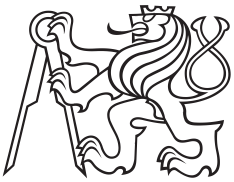
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Geometrical consistency for object pose estimation from images

Martin Malenický

Supervisor: Ing. Vladimír Petrík, Ph.D.
Study program: Cybernetics and Robotics
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Malenický Martin** Personal ID number: **507632**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Geometrical consistency for object pose estimation from images

Bachelor’s thesis title in Czech:

Geometrická konzistence při odhadu polohy a orientace objektu z obrázků

Guidelines:

1. Analyze the geometrical consistency of the scene composed of objects for which pose was estimated with state-of-the-art pose estimation methods, e.g. [1, 2].
2. Use differential collision detection algorithm [3] for fixing geometrical inconsistency.
3. Analyze one of the BOP datasets [4] (e.g., YCB-V [5]) and extract information about the surrounding environment that can be used to model additional geometry (e.g., use depth measurements to model the plane representing the desk).
4. Compare the accuracy of the predictions (SE3 distance between the poses) with and without enforcing the geometrical consistency on the selected YCB-V dataset.

Bibliography / sources:

[1] Labbé, Y., Manuelli, L., Mousavian, A., Tyree, S., Birchfield, S., Tremblay, J., Carpentier, J., Aubry, M., Fox, D. and Sivic, J., 2022. Megapose: 6d pose estimation of novel objects via render & compare. arXiv preprint arXiv:2212.06870.
[2] Labbé, Y., Carpentier, J., Aubry, M. and Sivic, J., 2020. Cosypose: Consistent multi-view multi-object 6d pose estimation. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16 (pp. 574-591). Springer International Publishing.
[3] Montaut, L., Le Lidec, Q., Bambade, A., Petrik, V., Sivic, J. and Carpentier, J., 2023, May. Differentiable collision detection: a randomized smoothing approach. In 2023 IEEE International Conference on Robotics and Automation (ICRA) (pp. 3240-3246). IEEE.
[4] Sundermeyer, M., Hodaň, T., Labbe, Y., Wang, G., Brachmann, E., Drost, B., Rother, C. and Matas, J., 2023. Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2784-2793).
[5] Xiang, Y., Schmidt, T., Narayanan, V. and Fox, D., 2017. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199

Name and workplace of bachelor's thesis supervisor:

Ing. Vladimír Petrík, Ph.D. Intelligent Machine Perception CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **26.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Vladimír Petrík, Ph.D. Supervisor's signature	prof. Dr. Ing. Jan Kybic Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my gratitude to my thesis supervisor, Ing. Vladimír Petrík, Ph.D., for his excellent guidance, provision of expert advice, insightful feedback, and great patience. I would like to thank Dr. Méderic Fourmy for his brilliant theoretical advice and practical ideas without which this thesis could not have been made. I would also like to thank my colleagues at CIIRC RMP, especially Mgr. Martin Cífka, for their help during the development of this thesis. Finally, I would like to thank my family, friends, and girlfriend for the support they have provided during my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

I further declare that the artificial intelligence used for text translation (DeepL), grammar checking (Writefull) and code snippet generation (GithubCopilot) was used in accordance with the Guidelines for the use of Artificial Intelligence at CTU.

In Prague, 20. May 2024

_____

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací.

Dále prohlašuji, že umělá inteligence použitá pro překlad textu (DeepL), kontrolu gramatiky (Writefull) a generování úryvků kódu (GithubCopilot) jsem použil v souladu s Rámcovými pravidly používání umělé inteligence na ČVUT.

V Praze, 20. května 2024

_____

Abstract

Object pose estimation from an image is an important task in robotics required for automatic interaction with an environment. However, current methods for pose estimation do not account for the physical constraints of the real world, resulting in physically infeasible estimates. This work aims to improve the object pose estimation from images by introducing physical consistencies into the scene. The desired consistencies include the application of gravity and resolution of collisions between objects in the scene. The poses of the objects are either estimated by pose estimators or are known a priori from the model of the environment, *e.g.*, the table on which the objects are placed. We formulate the physical consistency as an optimization problem for which we derive analytical gradients. Two synthetic rendered datasets and three real-world datasets from the BOP challenge were used to analyze the effect of enforcing physical consistency. Our approach improves the BOP metrics on average by 27% on our synthetic datasets and by 13% on the three BOP datasets. This shows that physical consistency has a significant effect on the BOP metrics for the object pose estimation. To demonstrate the effect of physical consistency in robotic applications, we perform a pick-and-place task on a Panda robot. Our approach results in a more stable operation, increasing the grasping success rate from 20% to 80% on challenging grasps.

Keywords: object pose estimation, physical consistency, geometrical consistency, robotic manipulation, BOP challenge

Supervisor: Ing. Vladimír Petrík, Ph.D. Intelligent Machine Perception CIIRC

Abstrakt

Odhad polohy a orientace objektu z obrázku je důležitou úlohou v robotice, která je nutná pro automatizovanou interakci s prostředím. Stávající metody odhadu polohy a orientace však nezohledňují fyzikální omezení reálného světa, což vede k fyzikálně nerealistickým odhadům. Cílem této práce je zlepšit odhad polohy a orientace objektu ze snímků zavedením fyzikálních konzistencí do scény. Požadované konzistence zahrnují uplatnění gravitace a odstranění kolizí mezi objekty ve scéně. Polohy a orientace objektů jsou buď odhadovány pomocí *pose estimatoru*, nebo jsou a priori známy z modelu robota a prostředí, např. stůl na kterém jsou objekty umístěné. Fyzikální konzistenci formulujeme jako optimalizační problém, pro který odvodíme analytické gradienty. K analýze vlivu fyzikální konzistence byly použity dva syntetické datasety a tři reálné datasety z BOP Challenge. Naše metoda zlepšuje BOP metriky v průměru o 27% na našich syntetických datasetech a o 13% na třech použitých BOP datasetech. To ukazuje, že fyzikální konzistence má významný vliv na BOP metriky pro odhad polohy a orientace objektu. Abychom demonstrovali vliv fyzikální konzistence v robotických aplikacích, provedli jsme úlohu uchop a polož s robotem Panda. Náš přístup vede ke stabilnějšímu fungování a zvyšuje úspěšnost náročných úchopů z 20% na 80%.

Klíčová slova: odhad polohy a orientace objektu, fyzikální konzistence, geometrická konzistence, robotická manipulace, BOP Challenge

Překlad názvu: Geometrická konzistence při odhadu polohy a orientace objektu z obrázků

Contents

Chapter 1

Introduction

1.1 Motivation

Accurately estimating the position and orientation, *i.e.*, 6D pose, of objects is a crucial area in computer vision and robotics. Reliable pose estimation is essential for various applications, such as autonomous vehicle control, robotic manipulation, motion tracking, augmented reality, and many others. In this thesis, we focus on object pose estimation for robotic manipulation.

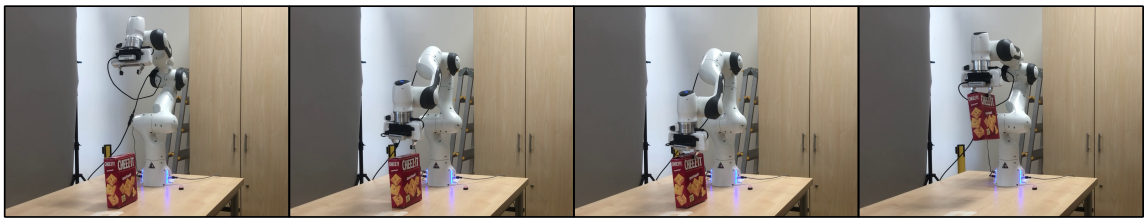
Current state-of-the-art solutions, such as the render-and-compare approach used in CosyPose [3] and MegaPose [2], or approaches that use both RGB and depth images, such as FoundationPose [4], do not explicitly enforce the geometric and physical constraints between the objects for which the pose estimation is performed. This results in geometric and

Figure 1.1: The left part of the image shows the input image taken from YCB-V dataset [1]. The right part of the image shows the object in estimated poses computed by MegaPose [2]. It is clearly visible that the objects in the estimated poses are colliding with each other and with the table.

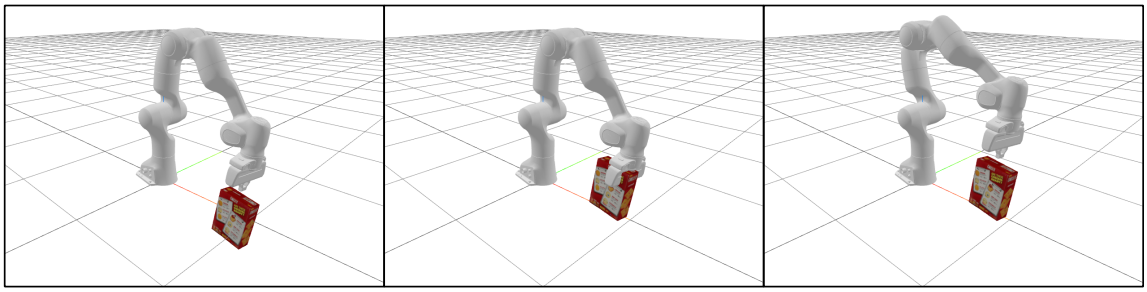
physical inconsistencies, as illustrated in Fig. 1.1. Objects in such estimated poses may float above the floor on which they should properly lie or be in collisions with each other, or with other objects in the scene, for which pose estimation has not been made but are modeled based on the real environment. A typical example of a modeled environment is table-top robotic manipulation. To compute the collision-free trajectory of the robot, a surrounding environment needs to be accurately modeled. In table-top manipulation, pose estimation is often done for objects lying on a table or other modeled surface. Our motivation is to accurately perform this table-top robotic manipulation based on the monocular image captured by the camera mounted on the robot. To achieve the desired accuracy, we propose to enforce physical and geometrical consistency by optimization. This pick-and-place setup is shown in Fig. 1.2.

1.2 Goals

The goal of this thesis is to use an iterative algorithm to enforce physical consistency in a scene where the poses of objects have been predicted by a state-of-the-art pose estimator



(a) : Real robotic experiment: picking an YCB-V object with physically consistent estimation.



(b) : Simulated scenes showing a possible effect of physical inconsistencies.

Figure 1.2: Picking of an YCB-V object based on a monocular camera. The first row illustrates the task we want to achieve, *i.e.*, picking an YCB-V object based on the monocular camera mounted on the robot arm. The robot first captures the image and then approaches the object from the top based on the pose estimated from the image. If the predicted pose is accurate, the picking is successful, as shown by the consecutive frames in the first row. However, if we apply MegaPose to the captured image, the obtained pose might put an object into collision with the desk, as shown in the first frame of the bottom row. The real execution would then end up in collision with the object, as illustrated in the middle frame of the bottom row. If our proposed approach is applied, the collision is resolved, and the robot approaches the object correctly, as illustrated in the last frame of the bottom row. This physically consistent pose was used to perform a real robot experiment shown in the top row.

MegaPose [2]. The scene contains estimated objects, static environmental objects, and a gravitational field. The geometric inconsistencies between pairs of objects are resolved together with the "floating" of objects in the air. At the same time, we will not move the objects too far away from their original estimates to keep the perceptual appearance of the image.

We distinguish several variants of our algorithm, depending on the amount of a priori information:

- if no information about the environment surface and gravity direction is known, we enforce only geometrical consistency (*i.e.* resolve collisions) between objects;
- if the environment is modeled, we consider also collisions between the static environment and the objects; and
- if the gravity direction is known we also enforce a gravitational field that resolves "floating" objects.

We compare the standard BOP metrics [5] of the initial poses predicted by MegaPose [2] and our geometrically and physically consistent poses. The comparison is performed on our synthetically generated dataset and on standard BOP datasets.

The contributions of the thesis are:

- We enforce the physical consistency of the scene whose poses were estimated from RGB images.
- We render a synthetic dataset for benchmarking.
- We benchmark the improvement in BOP metrics gained by enforcing physical consistency on both our synthetic dataset and on real BOP datasets. Our approach increases the BOP average recall by 13 % on average for 3 real datasets and by 27 % on average for two synthetic datasets.
- We made the code open source [6].

Chapter 2

Related works

Pose estimation of rigid objects from images is a topic whose first solutions began to be explored decades ago [7–10]. However, new methods based on different principles continue to emerge. Earlier methods were based on 2D-3D correspondences computed by manually designed algorithms [11–14], more recently, convolutional neural networks are used to identify correspondences [1, 15–17]. Current state-of-the-art methods often use the render-and-compare approach [2, 3, 18, 19]. This approach renders meshes of objects in different poses; these renderings are then compared with the image for which we want to perform the estimation, and the pose in the rendering closest to the original image is chosen to be the result. Our method is built on top of the output of the pose estimators and further refines the poses to get physically consistent results. We use MegaPose [2] in our experiments.

For the purpose of benchmarking 6D pose estimation methods, there are many metrics that test different aspects of the estimation. Hodan et al. [5] propose a standardized methodology to compare pose estimation methods that has been used in the BOP Challenge for several years [5, 20, 21]. This methodology tests pose estimators on a variety of metrics and datasets to provide a broad comparison. We tested our method on the metrics and several datasets used in the BOP Challenge.

Collision avoidance in object pose estimation

The collision is avoided in some methods that are based on point-pair-features [22]. These methods compute the hash map of features from the meshes in an offline phase. Online estimation is formulated as the matching of the depth map to the computed features.

Deng et al. [23] resolve duplicate estimates by rendering depth maps of objects in estimated poses, then filtering out objects whose depth maps overlap, thus avoiding the creation of a potential collision of an object with itself. Wang et al. [24] create a point cloud from the depth map used for estimation, then the RANSAC [25] algorithm is used to find the pose of the table. They use the table pose to down-sample the point cloud by removing points that are below the table plane. This method implicitly avoids estimating

objects in poses that would collide with the table. Fu el al. [26] solves the same problem as Wang et al. but explicitly. The pose of the table is also found using RANSAC. They then estimate the pose of the objects and delete those estimates that are likely to collide with the table based on the normal to the table.

However, these methods do not aim to resolve collisions directly; they only attempt to avoid them, and if they do occur, they do not seek further solutions. On the other hand, our method solves exactly this problem, *i.e.*, resolves collisions if they arise from the initial estimation.

Collision resolution in human hand and body pose estimation

Collisions are also widely studied in the field of human pose estimation and hand pose estimation. Rong et al. [27] estimate not only the pose of the hand, but also its shape, which is controlled by ten parameters. As a result, they are able to accurately resolve collisions without changing the pose of the hand. Smith et al. [28] also estimate the pose of the hand while considering collisions. However, the shape of the hand is not given by the parameterization, but is created by a deformable model. This approach allows the collisions to be solved by bending the fingers and compressing the skin.

Works on human pose estimation address collisions that can occur either between different body parts of a single person (self-collision) or between a person and the environment. Similarly to the estimation of hand pose, the shape of the body can also change. Some solutions use soft constraints and introduce a collision-penalizing term into their loss functions [29–32], thus creating a so-called collision-aware loss function. Other methods directly prohibit collisions either by differential methods [33], or introduce a collision potential between body parts [34].

These works directly address the collision problem, but they all use objects with parameterized shapes or elastic models. Our work focuses on the pose estimation of rigid objects with 6 degrees of freedom for the purpose of the object pose estimation in robotics.

Rigid object collision resolution

Some works address collisions between rigid objects. Wada et al. [35] use differential collision detection. They do not directly use meshes of objects to resolve collisions; they sample the meshes into points, which they then voxelize and compute loss functions based on voxel intersection. They then minimize the loss using gradient descent. Lee et al. [36] suggest using the internal expanding algorithm to compute the distance between two convex sets. In order to use it efficiently, they define the geometry of objects using support functions, from which they express differentiable contact features. They then solve a gradient optimization problem based on these derivatives. Landgraf et al. [37] trained a generative model to predict segmentations and poses of multiple convex objects from a single RGBD image.

They resolve collisions by fitting superquadratic shapes to a point cloud generated from the mesh and minimizing a loss function penalizing collisions of these superquadratic shapes.

These methods resolve collisions for uncertain object poses. Each method takes a different approach to representing objects and collisions. The previously mentioned methods use voxelization, superquadratic fitting, and support functions. Our method works directly with meshes and the derivative of their signed distance. Gravity was included only in the [37] by training the model on the dataset in which a simulated physics was applied. These methods also consider different use cases than we do.

Chapter 3

Enforcing physical consistency

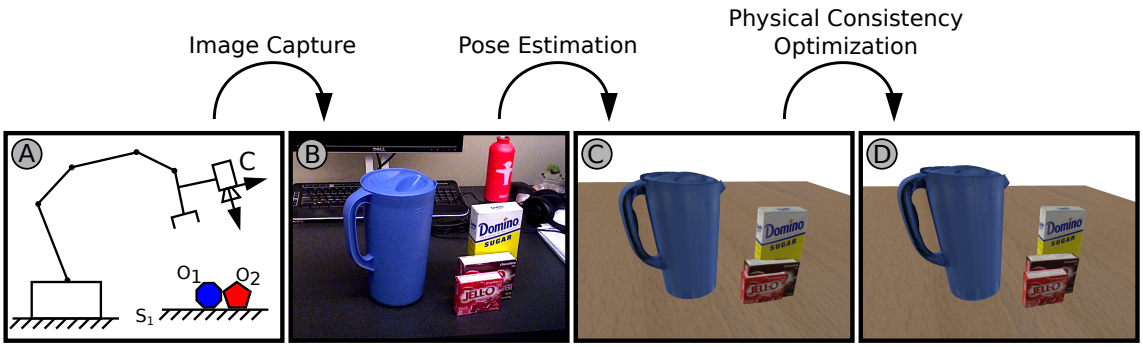


Figure 3.1: The goal of our method is to refine the poses of objects by enforcing physical consistency in the scene. A robot with an external camera, or a camera attached to a gripper such as the one shown in A, captures an images of the scene, as shown in B. From this image, we obtain an initial estimate of the object poses using a pose estimator. From these poses and the poses of other known static objects, we create a virtual representation of the scene, as in Figure C. There, we see that the three paper boxes are in collision with each other and the blue pitcher is in collision with the table. We resolve these collisions and other physical inconsistencies such as objects floating in the air using gradient optimization. Figure D then shows a virtual scene with the object poses already optimized. These physically consistent poses are then the output of our method.

The goal of this thesis is to enforce the physical consistency in a scene in which the poses of the objects are initially estimated by pose estimator, in our case MegaPose [2]. In case of no additional information about the scene, our goal is to find poses that minimize the distances from their initial estimates and resolve collisions between objects. If additional information about static objects is available for a given scene, collisions between objects and static objects as well as gravity can be added to our optimization-based method. We target robotics applications for which the static objects are often modeled, for example, for collision free path planning. We illustrate the overall pipeline of the method in Fig. 3.1.

3.1 Problem formulation and notation.

To achieve physical consistency, we formulate an optimization problem that is solved using gradient descent (GD). In this section, we introduce the notation that is used in the rest of the chapter to compute gradients for GD analytically. Several inputs are considered for our method: the poses of movable objects, the poses of static objects, and spatial representations of the objects.

Poses of movable objects. The proposed method takes as input the initial poses of the objects; these are estimated using an external pose estimator that has predicted the poses from an image captured by the camera. The poses of the objects are expressed in the reference frame of the camera, denoted by the subscript C. The objects are then denoted by a subscript O, i, where O stands for an object and i denotes the i-th object out of N objects in the scene. The transformation from the camera frame to the i-th object frame, estimated by the pose estimator, is then denoted by $\widetilde{T}_{C,Oi}$, where the tilde over T indicates that it is the initial (measured) pose. We will often use the translation and rotation parts of this transformation separately. These are successively denoted by $\widetilde{\boldsymbol{t}}_{C,Oi}$ and $\widetilde{R}_{C,Oi}$. The conversion between the components of the transform and the whole transform can be done by writing the transform in homogeneous coordinates, as follows:

$$\widetilde{T}_{C,Oi} = \begin{bmatrix} \widetilde{R}_{C,Oi} & \widetilde{\boldsymbol{t}}_{C,Oi} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.1}$$

Poses of static objects. The next input to the method, the poses of static objects, are denoted in a similar way: static objects are denoted by the subscript S, i, where i denotes the i-th static object of M known static objects. Transformation from camera frame to i-th static object frame is denoted as $\widetilde{T}_{C,Si}$ and can again be expressed in homogeneous coordinates as:

$$\widetilde{T}_{C,Si} = \begin{bmatrix} \widetilde{R}_{C,Si} & \widetilde{\boldsymbol{t}}_{C,Si} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.2}$$

Spatial representation of objects. Another required inputs to the method are the shapes of objects and static objects. We represent these shapes using meshes. Most algorithms that work with collisions, including the one that we used called Diffcol [38], require the meshes to be convex. This is mainly because of the high computational complexity of working with non-convex meshes. Therefore, at the beginning of the method, all meshes are converted to their convex representation; this is described in more detail in Sec. 3.4.

Optimization. During the optimization process, the poses and thus the transformation of the objects are gradually changed at each step until the last optimization step. We denote the number of steps by the symbol K. We denote the transformation from the camera frame to the i-th object frame at the j-th step by $T^j_{C,Oi}$. This can be expressed as:

$$T^j_{C,Oi} = \begin{bmatrix} R^j_{C,Oi} & \boldsymbol{t}^j_{C,Oi} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.3}$$

Note that this means that $T^0_{C,Oi} = \widetilde{T}_{C,Oi}$. And for static objects, $\widetilde{T}_{C,Si} = T^0_{C,Si} = T^1_{C,Si} = \ldots = T^K_{C,Si}$, since their poses do not change during optimization. The result of the method are the optimized poses of the objects in the last iteration of GD, *i.e.*, $T^K_{C,O1}$, $T^K_{C,O2}$, $\ldots$, $T^K_{C,ON}$. We present more details on optimization using this notation in Sec. 3.2.

3.2 Optimization loop for physical consistency

The gradient used in the GD method consists of three sub-gradients: the collision gradient $\boldsymbol{\nabla} C$, the gravity gradient $\boldsymbol{\nabla} G$ and the perception gradient $\boldsymbol{\nabla} P$. These three gradients represent the influence of collision resolution, the influence of the gravity term, and the influence of the initial estimation. We can change the ratio between the gradients using weight of the collision gradient ζ_C and weight of the gravity gradient ζ_G, the perception gradient has a fixed weight of one. The weights ζ_C and ζ_G are considered to be hyperparameters that are tuned to achieve stable convergence and accurate physical consistency. All presented gradients are represented by 6D vectors, where the first three components describe translation and the last three rotation. Detailed descriptions of the cost functions, the computation of the gradients, and the motivation for their use are presented in Sec. 3.3, 3.5, and 3.6. The computation of the entire gradient for the i-th object in the j-th iteration goes as follows:

$$\boldsymbol{\nabla} T^j_{C,Oi} = \zeta_C \boldsymbol{\nabla} C^j_i + \zeta_G \boldsymbol{\nabla} G^j_i + \boldsymbol{\nabla} P^j_i \in \mathbb{R}^6. \tag{3.4}$$

Due to possible numerical inaccuracies, especially in the calculation of the collision gradient, the entire gradient is clipped. It is then multiplied by the learning rate α. This yields a vector representing an iterative step in 6D space:

$$\mathrm{d}\boldsymbol{x}^j_i = -\alpha \ \mathrm{clip}(\boldsymbol{\nabla} T^j_{C,Oi}) \in \mathbb{R}^6. \tag{3.5}$$

We do the gradient clipping separately for the translation and rotation parts to allow for different clipping thresholds, ρ_t (translation threshold) and ρ_R (rotation threshold). This is useful, for example, because translation and rotation are in different units. We do not clip the vector by components, but by norm division to preserve the direction of the original vector. For an arbitrary 6D vector $\boldsymbol{v}$ consisting of two three-dimensional vectors $\boldsymbol{t}$ and $\boldsymbol{\omega}$ such that $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{t} & \boldsymbol{\omega} \end{bmatrix}^T$ the clipping looks like this:

$$\begin{aligned} \mathrm{clip}(\boldsymbol{v}) &= \begin{bmatrix} \boldsymbol{t}' \\ \boldsymbol{\omega}' \end{bmatrix}, \\ \boldsymbol{t}' &= \begin{cases} \rho_t \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} & \text{if } \|\boldsymbol{t}\| > \rho_t, \\ \boldsymbol{t} & \text{otherwise.} \end{cases} \\ \boldsymbol{\omega}' &= \begin{cases} \rho_R \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} & \text{if } \|\boldsymbol{\omega}\| > \rho_R, \\ \boldsymbol{\omega} & \text{otherwise.} \end{cases} \end{aligned} \tag{3.6}$$

The pose update of the i-th object in the j-th optimization iteration is computed by multiplying the pose of the object with the update transformation based on iterative step (3.5) in the following way:

$$T_{C,Oi}^{j+1} = T_{C,Oi}^{j} \exp\left(\mathrm{d}\boldsymbol{x}_i^j\right) \in SE(3). \tag{3.7}$$

The function exp [39] converts the 6D gradient to the form (3.3). All of the above calculations are performed in each of the K iterations of GD for each of the N objects.

For simplicity, we will use the notation without the superscript j for the remainder of this chapter, just note that all following calculations are performed in each iteration of the optimization loop.

3.3 Perception gradient

Since the initial poses are the best initial estimate we have, we do not want to deviate from it more than necessary to ensure physical consistency. For example, if the collision gradient between two objects is large in a certain iteration, this could cause the objects to suddenly move farther apart than needed to resolve the collision. The perception gradient will gradually bring the objects closer together until they touch again, at which point the collision gradient will begin to work against it. In this way, our aim is to find an equilibrium.

3.3.1 Perception cost and gradient

At each iteration step, we compute the perception gradient based on the perception cost. The cost function underlying the perception gradient is given by the Mahalanobis norm [40] of error. The error is computed as SE(3) difference [39] between the pose in the current iteration and the estimated pose. The norm is scaled by a covariance matrix of the estimator to allow for more divergence in the dimensions, where estimation is more difficult for image-based pose estimators. Description of calculation of the covariance matrix is in Sec. 3.3.2. We will denote the error for the i-th object as $\boldsymbol{e}_i$. Its calculation is as follows:

$$\Delta\boldsymbol{t}_i = \boldsymbol{t}_{C,Oi} - \widetilde{\boldsymbol{t}}_{C,Oi}\,, \tag{3.8}$$

$$R_{\widetilde{Oi},Oi} = \widetilde{R}_{C,Oi}^{-1} R_{C,Oi} = \widetilde{R}_{C,Oi}^{T} R_{C,Oi}\,, \tag{3.9}$$

$$\Delta\boldsymbol{\omega}_i = \log\left(R_{\widetilde{Oi},Oi}\right), \tag{3.10}$$

$$\boldsymbol{e}_i = \begin{bmatrix} \Delta\boldsymbol{t}_i \\ \Delta\boldsymbol{\omega}_i \end{bmatrix}. \tag{3.11}$$

The translation vectors and rotation matrices used in Eq. (3.8) and (3.9) defines the object pose in camera frame as defined in (3.1) and (3.3). In Eq. (3.10), an SO(3) logarithmic mapping [41] of the rotation matrix to the rotation vector is used. The perception cost is

then half of the squared Mahalanobis distance of the calculated error (3.11):

$$P_i = \frac{1}{2}\|\boldsymbol{e}_i\|^2_{\Sigma_{Ci}} = \frac{1}{2}\boldsymbol{e}_i{}^T\Sigma_{Ci}^{-1}\boldsymbol{e}_i = \frac{1}{2}\boldsymbol{e}_i{}^T H_i \boldsymbol{e}_i\,, \tag{3.12}$$

where Σ_{Ci} is the covariance matrix we mentioned earlier and the H_i is a so-called precision matrix [42], which can be calculated as inverse of the covariance matrix:

$$H_i = \Sigma_{Ci}^{-1}\,. \tag{3.13}$$

The precision matrix multiplies the error by a large number in the direction for which we assume a good initial estimate of the pose and a small number in the direction in which the estimate is typically flawed, *e.g.*, depth direction.

Since the cost is a quadratic form, its derivative will depend on the size of the error, so if the error is large, the derivative will also be large and vice versa. Now we calculate the perception gradient, that is, the derivative of the perception cost with respect to the pose of the object. We first decompose this as the derivative of the cost with respect to the error and the derivative of the error with respect to the pose, as follows:

$$\boldsymbol{\nabla} P_i = \frac{\partial P_i}{\partial T_{C,Oi}} = \frac{\partial P_i}{\partial \boldsymbol{e}_i}\frac{\partial \boldsymbol{e}_i}{\partial T_{C,Oi}}\,. \tag{3.14}$$

We call the derivative of the pose error $\frac{\partial \boldsymbol{e}_i}{\partial T_{C,Oi}}$ the Jacobian and denote it by J_i. The Jacobian tells us how much the individual components of the error vector $\boldsymbol{e}_i$ change as the pose of the object changes. How the Jacobian is computed is described in Sec. 3.3.3, for now we assume that the Jacobian is known. After substituting the cost (3.12) and the Jacobian into formula (3.14), the derivative can be further adjusted:

$$\boldsymbol{\nabla} P_i = \frac{\partial(\frac{1}{2}\boldsymbol{e}_i{}^T H_i \boldsymbol{e}_i)}{\partial \boldsymbol{e}_i} J_i = \frac{1}{2}\boldsymbol{e}_i{}^T(H_i + H_i^T)J_i\,. \tag{3.15}$$

The precision matrix is symmetric and therefore $H_i + H_i^T = 2H_i$, which allows us to further simplify the expression:

$$\boldsymbol{\nabla} P_i = \boldsymbol{e}_i{}^T H_i J_i\,. \tag{3.16}$$

Equation (3.16) is the final form of the perception gradient used in the optimization loop (3.4).

3.3.2 Covariance between pose estimation error and the estimation axis

Although we consider the initial estimation to be approximately correct, typically pose estimators have more difficulty estimating depth than estimating rotation and position in the image plane. Thus, we need to allow more freedom to move objects in axis where pose estimators make larger errors. To achieve that, we model the covariance in a coordinate frame shown in Fig. 3.2 in which the camera z-axis points toward the center of the estimated object. We assumed diagonal covariance matrix for translation and rotation, *i.e.* we assume

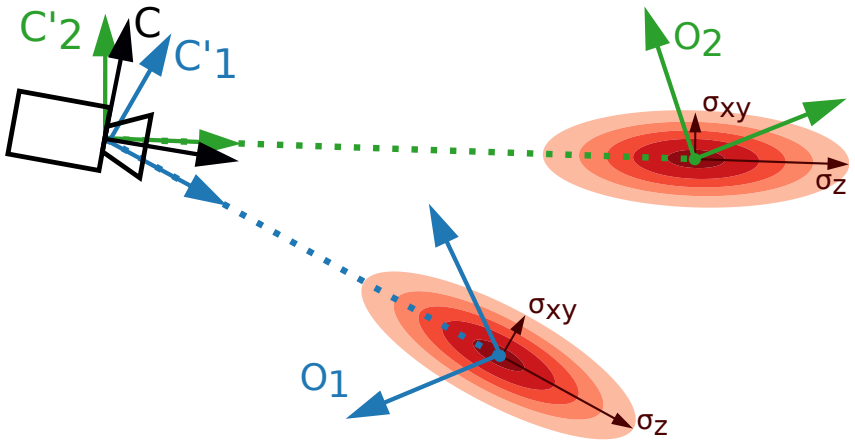


Figure 3.2: 2D illustration of translation covariance error ellipses for two objects O_1 and O_2 in rotated camera frames C_1' and C_2'. The standard deviation is larger along the axis pointing to the object than along the axis perpendicular to it. This reflects the difficulty of estimating depth using pose estimators. We therefore allow the objects more freedom of movement in this axis.

uncorrelated error among the axes of translation and axes of rotations. As in [43], we estimated the standard deviation σ_z for the z-axis error, together with the standard deviation σ_{xy} for the x and y axis, and finally the single standard deviation σ_θ for all axis of rotations. Together, these standard deviations form the covariance matrices for translation and rotation.

$$\Sigma^t_{C'i} = \begin{bmatrix} \sigma_{xy}^2 & 0 & 0 \\ 0 & \sigma_{xy}^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}, \tag{3.17}$$

$$\Sigma^R_{Oi} = \begin{bmatrix} \sigma_\theta^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{3.18}$$

The covariance matrix for translation is the same for all objects as long as it is expressed in the C_i' frame, *i.e.*, the frame of the camera rotated towards the i-th object, as shown in Fig. 3.2. Similarly, the covariance matrix for rotation is the same for all objects as long as it is expressed in the frame of the objects in question.

Since all poses and therefore the perception gradient are expressed in the unrotated camera frame, we have to transform the diagonal covariance matrices into it. To do this, we use the initial poses of the objects $\widetilde{T}_{C,Oi}$ for which the covariance matrix is measured. We first find the axis and angle of rotation between the z-axis of the unrotated camera, that is, $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, and the z-axis of the rotated camera pointing to the i-th object, that

is, the normalized translation $\widetilde{\boldsymbol{t}}_{C,Oi}$. Therefore, the rotation axis and the relative angle are:

$$\hat{\boldsymbol{\omega}}_i = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \frac{\widetilde{\boldsymbol{t}}_{C,Oi}}{\left\| \widetilde{\boldsymbol{t}}_{C,Oi} \right\|}, \tag{3.19}$$

$$\theta_i = \arccos\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot \frac{\widetilde{\boldsymbol{t}}_{C,Oi}}{\left\| \widetilde{\boldsymbol{t}}_{C,Oi} \right\|} \right). \tag{3.20}$$

The Eq. (3.19) and (3.20) can be combined into an exponential representation [41], as follows:

$$\boldsymbol{\omega_i} = \theta_i \hat{\boldsymbol{\omega}}_i \,. \tag{3.21}$$

Using the SO(3) exponential mapping [41], we convert the Eq. (3.21) into a rotation matrix:

$$R_{C,C'i} = \exp(\boldsymbol{\omega}_i) \,. \tag{3.22}$$

Using the rotation matrix (3.22) we can convert the translation part of the covariance matrix (3.17) to the unrotated camera frame:

$$\Sigma^t_{Ci} = R_{C,C'i} \Sigma^t_{C'i} R^T_{C,C'i} \,, \tag{3.23}$$

in the same way, we can transform the rotation part of the covariance matrix (3.18) into the camera frame by using the rotation of the object relative to the camera $\widetilde{R}_{C,Oi}$:

$$\Sigma^R_{Ci} = \widetilde{R}_{C,Oi} \Sigma^R_{Oi} \widetilde{R}^T_{C,Oi} \,. \tag{3.24}$$

We can now combine the translation covariance (3.23) and the rotation covariance (3.24) into a single covariance matrix of dimension 6×6:

$$\Sigma_{Ci} = \begin{bmatrix} \Sigma^t_{Ci} & 0_3 \\ 0_3 & \Sigma^R_{Ci} \end{bmatrix}, \tag{3.25}$$

where 0_3 denotes 3×3 matrix full of zeros.

3.3.3 Perception Jacobian

In this subsection, we describe the computation of the Jacobian J_i used in Eq. (3.16). As mentioned earlier, the Jacobian in our case is the derivative of the perception error of the object pose (3.11) with respect to the pose. Since we can split the error into the translation part (3.8) and the rotation part (3.10), we can split its derivative and hence the Jacobian:

$$J_i = \frac{\partial \boldsymbol{e}_i}{\partial T_{C,Oi}} = \frac{\partial \begin{bmatrix} \Delta \boldsymbol{t}_i & \Delta \boldsymbol{\omega}_i \end{bmatrix}^T}{\partial T_{C,Oi}} = \begin{bmatrix} \frac{\partial \Delta \boldsymbol{t}_i}{\partial T_{C,Oi}} & \frac{\partial \Delta \boldsymbol{\omega}_i}{\partial T_{C,Oi}} \end{bmatrix}^T, \tag{3.26}$$

we will calculate these components separately.

Modification of the translation part of the Jacobian. We will modify the derivative of the translation part using Eq. (3.8):

$$\frac{\partial \Delta \boldsymbol{t}_i}{\partial T_{C,Oi}} = \frac{\partial \boldsymbol{t}_{C,Oi} - \widetilde{\boldsymbol{t}}_{C,Oi}}{\partial T_{C,Oi}} = \frac{\partial \boldsymbol{t}_{C,Oi}}{\partial T_{C,Oi}} - \frac{\partial \widetilde{\boldsymbol{t}}_{C,Oi}}{\partial T_{C,Oi}} = \frac{\partial \boldsymbol{t}_{C,Oi}}{\partial T_{C,Oi}}. \tag{3.27}$$

We can make this modification because $\widetilde{\boldsymbol{t}}_{C,Oi}$ is a constant. To compute the derivative of the position with respect to the pose, we use [39], in this paper, Solà et al. deduced the derivative:

$$\frac{\partial T\boldsymbol{p}}{\partial T} = \frac{\partial R\boldsymbol{p} + \boldsymbol{t}}{\partial T} = \begin{bmatrix} R & -R[\boldsymbol{p}]_\times \end{bmatrix}, \tag{3.28}$$

where T is an arbitrary frame transformation consisting of a rotation part R and a translation part $\boldsymbol{t}$, $\boldsymbol{p}$ is an arbitrary 3D vector and $[\boldsymbol{p}]_\times$ creates a skew-symmetric matrix from vector $\boldsymbol{p}$. For $T := T_{C,Oi}$ and $\boldsymbol{p} := \mathbf{0}$, where $\mathbf{0}$ is a 3D zero vector, we can rewrite the equation (3.28) as:

$$\frac{\partial T_{C,Oi}\mathbf{0}}{\partial T_{C,Oi}} = \frac{\partial R_{C,Oi}\mathbf{0} + \boldsymbol{t}_{C,Oi}}{\partial T_{C,Oi}} = \frac{\partial \boldsymbol{t}_{C,Oi}}{\partial T_{C,Oi}} = \begin{bmatrix} R_{C,Oi} & -R_{C,Oi}[\mathbf{0}]_\times \end{bmatrix} = \begin{bmatrix} R_{C,Oi} & 0_3 \end{bmatrix}. \tag{3.29}$$

Note that the third term of Eq. (3.29) is the desired derivative from Eq. (3.27), so the resulting translation part of the Jacobian is:

$$\frac{\partial \boldsymbol{t}_{C,Oi}}{\partial T_{C,Oi}} = \begin{bmatrix} R_{C,Oi} & 0_3 \end{bmatrix}. \tag{3.30}$$

Modification of the rotation part of the Jacobian. The rotational part of the Jacobian can be rewritten using equations (3.9) and (3.10):

$$\frac{\partial \Delta \boldsymbol{\omega}_i}{\partial T_{C,Oi}} = \frac{\partial \Delta \boldsymbol{\omega}_i}{\partial R_{\widetilde{O}i,Oi}} \frac{\partial R_{\widetilde{O}i,Oi}}{\partial R_{C,Oi}} \frac{\partial R_{C,Oi}}{\partial T_{C,Oi}} = \frac{\partial \log\left(R_{\widetilde{O}i,Oi}\right)}{\partial R_{\widetilde{O}i,Oi}} \frac{\partial \widetilde{R}^T_{C,Oi} R_{C,Oi}}{\partial R_{C,Oi}} \frac{\partial R_{C,Oi}}{\partial T_{C,Oi}}. \tag{3.31}$$

The calculation of the first term of the product, *i.e.*, the derivative of the logarithmic mapping with respect to its argument, is analytically implemented in the Pinocchio library [44] and we consider this derivative to be known. Solà et al. [39] solved the other two terms of the product as follows:

$$\frac{\partial \widetilde{R}^T_{C,Oi} R_{C,Oi}}{\partial R_{C,Oi}} = I_3\,, \tag{3.32}$$

$$\frac{\partial R_{C,Oi}}{\partial T_{C,Oi}} = \begin{bmatrix} 0_3 & I_3 \end{bmatrix}, \tag{3.33}$$

where I_3 is a 3×3 identity matrix. Using the identities mentioned above, we modify the equation (3.31) to form:

$$\frac{\partial \Delta \boldsymbol{\omega}_i}{\partial T_{C,Oi}} = \frac{\partial \log\left(R_{\widetilde{O}i,Oi}\right)}{\partial R_{\widetilde{O}i,Oi}} I_3 \begin{bmatrix} 0_3 & I_3 \end{bmatrix} = \begin{bmatrix} 0_3 & \frac{\partial \log\left(R_{\widetilde{O}i,Oi}\right)}{\partial R_{\widetilde{O}i,Oi}} \end{bmatrix}. \tag{3.34}$$

Combining the equations (3.30) and (3.34) yields the formula for the Jacobian used in (3.16):

$$J_i = \begin{bmatrix} R_{C,Oi} & 0_3 \\ 0_3 & \frac{\partial \log\left(R_{\widetilde{Oi,Oi}}\right)}{\partial R_{\widetilde{Oi,Oi}}} \end{bmatrix}. \tag{3.35}$$

3.4 Convex meshes and convex decomposition

Before we explain how collisions between meshes are computed, we need to describe how to convert the original concave meshes to their convex representation. For this we use two kinds of representation, convex hull and convex decomposition.

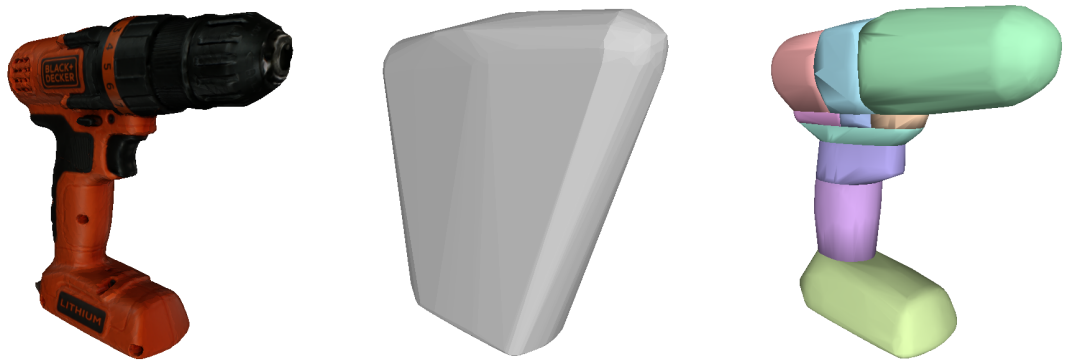


Figure 3.3: Visualization of object representation. From left to right: the original concave mesh from the YCB-V dataset, its convex hull, and a convex decomposition consisting of ten differently colored parts.

A convex hull is the smallest convex mesh that contains the entire original mesh. However, it is only a very rough approximation of the original shape, which we will use for approximate and fast computations. A major problem with the convex hull is the neglect of mesh details such as holes and dips in the mesh, as seen in Fig. 3.3. When calculating the distance, or derivative of collision, between meshes where some holes have been approximated in this way, a large error would occur. For the conversion of the original mesh to a convex hull, we use the Quickhull algorithm [45] implemented within HPP-FCL [46].

We use a convex decomposition of the meshes to obtain greater resemblance to the original object. The convex decomposition algorithm produces a set of smaller convex meshes whose union no longer needs to be convex, contains the entire original mesh, but is "smaller" than the convex hull. This can be seen in Fig. 3.3 on the right. For this task, we used a state-of-the-art method for convex decomposition CoACD [47].

We denote the convex hull of the i-th object out of N objects by $\mathcal{O}_i$, and the convex hull of the i-th static object out of M static objects by $\mathcal{S}_i$. The pose of these meshes is identical

to that of their associated objects, *i.e.*, the mesh $\mathcal{O}_i$ is located in $T_{C,Oi}$ and the mesh $\mathcal{S}_i$ is located in $T_{C,Si}$. For the decomposition, we will use a convention in which the k-th decomposed part of the i-th object will be denoted by $\mathcal{O}_{ik}$ and the number of decomposed parts of the i-th object is N_i. Similarly, the k-th decomposed part of the i-th static object will be denoted by $\mathcal{S}_{ik}$ and the number of decomposed parts of the i-th static object is M_i. Individual decomposed parts have a coordinate system at the same location as the original mesh, but the mesh vertices themselves are shifted to match the original undecomposed mesh. The pose notation of the decomposed part is the same as that of the convex hull, *i.e.*, the part $\mathcal{O}_{ik}$ is in the pose $T_{C,Oi}$ and $\mathcal{S}_{ik}$ is in $T_{C,Si}$.

3.5 Collision gradient

First, we introduce the notion of witness-points, in the same way as in Diffcol [38]. Witness-points $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ are a pair of vertices of arbitrary meshes $\mathcal{A}_1$ and $\mathcal{A}_2$ that are closest to each other among all pairs of vertices, *i.e.*:

$$\boldsymbol{x}_1^*, \boldsymbol{x}_2^* = \underset{\boldsymbol{x}_1 \in \mathcal{A}_1, \boldsymbol{x}_2 \in \mathcal{A}_2}{\operatorname{argmin}} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 . \tag{3.36}$$

Next, we define the signed distance of two arbitrary meshes $\mathcal{A}_1$ and $\mathcal{A}_2$. The signed distance of the meshes is the distance of their witness-points, the sign of the distance is positive if the meshes are not in collision and negative if they are. We can write this as follows.

$$d(\mathcal{A}_1, \mathcal{A}_2) = \begin{cases} \|\boldsymbol{x}_1^* - \boldsymbol{x}_2^*\|_2 & \text{if } \mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset , \\ -\|\boldsymbol{x}_1^* - \boldsymbol{x}_2^*\|_2 & \text{otherwise.} \end{cases} \tag{3.37}$$

Collision cost for convex shapes. The collision cost function consists of two cost functions, the first calculates collisions between pairs of objects and the second between pairs of objects and static objects. The total cost for the i-th object is:

$$C_i = C_{Oi} + C_{Si} . \tag{3.38}$$

The collision cost of movable objects is computed as

$$C_{Oi} = \sum_{\substack{n=1 \\ n \neq i}}^{N} \frac{1}{\psi} \exp(-\psi \min(0, d(\mathcal{O}_i, \mathcal{O}_n))) \tag{3.39}$$

and for static objects:

$$C_{Si} = \sum_{m=1}^{M} \frac{1}{\psi} \exp(-\psi \min(0, d(\mathcal{O}_i, \mathcal{S}_m))). \tag{3.40}$$

Each of the two cost functions for the i-th object is the sum of exponentials of the negative signed distances between the i-th object and all other objects (or static objects). The negative sign ensures that the cost function is positive if the objects are in collision and

negative if they are out of collision, thus penalizing collisions. The distance is only included in the cost function if the objects are in a collision, *i.e.*, the signed distance is negative. If we added the distance in both cases, the objects would endlessly drift apart when minimizing the cost function. The exponential ensures that the cost increases rapidly as the collision distance increases, preventing large penetrations. A hyperparameter ψ allows us to control growth rate of the cost.

Collision cost for convex decompositions. In the previously proposed way, we considered collision between convex hulls only. To achieve a finer collision resolution, we refine the cost using a convex decomposition. This way, our final cost accounts for all collisions between decomposed parts of all mesh pairs:

$$C_{Oi} = \sum_{\substack{n=1 \\ n \neq i}}^{N} \frac{1}{\psi} \exp\left(-\psi \sum_{l=1}^{N_i} \sum_{k=1}^{N_n} \min(0, d(\mathcal{O}_{il}, \mathcal{O}_{nk})) \right). \tag{3.41}$$

Again, we write a similar cost function for static meshes:

$$C_{Si} = \sum_{m=1}^{M} \frac{1}{\psi} \exp\left(-\psi \sum_{l=1}^{N_i} \sum_{k=1}^{M_m} \min(0, d(\mathcal{O}_{il}, \mathcal{S}_{mk})) \right). \tag{3.42}$$

Differentiation through collisions. To partially differentiate the cost function with respect to the object pose, we first need to calculate the partial derivative of the signed distance with respect to the object pose. For this task, we use Diffcol [38]. Meshes consist of vertices connected in triangles, and together they can form detailed curved shapes, but locally around each vertex the mesh is made up of triangle faces that are flat, and this makes it hard to get a good estimate of the derivative of the signed distance. Diffcol uses a randomized smoothing approach to obtain more informative gradients. Randomized smoothing is a technique that involves convolution of a function with a probability distribution in order to approximate the function using a smoothed version of it. This is achieved by adding noise to the function. The advantage of this technique is the detailed capture of the local geometry and curvature of the mesh around the witness-point, which can be thought of as a function of pose and shape of the mesh.

Knowing the derivatives of the signed distance, we can calculate the derivatives of the cost functions (3.41) and (3.42) with respect to pose. First we show the derivative of (3.41):

$$\boldsymbol{\nabla} C_{Oi} = \sum_{\substack{n=1 \\ n \neq i}}^{N} \exp\left(-\psi \sum_{l=1}^{N_i} \sum_{k=1}^{N_n} \min(0, \delta_{ilnk}) \right) \sum_{l=1}^{N_i} \sum_{k=1}^{N_n} \begin{cases} -\frac{\partial \delta_{ilnk}}{\partial T_{C,Oi}} & \text{if } \delta_{ilnk} < 0, \\ \mathbf{0} \in \mathbb{R}^6 & \text{otherwise,} \end{cases} \tag{3.43}$$

where $\delta_{ilnk} = d(\mathcal{O}_{il}, \mathcal{O}_{nk})$. Next is the derivative of (3.42):

$$\boldsymbol{\nabla} C_{Si} = \sum_{\substack{m=1 \\ n \neq i}}^{M} \exp\left(-\psi \sum_{l=1}^{N_i} \sum_{k=1}^{M_m} \min(0, \delta_{ilmk}) \right) \sum_{l=1}^{N_i} \sum_{k=1}^{M_m} \begin{cases} -\frac{\partial \delta_{ilmk}}{\partial T_{C,Oi}} & \text{if } \delta_{ilmk} < 0, \\ \mathbf{0} \in \mathbb{R}^6 & \text{otherwise,} \end{cases} \tag{3.44}$$

where $\delta_{ilmk} = d(\mathcal{O}_{il}, \mathcal{S}_{mk})$.

Although calculating the distances between meshes is not time consuming, calculating the distances between all decomposed parts pairs of all meshes in the scene would result in tens of thousands of calculations in a single iteration[1]. Therefore, we try to minimize the number of these operations, and hence the computation does not proceed exactly as in formulas (3.43) and (3.44). For two objects $\mathcal{O}_i$ and $\mathcal{O}_n$, we first calculate the distance of the convex hulls $d(\mathcal{O}_i, \mathcal{O}_n)$, if the distance is positive, it means that they are not in collision and thus their convex decompositions are not in collision either, the collision gradient is zero in this case, and there is no need to continue calculating the distances of decomposed parts. If the convex hulls are in a collision, the distances $d(\mathcal{O}_{il}, \mathcal{O}_n)$ and $d(\mathcal{O}_i, \mathcal{O}_{nk})$ are pre-calculated, *i.e.* the distances of the decomposed part of the first object against the convex hull of the second object and vice versa. Then the calculations of the distances between the decomposed parts, as done in (3.43) and (3.44), are performed only for those parts that are in collision with the convex hull because if the decomposed part is not in collision with the convex hull, it is not in collision with the decomposed parts inside the hull either.

In the end, we sum the gradients (3.43) and (3.44) to get the total collision gradient containing collisions between objects and static objects.

$$\boldsymbol{\nabla} C_i = \boldsymbol{\nabla} C_{Oi} + \boldsymbol{\nabla} C_{Si} \in \mathbb{R}^6. \tag{3.45}$$

This gradient (3.45) is then used in gradient descent optimization loop, *i.e.* Eq. (3.4).

3.6 Gravity gradient

The last part of the total gradient is gravity. We simulate gravity by minimizing the potential energy of objects relative to a selected static object. We denote the static object (usually a table) to which the potential energy (and hence gravity) is related as the first static object S_1 in the scene. We define the gravitational cost function as the potential energy of the object with respect to S_1, for the i-th object this will be:

$$G_{i,S1} = m_i \mathrm{g} z_i \,, \tag{3.46}$$

where m_i is the mass of the object, but we usually do not have access to it and therefore we consider it to be one, g is the gravitational acceleration and z_i is the z translation component of $T_{S1,Oi}$ transformation, *i.e.* the height of the object above S_1. For simplicity, we assumed that gravity acts against the z-axis of the local coordinate frame of the first static object. This could be always achieved by rotating the frame of reference of the first static object in an appropriate direction.

The derivative of this cost function for unit mass, with respect to the pose of the i-th object in the S_1 frame yields the gradient:

$$\frac{\partial G_{i,S1}}{\partial T_{S1,O1}} = \boldsymbol{\nabla} G_{i,S1} = \begin{bmatrix} 0 & 0 & \mathrm{g} & 0 & 0 & 0 \end{bmatrix}^T. \tag{3.47}$$

[1] For 6 objects in a scene with 64 decomposed parts each, the number of distance calculations would be 61 440 in a single iteration. With 1000 iterations, the total number of calculations would then be more than 61 million

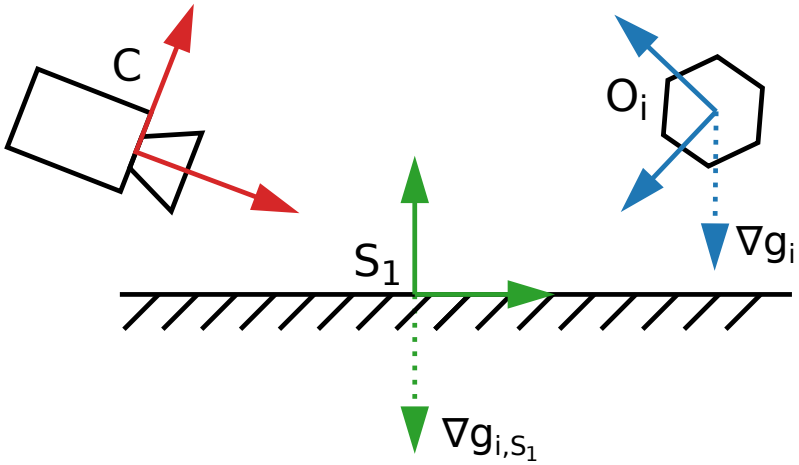


Figure 3.4: Illustration of the gravity direction. The gravity direction is defined with respect to the first static object S_1, acting against z-axis of its coordinate frame. This coordinate frame is usually defined on the ground or on the table desk. The gravity direction is independent of camera frame (red) and it is also independent of object rotation (blue frame), *i.e.*, gravity always acts towards the ground (green frame). The direction of the gravity is shown in dashed line.

Now we will transform the gradient into the object frame using known transformations. Since the cost function does not depend on the rotation of the object, we will transform only the translation part of the gradient, $\nabla g_{i,S1} = \begin{bmatrix}0 & 0 & \mathrm{g}\end{bmatrix}^T$, as shown in Fig. 3.4. The translational part of the gradient expresed in the i-th object coordinate frame is:

$$\nabla g_i = R_{C,Oi}^T \widetilde{R}_{C,S1} \nabla g_{i,S1} \,. \tag{3.48}$$

The 6D gradient is obtained by appending zeros representing rotation to the translation part of the gravity gradient:

$$\nabla G_i = \begin{bmatrix} \nabla g_i \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^6 \,. \tag{3.49}$$

We want to use gravity only for collision-free objects. If an object is in collision with another object or with a static object, the gravity gradient for that object is equal to the zero vector. This helps converge better after the objects have landed and prevents objects from falling through each other or changing the order they were originally in before they fell. The final version of the gravity gradient used in (3.4) is:

$$\nabla G_i = \begin{cases} \begin{bmatrix} \nabla g_i \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^6 & \text{if } i\text{-th object is not in collision,} \\ \mathbf{0} \in \mathbb{R}^6 & \text{otherwise.} \end{cases} \tag{3.50}$$

Chapter 4

Experiments

In this chapter, we describe how we tested the functionality and performance of our method. For initial coarse testing, we created two synthetic datasets. We then performed more detailed tests on three real datasets from the BOP challenge [21] widely used for benchmarking of pose estimation from images. Since our method contains 10 hyperparameters, we performed an extensive ablation study to fine-tune them. We compared the set of parameters that performed best on average on all datasets with a state-of-the-art method for pose estimation called MegaPose [2].

4.1 Experimental setup

In order to consistently compare our method with the MegaPose method, we always followed the same procedure. First, before running MegaPose, it is necessary to detect objects of interest in the RGB image and obtain their bounding boxes and segmentation masks. The bounding boxes are then input to MegaPose along with the RGB image. To eliminate the influence of the external detector in order to compare only pose estimation methods, we used ground-truth detection. We can obtain these for synthetic datasets during rendering, and manually annotated ground-truth detections and segmentations are available for real BOP datasets. However, these ground-truth detections have a drawback, objects are detected in the image even if only a small part of them is visible. Therefore, we remove those detections for which the ratio of the visible part of the object to the whole object is less than 50%; an example of this is shown in Fig. 4.1. With these filtered detections and the original image, we obtain the pose estimation using MegaPose.

Next, we need to select ten hyperparameters that can be tuned for our method. We will evaluate our method for three different settings, using only collisions between objects, using collisions between objects and between static objects, and using both types of collisions as well as gravity. For each of these settings, we will select one set of ten parameters which we will then use to evaluate all datasets, both synthetic and real. To find good combinations of parameters, we performed an ablation study, described in Sec. 4.6. The parameters we

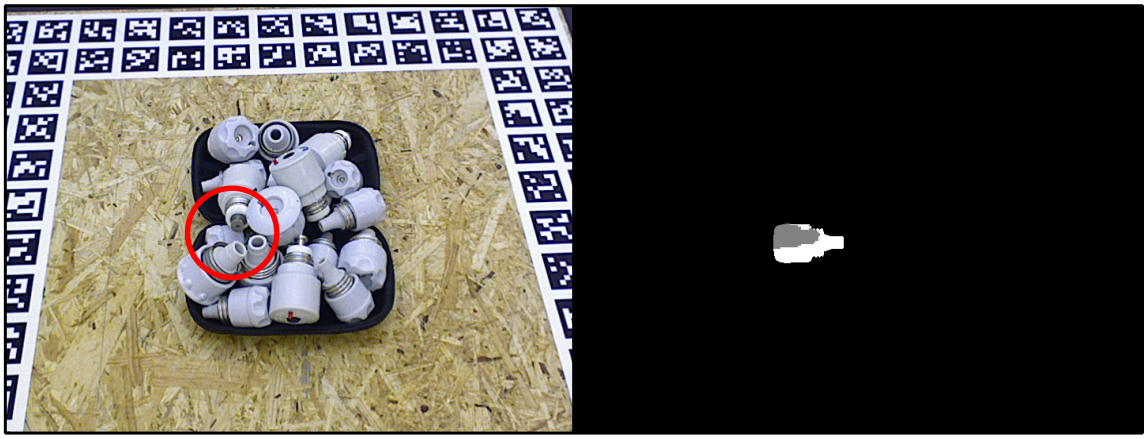


Figure 4.1: This figure shows an example of low visibility detection in a real dataset. In the left part we can see the original image, one of the badly visible objects is marked in red circle. In the right part are the segmentation masks of the marked object. The modal segmentation mask is colored gray and the amodal mask is colored white. The ratio of the number of pixels of the modal to the amodal mask in this case is 0.424.

decided to use are shown in Tab. 4.1.

	Coll	Coll+Table	Coll+Table+Gravity
K - number of iterations	1000	1000	1000
α - learning rate (LR)	0.0001	0.0001	0.0001
σ_{xy} - standard deviation in x,y axis	0.05	0.05	0.05
σ_z - standard deviation in z axis	0.49	0.49	0.49
σ_θ - standard deviation in rotation	0.26	0.26	0.26
ζ_C - weight of collision gradient	5	2	1
ζ_G - weight of gravity gradient	0	0	1
ψ - scale of distance exponent	10	0	0
ρ_t - translation clipping thresholds	100	100	100
ρ_R - rotation clipping thresholds	100	100	100

Table 4.1: Combination of parameters used for evaluation. Three sets of ten parameters are selected for three different variants of our methods: (i) *Coll* - only collision between movable objects, (ii) *Coll+Table* - collision between movable objects and between movable objects and static table, and (iii) *Coll+Table+Gravity* - all collisions with gravity.

To achieve stable convergence, a three-step optimization procedure is designed for situations involving gravity, *i.e.*, if $\zeta_G > 0$. This procedure consists of a sequential execution of three optimization loops. In the first phase, *i.e.*, the first optimization loop, only collisions are solved. We do this by setting $\zeta_G = 0$. This will resolve the large initial collisions that we do not want to resolve when optimizing with gravity; if the objects were in large collisions the gravity application might not get done in time. In the second phase, we include gravity by setting ζ_G to the desired value and start the optimization. This will put the objects in contact with the table and/or with each other. Finally, we start the third phase, in which we again resolve only collisions that may have occurred during the application of gravity.

This will ensure that gravity does not push objects even into a small collision. Note that if gravity is not considered ($\zeta_G = 0$), we use a single optimization loop.

4.2 Metrics

To benchmark our algorithm, we used the same methodology as in the BOP Challenge. We use the same error functions:

- MSSD (Maximum Symmetry-Aware Surface Distance): Consider mesh in two poses, ground-truth and our estimated one. The MSSD is the largest distance between the respective vertices of the two meshes in the given poses. If we have information about the symmetry of the mesh, *i.e.*, we know the poses such that the mesh in these different poses is identical (or sufficiently similar), the MSSD will be the minimum of all calculated MSSDs for symmetric mesh poses.
- MSPD (Maximum Symmetry-Aware Projection Distance): MSPD also calculates the maximum vertex distance considering symmetries as in MSSD, but the distance is not in 3D, it is the distance of the pixels on which the vertices are projected.
- VSD (Visible Surface Discrepancy): By rendering the mesh in ground-truth pose and our estimated pose, we get two distance maps (not depth maps). We filter these distance maps using the real distance map of the scene taken by the camera by keeping only pixels in the rendered maps whose value is smaller, or up to a tolerance larger than the value of the same pixels in the real depth map. This means that we remove pixels that are occluded by another object in the real scene. Then we compare whether the difference in the values of the same pixels of the two filtered distance maps is less than a given threshold. VSD is the ratio of the correctly estimated distances to the number of pixels in the intersection of the filtered distance maps.

A more detailed explanation and the exact calculation procedure can be found in [5, 48]. In addition, we calculate the following error functions:

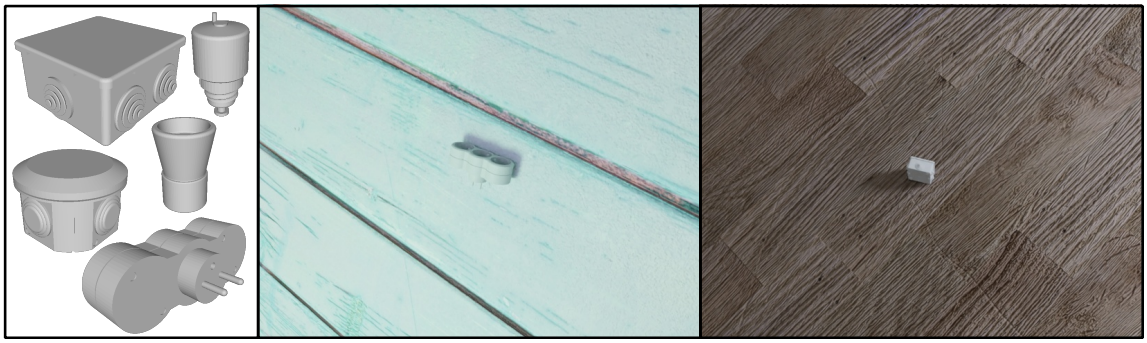
- TE (Translation error): TE is the norm of the difference between the translations of the ground-truth pose and our estimated pose.
- RE (Rotation error): We convert the product of the ground-truth rotation matrix and the inverse of our estimated rotation matrix into an axis-angle representation. Then RE is the angle converted to degrees.

For these five types of errors, we compute the recall for different true-positive thresholds. For each type of error, we obtain a single average recall for all the thresholds used. As in the BOP Challenge, we will consider AVG as the most important metric. The AVG is the average of the average MSSD, MSPD and VSD recalls. The AVG recall is therefore

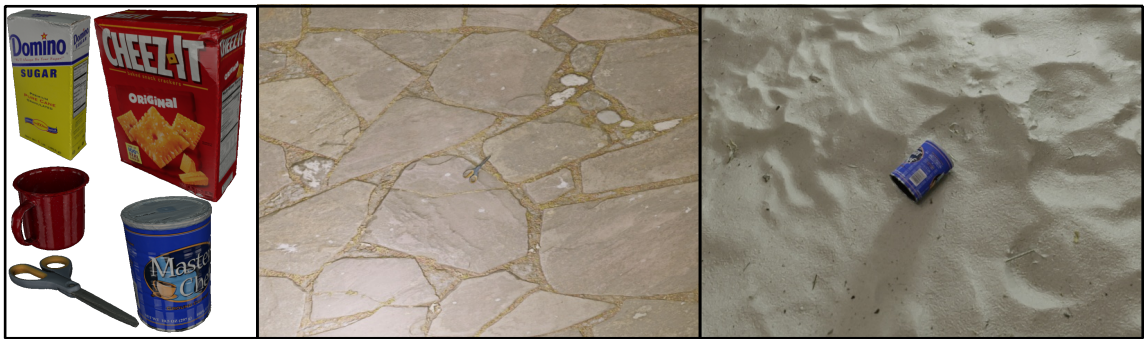
computed as an average over the metrics and over all the thresholds for the given metrics. The tables presented within this chapter will always show the average recall of the error function listed in the column label. We used the BOP Toolkit available as part of the BOP Challenge to calculate the metrics.

4.3 Synthetic toy datasets

For testing purposes, we created two synthetic datasets. The first dataset is based on objects from the YCB dataset [49], and the second contains objects from the T-LESS dataset [50]. Each dataset consists of 500 images with a resolution of 640×480 pixels, each image contains one object lying on the floor. There are five types of objects in each dataset, each object is rendered on 100 images, while the texture of the floor changes every 10 images. The camera is from 0.75 m to 3.5 m away from the object and the angle between the camera and the floor ranges from 20° to 90°. Since the meshes from the T-LESS dataset do not include textures, we used a random shade of gray as a mesh color during rendering. We rendered the datasets using Blenderproc [51]. Since the datasets are synthetic, we have complete information about the layout of each scene, *i.e.* the exact pose, segmentation, and bounding-box of the objects, the pose of the floor, and the camera intrinsics and extrinsics. An example of renders and a preview of all objects used is in Fig. 4.2.



(a) : Synthetic T-LESS dataset examples



(b) : Synthetic YCB-V dataset examples

Figure 4.2: The figures show an example of our rendered datasets. The left images show the objects used to render the datasets. The images on the right show a sample of the renders.

4.3.1 Results

In this section, we present results and findings from the evaluation of synthetic datasets. We first focus on the analysis of the physical inconsistencies predicted by MegaPose and their subsequent correction by our method. We then look at the quantitative evaluation using metrics from chapter 4.2.

For both synthetic datasets, we made initial pose estimates using MegaPose and then refined these estimates using our method. From both our and MegaPose's estimated object poses and ground-truth floor poses, we computed the signed distances between the object and the floor. As we described in the previous section 4.3, objects are always rendered directly on the floor and therefore the signed distance should always be zero. We expect the pose estimates, and hence the absolute value of the signed distance, to deteriorate as the visibility of the object gets worse. In our case, the object is always fully visible, but its size (number of pixels) decreases as the distance of the object from the camera increases.

The dependence of the object-floor signed distance on the object-camera distance for both synthetic datasets is shown in Fig. 4.3. What we can see here is a comparison between MegaPose and the subsequent refinement of the poses by our method using only the collision and perception gradient, ζ_G is therefore zero. We can observe the expected behavior of MegaPose, *i.e.*, the magnitude of the signed distance increases as the distance of the object from the camera increases. We can also notice that the T-LESS dataset has generally larger signed distances than the YCB-V dataset; this is mainly due to two factors, the objects from the T-LESS dataset are smaller than those from the YCB-V dataset and they do not have textures, both of which contribute to worse pose estimation. Furthermore, we can see that all collisions (negative values on the y-axis) have been resolved for both datasets using our method.

However, using only the collision part, it is not possible to ensure complete physical consistency; the objects still float above the floor without change using our method. This is why we also included the gravity part of the gradient, so ζ_G is a positive value. For this case, the results are shown in Fig. 4.4. We see that all object-floor collisions and all of the objects floating above the floor are resolved.

We will now look at the results of the evaluation using the metrics from section 4.2. We evaluated the output of MegaPose and our method using two different sets of parameters. Although we mentioned in Sec. 4.1 that we will perform the evaluation for three different sets of parameters, in this case we will only use the two that count with the floor. Since the two synthetic datasets contain only one object per scene, it would be redundant to only consider collisions between objects. The first parameter set used has the gravity gradient weight fixed to zero, $\zeta_G = 0$, and thus uses only collisions with the table for gradient pose optimization. The second set uses both collision and gravity gradients.

The result of the evaluation is shown in Tab. 4.2. For both datasets, we can notice a large improvement in the average recall when using the gravity gradient. On the other hand, when using collisions only, the improvement is only marginal, the reason is that MegaPose predictions were rarely in collision, the objects were mostly floating above the floor, as we

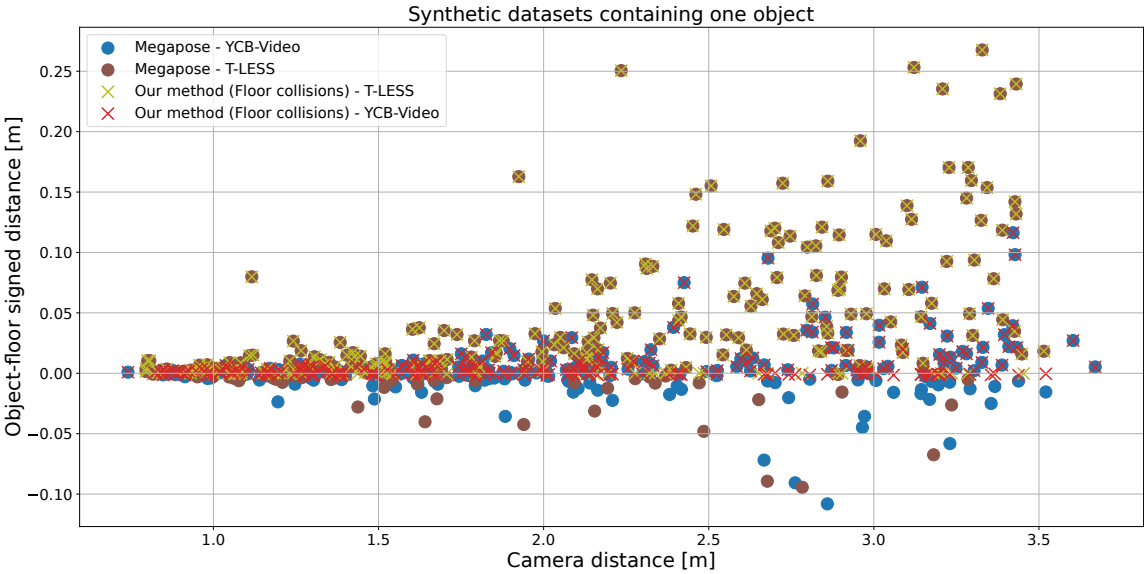


Figure 4.3: The graph shows the dependence of the signed distance of the object from floor on the distance of the camera from the object for both synthetic datasets used in this thesis. In this case, we used MegaPose and our method with collision and perception terms only (*i.e.*, without gravity).

can see in Fig. 4.3 and Fig. 4.4, therefore, the room for improvement was small. We also see little or no change in the average MSPD recall, but this is not surprising since MSPD calculates the projection of the mesh vertices onto the image plane and is not strongly affected by object depth estimation error. Rotation is not affected when applying gravity and is often not needed to resolve collisions, so there is no change in RE recall. Instead, the TE, MSSD and VSD recalls improve significantly, as they reflect the improvement in depth estimation.

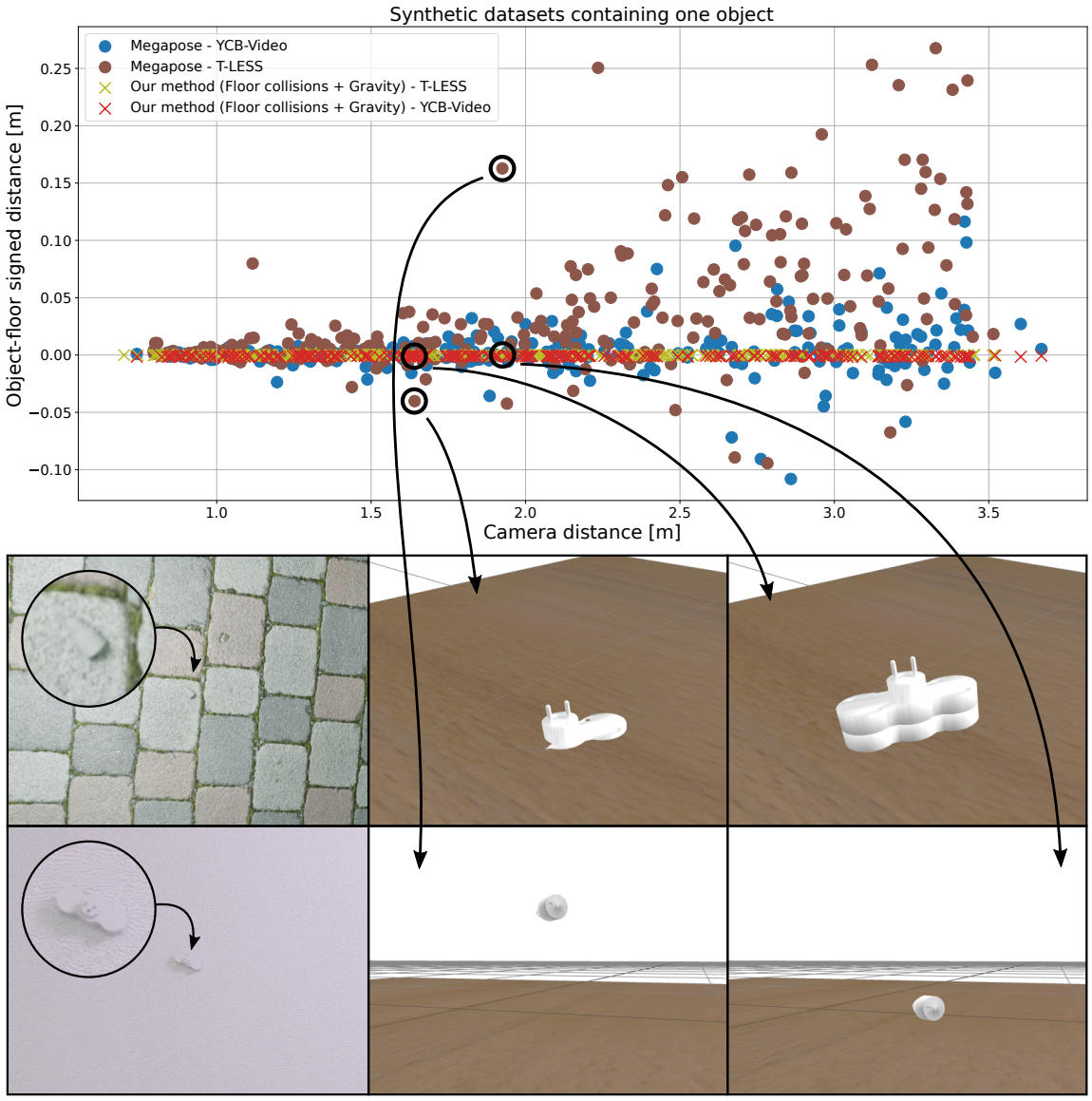


Figure 4.4: The graph shows the dependence of the signed distance of the object from floor on the distance of the object from the camera for both synthetic datasets used in this thesis. We report MegaPose and our method that resolves collision and the gravity. The points marked with black circles in the graph are visualized in the images at the bottom of the figure. The left column shows the original images, the middle the physically inconsistent estimates by MegaPose, and the right the estimates by our method.

	AVG	MSPD	MSSD	VSD	TE	RE
MegaPose	0.6224	0.9892	0.4349	0.4430	0.5030	0.1145
Ours - Coll+Table	0.6387	0.9892	0.4614	0.4656	0.5217	0.1145
Ours - Coll+Table+Gravity	**0.9170**	0.9892	**0.8807**	**0.8811**	**0.9747**	0.1145
Improvement [%]	47.3	0.0	102.5	98.9	93.8	0.0

(a) : Average recalls for synthetic T-LESS dataset.

	AVG	MSPD	MSSD	VSD	TE	RE
MegaPose	0.8928	**0.9981**	0.8532	0.8271	0.7920	0.7909
Ours - Coll+Table	0.8934	0.9970	0.8559	0.8272	0.7947	0.7909
Ours - Coll+Table+Gravity	**0.9515**	0.9973	**0.9502**	**0.9070**	**0.9536**	0.7909
Improvement [%]	6.6	-0.1	11.4	9.7	20.4	0.0

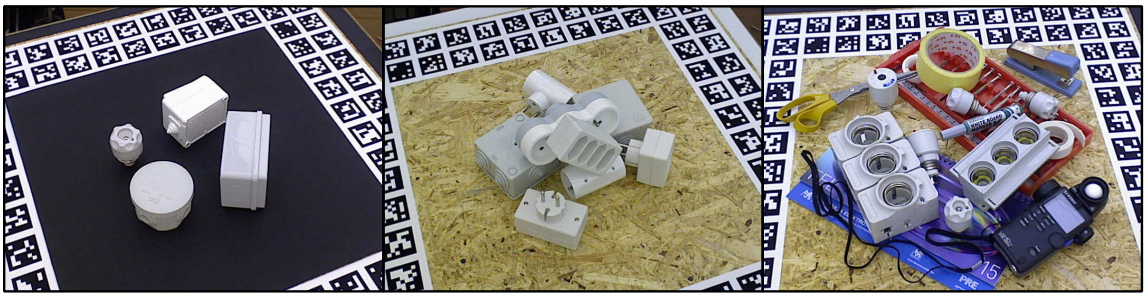
(b) : Average recalls for synthetic YCB-V dataset.

Table 4.2: The tables show the output of the BOP evaluation for the two synthetic datasets. The evaluation was performed for MegaPose and our method. Our method uses only table collisions in one case and both collisions and gravity in the other case. The last row shows the percentage improvement of our method using gravity compared to the MegaPose. If a column has a strict maximum it is in bold.

4.4 BOP Datasets

In addition to the two synthetic datasets, we also used three real datasets that capture tabletop scenes. Each of these datasets has different specific characteristics that can affect both the initial pose estimation from the images and our physical optimization. These properties are described in the next section and shown in Fig. 4.5.

(a) : HOPE-Video dataset examples



(b) : T-LESS dataset examples

(c) : YCB-V dataset examples

Figure 4.5: The images show a sample of the used BOP datasets. The first row shows the HOPE-Video dataset. In the left image of the first row, we can see blur caused by the camera movement while taking the video; in the middle image, we can see that some objects are only partially visible. In the second row, T-LESS dataset is shown and we can see the different layouts of the scenes, these can have objects far apart like in the image on the left, on top of each other, and inside each other like in the middle or laid on top of distractor objects like on the right. The last row shows the YCB-V dataset. The left image shows a scene with poor lighting conditions, and as in the image on the right, there are objects on top of each other. A lot of camera noise can be seen in the middle image.

4.4.1 BOP datasets description

The first dataset is Household Objects for Pose Estimation Video Dataset (HOPE-Video) [52]. It contains 10 video sequences taken by a camera attached to a robot gripper. In total, the video sequences are divided into 2038 images. Because the images are taken by sampling the videos, they tend to be blurry. In the scenes, the objects are never on top of each other and are placed on a flat (table) or slightly deformed (couch) surface. There are 28 high-quality textured meshes available with the dataset. HOPE-Video features accurate manually annotated ground-truth poses of objects.

Another used dataset is T-LESS [50]. It consists of 20 scenes with 50 images each. The images in one scene are taken from different angles, mainly varying in the camera elevation. In the scenes, objects can be directly on the table, on top of each other, inside each other (sockets plugged into each other) or on other objects that are not part of the dataset (distractor objects). There are 30 meshes available for the dataset produced using CAD, the meshes do not have textures, which is what characterizes T-LESS.

The last dataset that we use is the Yale-CMU-Berkeley Video Dataset (YCB-V) [1], which is based on 21 objects selected from the standardized object dataset YCB [49]. BOP Challenge uses a manually selected subset of YCB-V to reduce computational burden. We use the same subset as the BOP Challenge and refer to it as YCB-V. The subset consists of 12 scenes with 75 images each. The images are usually affected by camera noise and/or poor lighting. The objects are placed on a table or on top of each other. As mentioned before, the dataset includes 21 objects and thus 21 meshes. The meshes were created using laser scanning.

4.4.2 Table pose estimation

Since we know that all the datasets used are in tabletop layout, we can use this information to add additional geometric constraints in the form of a table. We use the depth maps, the ground-truth camera matrix, the object segmentation masks, and the object poses available for the datasets to estimate the pose of the table. The ground-truth data are only used to estimate the table pose, which is often available in robotic applications without estimating it, *i.e.*, we assume this is known a priori. In scenes containing distractor objects, such as the right image in Fig. 4.5b, it is not advantageous to estimate the table pose at all; using gravity, the objects would fall on the table plane without collision, but in a real scene they would be in collision, which we cannot simulate without knowing the poses and meshes of the distractor objects. For these scenes, the static object pose was not estimated, and therefore collision checking with static objects and gravity was not used.

To estimate the plane of the table, we use depth maps, from which we filter out the points corresponding to the objects. Ground-truth segmentations are used for the filtering. The point cloud is calculated from the depth map and the table plane is fitted using the RANSAC algorithm [25] implemented in Sklearn [53]. We ensure that the plane corresponding to the table is fitted (instead of the walls, ground, etc.) by measuring the distance from the

plane to the ground-truth object poses. The pose of the plane is further refined so that ground-truth objects lie on top of it. With this approach, we use ground-truth data to obtain the pose of the table. This is the only place where our method uses ground-truth information. Note that this procedure is needed only to be able to evaluate on BOP datasets; in a standard robotics setup, information about the table is often available.

4.4.3 Results

In this section, we will show and review the evaluation results of the three used BOP datasets. To evaluate all three datasets, we first made an initial estimate using MegaPose and then refined this estimate using our method. We compared three different approaches for our refinement: the first approach did not account for the table; thus, neither gravity nor collisions between objects and static objects were used, only collisions between objects were considered; the second approach used the knowledge of the pose of the table, but we did not apply gravity, so we only accounted for collisions (both object-object and object-table); the last approach considered both collisions and gravity.

The results of the evaluations are shown in Table 4.3. We see that the best results were obtained when using gravity, except for the YCB-V dataset, where a slightly better result was achieved by using only collisions with a table. Our method that enforces gravity achieved significant improvement with respect to MegaPose by 16.7% for HOPE-Video, 6.6% for T-LESS and 15.8% for YCB-V. This validates the hypothesis that physical consistency is an important property to consider for accurate pose estimation in robotics. As in our analysis of synthetic datasets 4.2, we observe only slight changes in MSPD and RE recalls, but a significant improvement in TE, MSSD and VSD recalls. This is caused by the small influence of physical consistency on the MSPD and RE metrics and the significant influence of physical consistency on depth refinement captured by the TE, MSSD and VSD recalls. A selection of qualitative results of successful optimization is shown in Fig. 4.6.

Although the proposed refinement significantly improved the results, there are a few failure cases that could arise. For example, false positive and false negative detections can cause errors in our method. Other methods perform pose estimation for each object separately, so correctly estimated objects are not affected by incorrectly estimated ones. However, our method creates a scene from all objects and enforces the physical consistency on it. If an object in a scene is detected as a false positive, the other objects cannot fill the space where this false positive object is located, although the geometric consistency would not be broken in reality. In contrast, it may violate physical consistency if this false positive object was between another object and the floor. The other object would float in air in reality. False negative detections cause the exact opposite, objects in the virtual scene can get, either due to gravity or collisions with other objects, to the location of this false negative object and thus break geometric consistency in reality. The wrongly estimated object pose can affect the other poses in the same manner even in the cases where the detections are correct. A sample of unsuccessful refinements is qualitatively shown in Fig. 4.7.

	AVG	MSPD	MSSD	VSD	TE	RE
MegaPose	0.6236	0.7170	0.5716	0.5823	0.6850	0.5970
Ours - Coll	0.6239	0.7170	0.5719	0.5828	0.6858	0.5970
Ours - Coll+Table	0.6234	0.7166	0.5714	0.5822	0.6882	0.5970
Ours - Coll+Table+Gravity	**0.7275**	**0.7337**	**0.6972**	**0.7516**	**0.8681**	**0.5974**
Improvement [%]	16.7	2.3	22.0	29.1	26.7	0.1

(a) : HOPE-Video

	AVG	MSPD	MSSD	VSD	TE	RE
MegaPose	0.8285	0.9625	0.7700	0.7530	0.8575	0.1500
Ours - Coll	0.8309	**0.9650**	0.7750	0.7527	0.8600	0.1500
Ours - Coll+Table	0.8307	0.9600	0.7800	0.7520	0.8575	0.1500
Ours - Coll+Table+Gravity	**0.8829**	0.9625	**0.8600**	**0.8262**	**0.9300**	0.1500
Improvement [%]	6.6	0.0	11.7	9.7	8.5	0.0

(b) : T-LESS

	AVG	MSPD	MSSD	VSD	TE	RE
MegaPose	0.6769	0.8414	0.6370	0.5524	0.6661	0.5670
Ours - Coll	0.6870	0.8423	0.6525	0.5662	0.6889	0.5670
Ours - Coll+Table	**0.7914**	**0.8473**	**0.8067**	**0.7203**	0.8320	0.5670
Ours - Coll+Table+Gravity	0.7836	0.8423	0.8033	0.7053	**0.8331**	**0.5709**
Improvement [%]	15.8	0.1	26.1	27.7	25.1	0.7

(c) : YCB-V

Table 4.3: The tables show the output of the BOP evaluation for the three BOP datasets. The evaluation was performed for MegaPose and our method. Our method used only collisions between objects in the first case, collisions between objects and collisions between objects and table in the second case and all collisions and gravity in the third case. The last row shows the percentage improvement of our method using gravity compared to the MegaPose. If a column has a strict maximum it is in bold.

Figure 4.6: This figure shows a qualitative comparison of the initial poses estimated by MegaPose (middle column) and their subsequent refinement by our method (right column). The left column shows the images for which the estimation was performed. In the first row, notice the collision of the largest object in the middle with the floor, which was successfully resolved by our method. In the second row, all objects except the sugar box are in collision with the floor for MegaPose estimates. In the third row, the can and banana are in collision with the table. Also, notice the incorrectly estimated orientation of the banana. Although our method resolves the collision, the orientation of the banana is not changed and remains incorrect after our refinement, since a change of the orientation is not needed for the collision to be resolved. In the last row, there is a large collision between the wooden cube and the floor, and also between the cube and the white bottle; all resolved by our method.

Figure 4.7: This figure shows examples of failure cases for our method. Input image is shown on the left, MegaPose predictions in the middle, and our refinement on the right. The first row shows the pizza box (*i.e.* box behind the juice) in collision with the table and incorrectly rotated after the MegaPose prediction. During the optimization this incorrect prediction caused that the juice box is pushed towards the camera in order to resolve the collision. This can be seen in the last column of the first row, where refined juice box is bigger after rendering. In the second row, the wooden cube and the white bottle were predicted in a big collision by MegaPose. During our refinement, these two objects switch order compared to the input image.

4.5 Real robotic experiment

In this section, we describe a real robotic experiment we performed on the Franka Emika Panda robot with the panda-py controller [54]. We performed a pick-and-place experiment, in which the task was to grasp an object and move it to the desired location.

We solved the task as follows: first, we moved the robot from the starting pose to a pose from which the manipulation area is clearly visible. From there we take an image and run the detection algorithm; specifically, we used the detector implemented in the CosyPose framework pre-trained on the YCB-V dataset. We extract a bounding box from the detection and pass it, along with the image, to the MegaPose input to estimate the pose of the object. We then send the robot to a predefined pre-grasp pose that is few centimeters above the object, and from there to a grasp pose in which we grasp the object with the gripper. Next, we send the robot to the starting pose in which it releases the object. Note that the grasp pose was defined approximately 5 mm below the top of the object to make grasping challenging. We then place the same object in an identical location which we mark in advance. We run our method on MegaPose prediction and perform the same grasping sequence as for MegaPose. An example of successful and unsuccessful grasp is shown in Fig. 4.8, and another is shown in Fig. 1.2.

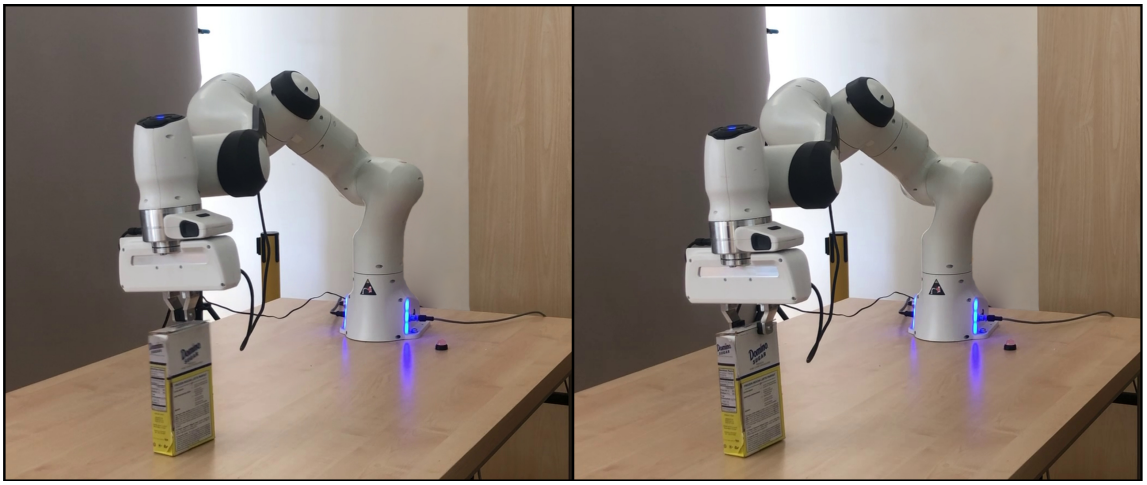


Figure 4.8: The left image shows an unsuccessful attempt to grasp an object whose pose was estimated using MegaPose. Because the MegaPose prediction was not accurate, robot closed the gripper few millimeters above the object. The right image shows a successful grasp for which the pose was estimated by refinement of the former pose using our method.

We performed the experiment five times for three different objects from the YCB-V dataset. We placed the objects in such positions relative to the camera for which it typically is difficult to make an accurate pose estimation. This is, for example, at the edge of the camera's field of view, where part of the object is not visible, directly under the camera, or rotated with its narrower side to the camera. Images of the objects, as they were on the MegaPose input, can be seen in Fig. 4.9. We defined the grasping poses on the edge of the objects, so the pose estimate has to be accurate in order to successfully grasp the

object. The grasp success rates are shown in Tab. 4.4. In it, we can see that our method consistently improved the grasp success rate.

Figure 4.9: Images used as input to MegaPose in real robotic experiment. The first row depicts the poses of Cracker box, the second row Mustard bottle, and the third row Sugar box.

	Cracker box	Mustard bottle	Sugar box
MegaPose	0%	60%	0%
Ours	80%	80%	80%

Table 4.4: The table shows the grasp success rates for the three objects and the two methods for pose estimation used in the robotic experiment. For each object, five grasping attempts were made by each method.

4.6 Ablation study

In this section, we describe the procedure for selecting the different parameters we used for inference. First, in Sec. 4.6.1 we describe how we qualitatively chose various hyperparameters used in our method, and in Sec. 4.6.2 we describe the process of quantitatively choosing the rest of the hyperparameters. Then, in Sec. 4.6.3, we describe how different approaches for computing the signed distance derivatives can affect the methods performance.

4.6.1 Qualitatively estimated hyperparameters

Our method contains 10 adjustable hyperparameters. Most of them do not need to be fine-tuned, as we can estimate a reasonable value directly, or we can measure them. The numerical values are listed in Tab. 4.1.

The parameters whose values we measured include the standard deviations for the perception cost σ_{xy}, σ_z and σ_θ, the procedure for finding them has already been discussed in Sec. 3.3.2. The easily estimable parameters are the clipping thresholds ρ_t and ρ_R. For these, we just need to determine the maximum gradient magnitude we will tolerate. The

next estimated parameter is the number of optimization iterations K and the learning rate α. For the parameter K, in general, a higher number leads to better convergence, but at the cost of the runtime of the algorithm. Conversely, a higher α will reach the local minimum faster but then oscillate around it with a large deviation. Therefore, we qualitatively chose the largest value of α for which the oscillations are perceptually as small as possible and K for which we can reach the minimum with a given α even for cases where objects are farther away from the minimum.

4.6.2 Quantitatively estimated hyperparameters

Next, we need to find the optimal values for the remaining hyperparameters, these are the weights ζ_C and ζ_G and the scale ψ. These values are harder to tune manually, and therefore we performed an ablation study in which we selected the values maximizing the BOP AVG recall. First we find the gradient weights ζ_C and ζ_G for three analyzed cases, in the first case we use collisions between objects and the table and gravity, in the second case we use collisions between objects and the table, and in the third case we use only collisions between objects. We then determine the scale ψ for these three cases. We calculate AVG recall for each parameter combination for all five (3 BOP and 2 synthetic) datasets and average the resulting AVG recalls across datasets. The only exception is the case of resolving only collisions between objects, in which synthetic datasets are not used, as they contain only a single object per scene.

Ablation of the weights ζ_C and ζ_G. To determine the gradient scales, we analyze seven values for the collision gradient weight and seven values for the gravity gradient weight. For the first case with gravity, we evaluate all combinations of pairs of weight values. For the other two cases (*i.e.*, without gravity), we only evaluated the seven selected weights for ζ_C. For now, we set the value of the ψ scale to zero; we will determine its value later. The evaluation of the first two cases when the table is used is shown in Fig. 4.10. The second case where gravity is not used is visible in the last row of the plot, where $\zeta_G = 0$. Using this plot, we can determine the optimal values of the gradient weights. The best average recall for the first case is for $\zeta_C = 1$ and $\zeta_G = 1$ with a recall of 0.8525. For the second case, it is for $\zeta_C = 2$ and $\zeta_G = 0$ with a recall of 0.7555. Next, we performed an evaluation for the third case without using a table, this is shown in Fig. 4.11. Here we can see that the largest average AVG recall is 0.7135 for $\zeta_C = 5$ and $\zeta_G = 0$.

Ablation of the scale ψ. For the scales ζ_C and ζ_G, for which we have already found the optimal values, we now determine the scale ψ. We have selected ten values for ψ. For the first case, using gravity we see the graph in Fig. 4.12 and for the second case using collisions between objects and the table we see the graph in Fig. 4.13. In both cases, we obtain the highest average recall for $\psi = 0$. This gives us two sets of parameters, the first $\zeta_C = 1$, $\zeta_G = 1$, $\psi = 0$, and the second $\zeta_C = 2$, $\zeta_G = 0$, $\psi = 0$. The graph for the third case, considering only collisions between objects, is shown in Fig. 4.14. Here we see that the maximum average recall is achieved for $\psi = 10$. Thus, the last set of parameters is $\zeta_C = 5$, $\zeta_G = 0$, $\psi = 10$.

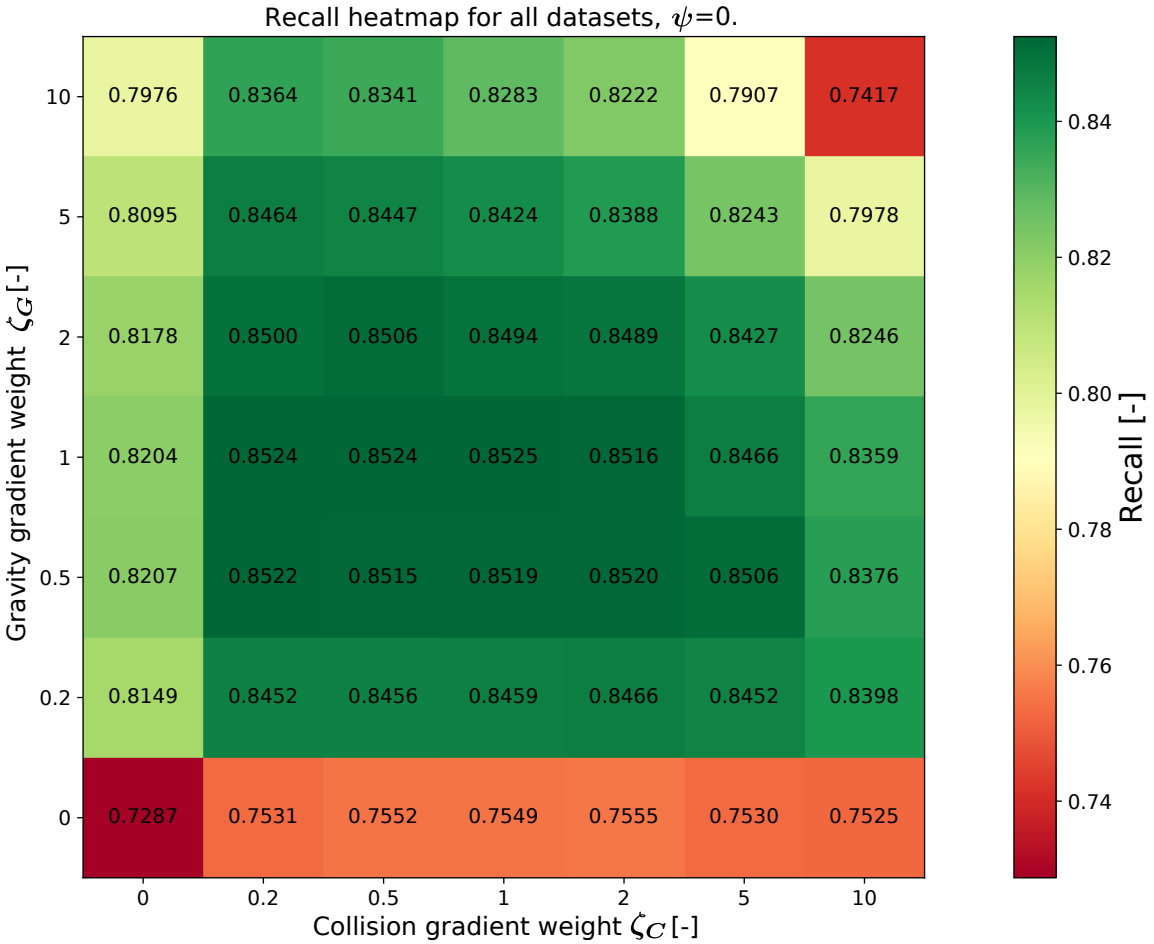


Figure 4.10: The graph shows evaluation for different values of ζ_C and ζ_G with $\psi = 0$. The values in the graph are averages of AVG recall across all five datasets. The bottom row does not use gravity and the left column does not use collisions. In the bottom left corner, is the result for MegaPose. Maximum when using gravity is at $\zeta_C = 1$ and $\zeta_G = 1$. Maximum when not using gravity (*i.e.* $\zeta_G = 0$) is at $\zeta_C = 2$.

4.6.3 Collision derivative

The computation of the derivatives of signed distances is the mainstay of this work. Finite Differences-based algorithms are commonly used for this task, but these can fail to converge well or lead to uninformative gradients. Therefore, we use a method to calculate the first-order estimate of collision derivatives implemented by Montaut et al. in Diffcoll [38]. We have already described this method in Sec. 3.5. To smoothen the local geometry around the mesh vertices, we use Gaussian noise implemented in the Diffcoll library; we therefore call the method "First-order Gaussian" for short.

To compare the effect of the methods for computing the derivatives of the signed distances, we executed our program with the same set of hyperparameters on three BOP datasets, using different methods for computing the derivatives. The result of the evaluation is shown in Table 4.5. In the tables for the HOPE-Video and YCB-V datasets, we see that the

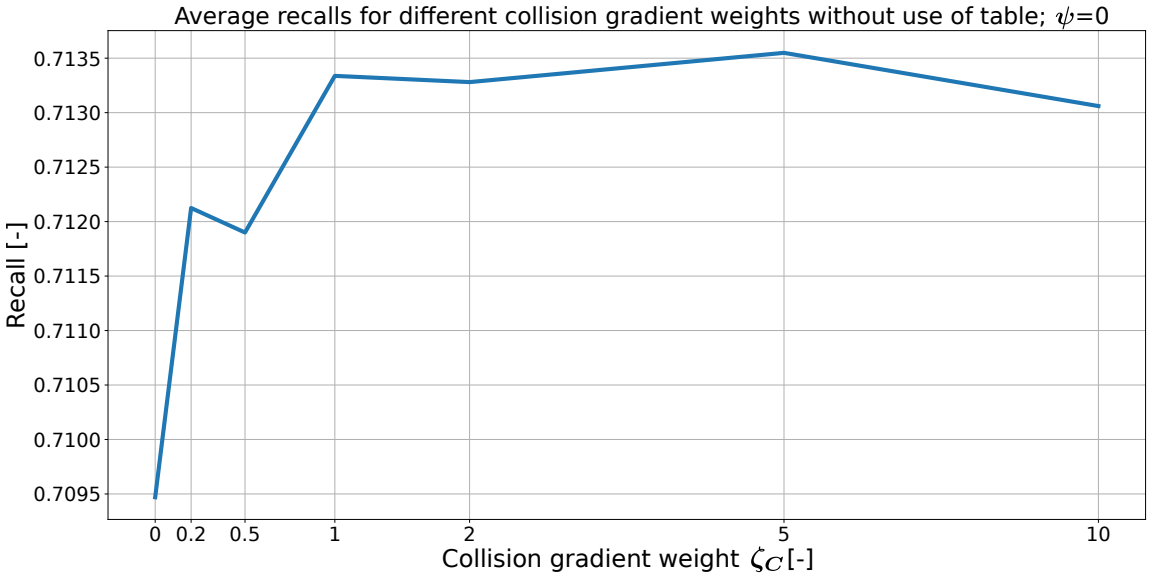


Figure 4.11: The graph shows evaluation for different values of ζ_C. In this case, the pose of the table is unknown. Scale ψ is set to zero. Global maximum is achieved for $\zeta_C = 5$.

difference between the methods is only in the region of 1% around zero for all metrics. In contrast, for the T-LESS dataset, we see a significant improvement when using First-order Gaussian. As we said in Sec. 4.4.1, working with the T-LESS dataset is different because we use CAD models of the objects instead of meshes obtained by 3D scanning. These CAD meshes are very accurate and thus have sharp edges and fewer vertices than if they were obtained by 3D scanning. We hypothesis that this is the reason why the First-order Gaussian outperformed the Finite Differences on the T-LESS dataset. Because of these ablation results, we decided to use First-order Gaussian smoothing for the collision gradient computation.

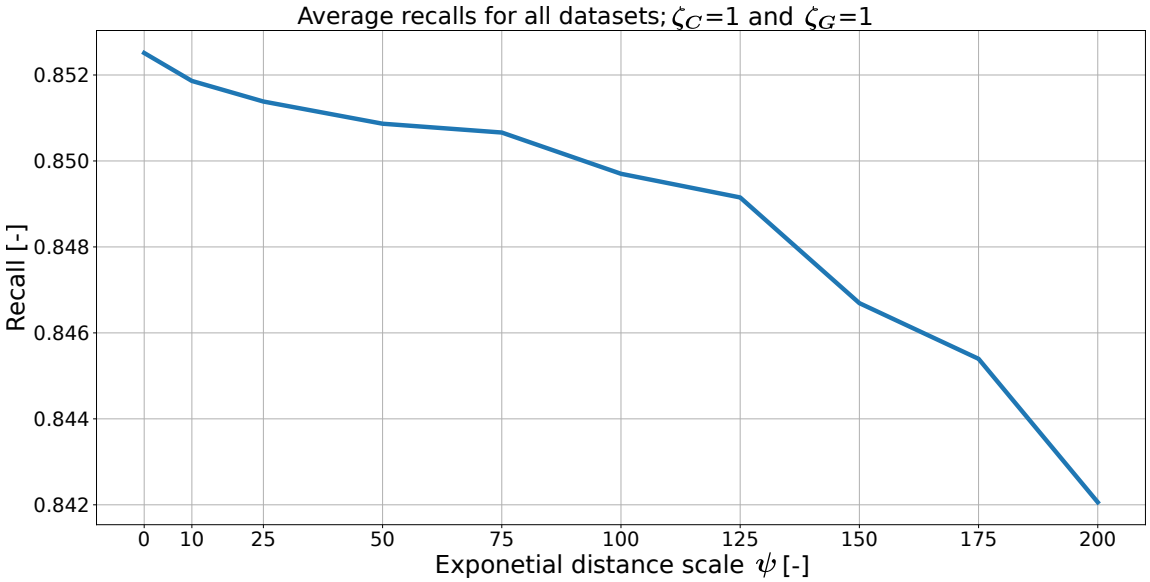


Figure 4.12: The graph shows evaluation for different values of ψ while using gravity. The gradient weights used are $\zeta_C = 1$ and $\zeta_G = 1$. The global maximum is for $\psi = 0$.

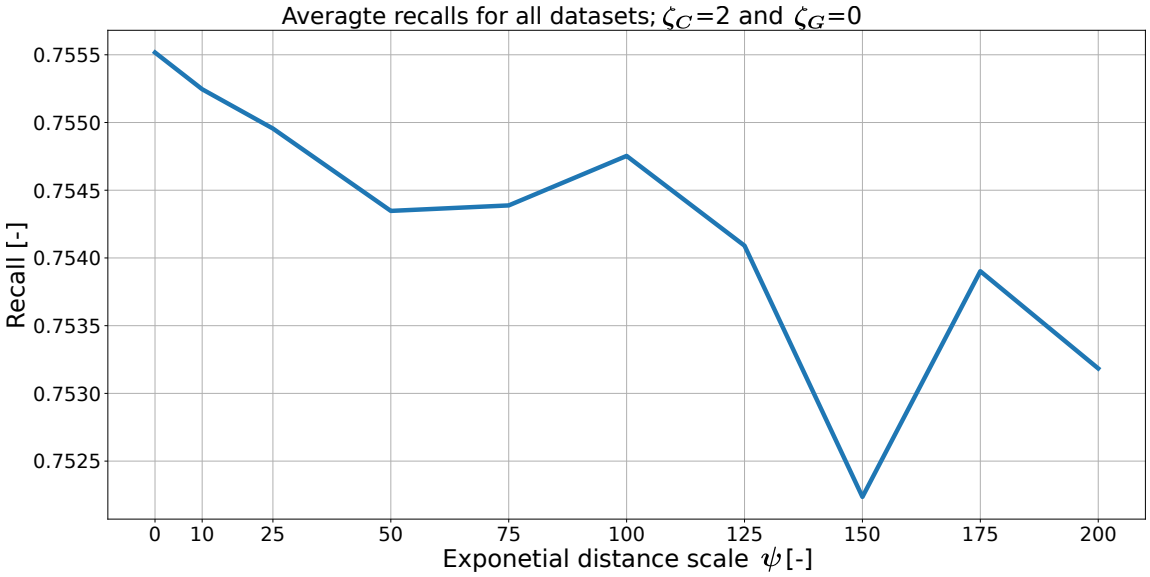


Figure 4.13: The graph shows evaluation for different values of ψ while not using gravity, but the table is used for collisions. The gradient weights used are $\zeta_C = 2$ and $\zeta_G = 0$. The global maximum is for $\psi = 0$.

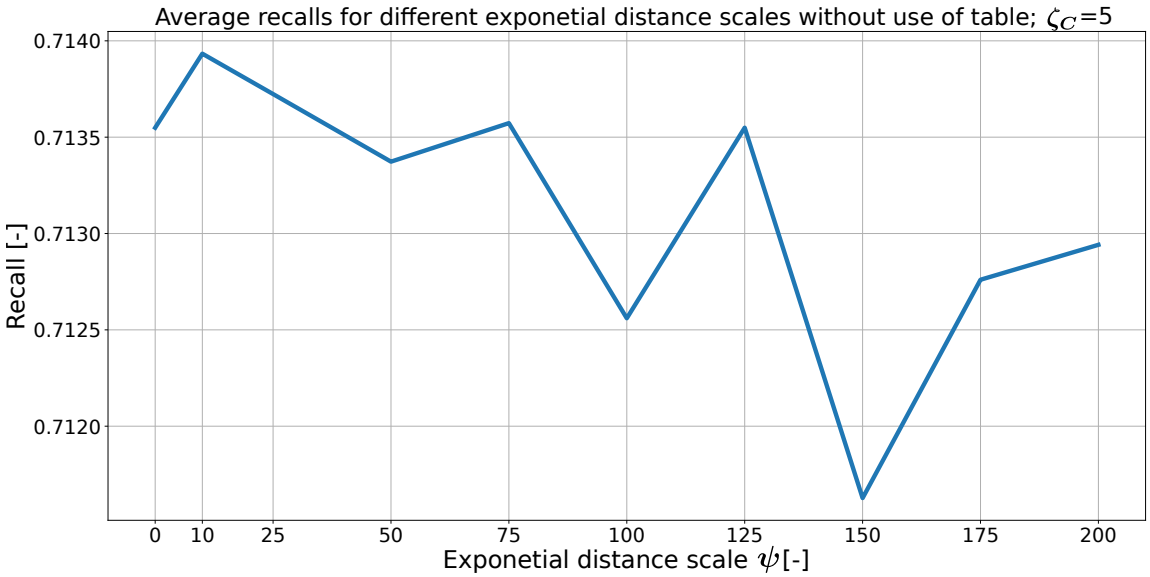


Figure 4.14: The graph shows evaluation for different values of ψ while table is not used. The gradient weights used are $\zeta_C = 5$ and $\zeta_G = 0$. The global maximum is for $\psi = 10$.

	AVG	MSPD	MSSD	VSD	TE	RE
Finite Differences	**0.7277**	**0.7339**	**0.6978**	**0.7514**	0.8682	**0.5978**
First-order Gaussian	0.7268	0.7325	0.6975	0.7503	**0.8688**	0.5971
Improvement [%]	-0.1	-0.2	0.0	-0.1	0.1	-0.1

(a) : HOPE-Video

	AVG	MSPD	MSSD	VSD	TE	RE
Finite Differences	0.8359	0.9575	0.7975	0.7528	0.8825	0.1500
First-order Gaussian	**0.8723**	**0.9650**	**0.8425**	**0.8095**	**0.9250**	0.1500
Improvement [%]	4.4	0.8	5.6	7.5	4.8	0.0

(b) : T-LESS

	AVG	MSPD	MSSD	VSD	TE	RE
Finite Differences	0.7831	**0.8441**	0.8021	0.7030	0.8303	0.5670
First-order Gaussian	**0.7850**	0.8408	**0.8075**	**0.7068**	**0.8320**	**0.5728**
Improvement [%]	0.3	-0.4	0.7	0.5	0.2	1.0

(c) : YCB-V

Table 4.5: The tables compares two approaches for collision derivative calculation: Finite Differences and Diffcol implementation of First-order estimator using Gaussian noise. The comparison is done for three real BOP datasets, the optimization is done using gravity. The last line in each table shows improvement of First-order Gaussian with respect to Finite Differences.

Chapter 5

Conclusions

In this thesis, we presented a method for introducing physical consistency into a scene for which object poses were estimated using an external pose estimator. We used gravity and collisions between objects in the scene to achieve physical consistency. Our experiments have shown that physical consistency has a significant impact on the standardized metrics used in the BOP Challenge, *i.e.* average of MSPD, MSSD and VSD recalls. Although we focused on the state-of-the-art pose estimator MegaPose, our method can be implemented on top of output of an arbitrary pose estimator in order to refine the poses by enforcing physical consistency.

To achieve consistency, we first defined three cost functions: collision, perception, and gravity. The collision cost penalizes the poses of objects that cause the intersection of their convex hulls. For better accuracy, we performed a convex decomposition of all the objects used, which was also reflected in the cost function. The perception cost aims to keep the estimate as it was initially predicted by the pose estimator since this initial estimate is the best we have. However, this estimate is usually accurate only from a perceptual point of view, *i.e.*, after projecting the object onto the image plane; the depth estimate is often wrong. Our formulation accounts for this imbalance of accuracy. The gravitational cost function penalizes the potential energy of objects relative to a selected stationary object from the scene, which is usually a table or floor. Then we analytically derived gradients for these three cost functions, which we then used in the gradient descent method.

We evaluated our proposed method and its variants on two synthetically generated datasets and on three standardized BOP datasets. Our method significantly improved the BOP metrics on all datasets relative to the original poses estimated by MegaPose. In addition, we conducted an experiment on a real Franka Emika Panda robot for a pick-and-place task. Using our method, the robustness in grasping manipulated objects was increased.

Bibliography

[1] 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.

[2] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, "MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare," in *CoRL*, 2022.

[3] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[4] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6d pose estimation and tracking of novel objects," in *CVPR*, 2024.

[5] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "BOP challenge 2020 on 6D object localization," *European Conference on Computer Vision Workshops (ECCVW)*, 2020.

[6] M. Malenický, "collision-pose," 2024. [Online]. Available: https://github.com/malenickymartin/collision-pose

[7] L. Roberts, "Machine perception of three-dimensional solids." Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[8] I. Sobel, "On calibrating computer controlled cameras for perceiving 3-d scenes," *Artificial Intelligence*, vol. 5, no. 2, pp. 185–198, 1974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370274900290

[9] R. Horaud, "New methods for matching 3-d objects with single perspective views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 3, pp. 401–412, 1987.

[10] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370287900701

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 404–417.

[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1150–1157.

[13] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The international journal of robotics research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[14] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*. IEEE, 2011, pp. 858–865.

[15] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[16] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4561–4570.

[17] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7668–7677.

[18] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[19] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1941–1950.

[20] M. Sundermeyer, T. Hodaň, Y. Labbé, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas, "Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects," *Conference on Computer Vision and Pattern Recognition*, pp. 2785–2794, June 2023.

[21] T. Hodaň, M. Sundermeyer, Y. Labbé, V. N. Nguyen, G. Wang, E. Brachmann, B. Drost, V. Lepetit, C. Rother, and J. Matas, "BOP challenge 2023 on detection, segmentation and pose estimation of seen and unseen rigid objects," *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

[22] 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*, 2010, pp. 998–1005.

[23] J. Deng, W. Qu, and S. Fang, "A high accuracy and recall rate 6d pose estimation method using point pair features for bin-picking," in *2022 34th Chinese Control and Decision Conference (CCDC)*, 2022, pp. 6056–6061.

[24] H. Wang, H. Wang, and C. Zhuang, "6d pose estimation from point cloud using an improved point pair features method," in *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, 2021, pp. 280–284.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[26] H. Fu, X. Mei, Z. Zhang, W. Zhao, and J. Yang, "Point pair feature based 6d pose estimation for robotic grasping," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, 2020, pp. 1803–1808.

[27] Y. Rong, J. Wang, Z. Liu, and C. C. Loy, "Monocular 3d reconstruction of interacting hands via collision-aware factorized refinements," in *2021 International Conference on 3D Vision (3DV)*, 2021, pp. 432–441.

[28] B. Smith, C. Wu, H. Wen, P. Peluse, Y. Sheikh, J. K. Hodgins, and T. Shiratori, "Constraining dense hand surface tracking with elasticity," *ACM Trans. Graph.*, vol. 39, no. 6, nov 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417768

[29] W. Jiang, N. Kolotouros, G. Pavlakos, X. Zhou, and K. Daniilidis, "Coherent reconstruction of multiple humans from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] Y. Hasson, G. Varol, C. Schmid, and I. Laptev, "Towards unconstrained joint hand-object reconstruction from rgb videos," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 659–668.

[31] M. Hassan, V. Choutas, D. Tzionas, and M. J. Black, "Resolving 3d human pose ambiguities with 3d scene constraints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[32] J. Y. Zhang, S. Pepose, H. Joo, D. Ramanan, J. Malik, and A. Kanazawa, "Perceiving 3d human-object spatial arrangements from a single image in the wild," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer, 2020, pp. 34–51.

[33] A. Davydov, M. Engilberge, M. Salzmann, and P. Fua, "Cloaf: Collision-aware human flow," *arXiv preprint arXiv:2403.09050*, 2024.

[34] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3d pictorial structures for multiple human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[35] K. Wada, E. Sucar, S. James, D. Lenton, and A. J. Davison, "Morefusion: Multi-object reasoning for 6d pose estimation from volumetric fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020, pp. 14 540–14 549.

[36] J. Lee, M. Lee, and D. Lee, "Uncertain pose estimation during contact tasks using differentiable contact features," *arXiv preprint arXiv:2305.16778*, 2023.

[37] Z. Landgraf, R. Scona, T. Laidlow, S. James, S. Leutenegger, and A. J. Davison, "Simstack: A generative shape and instance model for unordered object stacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 012–13 022.

[38] L. Montaut, Q. L. Lidec, A. Bambade, V. Petrik, J. Sivic, and J. Carpentier, "Differentiable collision detection: a randomized smoothing approach," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3240–3246.

[39] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2021.

[40] H. Ghorbani, "Mahalanobis distance and its application for detecting multivariate outliers," *Facta Universitatis, Series: Mathematics and Informatics*, pp. 583–595, 2019.

[41] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st ed. USA: Cambridge University Press, 2017.

[42] M. Taboga, "Precision matrix," https://www.statlect.com/glossary/precision-matrix, 2021, lectures on probability theory and mathematical statistics. Kindle Direct Publishing.

[43] V. Priban, M. Fourmy, J. Sivic, and V. Petrik, "Temporally consistent object 6d pose estimation for robot control," *in submission of IEEE RA-L*, 2024.

[44] J. Carpentier, N. Mansard, F. Valenza, J. Mirabel, G. Saurel, and R. Budhiraja, "Pinocchio - Efficient and versatile Rigid Body Dynamics algorithms," Jul. 2021. [Online]. Available: https://github.com/stack-of-tasks/pinocchio

[45] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, p. 469–483, dec 1996. [Online]. Available: https://doi.org/10.1145/235815.235821

[46] J. Pan, S. Chitta, D. Manocha, F. Lamiraux, J. Mirabel, J. Carpentier *et al.*, "HPP-FCL: an extension of the Flexible Collision Library," https://github.com/humanoid-path-planner/hpp-fcl, 2015–2022.

[47] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[48] T. Hodaň, "Pose estimation of specific rigid objects," PhD thesis, Czech Technical University in Prague, Prague, February 2021, available at https://dspace.cvut.cz/handle/10467/93910.

[49] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics and Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[50] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[51] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering," 2020.

[52] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield, "Multi-view fusion for multi-level robotic scene understanding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6817–6824.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[54] J. Elsner, "Taming the panda with python: A powerful duo for seamless robotics programming and integration," *SoftwareX*, vol. 24, p. 101532, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711023002285