robotic hand’s fingertips can cover, a good triplet must minimize the following discontinuous function:

\[
d(T) = \begin{cases} 
  k_1(a_{\text{max}} - a)^2, & \text{if } a \geq a_{\text{max}} \\
  k_2a, & \text{otherwise}
\end{cases}
\]

where \( k_1 \) and \( k_2 \) are parameters empirically found. Specifically, if the area covered by the triplet is less than the maximum area, the function has a paraboloid shape. Conversely, when the area covered by the triplet is larger than the maximum area, the function increases linearly (see Fig. 2).

3) Hand Shape: Another crucial information that, as far as we know, has never been taken into account, is the specific shape of the hand. Indeed, different robotic hands will have different grasping capabilities. It has been demonstrated in [7] that there exists four type of force-closure grasps with three hard-finger contacts, and they are dependent on the number of friction cones that pairwise counter-overlap. Specifically, two friction cones \( F_1 \) and \( F_2 \) are said to counter-overlap if the angle between their principal axes \( n_{i1} \) and \( n_{j1} \) is less than the angle of the friction cone \( 2 \arctan(\mu) \), where \( \mu \) is the friction coefficient. The four classes are depicted in Fig. 3, row A, and represent, respectively, a force-closure grasp with 0 counter-overlapping cones, which we define as shape0, a force-closure grasp with 1 counter-overlapping cone, termed shape1, a third force-closure grasp with 2 counter-overlapping cones, labelled as shape2 and a last force-closure grasp with 3 counter-overlapping cones, defined shape3. Fig. 3 also illustrates the four grasps performed by a human hand, in row B. In Fig. 3, row C the iCub’s hand and a Barrett hand are depicted. From the images it is possible to infer that a humanoid robotic hand will have some difficulties in realizing a shape3 grasp, because of the limited displacement between the index and the middle fingers, which cannot face each other. This shape can be realized only when the friction coefficient takes very high values. Similarly, the iCub’s most comfortable force-closure grasp classes will probably be shape1 and shape2. This means that, when a good triplet is selected, it should also have a shape that adapts to the robotic hand that is performing the grasp. A very good example is represented by a spherical object. It will certainly present stable triplets belonging to shape3, but for a humanoid robotic hand a triplet belonging to shape2 or shape1 will be preferable in terms of feasibility.

Inspired by these considerations, we develop a new measure, which depends on the shape that is wanted to realize. Specifically, given a triplet \( T \), a number \( n \) of desired counter-overlapping cones, and the actual number \( c(T) \) of counter-overlapping cones in the triplet \( T \), we compute the quantity \( d = n - c(T) \). If \( d \neq 0 \) we build two vectors: a vector \( u \) containing the angles between the pairs of friction cones that counter-overlap, sorted from the smallest to the biggest value, and a second vector \( v \) that contains the sorted angles between the pairs of friction cones that do not counter-overlap. The proposed measure is the following:

\[
s(T) = \begin{cases} 
  0, & \text{if } d = 0 \\
  \sum_{i=1}^{d} v_i, & \text{if } d > 0 \\
  \sum_{i=1}^{d} u_i, & \text{if } d < 0
\end{cases}
\]
The above-mentioned formulation represents how much the current triplet is far from belonging to the class of the desired shape. As we are going to minimize these quantities, the smallest the measure is, the better the triplet is evaluated.

\section*{C. Discrete Particle Swarm Optimization}

Given a set of points $P$, we aim at finding a triplet $T = \{(p_i, n_i), (p_j, n_j), (p_k, n_k)\}, p_i, p_j, p_k \in P$ that satisfies the properties illustrated in Sec. III-B. In particular, we want to solve the following problem:

$$\min_T \quad d(T) + ws(T) \quad \text{s. t. } T \text{ is force-closure},$$

which summarizes the properties that a candidate triplet should present. Specifically, the stability is formulated as a constraint, as it is necessary to consider a triplet acceptable. At the same time, we wish to select candidate triplets of a specific shape; this is defined by $s(T)$, which corresponds to Eq. (5). Then, $d(T)$ is Eq. (4), and it assures that the area covered by the triplet is large, but does not exceed the maximum area that can be covered by the hand. Finally, $w$ is a variable chosen empirically, which regulates the contribution of $d(T)$ and $s(T)$. Even though the number of possible triplets is finite, and thus it would be possible to analyze them all to find the optimal one, an exhaustive search cannot be performed if there is a large number of points. Furthermore, Borst [28] demonstrated that there is no need to find the globally optimal grasp, because most of the times an average grasp (in the force-analysis sense) is acceptable. Therefore we resort to a local optimization algorithm.

We have a discrete number of points $P$, thus a discrete, derivative-free optimization algorithm is requested. On the other hand, our downsampled point cloud is an approximation of a surface, and it holds some nice properties that we would like to exploit. For instance, the fact that close points are likely to have similar normals. This assumption would help the optimization algorithm to find a good solution fast. Among the discrete optimization algorithms, a technique that satisfies all our needs is the Discrete Particle Swarm Optimization (DPSO), in the variant presented by [21].

Particle swarm optimization (PSO), originally presented in [29], is a population-based optimization method. It initializes a set of particles that explore the space domain on the basis of historical information learnt from the swarm population. In our setting a particle $p$ is represented by three points $p_1$, $p_2$ and $p_3$, therefore $p \in \mathbb{R}^3$. At each point the normal computed as in Sec. III-A is associated, therefore a triplet $T$ can be represented as the couple $\{p, n\}$, where $n$ is the set $\{n_1, n_2, n_3\}$ of normals on the points in $p$. Each particle has its position and its velocity; it also keeps track of the best global position $g^*$ ever visited from the population, as well as the best position $l^*$ it has visited in the past. For each particle a fitness function on its associated triplet is computed. In our case, the fitness function is in the form $d(T) + ws(T) + f c(T)$ (see Eq. (6)), where $f c(T)$ returns 1 if the triplet is force-closure, a very large number otherwise. At each iteration, all the particles update their positions modifying their velocities on the basis of their past velocity, $g^*$ and $l^*$. As in [21], we use the velocity formula that is used in the standard version of PSO:

$$v_{t+1} = \alpha v_t + c_1 r_1 (l^* - p) + c_2 r_2 (g^* - p),$$

where $\alpha$, $c_1$ and $c_2$ are constant values, and $r_1$ and $r_2$ are random values. It is worth noting that only the points explore the space, whereas the normals are associated to the points in a fixed way. Therefore the velocity corresponds to an actual velocity vector in the space. We can use this formulation under the assumption that close points have similar normals; therefore if the velocity is small, the particle computed at the next iteration along with its associated normals will be similar to the current one. Obviously the velocity is a continuous variable and may return a particle $p$ whose points are not contained in our set $P$; therefore, as in [21], at each iteration we select the set of points in $P$ that are closest to $p$. When the algorithm terminates, the best particle found is returned.

An ideal situation would be to analyze every acceptable triplet through inverse kinematics, in order to check whether it is actually feasible for the robot. This is not possible for computational reasons; indeed, even though the inverse kinematics algorithm is fast and reliable, it takes 500 milliseconds to find a solution. Therefore, running it for every triplet found by DPSO – which could be a very large number – is not computationally feasible. We need to develop an algorithm that can be performed in real-time, thus we rely on computing a limited number of the best triplets that DPSO finds, and we let the kinematics pick the best one. In the current setting we run 4 parallel DPSO procedures, one for each hand shape, and we select the best one for each DPSO routine. If requested, it would be possible to keep track of a bigger number of the best ranked triplets for each DPSO routine. These triplets are analyzed by the kinematics of both the left and the right hand, thus the algorithm can choose among 8 different solutions in total.

\section*{IV. INVERSE KINEMATICS}

Given a triplet $T$, we are asked to find a feasible configuration $(x, o, q)$ for the robot’s hand. Specifically, the inverse kinematics algorithm should provide the end-effector position $x \in \mathbb{R}^3$ and orientation $o \in \mathbb{R}^3$ in the Cartesian space, and finger configuration $q \in \mathbb{R}^8$ in the joint space. We use the thumb, the index and the middle finger of the iCub’s hand, which are equipped respectively with 3, 3 and 2 joints. The end-effector position and orientation correspond to the region of the object towards which the hand will move. The finger configuration (or hand preshape) corresponds to the angles to which finger joints are set to reach the contact points.

We formulate the inverse kinematics (IK) problem as an optimization problem, similarly to [19] and [13], and we use IPOPT [30] to solve it. However, our problem is slightly different with respect to [19] and [13]. Indeed, we are dealing with discrete point clouds, therefore we cannot use a model of the object, as [19] and [13] do. Furthermore, we are only interested in the inverse kinematics part, whilst [19] and [13] take into account a larger number of variables. In particular,
given the triplet $T = \{\{p_1, n_1\}, \{p_2, n_2\}, \{p_3, n_3\}\}$, the object center $x_o$ and its covariance matrix $D$ containing information about its dimension, we minimize with respect to $x$, $o$ and $q$ the following problem:

$$
\min_{x,o,q} \sum_{i=1}^{3} z_i n_i
$$

s. t.  
\begin{align*}
&||f_i - p_i|| < \epsilon, \text{ for } i = 1, \ldots, 3 \\
&(x - x_o)^T D^{-1} (x - x_o) > 1 \\
&l_i < q_i < u_i, \text{ for } i = 1, \ldots, m,
\end{align*}

where $l_i$ and $u_i$ are the lower and upper limits for each joint, $m$ is the total number of joints, $z_i$ is the direction of the force exerted by the fingertip and $f_i$ is the position of the $i$-th fingertip, computed via forward kinematics of the hand. The cost function represents the fact that the force exerted by the fingertip should be opposite with respect to the normal on the point. The first set of constraints takes care of minimizing the distance between the fingertip position and the contact point. The second constraint prevents the hand to collide with the object, imposing that the end-effector should lie outside the object. The last set of constraints controls that the joint angles do not exceed their physical bounds.

This minimization problem is solved for all the triplets retrieved by the step described in Sec. III, and 8 solutions are found – 4 for the right hand and 4 for the left hand. Besides the value of the cost function, which already provides a good measure of a configuration’s feasibility, we would need to consider also the whole robot’s arm. Most of the methods that deal with precision grasp work in simulation, therefore they do not account for the robot’s arm, assuming that the hand can move freely. Differently, we would like to use the proposed framework on real robots in the future, therefore we take a preliminary step in considering the hand as being connected to the arm, which is bound to the rest of the robot’s body. A good measure of how a certain configuration is suitable for a robot’s joints configuration is provided by the standard manipulability [31], [32]. To compute this quantity, we make use of the method in [33] to solve the inverse kinematics (IK) of the arm and the torso. It results in the joint configuration that satisfies the desired position and orientation of the hand using 10 degrees of freedom of the robot (7 for the arm and 3 for the torso). The solution that presents the best cost function and the higher manipulability is finally selected.

V. EXPERIMENTAL RESULTS

We first qualitatively show that our method is suitable for dealing with complete 3D models of the objects. To this end, we present some results on the KITObjectModels WebDatabase [34] in Fig. 4, which contains complete 3D point clouds of several objects of different shapes.

The scope of our paper though, is showing that our framework is able to handle also incomplete, raw 3D point clouds. Therefore we assessed our method on the 8 real objects showed in Fig. 5. The objects have been reconstructed using the stereo vision of the iCub. As the objects are unknown, we do not know their friction coefficients; we then employed a friction coefficient of 0.6. Since we retrieve triplets only on the visible part of the object, using a too small friction coefficient can prevent from finding stable triplets; in this case, the friction coefficient is increased until 0.9, which is the coefficient between rubber and paving. If no stable triplets are found, likely the visible part of the object does not present suitable contact points. We perform 10 different trials on each object, for a total of 80 trials. For each run we launched 4 parallel DPSO routines, resulting in 4 triplets. The inverse kinematics of the hand was computed for both left and right arms, obtaining 8 different solutions for each trial. Among these configurations, the one presenting the lowest value of the cost function and the highest manipulability was selected. The entire pipeline, from the object reconstruction to the retrieval of the grasp configuration, takes around 6 seconds in total with a standard deviation of 1 second.

As grasping lacks of a standardized benchmark, we proceeded analyzing the resulting configurations as follows: if the fingertips are in contact with the object and there is no interpenetration among the fingers and the object, each finger is lying in a feasible position without crossing the other fingers, and, finally, the hand is in a suitable position for the robot, then the grasp is labelled as successful. We achieved an overall success rate of 85%. A detailed analysis of such accuracy is reported in Fig. 5. It is worth noting that the lowest accuracy is reported by the watermelon, which is harder to reconstruct because of its flat sides; indeed, using stereo vision on flat, continuous surfaces may bring to inaccurate point projections. The only other object that reported a low accuracy (the monkey), is the biggest object we have used. In all the failure cases, the retrieved triplets covered a too big area for the hand; probably the DPSO algorithm was not able to find triplets covering a smaller area. Notably, thanks to the manipulability measure included in the second stage of our framework, all the selected grasps...
are feasible for the robot’s position and do not entail singular or unnatural configuration.

To better understand the reasons behind the small percentage of failures recorded in our simulations, we report hereinafter on further evaluations carried out on the two components of the proposed pipeline: the DPSO and the IK routines.

A. DPSO

The first stage of our algorithm is extremely accurate in retrieving only stable contact points. Indeed, on 80 trials there was only a single case where DPSO did not find any stable triplet. This was due to the reconstruction of the object, which provided a distorted point cloud. This lead to an inaccurate computation of the surface normals, and therefore to a failure in retrieving stable triplets. On the other hand, we aim at measuring how many triplets retrieved by DPSO were actually feasible for the robot’s hand. In Table I we present, for each object, the percentage of feasible triplets provided by DPSO. Comparing these results with the overall accuracy achieved on our dataset, we highlight that our algorithm retrieved unfeasible triplets on the car, the carrot, the monkey and the watermelon. In Table I it is also possible to review the number of times a specific grasp shape has been chosen. Notably, shape3 is not mentioned in the table because it is never selected by the iCub’s hand. Differently, shape2 and shape1 are the most commonly selected, as we were expecting.

B. Inverse Kinematics

Finally, we assess the accuracy of the IK stage. If the reconstruction is actually accurate, and the triplets are feasible, there is the possibility that the inverse kinematics algorithm does not converge to adequate hand configurations. In Table II we provide information in this regard: we compute the mean distance between the desired contact point locations and the fingertip positions, and the mean angle between the normal on the point and the direction of the force exerted by the fingertip on each object, when the grasp has been successful. We remind that the angle should be close to π. In the last column we also report how many times the inverse kinematics algorithm failed for each object, when feasible triplets were found.

### Table I

<table>
<thead>
<tr>
<th>Object</th>
<th>Feasible Triplets</th>
<th>shape0</th>
<th>shape1</th>
<th>shape2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0.9</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Carrot</td>
<td>0.9</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Dog</td>
<td>0.9</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Grapefruit</td>
<td>1.0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Monkey</td>
<td>0.7</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Octopus</td>
<td>1.0</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Tomato</td>
<td>0.9</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Watermelon</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Object</th>
<th>Norm [mm]</th>
<th>Angle [rad]</th>
<th>Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>4.4</td>
<td>2.13</td>
<td>1</td>
</tr>
<tr>
<td>Carrot</td>
<td>3.2</td>
<td>2.17</td>
<td>1</td>
</tr>
<tr>
<td>Dog</td>
<td>3.1</td>
<td>2.19</td>
<td>0</td>
</tr>
<tr>
<td>Grapefruit</td>
<td>3.5</td>
<td>2.13</td>
<td>0</td>
</tr>
<tr>
<td>Monkey</td>
<td>3.9</td>
<td>2.23</td>
<td>0</td>
</tr>
<tr>
<td>Octopus</td>
<td>3.5</td>
<td>2.36</td>
<td>0</td>
</tr>
<tr>
<td>Tomato</td>
<td>3.3</td>
<td>2.25</td>
<td>1</td>
</tr>
<tr>
<td>Watermelon</td>
<td>3.3</td>
<td>2.24</td>
<td>0</td>
</tr>
</tbody>
</table>
VI. CONCLUSIONS

We presented a complete, novel algorithm for three-finger precision grasp. The biggest contribution is the possibility to deal with raw incomplete 3D point clouds. Dealing with incomplete 3D point clouds instead of requiring complete models is very important as it allows grasping unknown, partially perceived objects. Furthermore, we avoid the approximation of the objects to simpler shapes, meaningfully decreasing the uncertainty about the contact point locations on the object. We thus explore the incomplete 3D point cloud, looking for a number of stable triplets through a variant of the Discrete Particle Swarm Optimization algorithm. The candidate triplets should also satisfy specific properties that are related to the robotic hand that is performing the grasp, as different robotic hands have different grasping capabilities. Therefore, a second contribution is the inclusion of the hand size and shape in the triplet retrieval step. We have also proposed a new quality measure on the hand shape. We then employ IPOPT to solve the hand inverse kinematics. Finally, we account for the manipulability of the arm pose when selecting the best grasp configuration.

The main future direction is the refinement of our framework in order to be used safely and reliably on real robots.

REFERENCES