Non-Penetration Iterative Closest Points for Single-View Multi-Object 6D Pose Estimation

Mengchao Zhang[1] and Kris Hauser[2]

***Abstract*— This paper presents a novel iterative closest points (ICP) variant, non-penetration iterative closest points (NPICP), which prevents interpenetration in 6DOF pose optimization and/or joint optimization of multiple object poses. This capability is particularly advantageous in cluttered scenarios, where there are many interactions between objects that constrain the space of valid poses. We use a semi-infinite programming approach to handle non-penetration constraints between complex, non-convex 3D geometries. NPICP is applied to a common use case for ICP as a post-processing method to improve the pose estimation accuracy of a rough guess. The results show that NPICP outperforms ICP, assists in outlier detection, and also outperforms the best result on the IC-BIN dataset in the Benchmark for 6D Object Pose Estimation.**

I. Introduction

Accurate detection and estimation of the six degree-of-freedom (6-DOF) pose (position and orientation) of objects in a scene is needed for augmented reality and robot manipulation tasks, so pose estimation has been an active topic of research for many years. However, state-of-the-art object detection [8] and pose estimation [20] systems are still noisy, often including many false positives, duplicate detections, and inaccurate pose estimates. These problems get worse when objects are cluttered in the scene, and when the same object appears multiple times. Under these circumstances, local information is often insufficient to disambiguate poses, whereas global information about the scene such as geometric non-penetration and support relationships can help narrow down the space of valid poses.

This paper addresses the use of non-penetration constraints to improve the accuracy of local optimization for single-object and multi-object pose estimation. Specifically, we address the problem that complex, non-convex 3D geometries impose a large number of constraints, which can number in the tens or hundreds of thousands. Rather than use standard nonlinear programming (NLP) techniques, we use the idea of the exchange method for semi-infinite programming (SIP) [17]. The optimizer progressively generates some constraints to be included in a smaller NLP, and the series of problems converges toward a true optimum of the original problem.

Our proposed NPICP algorithm implements our SIP non-penetration approach in an Iterative Closest Points (ICP) algorithm that minimizes a point cloud registration energy while respecting non-penetration constraints between objects and the free-space of an RGB-D image. It is evaluated on the single-view multi-object 6D pose estimation problem of the IC-BIN dataset [4]. As a post processor for a deep neural network (DNN) object detector/pose estimator, NPICP performs geometry-based "cleanup" of physically implausible poses. The advantages of NPICP are most apparent in cluttered scenes, such as the example from the IC-BIN dataset [4] shown in Fig. 1. The benefits of NPICP are most significant under perfect detections, since it is a local method and not able to overcome a poor initialization. Nevertheless, it outperforms ICP under both perfect and noisy detections, and outperforms the top performing pose estimator for the IC-BIN dataset in the Benchmark for 6D Object Pose Estimation (BOP Challenge) [1].

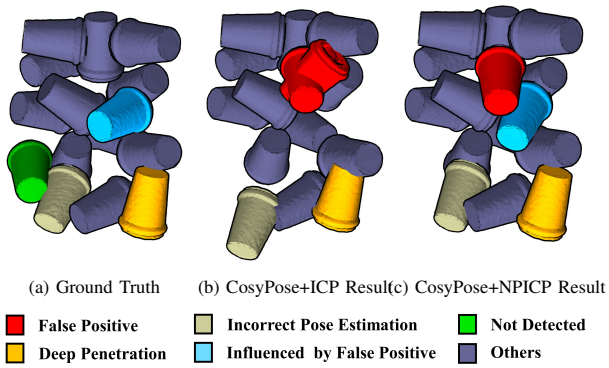


Fig. 1: Views from below of the reconstruction results of Scene 1 in the IC-BIN dataset, showing (a) ground truth, (b) results of the best performer on the IC-BIN dataset in the BOP Challenge which uses ICP to refine the result of the CosyPose detector, (c) NPICP used in place of vanilla ICP. Compared to the results in (b), NPICP resolves the deep penetration of the yellow cup with the cup next to it, and corrects the pose of the beige cup. It is also partially able to recover the pose of the cyan cup, but is influenced by a false positive from CosyPose (red). [Best viewed in color.]

II. Related work

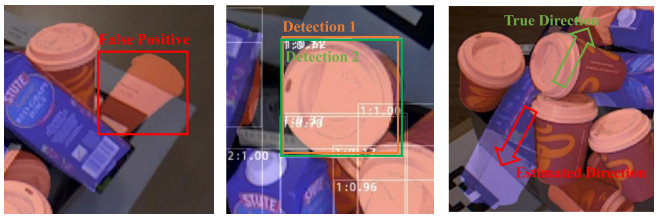
The BOP challenge is an open competition which aims to capture the state of the art in the field of 6D object pose estimation from an RGB-D image. Point pair features (PPF) methods [6] are a common approach in which pairs of oriented 3D points are matched between the test scene's point cloud and the model of the 3D object. The object's pose is then estimated through a voting scheme. However, PPF methods are not robust to background clutter or sensor noise, and their quadratic computational complexity is an important drawback [2].

DNN methods have made great progress in object detection [8], [18], which further motivated their use in pose estimation [20], [21]. After being dominated by the methods

* This work was supported by NSF Grant IIS #-1911087.

[1]Mengchao Zhang is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign. mz17@illinois.edu

[2]Kris Hauser is with the Department of Computer Science, University of Illinois at Urbana-Champaign. kkhauser@illinois.edu



(a) False positive (b) Duplicated detection (c) Flipped direction

Fig. 2: Common problems in DNN object detection and pose estimation results. The blue and red shapes are rendered objects at the estimated poses. (a) A coffee cup is incorrectly detected in free space. (b) The object is detected twice. (c) The estimated pose of the juice box is in a flipped direction compared with the ground truth. [Best viewed in color.]

based on PPF in the first several editions of the BOP challenge, in 2020, the performance of deep neural network (DNN) methods ultimately caught up [12]. However, as shown in Fig. 2, the results of DNN methods can be noisy. Thus, DNN pose estimation is often followed by local refinement, such as ICP [19], which performs local registration to improve the match between the point cloud and object geometries. Particularly when much of an object is occluded, local registration may not be enough to fix these errors. Our work enhances local optimization by including constraints to enforce non-penetration between objects, and between objects and free space.

Other authors have used object interactions to improve pose estimation accuracy. Mitash et al (2019) [14] propose a method which generates multiple candidate poses per object at first, and then searches over the Cartesian product of these individual object pose candidates to find the optimal scene hypothesis. However, due to the global optimization nature of this method, its run time increases exponentially when the number of objects in the scene increases, and in the paper's experiments, scenes containing at most 3 objects were tested. Compared with this method, our method only considers local interactions between objects, such that the run time increases roughly linearly with the number of objects in the scene. Our experiments on the IC-BIN dataset contains up to 19 objects in scenes 1 and 3, which would be intractable for prior work.

Mitash et al (2020) [15] use the penetration between objects to reject false positive pose hypotheses generated by a DNN. We use a similar approach to help refine the output of the CosyPose DNN. However, the primary focus of our work is to include penetration information in pose refinement.

III. Non-Penetration Iterative Closest Points

A. Problem Setup

We address the single-view multi-object 6D pose estimation problem, that is estimating the poses of N objects in an RGB-D image I, assuming known 3D object geometries. Since our method is most useful as a local registration technique for a DNN-based initializer, we assume an object detector generates a set of estimated objects $e_{1:n} = [e_1, \cdots, e_n]$, where $e_i = (e_i^c, e_i^q, e_i^s, e_i^M)$ contains the object class $e_i^c \in \{1, \cdots, C\}$, the 6D object pose $e_i^q \in SE(3)$ with respect to the camera, a confidence score $e_i^s \in [0, 1]$ and a binary mask image e_i^M. Our approach generates a refined set of object estimates $o_{1:m} = [o_1, \cdots, o_m]$ where $o_i = (o_i^c, o_i^q, o_i^s, o_i^M)$. Note that the first stage of our algorithm filters out some proposed object estimates, so m is often smaller than n.

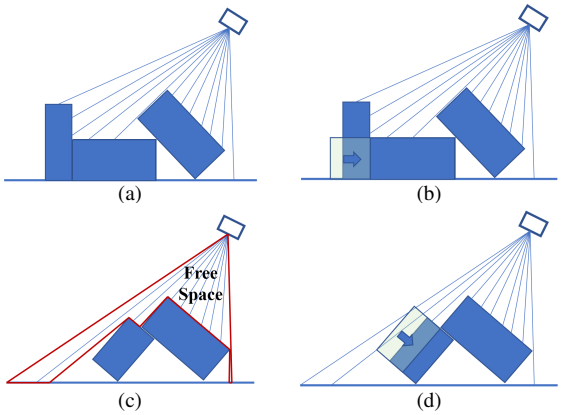


Fig. 3: Illustration of how non-penetration information is helpful to improve pose estimation accuracy. Pure ICP cannot distinguish between poses in (b) or poses in (d), whereas non-penetration constraints are able to disambiguate the correct pose. (a) and (b) illustrate how object non-penetration constraints disambiguate the correct pose. (c) and (d) illustrate how object-free space non-penetration constraints disambiguate the correct pose.[Best viewed in color.]

To represent the object's geometry, we use a dual representation, consisting of both the signed distance function (SDF) and point cloud sampled from the surfaces of the object. We denote the SDF of o_i in its local frame as $g_i(\cdot) : \mathbb{R}^3 \to \mathbb{R}$. The surface of o_i is also denoted as a point cloud in its local reference frame as PC_i^l.

B. Approach

Our approach uses two forms of non-penetration constraints to improve pose estimation accuracy: object-object penetration and object-free space penetration (Fig. 3). ICP only works to minimize the error of matches between points and object geometry, but for occluded objects or those with little texture, it has little information to leverage to distinguish the true pose.

As illustrated in Fig. 3 (b), object non-penetration constraints can find shift poses closer to the ground truth. Free-space constraints are also very informative, because each point in the depth image is the closest point between the camera and the objects in the scene along that direction, so all objects should be behind the observed surface. So, from the RGB-D image I and the intrinsic parameters K of the camera we build a free space object o_{free} consisting of these line segments in Fig. 3 (c). Fig. 3 (d) illustrates how object-free space non-penetration can improve pose estimation by pushing the left object downward under the observed surface.

However, directly adding all the estimated objects to NPICP and optimizing their poses jointly can lead to low accuracy and low efficiency. The false positives may take the space of correctly detected objects, and very deep penetration can cause trouble for NPICP. Besides, adding more object to the optimization will result in longer solve time.

Thus, we design a two-step method. In Step 1, Estimation

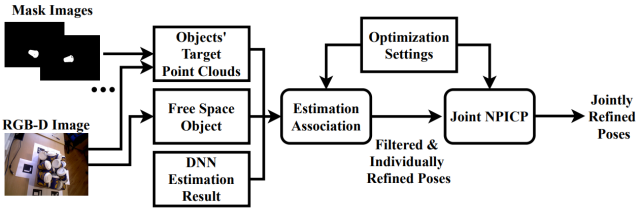


Fig. 4: The workflow of our method.

Association (EA), we filter out bad pose estimations using geometry information, and we run NPICP individually on each object to improve the estimated poses, which only considers the non-penetration between the object and the free space. In step 2, Joint NPICP, we run NPICP over all the kept objects to optimize their poses jointly, using the non-penetration both between objects and between objects and the free space. The pipeline is illustrated in Fig. 4.

Next, we discuss each of the steps in detail.

1) Step 1: Estimation association

DNN object detection/pose estimation results can include many false positives, and these are critical to remove because spurious objects can cause NPICP joint pose optimization to lose accuracy. Otherwise, NPICP will try to prevent penetration between phantom objects, which can cause poor estimates to cascade. Our first step (Alg. 1) filters out likely false positives, and it also generates the target (scene) point clouds which are used for the registration energy in NPICP. It proceeds by examining each estimated object, in order of most to least confident, by the following steps:

a) Low Quality Mask Removal

Line 3 removes estimations whose masks have fewer than n_{min} valid pixels. DNN object detection methods sometimes give detections that consist of very few pixels, which cannot generate good pose estimation through ICP or its variants, so these are filtered out.

b) Translation Adjustment

DNN pose estimation methods often produce large errors in the estimated distance from the object to the camera, so Lines 4–6 use the point cloud to adjust the initial translation estimation. To do so, we render the object into a depth image where the object is placed at the estimated configuration e_i^q and then extract points PC_{ren} belonging the mask e_i^M. We shift the estimated pose to match the centroid of the rendered point cloud PC_{ren} to the centroid of the portion of the scene point cloud containing e_i

This stage also extracts a target scene point cloud PC_{scn} to be used in the NPICP registration energy. The extraction is performed by $GetObjectPointcloud(I, K, e_i^M, n_{obj}, n_n, r_{std})$ in Algorithm 1. First, we use the object mask and camera intrinsics to determine a dense point cloud PC^s_{reg}, then we down-sample to n_{obj} points to accelerate the optimization. To make sure the down-sampled points are representative, the iterative farthest point method is used. Then, we run the statistical outlier removal method in Open3D [22] on the down-sampled point cloud, with parameters n_n and r_{std} specifying how many neighbors are taken into account in order to calculate the average distance for a given point, and r_{std} is an outlier rejection threshold level based on the standard deviation of distances.

c) Individual NPICP

To make sure that the object is roughly at the ground truth pose, such that the following estimation quality check step will not filter out good results, we run NPICP on each estimated object before we pass it to the quality check. This individual NPICP only considers the penetration between the object and the free space object.

d) False Positive Removal

Next, Line 8 tests the penetration with prior objects $o_{1:m}$ and o_{free}. If all the absolute penetration distances are smaller than a threshold d_1, the sum of the absolute penetration distances is smaller than a threshold d_2, and the absolute penetration distances with the free space is smaller than another threshold d_3, then we will go to the next step. Otherwise, we discard the estimation. All of d_1, d_2 and d_3 are determined by the size of the geometry to be examined.

e) Duplicated Detection Removal

DNN object detection methods sometimes detect an object instance multiple times. So, for the next estimated object e_i to be examined, Line 9 checks the distance $\delta q_{ij} = \|e_i^q.t - o_j^q.t\| \ \forall j \in 1, ..., m$, where $e_i^q.t$ is the translation of the estimated pose. If any δq_{ij} is smaller than some threshold δq_{min}, then we consider the estimation as a duplication.

Algorithm 1 Estimation Association

Input:

- RGB-D image I, camera intrinsic parameters K.
- Estimations $e_{1:n}$.
- Free space object o_{free}.
- Maximum iterations N_{max}^{single}, step size tolerance ϵ, index addition and deletion threshold g_{max}.
- Minimum # of pixels occupied in object mask n_{min}.
- Maximum # of points retained in object point clouds n_{obj}.
- Outlier removal parameters n_n, r_{std}, inlier percentage p.

1: Number of retained objects $m = 0$
2: **for** e_i in $Sort(e_{1:N})$ **do**
3: **if** $ValidPixel(e_i^M) > n_{min}$ **then**
4: $PC_{scn} \leftarrow GetObjectPointcloud(I, K, e_i^M, n_{obj}, n_n, r_{std})$
5: $PC_{ren} \leftarrow RenderPointcloud(e_i^c, e_i^q, e_i^M, K)$
6: $e_i^q.t \ +\!\leftarrow Centroid(PC_{scn}) - Centroid(PC_{ren})$
7: $e_i^q \leftarrow NPICP([e_i], [PC_{scn}], o_{free}, N_{max}^{single}, g_{max}, \epsilon, p)$
8: **if** not $FPRemoval(e_i, o_{1:m})$ **then**
9: **if** not $DuplicateRemoval(e_i, o_{1:m})$ **then**
10: $m \leftarrow m + 1$
11: $o_m \leftarrow e_i$
12: $PC_m^s \leftarrow PC_{scn}$
13: $o^s_{1:m} \leftarrow ModifyConfidenceScore(o^s_{1:m}, o^M_{1:m})$
14: **return** $o_{1:m}, PC^s_{1:m}$

f) Score Modification

For all retained objects, we adjust their confidence scores from the DNN object detection method according to a heuristic so that objects with fewer occlusions have higher confidence scores (these are used in the scoring function in BOP challenge; this step is not strictly necessary). We modify the score according to the number of pixels in the mask

image o_i^M, setting $o_i^s = o_i^s + \frac{ValidPixel(o_i^M)}{Pixel_{max}}\ \forall i = 1, \dots m$, where $Pixel_{max} = \max_{i=1}^m ValidPixel(o_i^M)$.

At the end of step 1, all objects are considered to be inliers and NPICP only modifies their poses hereafter.

2) Step 2: Non-penetration Iterative Closest Point

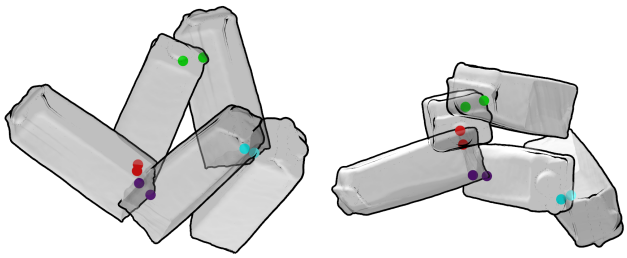
The next stage is the main contribution of this paper, which is an ICP implementation with non-penetration constraints. This includes the non-penetration constraints between each object and the free space, and the non-penetration constraints between each pair of objects.

We use the local SDF representations of each object, $g_i(\cdot)$, to measure penetration. Denoting the volume of the free space object o_{free} as $\mathcal{V}_{free}$, the non-penetration constraint between o_{free} and o_i is expressed as $g_i((o_i^q)^{-1} \cdot y) \geq 0\ \forall y \in \mathcal{V}_{free}$. The non-penetration constraint between object i and object j is expressed as $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \geq 0\ \forall y \in PC_j^l$. Although there are an intractable number of points in these constraints, we follow the SIP formulation of Hauser (2018) which uses fast deepest-penetrating-point techniques to construct a manageable number of constraints [7].

The point cloud registration energy of object o_i is defined by the sum of squared distances of the geometry to the scene point cloud PC_i^s. A standard technique used in ICP is to reject outliers by only registering the closest p percent of points in PC_i^s to o_i [19]. This improves the the accuracy and stability of pose estimation by rejecting invalid matches. Denote $d_i(o_i^q) = [g_i((o_i^q)^{-1} y_{i,1}^s)^2, \cdots, g_i((o_i^q)^{-1} y_{i,N_i^s}^s)^2]^T$ as the vector concatenating the squared distances of points in PC_i^s to o_i, with $N_i^s = |PC_i^s|$. Also, let $\mathbb{1}_p(d_i)$ be an inlier selection function, which selects the smallest p percent points from d_i. The returned vector has the same dimension as d_i, and $d_{ij} = 1$ if the j'th distance is an inlier and $d_{ij} = 0$ if it is an outlier. Then the NPICP optimization problem is given by the following form:

$$\begin{aligned} \arg\min_{o_{1:m}^q} \ & \frac{1}{2} \sum_{i=1}^m \mathbb{1}_p(d_i(o_i^q))^T d_i(o_i^q) \\ \text{s.t.} \quad & g_i((o_i^q)^{-1} y) \geq 0\ \forall i \in 1, \cdots, m\ \ \forall y \in \mathcal{V}_{free} \\ & g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \geq 0\ \ \forall y \in PC_j^l \\ & \forall i \in 1, \cdots, m,\ \forall j \in 1, \cdots, m,\ i \neq j. \end{aligned} \tag{1}$$

The primary challenge is that $\mathcal{V}_{free}$ and each of the PC_i^l may contain tens or hundreds of thousands of points, making direct optimization of (1) using an NLP solver intractable. We leverage semi-infinite programming (SIP) techniques to improve the speed of optimization. SIP allows constraints to depend on free variables, known as index points, which may range over an infinite set. Certainly, an infinite number of constraints cannot be solved directly in a standard NLP solver. Thus, an exchange method [9] is used to instantiate a series of finite dimensional optimization problems, each of which progressively adds some number of constraints. Also, through using a judicious index point selection procedure, called an oracle, the series of problems converges toward one that contains a true optimum of the original infinite dimensional problem. The same approach can be applied to greatly accelerate optimization in problems with a very large number of constraints, like ours.



(a) Front View (b) Top View

Fig. 5: Index points between objects generated by the maximum-violation oracle when the objects are at the configurations shown in the figure. The points are the closest points (or deepest penetration points) between each pair of objects. The index points between different pair of objects are represented by different colors. [Best viewed in color.]

We use a maximum-violation oracle [17], which identifies a parameter value that has a large effect on the next iterated solution, to instantiate index points. Specifically, on iteration k of the optimization, we calculate the most violating parameter of each non-penetration constraint:

$$y_{i,j}^{min} \leftarrow \arg\min_{y \in PC_j^l} g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \tag{2}$$

and similarly for the free-space violation constraint. An example of the index points between objects instantiated by the most-violation oracle is shown in Fig. 5.

We define the set of instantiated index parameters as Y, and choose an index addition and deletion threshold g_{max}. Then if $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y_{i,j}^{min}) \leq g_{max}$, we will add $(y_{i,j}^{min}, i, j)$ to Y. To simplify notation, we regard the free space object as object 0 ($o_{free} \equiv o_0$) and set its pose to the identity transform. We also delete constraints from the constraint set when they are not deemed necessary. To be specific, if any index parameter $(y_{i,j}^{min}, i, j)$ in Y that satisfies $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y_{i,j}^{min}) > g_{max}$, then we will delete it from Y to accelerate the following NLP solving.

Letting Y^k be the index set generated at iteration k, and freezing the inlier selection vector as $\mathbb{1}^k \leftarrow \mathbb{1}_p(d_i(o_i^{q,k-1}))$ for the poses at the start of the iteration, we solve the following the nonlinear program to determine a desired set of poses at iteration k:

$$\begin{aligned} \arg\min_{o_{1:m}^q} \ & \frac{1}{2} \sum_{i=1}^m (\mathbb{1}^k)^T d_i(o_i^q) \\ \text{s.t.} \quad & g_i((o_i^q)^{-1} \cdot o_j^q \cdot y_j^l) \geq 0\ \ \forall (y_j^l, i, j) \in Y^k. \end{aligned} \tag{3}$$

Simply taking the NLP solution as a step could cause infeasibility for a non-instantiated constraint, so instead we use a line search. The line search ensures a sufficient decrease in a merit function that penalizes the worst-case violation of a constraint. Let $g_{i,j}^*(o_i^q) \equiv \min_{y \in PC_i^l} g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \geq 0$ be the worst-case violation of the collision constraint between objects i and j. This is then used in the merit function $\phi(o_{1:m}^q, Y, w_1, w_2) = f(o_{1:m}^q) + w_1 \times |D(o_{1:m}^q, Y)^-| + w_2 \times |g^*(o_{1:m}^q)^-|$ (see Algorithm 2 Line 7). In this function, $D(o_{1:m}^q, Y)$ returns a stack of the distances of the instantiated index points, and $g^*(o_{1:m}^q)$ returns a stack of the closest distances between each pair of objects and the closest dis-

tance between each object and the free space object, where $(\cdot)^- \equiv \min(\cdot, 0)$. We also add the penetration points detected during line search to the index set in the next iteration.

NPICP is summarized in Algorithm 2.

Algorithm 2 Non-Penetration Iterative Closest Point

Input:

- Estimations $o_{1:m}$.
- Target point clouds $PC^s_{1:m}$.
- Free space object o_{free}.
- Maximum iteration number N_{max}, step size tolerance ϵ.
- Index addition and deletion threshold g_{max}.
- Inlier selection parameter p.

1: $k \leftarrow 0$
2: $o^k_{1:m} \leftarrow o_{1:m}$, $Y^k \leftarrow \{\}$, $Y^k_{LS} \leftarrow \{\}$
3: **for** $k = 1, \ldots, N_{max}$ **do**
4: $\quad Y^k \leftarrow DeleteIndex(Y^{k-1}, o^{k-1}_{1:m}, o_{free}, g_{max})$
5: $\quad Y^k \leftarrow Oracle(Y^k, Y^{k-1}_{LS}, o^{k-1}_{1:m}, o_{free}, g_{max})$
6: $\quad$ Solve Equation 3 to get $o^{q,k}_{1:m}$
7: $\quad$ Run line search on ϕ to get step size α and new penetration points detected during line search Y^k_{LS}
8: $\quad \Delta o^q_{1:m} \leftarrow \alpha(o^{q,k}_{1:m} - o^{q,k-1}_{1:m})$
9: $\quad o^{q,k}_{1:m} \leftarrow o^{q,k-1}_{1:m} + \Delta o^q_{1:m}$
10: $\quad \triangleright$ Test for convergence
11: $\quad$ **if** $\|\Delta o^q_{1:m}\|/m \leq \epsilon$ **then return** $o^k_{1:m}$

IV. Experiments and Results

We tested our method on the IC-BIN dataset, which contains two object types that appear in multiple locations with heavy occlusion in a bin-picking scenario (Fig. 6). These scenes also have severe foreground occlusions and background distractors.

We followed the evaluation methodology provided by the BOP Challenge to evaluate the pose estimation results. The performance of a method on a dataset is measured by the Average Recall: $AR = (AR_{VSD} + AR_{MSSD} + AR_{MSPD})/3$. VSD is the Visible Surface Discrepancy [11], [10], which measures the distance difference in the depth image using only the visible part of the object in the image. MSSD is the Maximum Symmetry-Aware Surface Distance [5], which measures the maximum surface deviation and is relevant to robotic manipulation. MSPD [10] is the Maximum Symmetry-Aware Projection Distance, which changes the average distance in 2D projection [3] by the maximum distance and is relevant for augmented reality applications. An object is only considered in the evaluation if at least 10% of its surface is visible. All experiments were run on a single core of a 3.6 GHz AMD Ryzen 7 processor.

We use the following parameters in all the experiments:

- $N^{single}_{max} = 10$, $N^{joint}_{max} = 10$, $\epsilon = 5e-4$, $g_{max} = 5\ mm$.

(a) Scene 1 (b) Scene 2 (c) Scene 3

Fig. 6: Example images of the IC-BIN dataset. [Best viewed in color.]

Scene	Method	AR	Pen/obj (mm)
1	Init	0.376	10.84
	ICP	0.695	5.18
	NPICP	0.808	1.53
2	Init	0.425	9.04
	ICP	0.788	2.07
	NPICP	0.814	0.90
3	Init	0.400	11.50
	ICP	0.686	4.37
	NPICP	0.823	1.42

TABLE I: The results of ICP and NPICP given perfect detection on the three scenes of IC-BIN seperately.

- $n_{min} = 100$, $n_{obj} = 200$, $n_n = 50$, $r_{std} = 2$, $p = 95$.
- Denoting the smallest dimension of the bounding box of the object to be examined as d^*, we use $d_1 = \frac{d^*}{3}$, $d_2 = \frac{d^*}{2}$ and $d_3 = \frac{d^*}{3}$.

A. Perfect detection, disturbed ground truth pose

The first set of experiments compares ICP and NPICP when we have perfect detection but imperfect pose estimation. In this case, we know the true number of objects $m = n$ and their classes, and we skip the Estimation Association filters (Lines 3, 8, and 9 in Algorithm 1) and directly set $o_{1:m} := e_{1:n}$. The initial estimated poses are the ground truth poses plus random disturbances, with each disturbance sampled as a random rotation in the range $\pm 0.25\ rad$ and translation in the range $\pm 3\ cm$. We run both ICP and NPICP on three scenes of IC-BIN. The ICP baseline we used is the point-to-point ICP in Open3D [22]. For a fair comparison, we run the translation adjustment for both algorithms.

The results of this experiment are shown in Table I. Init is the initial guess. AR is the average recall and Pen/obj is the average penetration of an object with all its surrounding objects. Compared with ICP, NPICP greatly increases the AR and decreases the penetration between objects. Also, the improvement of NPICP over ICP is larger when the scene is highly cluttered, as in scenes 1 and scene 3.

To validate the design choice of our algorithm, we test the following NPICP variants:

- NPICP Ind: Run NPICP individually on each object (skipping Joint NPICP in Fig. 4).
- NPICP Joint: Run NPICP jointly over all the objects (skipping line 7 in Algorithm 1).
- NPICP: Run NPICP both individually and jointly.

Results on the IC-BIN dataset are shown in Table II. Time/img is the average time used for the pose optimization of each image. NPICP provides the best results of all the three variants while taking the most time. However, NPICP is only slightly slower than NPICP Joint, which we believe is because the individual optimization makes the initial condition of the joint optimization more ideal, which enables the joint optimization to converge more quickly.

The Pen/obj of NPICP Ind is larger than that of ICP because NPICP Ind only constrains penetration with the free space object, such that there is no guarantee that it will decrease the penetration between objects.

B. DNN-based detection, DNN-based pose estimation

The second experiment compares ICP and NPICP given DNN object detection and pose estimations. We use the

Method	AR	Pen/obj (mm)	T_{err}/obj		Time/img (s)
			μ	σ	
Init	0.393	10.88	31.3	7.6	-
ICP	0.704	4.40	19.2	14.4	17.6
NPICP Ind	0.743	4.85	14.9	17.5	26.8
NPICP Joint	0.812	1.59	12.0	18.0	106.5
NPICP	0.815	1.40	11.8	17.6	108.6

TABLE II: Results of ICP and three variants of NPICP given perfect detection on the IC-BIN dataset. Pen/obj is the average penetration of the objects selected by the BOP AR calculation procedure. T_{err} is the average pose error compared with the ground truth pose, which sums the translational error (in mm) and rotational error (in degrees).

results of CosyPose [13] as the initial guess.

The variant of CosyPose which has the best performance additionally applies an ICP refinement, and is currently the best result on the IC-BIN dataset in the BOP Challenge. We run all the three variants of NPICP as a post-processing method for CosyPose (without ICP), and compare their results with the ICP variant of CosyPose.

The results of this experiment are shown in Table III. CP is CosyPose for short. Given the results of CosyPose, all the NPICP variants increase the average recall and decrease the penetration between objects, even though the margin of performance increase is not as large as with perfect detection. We believe that this is caused by the following reasons: 1) Some objects are not detected, thus the interactions between that object and the objects surrounding it could not be used in the joint optimization. 2) The confidence score of the object detection module are not reliable. Some bad detections could take the place of good detections if they have higher scores, and could also influence its surrounding objects. Given imperfect detection, NPICP still performs the best among all the three NPICP variants.

An example of how the poses of objects are updated through iterations of NPICP is shown in Fig. 7.

C. Analysis

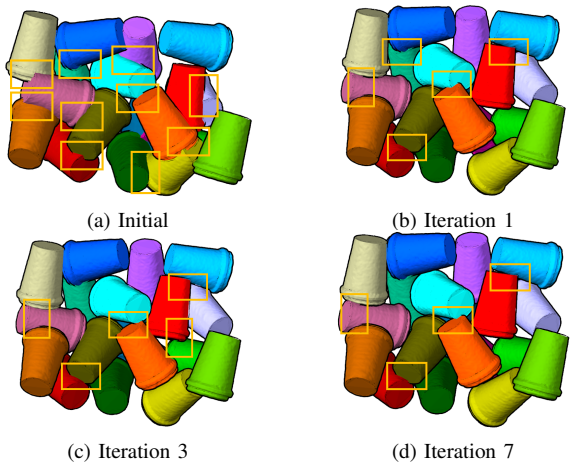
The advantages of NPICP are the best with perfect detection, and given imperfect detection, although the improvements beyond ICP degrade, it still outperforms the best result on the IC-BIN dataset in the BOP Challenge.

The run time of NPICP are shown in Table II and Table III, which only counts the optimization time and does not include the time used for processing the geometries. The run time depends roughly linearly on the following factors: 1) The number of initial estimation N. 2) The number of kept objects m. 3) The iteration number N_{max}^{joint}, N_{max}^{single}. 4) The number of point in $PC_{1:m}^{s}$.

Although the run time of the problem is tens of seconds,

Method	AR	Pen/obj (mm)	Time/img (s)
CP+ICP	0.647	4.24	11.4*
CP+NPICP Ind	0.667	2.83	40.5
CP+NPICP Joint	0.672	1.33	73.9
CP+NPICP	0.674	1.28	99.2

TABLE III: The results of the ICP variant of CosyPose and using three variants of NPICP as post-processing methods on CosyPose. *The average run time of CP+ICP is 11.358 s as listed on BOP Challenge's website. It was run on a machine with 20-core Intel Xeon 6164 @ 3.2 GHz CPU and Nvidia V100 GPU. As a reference, the average run time of CP without ICP is 0.678 s on the same machine.



(a) Initial (b) Iteration 1 (c) Iteration 3 (d) Iteration 7

Fig. 7: Visualization of iterations of NPICP. Deep penetrations are quickly resolved in the first few iterations and later iterations fine-tune to reach a balance between scene point cloud fitting and penetration elimination. Penetrations are marked with boxes. [Best viewed in color.]

the computational performance of the current implementation could be significantly improved. We coded in Python for the purpose of rapid prototyping, and the run time could be reduced if we rewrite the code in a compiled language such as C++. Also, although the run time is not ideal for real time application, as far as we know, methods which could deal with highly occluded scenes that have a similar number of objects, and could provide similar accuracy are all far from real time.

D. Parameter variations

To compare the influence of false positive removal, we tested different choices for d_1-d_3 values in Algorithm 1. When thresholds are large ($d_1 = 1/2\ d^*, d_2 = 3/4\ d^*, d_3 = 1/2\ d^*$), some deeply penetrated objects are not discarded by Algorithm 1, which decrease the AR to 0.657 and increases the penetration to 4.96 mm. When thresholds are small ($d_1 = 1/6\ d^*, d_2 = 1/4\ d^*, d_3 = 1/6\ d^*$), some good detections could be discarded, which decrease the AR to 0.667 but slightly decrease the penetration to 1.09 mm.

V. Conclusion and Future Work

We proposed NPICP, a method for including non-penetration constraints into pose estimation. Our method is particularly advantageous in highly cluttered scenarios, where there are many interactions between objects. NPICP outperforms the best result in the BOP Challenge for single-view multi-object 6D pose estimation on the cluttered IC-BIN dataset. Moreover, we suspect that non-penetration information will become progressively more important object detection accuracy increases, since with perfect detection NPICP improves the BOP AR score by 15% and reduces penetration by 68% beyond standard ICP. Future extensions might use information of the environment (such as the table in the IC-BIN dataset) to constrain the pose of the objects to further improve the estimation accuracy. Also, the proposed approach can accept other objective functions, and could work with other iterative pose estimation techniques such as Coherent Point Drift (CPD) [16].

REFERENCES

[1] Benchmark for 6d object pose estimation. [Online]. Available: https://bop.felk.cvut.cz/home/

[2] T. Birdal and S. Ilic, "Point pair features based object detection and pose estimation revisited," in *2015 International Conference on 3D Vision*. IEEE, 2015, pp. 527–535.

[3] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3364–3372.

[4] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3583–3592.

[5] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing mvtec itodd-a dataset for 3d object recognition in industry," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2200–2208.

[6] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee, 2010, pp. 998–1005.

[7] K. Hauser, "Semi-infinite programming for trajectory optimization with nonconvex obstacles," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 565–580.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] R. Hettich and K. O. Kortanek, "Semi-infinite programming: theory, methods, and applications," *SIAM review*, vol. 35, no. 3, pp. 380–429, 1993.

[10] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 606–619.

[11] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, *et al.*, "Bop: Benchmark for 6d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[12] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *European Conference on Computer Vision*. Springer, 2020, pp. 577–594.

[13] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *European Conference on Computer Vision*. Springer, 2020, pp. 574–591.

[14] C. Mitash, A. Boularias, and K. Bekris, "Physics-based scene-level reasoning for object pose estimation in clutter," *The International Journal of Robotics Research*, p. 0278364919846551, 2019.

[15] C. Mitash, B. Wen, K. Bekris, and A. Boularias, "Scene-level pose estimation for multiple instances of densely packed objects," in *Conference on Robot Learning*. PMLR, 2020, pp. 1133–1145.

[16] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[17] R. Reemtsen and S. Görner, "Numerical methods for semi-infinite programming: a survey," in *Semi-infinite programming*. Springer, 1998, pp. 195–275.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[19] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[20] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3343–3352.

[21] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[22] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.