Proceeding of the IEEE
International Conference on Robotics and Biomimetics (ROBIO)
Shenzhen, China, December 2013

A Geometrical Placement Planner For Unknown Sensor-Modelled Objects And Placement Areas

Johannes Baumgartl, Per Kaminsky, and Dominik Henrich

***Abstract*— A Personal Robot should be able to handle possible unknown objects in unknown environments. For a manipulation task the question what to do with an object once it had been grasped is one of the most essential ones beside the grasping task itself.**

We propose a placement planner for sensor-modelled objects in complex environments. The planner computes a stable position and orientation for the object in the environment. The algorithm uses only geometric information, most notably no force or torque sensor is required. In particular, we introduce a novel approach regarding the configuration computation.

By means of experiments with various household objects the robustness and performance are validated. Further on, we compare our approach to a configuration computation using a physics simulation framework.

I. Introduction

A flexible and skilled Personal Robot needs to perform various pick-and-place tasks in different domains like i.g. cleaning up worktops in households, laboratories, or workplaces in general. Therefore, a personal robot should be able to elaborate how and where to place objects.

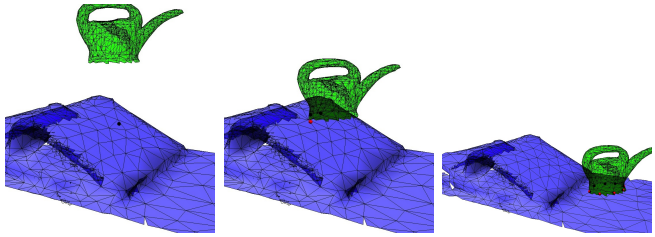
However, there is a wide range of situations, objects, and environments that affect this challenging task. Hence, the robot must be able to handle unknown objects and environments whose surface models are reconstructed using sensor data. The resulting models are potentially incomplete and noisy.

When it comes for the robot to place an object in the environment the robot has to be able to plan where to put down the object. An algorithm calculating stable placement configurations helps the robot to perform successfully.

The input to our algorithm is a placement area and an object, both given as surface models with convex planar surface patches. To reconstruct these surface models, the sensor data from a depth camera is integrated over time. These object models include holes of non considered regions. The integration over time of the noisy sensor data dramatically reduces the noise in the resulting object models.

The output of our algorithm is a configuration (position and orientation) for the object and several intermediate configurations that describe constructive steps to reach a stable goal configuration. The main steps of object placement planning are:

1.) **Select a location on the placement area** suitable for the object.
2.) **Orient the object** (Fig. 1(a)) with respect to its geometrical properties or semantical needs, e.g. an upright orientation [1].



(a) Object configuration after step 2.) (b) Intermediate configuration during step 3.) (c) Goal configuration after step 4.)

Fig. 1: Illustration of the general steps 2-4 of our algorithm.

3.) **Compute the placement configuration** (Fig. 1(b)) by tilting the object until it remains in a probably stable configuration ensuring enough contact points with the placement area.
4.) **Compute the quality metric** (Fig. 1(c)) of the reached configuration, including an evaluation of the stability. In case of instability we apply a **local optimisation** step using step 3.

The contribution of this paper is to offer a solution consideringall of the above steps for planning object placement configurations in the environment while using only the geometric information gained by a depth sensor. With a special focus is put on the question of how to compute candidate configurations and how to optimise them locally in case of static instability.

The remainder of this paper is organized as follows: At first we review related works (Section II). Then we present in detail the steps of the proposed algorithm, starting with a short nomenclature (Section III). Next, the strategies for the selection of the suitable floor points in the placement area are introduced (Section IV). Further on, the procedures to compute appropriate candidate goal configurations is explained (Section V). We continue presenting how stability and quality are evaluated in order to select a feasible goal configuration and we introduce our local optimisation step (Section VI). Finally, we introduce our object modelling pipeline and the experimental results (Section VII).

II. Related Work

The field of object placement planning in the context of pick-and-place applications [2] was not as such much focused on as was the grasp planning [3]–[5] during the past years. First, we mention approaches dealing with subproblems and related approaches. Second, we review approaches considering our own problem description.

978-1-4799-2744-9/13/$31.00 ©2013 IEEE

The focus of several works is on the two first steps of the placement problem. Especially, Schuster et al. [6] offers an approach to find free space on planar surfaces. Fu et al. [1] regard finding the upright orientation of man-made objects based on geometric features. And Glover [7] estimates poses of objects based on incomplete point clouds using an object database. Since these works do not consider finding a stable configuration of the object on the placement area, they are not regarded any further.

Arranging objects efficiently refers to the packing problem [8]–[10] considering optimal object arrangements in a limited space even for irregular shaped objects [11], [12]. These approaches focus on generic packing problems with known objects. That is why they are complementary to our approach. These ideas could be used in future to develop local optimisation criteria for the fourth step of the algorithm overview.

In the field of robotic applications, placing is about controlling the robot arm and placing the object gently using e.g. force control and passive compliance [13], or task-level planning [14]. However, the focus is here not on finding feasible placing configurations. Those approaches are subsequent operations after the placement planning.

Another related topic is push planning of objects on planar placement areas [15]. But again, the focus is not on computing placement configurations. However, the results could be used in an subsequent improvement step.

An approach that targets placement planning in the sense of packing is [16]. Since this approach uses only base areas of the unknown objects and known planar placement areas, it is not applicable for general placement areas, which we are interested in. Another approach considering known objects with an unknown placement area is introduced in [17]. Here, object placement considers the footprint of the contact area. The target location of the object is given as input by a human. Consequently, they introduce an interactive approach and not an autonomous one. Both approaches only consider planar placement areas. Contrary to that, we generalise from planar to complex placement areas.

One of the first overall solutions for placing novel objects in complex placement areas is introduced by Jiang et al. [18] and [19]. They outline a learning based framework that requires an object database together with semantic labels. Our work is different from that in so far as we use only geometric information about the object and do not use any kind of learning step.

III. Nomenclature

We assume that the setting does not change during the planning phase. Here and subsequently, we take that the direction of the gravity $d_{\mathrm{g}} \in \mathbb{R}^3$ and an approach direction $d_{\mathrm{vis}} \in \mathbb{R}^3$ as given. Surface models and their properties are defined as follows:

Definition Let $\mathcal{S} := \left\{v_1, ..., v_k \in \mathbb{R}^3\right\}$ be a *planar convex polygonal surface patch* with v_i as vertices ordered counter clockwise, (v_i, v_{i+1}) an edge of this surface, and $|\mathcal{S}| > 2$.

Definition Let $\mathcal{M} := \{\mathcal{S}_1, ..., \mathcal{S}_n\}$ be a *polygonal surface mesh* with convex polygonal surface patches $\mathcal{S}_i$.

Definition Let $k_1^{\mathcal{M}}, k_2^{\mathcal{M}}, k_3^{\mathcal{M}} \in \mathbb{R}^3$ be the *principal axis* of a surface mesh $\mathcal{M}$, with the corresponding eigenvalues $\lambda_1^{\mathcal{M}} > \lambda_2^{\mathcal{M}} > \lambda_3^{\mathcal{M}}$.

Henceforth, we represent the gripped object $\mathcal{M}_{\mathrm{O}}$ and the placement area $\mathcal{M}_{\mathrm{P}}$ as polygonal surface meshes. Further on, we compute the center of mass $c_{\mathrm{O}}^{\mathcal{M}}$ of the object using the vertices of the surface mesh.

IV. Pre-placement Configuration

This section considers the orientation of the object in advance to the placement planning and the selection of feasible candidate locations on the placement area, called *floor points*. The whole passage refers to step 1 and 2 of the algorithm overview in Section I.

A. Object Orientation

Currently we apply a heuristic to achieve the desired object orientation. We use the dedicated principal axis $k_1^{\mathcal{M}_{\mathrm{O}}}$ of the most dominant eigenvalue. We rotate the object such that its first dominant principal axis $k_1^{\mathcal{M}_{\mathrm{O}}}$ is parallel to the approach direction d_{vis}.

By this procedure we are able e.g. to reach the bottom of a box filled with pens or to place a plate in a disk rack. In case of planar placement areas, a configuration with $k_1^{\mathcal{M}_{\mathrm{O}}}$ parallel to this plane is still feasible.

If the upright orientation of the object can be estimated with the help of the geometrical observations introduced by Fu et al. [1], we are able to get an idea of the semantically correct orientation and align the object accordingly.

B. Estimate Floor Points

Our algorithm requires a desired location for an object on the placement area, called *floor point*. We select those floor points according to two observations that we had made in the context of household objects. The first observation is that humans tend to place objects at visible locations. The second one is that humans tend to place objects at the lowest position possible, e.g. pens in pencil cups, flowers in vases or dishes in a dish rack.

According to the first observation, we use a ray casting approach featuring rays parallel to the approach direction d_{vis}. The intersection points between rays and placement area are the visible points. In case of a box-like placement area, we select the n lowest points (with respect to d_{g}) from the ones visible, in order to render it more likely that the object reaches the ground of the placement area. The remaining set of vertices is our floor point set.

V. Stable Configuration

Based on the computed floor points, we calculate placement configurations that are likely to be stable. This section refers to step 3 of the algorithm overview.

First, we explain the procedure to establish a first contact between the object and the placement area (Section V-A).

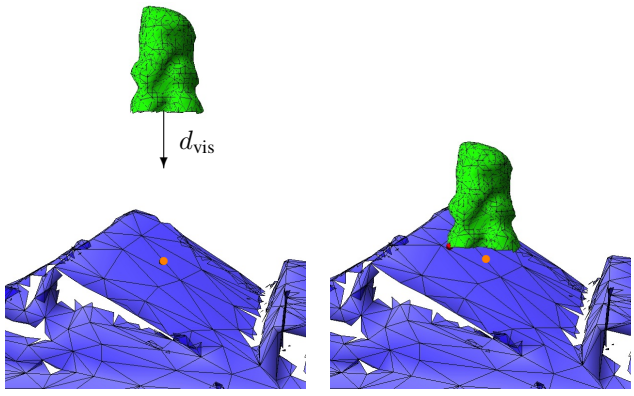
Second, we elaborate on the tilt computation run to establish more contact points (Section V-B). Those computations affect transformation matrices, which are applied to the object's center of mass and the principal axis. The resulting position of the center of mass is the position part $p^{\mathcal{C}}$ of the goal configuration $\mathcal{C}$ for the object. The accumulated rotations define the rotatory part $o^{\mathcal{C}}$. Every single transformation matrix itself can be written as intermediate configuration, too. Conveniently, those configurations can directly be used for a motion planner to approach the goal configuration.

A. First Object Contact

Fig. 2(a) and Fig. 2(b) show the two steps necessary to establish an initial contact between the object and the placement area.

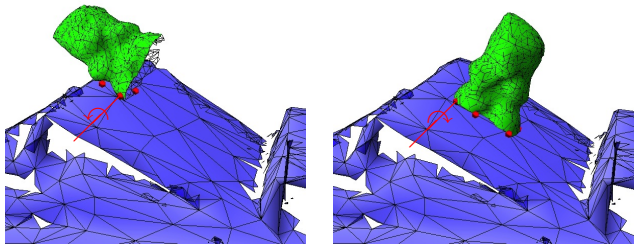
In the first step, the object is translated such that its center of mass $c_{\mathrm{O}}^{\mathcal{M}}$ and one of the previously computed floor points define a line parallel to the direction d_{vis} and in a distance to the placement area so that no rotation of the object would cause a collision with it.

In the second step, we compute the minimal translational distance along d_{vis} needed for the object to make contact with the placement area using a ray cast technique.



(a) Location of the pre-orientated object above the selected and visible floor point (orange).

(b) Established first contact (red) between object and placement area.



(c) First result of the tilt computation (left-handed) based on the contact points from (b) around the red axis.

(d) Second result of the tilt computation (right-handed) based on the contact points from (b) around the red axis.

Fig. 2: Examples for the four main steps of the configuration computation for the object (green) relative to the placement area (blue). Colors: Contact points and rotation axis (red), and current floor point (orange).

B. Object Tilt Computation

To establish possible stable contact configurations, we tilt the object (Fig. 2(c) and 2(d)). Therefore, we need to compute a rotation axis based on the current contact situation (Section V-B.1). We also compute the tilt angle for this axis producing new contact points (Section V). We continue this tilting procedure until at least three independent contact points between object and placement area are established, or a maximum amount of tilts was performed in a row without success. Since we compute two rotation angles (clockwise - Fig. 2(d) and counter clockwise - Fig. 2(c)) for the object during each tilt step, the recursive applying of tilt steps results in a binary tree. The maximum depth of this tree equals the maximum amount of tilts.

1) Tilt Axis: For the contact point computation we use the PQP library [20]. It determines the two triangle patches in contact. We choose one contact point on the intersection between the two patches of a pair. Depending on the size of the patches, the contact points can be located very close to each other. Therefore, we cluster contact points based on their Euclidean distance. Each cluster represents from now on a contact point. Based on the contact situation, we compute in the following the axis for the next rotation. The clustering may affect a small penetration between object and placement area. Its amount is bounded by the maximum diameter of a collision point cluster.

If only one contact point between object and placement area exists, we distinguish between three situations. If the main principal axis of the object $k_1^{\mathcal{M}_{\mathrm{O}}}$ is not parallel to the gravity direction d_{g}, we take $k_1^{\mathcal{M}_{\mathrm{O}}} \times d_{\mathrm{g}}$ as axis. Otherwise, we take the vector connecting the contact point and the center of mass of the object instead of the main principal axis of the object. If this solution is not suitable because the computed vector is also parallel to d_{g}, we chose an axis randomly. The reason for this rotation axis computation is that we try to tilt the object into its natural direction of dip, which is indicated by its main principal axis or its center of mass.

In case of two or more contact points we distinguish between two situations. If all contact points are located on a common line, we take this line as rotation axis. Otherwise, there is no further need for a rotation axis because the contact points are independent from each other. We then move on to the stability analysis and local optimisation (Step 4 of the algorithm overview).

2) Tilt Angle: For the computation of the tilt angle we merely need to consider vertex/surface contacts and edge/edge contacts, since we have convex surface patches.

Further on, we select the relevant surface patches using an octree in order to attain a more efficient computation. These patches serve us to compute the rotation angle based on the following calculations.

a) Vertex/Polygon Rotational Contact: Given are a vertex $v \in \mathcal{M}_{\mathrm{O}}$, a surface patch $\mathcal{S} \in \mathcal{M}_{\mathrm{P}}$, and a rotation axis $A := s_A + \lambda_A d_A$. The point s_A is a collision point and d_A is the axis computed in Section V-B.1. Now we seek the rotation angle around A and between v and $\mathcal{S}$.

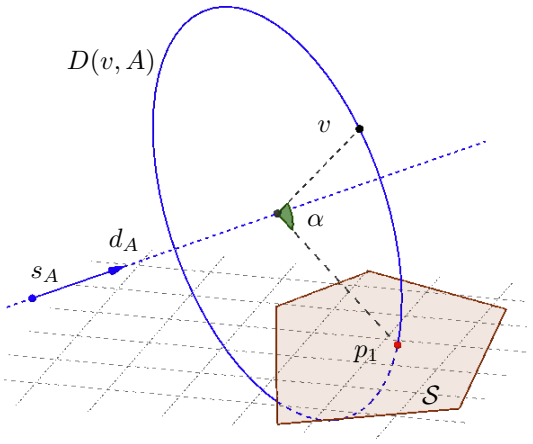


Fig. 3: Illustration of the angle computation between a vertex v and a surface patch $\mathcal{S}$ around $A := s_A + \lambda_A d_A$ including the auxiliary circle $D(v, A)$, the intersection point $p_1 = \mathcal{S} \cap D(v, A)$ and the computed rotation angle α.

Therefore, we introduce the plane $P(\mathcal{S})$ defined by the surface patch $\mathcal{S}$. Further on, we define the auxiliary circle $D(v, A)$ by the vertex v and axis A (Fig. 3):

$$D(v, A) := \Big\{ p \in \big\{ t \in \mathbb{R}^3 | (t - p_A) \cdot d_A = 0 \big\} | \exists p_A \in A : \min d(p_A, v), d(p, A) = d(p_A, v) \Big\}$$

We intersect the plane and the circle: $\{p_1, p_2\} = P(\mathcal{S}) \cap D(v, A)$. Subsequently we examine both intersection points p_1 and p_2 to check whether they are located inside of the surface patch $\mathcal{S}$. Finally, we compute the angles between the intersection points inside the surface patch and the vertex v according to the rotation axis A. In case that there are two intersection points we choose the smaller angle.

b) Edge/Edge Rotational Contact: Given are two given skew line segments $L_1 := s_{L_1} + \lambda_1(e_{L_1} - s_{L_1}) \in \mathcal{M}_{\mathrm{O}}$, $L_2 := s_{L_2} + \lambda_2(e_{L_2} - s_{L_2}) \in \mathcal{M}_{\mathrm{P}}$ and an axis $A := s_A + \lambda_A d_A$, with start points $s_{L_i} \in \mathbb{R}^3$, end points $e_{L_i} \in \mathbb{R}^3$, direction $d_A \in \mathbb{R}^3$ and $\lambda_i \in [0; 1]$. The angle α rotates L_1 around the axis A. The task is to compute α so that it brings L_1 in contact with L_2.

The computation of this rotation includes basically three steps: (1) First, we test if the claimed rotation angle does exist. Second, we compute the angle with the use of a surface of revolution constructed by the rotation of L_1 around the axis A. (2) For this task we transform the line segments and the axis to a more suitable coordinate system, which allows us to calculate the coefficients of the surface of revolution in an easier way. (3) At last, we compute these coefficients, and by the help of the intersection between the other line segment L_2 and the constructed surface we compute the requested rotation angle. Fig. 4 illustrates the whole process graphically. Here are the steps once more in detail:

(1) Pretest: Check if the distance of both end points of L_1 and L_2 alternates according to their distance to the rotation axis: Without loss of generality, let L_1 be closer to A than L_2. If the condition

$$\min_{p \in \{s_{L_2}, e_{L_2}\}} d(p, A) > \max_{p \in \{s_{L_1}, e_{L_1}\}} d(p, A)$$

holds, there will be no collision for the described rotation. Further on, the projected endpoints of L_1 and L_2 on A have to alternate along A such that there is no separating plane $(x - t) \cdot d_A = 0$ with $t \in A$ as an arbitrary point that separates both line segments.

(2) Transformation: For a more simple construction we transform L_1 and L_2 with the result that A is identical to the z-axis d_z. Furthermore, if there exists a single distinct point $t_0 := \mathrm{argmin}_{t \in A}\{d(t, L_1)\}$ on the rotation axis A that has the smallest distance to the rotating line, we use t_0 to define an affine transformation that transforms t_0 into the origin and aligns the direction d_A with the z-axis d_z. If the point t_0 is not distinct, i.e. $L_1 || A$, we can take any point on A for t_0 without limitation.

Henceforth let $\hat{L_1}$ and $\hat{L_2}$ be the transformed line segments.

(3) Geometric construction: We want to compute a surface of revolution H describing the rotating segment which will later be intersected with the second line segment. We use the line segment $\hat{L_1}$ as the generatrix of the surface.

If $s_{\hat{L_1}}$ and $e_{\hat{L_1}}$ are in a plane coplanar to the x-y-plane the resulting primitive is an annulus and can therefore be written as:

$$H : (x - s_{\hat{L_1}}) \cdot d_z = 0$$

Otherwise, the surface of revolution can either be a cylinder, cone or hyperboloid of one sheet. Those surfaces are

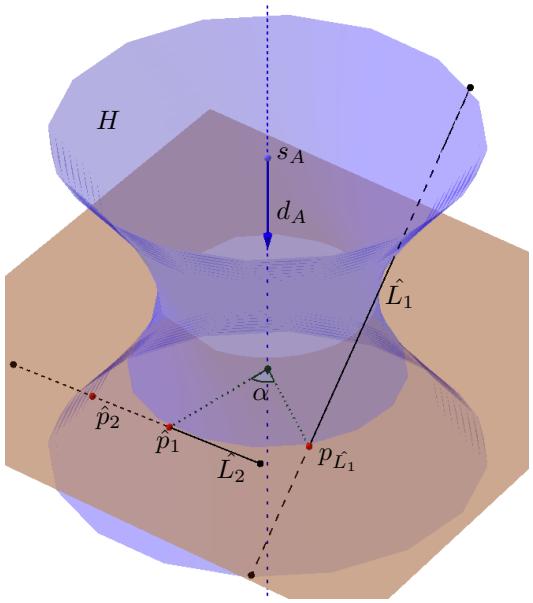


Fig. 4: The steps for the tilt angle computation between two skew line segments are: Construct a surface of revolution H using the line segment $\hat{L_1}$ as generatix and $A := s_A + \lambda_A d_A$ as axis. Compute the intersection points $\{\hat{p_1}, \hat{p_2}\} = H \cap \hat{L_2}$. Compute the intersection point $p_{\hat{L_1}}$ of the line segment $\hat{L_1}$ with the plane defined by p_1 and d_A used as normal. And compute the tilt angle α.

represented by a generalized quadric of the form:

$$H : \frac{x^2 + y^2}{a^2} - k \cdot \frac{z^2}{b^2} = c.$$

Hence, with the points $p_1 := \text{argmin}_{p \in \hat{L_1}}\{d(p, d_z)\}$ and $p_2 \in \hat{L_1}, p_2 \neq p_1$ and $d(p_2, d_z) \neq 0$ the quadric can be parametrized as shown in Table I.

TABLE I: Parametrization of different surfaces of revolution using the general quadric using $p_1 := \text{argmin}_{p \in \hat{L_1}}\{d(p, d_z)\}$ and $p_2 \in \hat{L_1}, p_2 \neq p_1$.

$\frac{x^2+y^2}{a^2} - k \cdot \frac{z^2}{b^2} = c$	k	c	a^2	b^2
cylinder	0	1	$d(p_2, d_z)^2$	-
cone	1	0	$d(p_2, d_z)^2$	$p_{2,z}^2$
hyperboloid	1	1	$d(p_1, d_z)^2$	$\frac{p_{2,z}^2}{\frac{d(p_2, d_z)^2}{d(p_1, d_z)^2} - 1}$

(4) Angle computation: Since the surface of revolution is now well defined, we are able to compute the intersection points $\{\hat{p}_1, \hat{p}_2\} = H \cap \hat{L_2}$ by inserting the line segment $\hat{L_2}$ into H. We expect $\hat{L_2}$ not to lie on H. Henceforth, we outline the computation steps using $\hat{p}_1$. For each intersection point $\hat{p}_1$ and $\hat{p}_2$ we define the plane $P : (x - \hat{p}_1) \cdot d_z$. The resulting rotation angle between the two line segments is now defined by $\sphericalangle_{d_z}(\hat{p}_1, P \cap \hat{L_1})$ (angle α in Fig. 4).

Note that in case of two intersection points, the smallest rotation angle represents the correct one. In addition, there is no need for an inverse transformation of the constructed surface of revolution H back into the original space because we only applied an affine transformation, which does not affect the value of the rotation angle.

VI. Configuration Selection

The previous steps computed a configuration for the object, were a contact situation with a minimum of three contact points is ensured. Now we apply different quality measurements, and in some cases we adjust the computed configuration (Step 4 of the algorithm overview).

The mandatory condition is that the configuration remains in static equilibrium. Therefore, we introduce in the following a quality metric and a local optimisation step based on the planar case.

If all contact points are located on a common plane P, we can compute the stability based on the planar case. We compute the 2D convex hull Ω of the contact points located on P. Then we project the center of mass $\hat{c}_\text{O}$ of the object along the gravity direction d_g onto P and check if it is inside Ω. Additionally, if the angle between the normal of P and d_g is smaller than the angle defined by the friction coefficient μ, then the object configuration remains in static equilibrium.

In case the plane defined by the collision points is not perpendicular to the gravitation and does not meet the friction condition, we compute the failure direction d_f parallel to P. Let $\hat{c}_\Omega$ be the center of Ω: $d_\text{f} = \hat{c}_\text{O} - \hat{c}_\Omega$. The new translation direction d_vis is d_f.

Using the new d_vis, we reconduct the translation of Section V-A as well as the contact computation of Section V.

In case the friction condition is not met and the plane defined by the collision points is not perpendicular to the gravitation, we just tilt the object again into the failure direction with the rotation axis defined by $d_A = d_\text{f} \times d_\text{g}$ and s_A as the farthest contact point of Ω with respect to d_A in the direction of d_f.

Our local optimisation steps generate contact points in such a way that they stabilise the object with respect to its old failure direction. Those optimised contact situations can usually no longer be handled with the planar case. Whether those configurations are in static equilibrium, is currently checked using the Newton-Euler equations also used by physic simulation frameworks [21].

For stable configurations regarding the planar case, our quality metric is:

$$Q = 1 - \frac{d(\hat{c}_\text{O}, \hat{c}_\Omega)}{\hat{l}_\Omega} - \frac{d(c_\text{O}, \Omega)}{A(\Omega)}$$

with $\hat{l}_\Omega$ as distance between $\hat{c}_\Omega$ and the boundary of Ω along the failure direction and the area of the convex hull $A(\Omega)$. The best quality value is one. This is the case, if and only if the two quotients are zero. The smaller a quality value, the worse is the computed object configuration.

The first quotient is a measure for the tilt turn. The closer the projected center of mass $\hat{c}_\text{O}$ is to the boundary of the convex hull Ω the greater is the distance $d(\hat{c}_\text{O}, \hat{c}_\Omega)$. A configuration is more stable if the distance $d(\hat{c}_\text{O}, \hat{c}_\Omega)$ decreases and the distance $\hat{l}_\Omega$ increases. The second quotient is the ratio between footprint size and height of the object. The smaller this ratio, the larger is the footprint area at a constant height, or the smaller is the hight of the object at a constant footprint area. Both possibilities imply that the object configuration is more stable at a smaller quotient-value $\frac{d(c_\text{O}, \Omega)}{A(\Omega)}$.

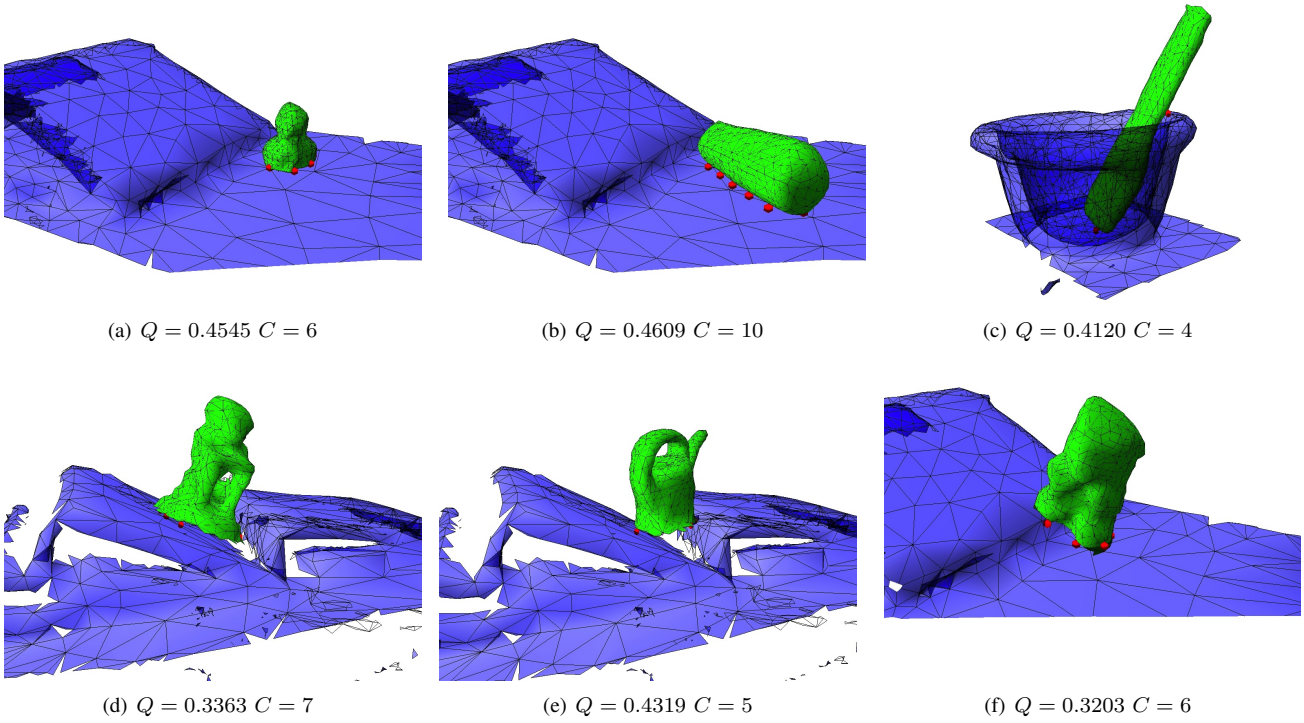
VII. Experiments

For the experimental evaluation we chose $d_\text{vis} = d_\text{g}$ as negative z-axis. All computations where performed on an Intel i7 CPU. First, we describe our object modelling pipeline (Section VII-A). Afterwards, we present planning results of various object and placement areas and discuss them (Section VII-B). At last we evaluate the computation time, and compare our approach to a physics simulation framework (Section VII-C).

A. Object Model

We now outline our object modelling pipeline used for the experiments. As sensor we use a hand held depth camera. The relative registration between two camera images is carried through using the *iterative closest point* algorithm. Since we know the initial position of the camera we are able to register the reconstructed object model to a robot's coordinate system.

Each point cloud of a depth image is integrated into a regular grid using a signed distance function. We convert this



(a) $Q = 0.4545$ $C = 6$ (b) $Q = 0.4609$ $C = 10$ (c) $Q = 0.4120$ $C = 4$

(d) $Q = 0.3363$ $C = 7$ (e) $Q = 0.4319$ $C = 5$ (f) $Q = 0.3203$ $C = 6$

Fig. 5: Selected experimental results for four different placement areas (blue) and seven objects (green). The value Q is the quality rating and C is the number of computed contact points (red).

grid-based model into a triangulated surface model using the *marching cube* algorithm. As a last step of our pipeline, we simplify these surface models to a certain amount of triangles (usually 1000).

Placement area and object are modelled independently from each other.

B. Results and Evaluation

Fig. 5 shows results of our placement planning for various objects and placement areas. All configurations are computed with respect to the same floor point. For the placement area in Fig. 5(c), the floor point is located almost at the center of the bottom. For those in Fig. 5(a) and 5(d), the floor point is located almost at the center of the slope (see Fig. 2(a)).

The planner computes generally stable configurations even if the chosen floor point is located in a region where the initial configuration computation had not calculate a stable configuration. Together with the local optimisation, a stable configurations is then reached. For planar and horizontal placement areas, the initial configuration computation already results in stable configurations (Fig. 5(a) and Fig. 5(b)).

To get an idea of the quality value of a configuration, we compare the configurations of Fig. 5(a) ($Q = 0.4545$) and Fig. 5(f) ($Q = 0.3203$). Both configuration have a comparable quotient between their area of the contact point convex hull and the distance of the object's center of mass to this hull. However, the object in Fig. 5(f) is much more tilted than the other one, which causes differences at the quality measures. Our experiments have shown that object configurations with a quality value under 0.3 should be considered unstable.

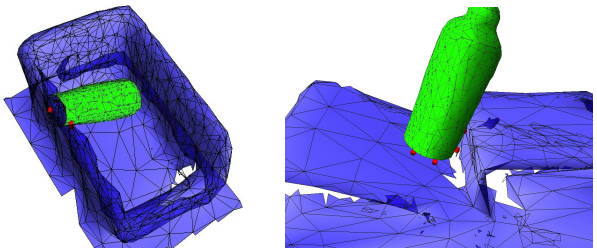
If we compare Fig. 5(d) and Fig. 5(e), we can see that the watering can (Fig. 5(e)) was first translated into the failure direction and then tilted, whereas the first object (Fig. 5(d)) was first tilted into the failure direction and then translated. This gives an explanation for the different intermediate construction order: The latter object generated three or more independent contact points after the first translation onto the placement area. Thereby, the configuration is not tilted once more Instead the local optimisation is applied, which leads to a translation into the failure direction.

Fig. 6 shows two placement results that are stable but do not meet the planar criteria (Section VI). Those configurations arise especially if the placement area has a box-like shape.

Especially Fig. 5(c) underlines that our approach computes stable configurations for complex placement areas. However, our planner is currently not able to hang objects onto the placement area. Furthermore, we can not rate the quality of non-planar contact point distributions. However, we can decide whether the configuration is stable.

C. Runtime Analysis

Our approach has two major parameters with high impact to the computation time: On the one hand the number of surface patches used for the models of placement area and object and on the other hand the number of tilt computations for one floor point.



(a) $C = 4$ (b) $C = 4$

Fig. 6: Two samples for stable configurations witch do not meet the plane case criteria. The placement area is colored blue, the object green and the collision points red.

We simplify the model of the placement area to 1000 triangles and vary the amount of triangles of the object model from 200 to 1000. Additionally, we rotate every object by ten times with equidistant angles around the second main axis to make the time measurement independent from the object orientation. We use the placement area displayed in Fig. 5(d) and all objects displayed in Fig. 5 and Fig. 6. We measure the computation time for the placement planning with one, three and five maximal allowed tilt computations in a row (respective 2^1, 2^3 and 2^5 maximum single tilt computations - compare Section V-B). All tilts before and after a possible improvement step are counted. Fig. 7 plots the mean measured computation times over the number of triangles of the object model.

For one and three tilts, the computation time is nearly linear in the number of surface patches. The amount of surface patches increases by a factor of five during our time measurement, but the computation time increases only by a factor less than two. The variation of the measured computation time also increases with the amount of allowed tilts. The first reason for this is that more translations become possible which is caused by the local optimisation step (compare Section VI). The second reason is, if a placement configuration has a complex contact situation or is considered stable, the algorithm terminates even if the amount of performed tilts is less than the maximum ones allowed.

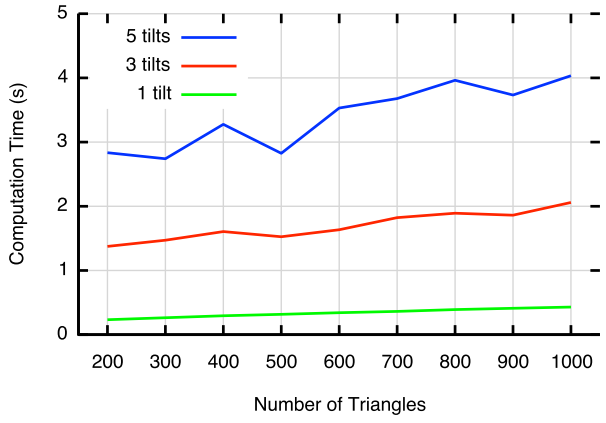


Fig. 7: Plot of computation times over number of triangles of the object model for three amounts of allowed object tilts (one, three, and five).

Both reasons are dependent on the object mesh, since the mesh simplification algorithm outputs slightly varying shapes dependent on the requested amount of triangles.

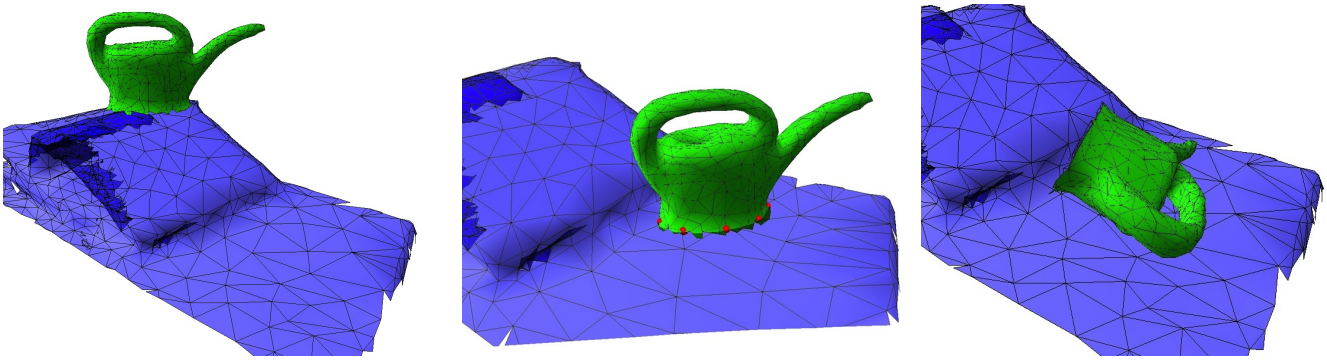
Compared to the computation time for a physics simulation framework [21], our algorithm is slower. The runtime of this simulation is about 200-700 ms until the object remains in equilibrium, which is mostly caused by the number of collision tests. The time needed to compute a new position and orientation in each time step is negligible. But physics simulation frameworks have one major drawback. Since the position and orientation change is affected by applying impulses or forces, it is difficult to encroach the computation as we do during the local optimisation step in Section VI. And since the placement will be performed by a robot, a physical correct placement process is not required, only a stable goal configuration is necessary. Fig. 8 shows for example the resulting stable configurations using our approach (Fig. 8(b)) and the rigid body simulation (Fig. 8(c)) with the same start configuration (Fig. 8(a)). Furthermore, the reconstructed object models include holes and are especially not watertight. This fact causes instable contact force results and, as a consequence, the computations of the rigid body simulation become unstable. During the experiments the penetration between object and placement area became sometimes so large that the object fell through the placement area.

VIII. Conclusion

We introduced an educated sample based placement planner for unknown sensor-modelled objects and placement areas. It uses only geometrical information in its computations. Therewith, we introduced a novel placement configuration calculation that offers the opportunity to influence the configuration computation easily. We use this opportunity for a local optimisation of unstable configurations, by recalculating the movement direction or tilt axis of the object. Not only planar placement areas are considered, but also for complex areas the planner performed well.

The experiments point out that the planner is able to cope with small holes in the surface of object models. However, if some parts of the object model are not reconstructed at all, the planner will not be able to overcome this issue. However, compared to a physics simulation framework our approach is more robust considering rough and noisy surfaces and holes. Further on, we indicate that different objects could be placed stable onto complex and planar placement areas. Runtime measurements show that the computation time only increases slightly for object models with more surface patches.

Future work includes completing the missing failure direction computation for an arbitrary contact point distribution. Additionally we carry out a user study to observe placements for objects to extract semantical criteria for the planning process. Moreover, one could target on the conservativeness of the planner for objects with huge holes or even complete unmodelled regions of the object. To improve the computation time of the algorithm, parts of the planner could be executed by the GPU.



(a) Start configuration (b) Our approach (c) Rigid body simulation

Fig. 8: Beginning with a common starting configuration in (a), our placement planner outputs (b) and a rigid body simulation outputs (c).

REFERENCES

[1] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer, "Upright orientation of man-made objects," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 42, 2008.

[2] T. Lozano-Pérez, J. L. Jones, E. Mazer, and P. A. O'Donnell, "Task-level planning of pick-and-place robot motions," *Computer*, vol. 22, no. 3, pp. 21–29, 1989.

[3] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka, "Rigid 3D geometry matching for grasping of known objects in cluttered scenes," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 538–553, Mar. 2012.

[4] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, Mar. 2012.

[5] M. a. Roa, M. J. Argus, D. Leidner, C. Borst, and G. Hirzinger, "Power grasp planning for anthropomorphic robot hands," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012, pp. 563–569.

[6] M. J. Schuster, J. Okerman, H. Nguyen, J. M. Rehg, and C. C. Kemp, "Perceiving clutter and surfaces for object placement in indoor environments," *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pp. 152–159, Dec. 2010.

[7] J. Glover, G. Bradski, and R. B. Rusu, "Monte Carlo Pose Estimation with Quaternion Kernels and the Bingham Distribution," *Robotics: Science and Systems VII*, p. 97, 2012.

[8] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, 2000.

[9] F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit, "A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing," *Annals of Operations Research*, vol. 179, no. 1, pp. 203–220, Oct. 2008.

[10] E. E. Bischoff and M. S. W. Ratcliff, "Issues in the development of approaches to container loading," *Omega*, vol. 23, no. 4, pp. 377–390, 1995.

[11] W. Han, J. A. Bennell, X. Zhao, and X. Song, "Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints," *European Journal of Operational Research*, pp. 1–18, Apr. 2013.

[12] A. Pasha, "Geometric Bin Packing Alogrithm for Arbitrary Shapes," Ph.D. dissertation, University of Florida, 2003.

[13] A. Edsinger and C. Kemp, "Manipulation in Human Environments," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, Dec. 2006, pp. 102–109.

[14] T. Lozano-Pérez, J. L. Jones, E. Mazer, and P. A. O'Donnell, "Task-Level Planning of Pick-and-Place Robot Motions," *Computer*, vol. 22, no. 3, pp. 21—-29, 1989.

[15] a. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sep. 2011, pp. 4627–4632.

[16] J. Baumgartl and D. Henrich, "Fast Vision-based Grasp and Delivery Planning for unknown Objects," in *7th German Conference on Robotics (ROBOTIK 2012)*, 2012.

[17] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, H. Onda, T. Yoshimi, and Y. Kawai, "Object placement planner for robotic pick and place tasks," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 980–985.

[18] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to Place New Objects in a Scene," *IJRR*, vol. 31, no. 9, Feb. 2012.

[19] Y. Jiang and A. Saxena, "Hallucinating humans for learning robotic placement of objects," in *ISER*, 2012.

[20] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Technical Report TR99-018, Department of Computer Science, University of North Carolina, Tech. Rep., 1999.

[21] Bullet Physics Library, "http://bulletphysics.org/," 2013.