공학석사 학위논문

푸싱 역학 학습에서의 물체 간 관통 방지

Avoiding Penetration in Pushing Dynamics Learning

2023년 2월

서울대학교 대학원

기계공학부

임 병 도

푸싱 역학 학습에서의 물체 간 관통 방지

Avoiding Penetration in Pushing Dynamics Learning

지도교수 박 종 우

이 논문을 공학석사 학위논문으로 제출함

2022 년 10 월

서울대학교 대학원

기계공학부

임 병 도

임 병 도의 공학석사 학위논문을 인준함

2022 년 12 월

위 원 장 :	이 경수
부위원장 :	박종우
위 원 :	이호원

ABSTRACT

Avoiding Penetration in Pushing Dynamics Learning

by

Byeongdo Lim

Department of Mechanical and Aerospace Engineering
Seoul National University

Making an accurate pushing dynamics model is critical in tasks such as pushing manipulation because it can accurately predict the changes in the poses of objects caused by the robot's push motion. In this paper, we argue that pushing dynamics model can be more accurate if it knows the law of physics in reality. In the real world, there is a physical law that objects do not penetrate each other, but existing pushing dynamics models cannot avoid penetration in predicted object poses. In this paper, we construct a repulsion module that receives and adjusts the poses predicted by the pushing dynamics model so that there is no penetration between the objects in the predicted poses. We train the repulsion module with simulation datasets and test it with simulation datasets and real-world datasets generated in real-world robot operation. The results from both datasets show that our method can avoid penetration and enhance dynamics accuracy in pushing dynamics learning.

Keywords: pushing dynamics learning, avoiding penetration

Student Number: 2021-28590

Contents

List of Figures

1 Introduction

Robotic pushing manipulation, in which a robot pushes objects to perform targeting tasks, has been growing interest. Pushing manipulation, for example, can be used to move objects to make them graspable [1, 2, 3, 4, 5], rearrange objects [1, 5, 6, 7], and move an occluded object to make it visible [8, 9].

For performing pushing manipulation, model-free and model-based methods are available. Model-free pushing manipulation constructs policy that takes observation of the environment as input and directly outputs best pushing action. They typically learn task-specific reward functions to evaluate each action, or they directly train the policy. Using model-free approaches, researchers have studied grasping [2, 3, 4], sorting [6], rearranging [7], and finding the invisible [8, 9] objects. However, their policy must implicitly learn how the environment changes as it interacts with the robot, which requires exploration of the environment and a large amount of data. Furthermore, because they use a task-specific reward function or policy, if a new task is assigned, the entire algorithm must be re-trained.

Model-based approach, on the other hand, is used to construct a model that understands how the environment changes as a result of the robot's push action. In other words, the model describes *pushing dynamics* which explains how the poses of objects transform according to the robot's push action. Through the model's prediction, the robot can determine which action is best suited for the target task and perform manipulation using the best action. Furthermore, because the model learns the general relationship between the environment and action, it can be applied to other tasks.

Analytic methods [10, 11, 12] had been extensively researched to precisely predict the pushing dynamics by constructing the model using the dynamic properties (e.g., shape, mass distribution, and friction distribution) of the manipulated objects. They cannot be used, however, if one of these properties is unknown. These methods are especially difficult to be applied when only vision data is provided, as the mass and friction distribution of the objects cannot be determined.

Recently, in situations where only vision observations were provided, data-driven approaches [1, 13, 14, 15] that learn pushing dynamics from data have been used. They use neural network to learn pushing dynamics from vision input. SE3-Nets [13] and SE3-Pose-Nets [14] take a raw point cloud of the scene, segment it into objects, predicts each object's pose transformation, and output the transformed raw point cloud. However, because this work is limited to raw point cloud, the model cannot recognize object shapes and learning accurate pushing dynamics is hindered.

DSR-Net [15] solves this problem by recognizing the object shape and predicting the object pose transformation in voxel space. It takes a truncated signed distance function (TSDF) of the voxelized workspace, segments the voxels to each object to recognize the object shape, predicts the pose transformation of each object, and outputs the voxelized scene flow. However, because recognition in voxel space is quite complicated, object shape recognition performance is far-less-than-satisfying, and thus learned dynamics performance is as well.

SQPD-Net [1] recognizes object shape by employing a shape class known as *superquadrics*, which can represent a wide range of shapes such as boxes, cylinders, ellipsoids, octahedrons, and so on. It takes a point cloud of the scene within workspace, segments it into each objects, recognizes object shapes and poses, and predicts the pose transformation of each object. Thanks to the simple object recognition in superquadric parameters, its object shape recognition performance is improved, and its pushing dynamics accuracy achieves state-of-the art.

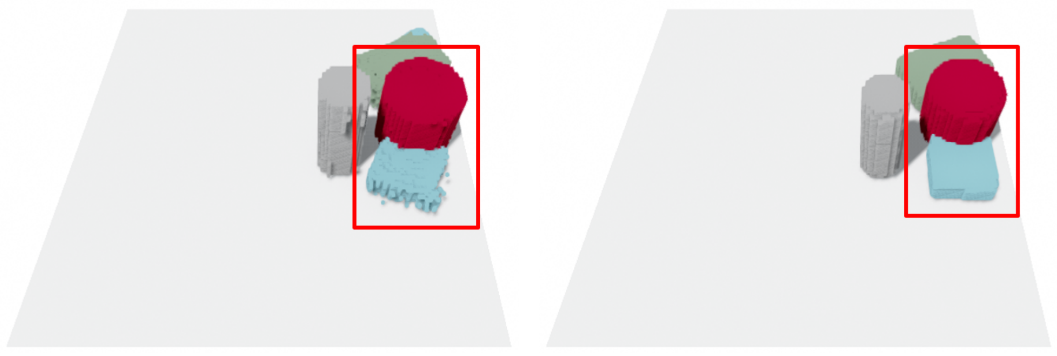

(a) DSR-Net [15].   (b) SQPD-Net [1].

Figure 1.1: Penetration between objects in the poses predicted by existing pushing dynamics model.

Such approaches have sought to improve the accuracy of the learned dynamics; however, we argue that few studies have attempted to improve dynamics accuracy by learning the realistic aspects of real-world physics. In particular, there is a physical law that objects cannot penetrate each other in the real world. However, existing state-of-the-art pushing dynamics models cannot avoid penetration and thus cannot reflect the physical law. Figure 1.1 shows the objects in the poses predicted by DSR-Net [15] and SQPD-Net [1], and the red cylinder and sky-blue box penetrate each other. Other pushing dynamics models are not shown because they do not recognize the shapes of objects. An ideal pushing dynamics model should reflect real-world physics and should not have penetration problem between the objects, so a model made to avoid the penetration will become more accurate.

In this work, we construct a repulsion module that prevents the object in the poses predicted by a pushing dynamics model from penetrating. The repulsion module receives the object poses predicted by the dynamics model, calculates a potential indicating the degree of penetration between objects, and adjusts the poses to make the potential zero. The adjustment process is implemented as a gradient descent of the potential with respect to the object poses, and it serves as a repulsion between penetrated objects. The repulsion module is designed to be trainable, in order to achieve a balance between the repulsion of the objects' positions and orientations.

The structure of this paper is as follows. Chapter 2 discusses details about the preliminary work, SQPD-Net; details about superquadrics, shape recognition using superquadrics, and transformed pose prediction using superquadric parameters as input. In Chapter 3, the theoretical structure of our repulsion module is explained, and in Chapter 4, we explain the experiments and results of our repulsion module. Finally, Chapter 5 provides a conclusion as well as opportunities for improvement of our repulsion module.

2 Preliminaries

Our repulsion module is constructed upon the state-of-the-art pushing dynamics model, SQPD-Net [1]. This model utilizes superquadrics for both recognizing the shapes of objects and predicting their transformed poses. As a result, the repulsion module detects penetration between objects using these superquadrics. Therefore, before explaining the repulsion module, it is necessary to provide an overview of superquadrics and overall structure of the SQPD-Net [1], which will be used in experiments with our repulsion module.

2.1 Superquadrics

Superquadrics are a shape class that can represent various shapes, such as boxes, cylinders, ellipsoids, octahedrons and so on, using a single continuous parameter space [16]. Their parameters consist of two shape parameters e_1, e_2 determining the shape, and three size parameters a_1, a_2, a_3 determining the size of the shape. Figure 2.1 shows example of superquadrics when shape parameters are changed.

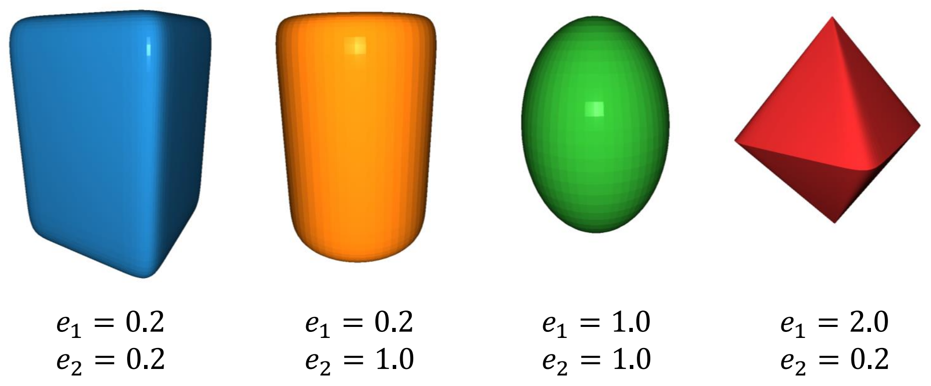


Figure 2.1: Examples of shapes represented by superquadric.

2.1.1 Explicit and implicit equations of superquadrics

The surface of superquadrics is defined by explicit or implicit equations. The explicit equation defines the surface vector $\mathbf{r}$ from the center of superquadric as

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} a_1 \cos^{e_1} \eta \cos^{e_2} \omega \\ a_2 \cos^{e_1} \eta \sin^{e_2} \omega \\ a_3 \sin^{e_1} \eta \end{bmatrix} \qquad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega \le \pi \end{matrix} \tag{2.1.1}$$

where η denotes the inclination which represents the signed angle between the vector $\mathbf{r}$ and xy plane, and ω denotes the azimuth which represents the signed angle between the vector $\mathbf{r}$ and positive x axis [16]. The implicit function is defined as

$$f(\mathbf{x}; \mathbf{s}) = \left(\left| \frac{x_1}{a_1} \right|^{\frac{2}{e_2}} + \left| \frac{x_2}{a_2} \right|^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} + \left| \frac{x_3}{a_3} \right|^{\frac{2}{e_1}} \tag{2.1.2}$$

where $\mathbf{x} = [x_1 \ x_2 \ x_3]^\top \in \mathbb{R}^3$ denotes the coordinates of a point and $\mathbf{s} = (a_1, a_2, a_3, e_1, e_2)$ denotes superquadric parameters. The implicit function indicates relative position of a point and a superquadric surface; if $f(\mathbf{x}; \mathbf{s}) = 1$, the point lies on the surface, if $f(\mathbf{x}; \mathbf{s}) < 1$, the point is inside of the superquadric, and if $f(\mathbf{x}; \mathbf{s}) > 1$, the point is outside of the superquadric.

2.1.2 Distance between point and superquadric surface

Also, distance between a point $\mathbf{x}$ and a superquadric surface defined by a superquadric parameters $\mathbf{s}$ is calculated as follows:

$$\delta_s(\mathbf{x}, \mathbf{s}) = \|\mathbf{x}\| \left| 1 - f^{-\frac{e_1}{2}}(\mathbf{x}; \mathbf{s}) \right| \tag{2.1.3}$$

where $\|\cdot\|$ denotes the Euclidean norm [17]. Similar with the implicit function, the superquadric distance indicates that the point is inside of the superquadric if $\delta_s(\mathbf{x}, \mathbf{s}) < 0$.

2.2 Superquadric pushing dynamics network

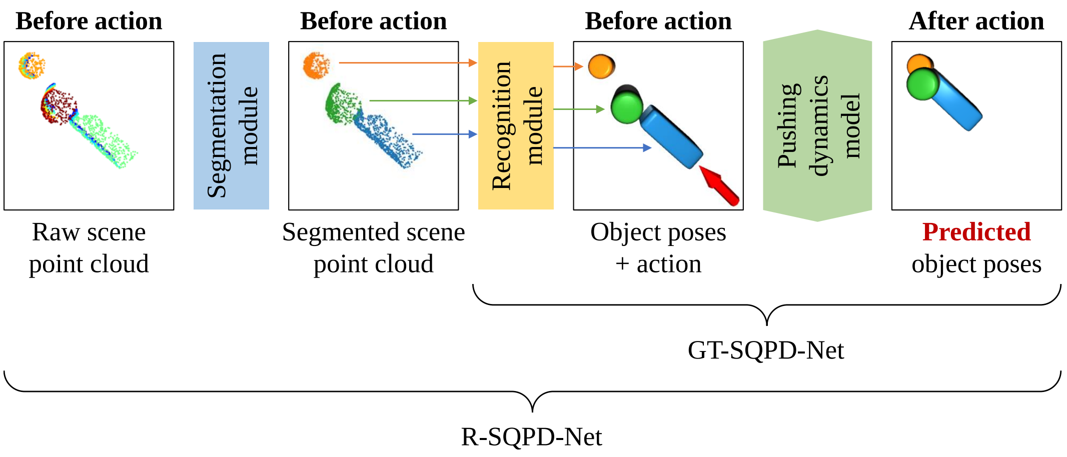


Figure 2.2: The overall structure of R- and GT-SQPD-Net [1].

Superquadric pushing dynamics network [1], abbreviated to SQPD-Net, is a pushing dynamics model that predicts objects' transformed poses according to robot's push action and uses superquadrics to recognize the objects. It is classified into *R-SQPD-Net* and *GT-SQPD-Net* according to the presence or absence of segmentation and recognition modules as shown in Figure 2.2. R-SQPD-Net takes a scene point cloud as input, segments it to each object's point cloud, recognizes the objects using superquadric parameters and poses, and predicts all objects' transformed poses using the recognized information and robot's push action. GT-SQPD-Net, on the other hand, which does not recognize objects, takes ground-truth superquadric parameters and poses of objects, and applies them directly to transformed pose prediction. Point cloud segmentation, object shape recognition, and transformed pose prediction modules is explained in order in the following sections.

2.3 Point cloud segmentation

SQPD-Net [1] utilizes a simple segmentation algorithm to segment a scene point cloud into object point clouds. It takes the scene point cloud as input and outputs segmentation labels for each point. Mathematically, given the scene point cloud $\mathcal{P} := \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^{N_{pc}}$, the module predicts segmentation label vector for each point $\hat{\mathbf{y}}_i = (y_{i1}, \ldots, y_{1N_o}) \in \mathbb{R}_+^{N_o}$ where N_o denotes the number of objects.

2.3.1 Structure

The overall structure of point cloud segmentation module of SQPD-Net [1] is shown in Figure 2.3. First, EdgeConv layers from the Dynamic Graph Convolution Neural Network (DGCNN) [18] is used as a backbone to maintain permutation-invariant characteristic of point cloud; even if the order of points changes in the point cloud, it indicates same point cloud, so the output should not be changed. Using five EdgeConv layers with point-wise latent space dimensions (64, 64, 128, 256) and a max pooling layer as a backbone, the input point cloud $\mathcal{P}$ is transformed into a

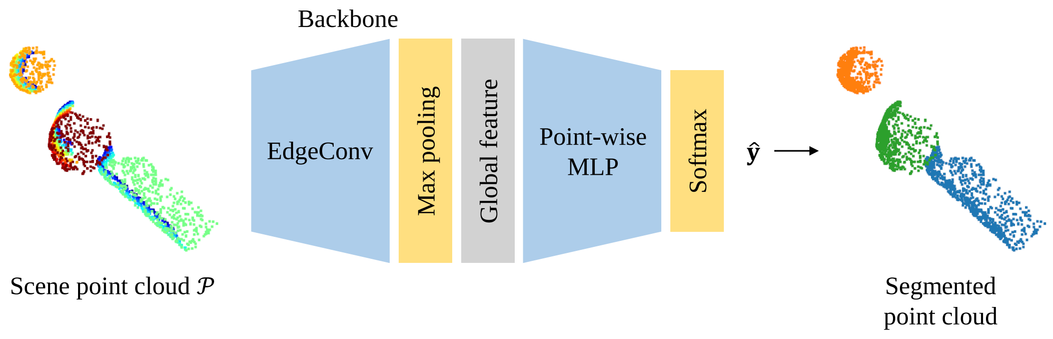


Figure 2.3: Structure of point cloud segmentation module in SQPD-Net [1].

global feature vector with 1024 dimension.

The global feature vector generated by the backbone is transformed into segmentation labels through additional networks. The networks are implemented by point-wise Multi-Layer Perception (MLP) layers, each of them followed by leaky Rectified Linear Unit (ReLU) with the negative slope angle of 0.2. Their latent space dimensions are (512, 256, 128) and they outputs N_o-dimensional segmentation labels $\hat{\mathbf{y}} = \{\hat{\mathbf{y}}_i\}_{i=1}^{N_{pc}} \in \mathbb{R}_+^{N_{pc}\times N_o}$ for each point after softmax activation.

2.3.2 Loss function

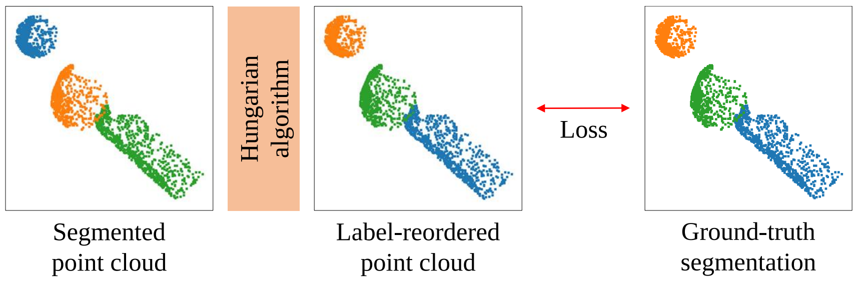


Figure 2.4: Reordering predicted segmentation labels and comparison with the ground-truth segmentation label in SQPD-Net [1].

The loss function used to train the point cloud segmentation module in SQPD-Net [1] evaluates the difference between the prediction and ground-truth segmentation labels. First, segmenting the scene point cloud for each object is not a semantic segmentation but close to a instance segmentation, so permutation of the segmentation labels between the objects is possible. Thus, assigning the predicted segmentation labels for the ground-truth segmentation labels is necessary. This assignment process is implemented by Hungarian algorithm which solves an optimal assignment problem, as shown in Figure 2.4. Denoting the ground-truth segmentation labels as $\mathbf{y}$, details about the assignment process is as follows:

(i) Define cost matrix $\mathbf{C} \in \mathbb{R}_+^{N_o \times N_o}$ by its entities as

$$C_{jk} = \frac{\sum_{i=1}^{N_{pc}} \hat{y}_{ij} y_{ik}}{\sum_{i=1}^{N_{pc}} \hat{y}_{ij} + \sum_{i=1}^{N_{pc}} y_{ik} - \sum_{i=1}^{N_{pc}} \hat{y}_{ij} y_{ik}} \tag{2.3.4}$$

where $j = 1, \ldots, N_o$ and $k = 1, \ldots, N_o$.

(ii) Assign the j-th predicted labels to the k-th ground-truth labels by solving following assignment problem.

$$j^*, k^* = \arg\max_{j,k} \sum_j \sum_k C_{jk} X_{jk} \tag{2.3.5}$$

where $X_{jk} = 1$ iff j is assigned to k.

(iii) Build a Boolean matching matrix $\mathbf{M}$ where $M_{jk} = 1$ iff $j = j^*, k = k^*$ and reorder the predicted segmentation labels as $\hat{\mathbf{y}}' = \hat{\mathbf{y}}\mathbf{M}$.

Following the assignment of predicted labels to the ground-truth labels, the loss L_s evaluates the difference between the prediction and ground-truth through a cross-entropy as follows:

$$L_s = \frac{1}{N_{pc}} \sum_{i=1}^{N_{pc}} \left[-\sum_{j=1}^{N_o} \left(y_{ij} \log \hat{y}'_{ij} + (1 - y_{ij}) \log (1 - \hat{y}'_{ij}) \right) \right] \tag{2.3.6}$$

2.4 Superquadric-based shape recognition

SQPD-Net [1] employs superquadric-based shape recognition, which is derived from its previous work, DSQNet [19]. It takes partially observed point cloud of a object as input and outputs best superquadric parameters and pose that describes the object. Mathematically, given the partially observed point cloud of a object $\mathcal{P}_o := \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^{N_{pc,o}}$, the module predicts superquadric parameters $\hat{\mathbf{s}}$ and pose $\hat{\mathbf{T}} \in \mathrm{SE}(3)$ which are best-fitted to the full point cloud of the object $\mathcal{P}_g := \{\mathbf{x}_{g,i} \in \mathbb{R}^3\}_{i=1}^{N_{pc,g}}$, which is the ground-truth.

2.4.1 Structure

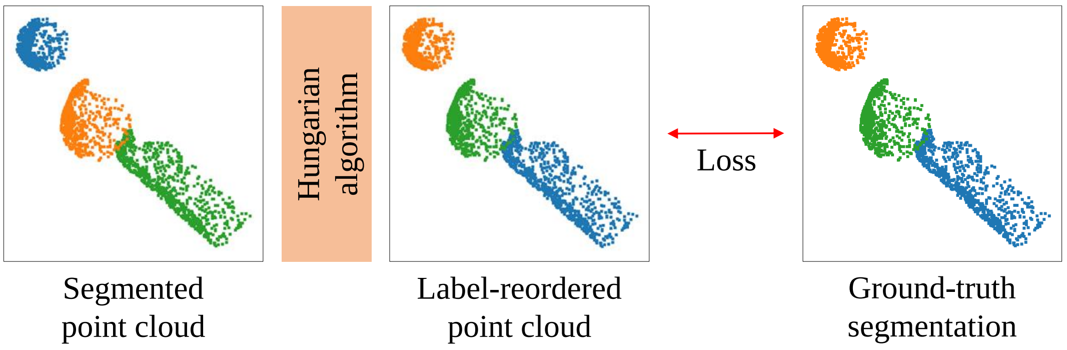


Figure 2.5: Structure of shape recognition module in SQPD-Net [1].

Figure 2.5 shows the overall structure of shape recognition module in SQPD-Net [1]. The structure of the module is the same as that of DSQNet [19] except that there is no deformation in superquadrics.

A backbone of the same structure with the one in segmentation module is used to produce the 1024-dimensional global feature. The superquadric parameters and pose are then generated when the global feature vector is passed through additional networks. The networks are implemented by fully-connected MLP layers, each of them is followed by leaky ReLU with the negative slope angle of 0.2. Their descriptions are as follows:

- The shape network consists of MLP layers with latent space dimensions (512, 256) and outputs 2-dimensional shape parameters after sigmoid activation. To avoid the divergence of equation 2.1.2 as e_1 or e_2 approaches zero and to avoid the superquadrics from becoming non-convex shapes, the shape parameters $\hat{\mathbf{e}} = (e_1, e_2)$ is imposed a lower bound of 0.2 and an upper bound of 1.7.
- The size network has the same structure as the shape network except that the output dimension is 3. To cover the size of the targeting objects, the size parameters $\hat{\mathbf{a}} = (a_1, a_2, a_3)$ are bounded in the interval [0.03, 0.53].
- The translation network outputs position part $\hat{\mathbf{t}} \in \mathbb{R}^3$ of the pose $\hat{\mathbf{T}}$, and it consists of MLP layers with latent space dimensions (512, 256). It has no activation layer and output dimension in 3.
- The rotation network outputs orientation part $\hat{\mathbf{R}} \in \mathrm{SO}(3)$ of the pose $\hat{\mathbf{T}}$ in the form of the unit quaternion representation $\hat{\mathbf{q}}$. Therefore, its structure is the same as that of the translation network except that the output dimension is 4. A normalization is applied to make the norm of the output to 1.

2.4.2 Loss function

In addition to the structure, the loss function used to train the shape recognition module in SQPD-Net [1] is the same as that of DSQNet [19]. It evaluates the suitability of the surface corresponding to the superquadric parameters $\hat{\mathbf{s}}$ and pose $\hat{\mathbf{T}}$ with the ground-truth point cloud $\mathcal{P}_g$. The suitability is implemented by the average of squared distances between the surface and the ground-truth point cloud. To measure the distances, points in the ground-truth point cloud are transformed with the inverse of pose $\hat{\mathbf{T}}^{-1}$, so that the point cloud is aligned with the surface with the superquadric's pose. The loss L_r is then evaluated by the mean squared error (MSE), where the error is the distance calculated by the equation 2.1.3.

$$L_r = \frac{1}{N_{pc,g}} \sum_{i=1}^{N_{pc,g}} \delta_s^2(\hat{\mathbf{T}}^{-1}\mathbf{x}_{g,i}, \hat{\mathbf{s}}) \tag{2.4.7}$$

2.5 Superquadric-based transformed pose prediction

Although SQPD-Net [1] was designed for the SE(2)-equivariant pushing dynamics model, but we will focus on how it utilizes superquadric and how it predicts transformed pose. It takes as input superquadric parameters and poses of given objects, as well as the robot's push action, and then predicts the transformed poses. Given the number of objects N_o, the superquadric parameters and poses are a set of tuples $\{(\mathbf{s}_i, \mathbf{T}_i)\}_{i=1}^{N_o}$, and the push action is represented by a tuple $(\mathbf{p}, \mathbf{v})$ where $\mathbf{p} \in \mathbb{R}^3$ is the position of its starting point and $\mathbf{v} \in \mathbb{R}^3$ is the unit vector indicating its direction. Since the distance traveled by the robot moves during the push action is fixed, SQPD-Net [1] is constructed as a discrete-distance pushing dynamics model that outputs transformed poses $\{\hat{\mathbf{T}}_i'\}_{i=1}^{N_o} = g(\{(\mathbf{s}_i, \mathbf{T}_i)\}_{i=1}^{N}, (\mathbf{p}, \mathbf{v}))$.

2.5.1 Structure

Since the module g predicts each object's transformed pose individually, it is divided into g_i's where $i = 1, \ldots, N_o$ and g_i predicts i-th object's transformed pose,

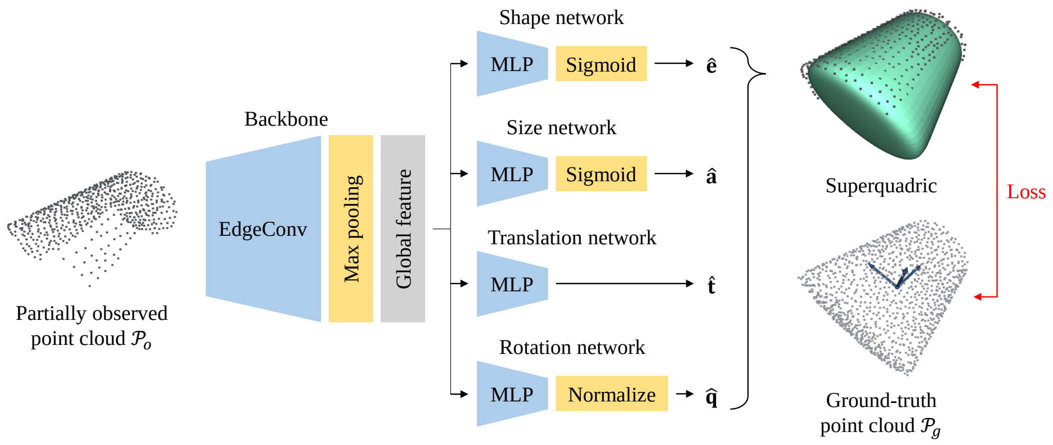


Figure 2.6: Structure of transformed pose prediction module in SQPD-Net [1].

i.e., $\hat{\mathbf{T}}_i' = g_i(\{(\mathbf{s}_i, \mathbf{T}_i)\}_{i=1}^{N_o}, (\mathbf{p}, \mathbf{v}))$. Figure 2.6 shows the overall structure of transformed pose prediction module for i-th object, g_i.

The module to predict i-th object's transformed pose will be explained. To impose SE(2)-equivariance on the module, the i-th object's pose is decomposed as $\mathbf{T}_i = \mathbf{C}_i\mathbf{U}_i$ and $\mathbf{C}_i^{-1}$ transforms the all poses and the action, but the rationale will be omitted because it is irrelevant to our work. $\mathbf{C}$ is defined as projection of $\mathbf{T}$ on the xy plane, so that it has the form of SE(2) as follows:

$$\mathbf{C} = \begin{bmatrix} \mathbf{Rot}(\hat{\mathbf{z}}, \theta) & \mathbf{t}_{xy} \\ 0 & 1 \end{bmatrix} \tag{2.5.8}$$

where $\mathbf{Rot}(\hat{\mathbf{z}}, \theta)$ is a 3×3 matrix for rotation along z-axis with angle θ and $\mathbf{t}_{xy} = (t_x, t_y, 0) \in \mathbb{R}^3$ is a translation vector on xy plane. $\mathbf{U}$ is then defined as $\mathbf{C}^{-1}\mathbf{T}$.

The action $\mathbf{C}_i{}^{-1}(\mathbf{p}, \mathbf{v})$, tuple of i-th object's own pose and superquadric parameters $(\mathbf{s}_i, \mathbf{U}_i)$, and tuples of every objects' poses and superquadric parameters $\{(\mathbf{s}_j, \mathbf{C}_i{}^{-1}\mathbf{T}_j)\}_{j=1}^{N_o}$ are entered into action, ego, and scene networks respectively. The action and ego networks are composed of fully-connected MLP layers, and the scene networks are composed of point-wise MLP layers. Every MLP layers are followed by leaky ReLU with the negative slope angle of 0.2, and their latent space dimensions are (64, 128) and they outputs 256-dimensional feature vector of action $\mathbf{a}_i$, ego $\mathbf{b}_i$, and scene $\mathbf{c}_i^j$. Scene feature vectors are max-pooled to generate global scene feature vectors $\mathbf{c}_i$.

The action, ego and global scene feature vectors are concatenated and entered into the motion network to generate pose transformation $\delta\mathbf{T}_i \in \mathrm{SE}(3)$. The pose transformation is generated in the form of position and orientation, and the motion network is divided into a position and orientation network. They consist of MLP layers with latent space dimensions (256, 128, 64). Since we assume planar movement of objects in non-stacked situation, outputs from position and orientation networks indicates 2-dimensional translation in xy coordinates and 1-dimensional rotation in z-axis. Rotation is generated in the form of cosine and sine of its angle, so its network generates 2-dimensional vector which is normalized to make the norm 1.

Finally, the pose transformation $\delta\mathbf{T}_i$ is multiplied to the pose before action $\mathbf{T}_i$ and the transformed pose is obtained as $\hat{\mathbf{T}}_i' = \mathbf{T}_i\delta\mathbf{T}_i$. This procedure is applied to all objects.

2.5.2 Loss function

The transformed pose prediction module in SQPD-Net [1] is trained using a loss that evaluates the difference between the prediction and ground-truth of transformed object poses. Denote the ground-truth pose of i-th object after action as $\mathbf{T}_i'$, and decompose it into 2-dimensional xy coordinates vector $\mathbf{t}_{xy,i}'$ and 1-dimensional z-axis rotation angle $\theta_{z,i}'$. The predicted pose of i-th object after action $\hat{\mathbf{T}}_i'$ is decomposed into $\hat{\mathbf{t}}_{xy,i}'$ and $\hat{\theta}_{z,i}'$ in the same way. The position difference is evaluated by the squared error between xy coordinates, and the orientation difference is evaluated by the squared error of cosine value of the difference in angle. At last, as shown in Equation 2.5.9, the loss is defined as a weighted sum of two differences, and summed to all objects. α is a weight hyperparameter.

$$L_d = \sum_{i=1}^{N_o} \left(\|\mathbf{t}'_{xy,i} - \hat{\mathbf{t}}'_{xy,i}\|_2^2 + \alpha(1 - \cos(\theta'_{z,i} - \hat{\theta}'_{z,i}))_2^2 \right) \tag{2.5.9}$$

3

Avoiding Penetration in Pushing Dynamics Learning

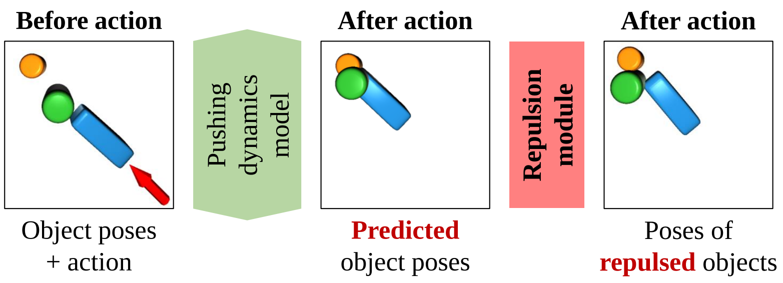


Figure 3.1: The overall structure of SQPD-Net [1] and our repulsion module.

Our repulsion module is constructed upon the SQPD-Net [1] as shown in Figure 3.1. The repulsion module receives SQPD-Net's object shape recognition and transformed pose prediction, calculates penetration potential, and adjusts the predicted poses using gradient descent of the potential which serves as a repulsion between penetrated objects. The repulsion module is designed to be trainable in order to achieve the best balance of position and orientation in repulsion. Section 3.1 describes the design of the penetration potential, Section 3.2 describes the repulsion algorithm, and Section 3.3 describes how to train the repulsion module.

3.1 Penetration potential

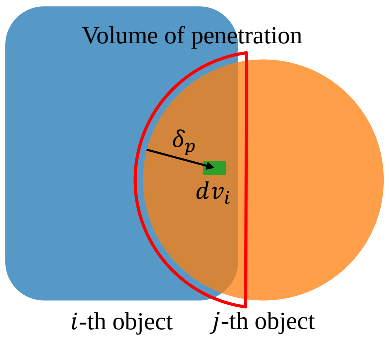
In designing the penetration potential, we determine two properties that the potential should have. First, to realize the repulsion process by gradient descent of the potential until it becomes zero, the potential must be zero when there is no penetration between objects, i.e., the penetrated volume is zero. Second, to ensure that the repulsive force is increased when objects are penetrated deeply, the potential is increased in that case. As a result, the penetration potential is mathematically defined as an integration of the penetrated volume weighted by the penetration distance.

3.1.1 Penetration distance

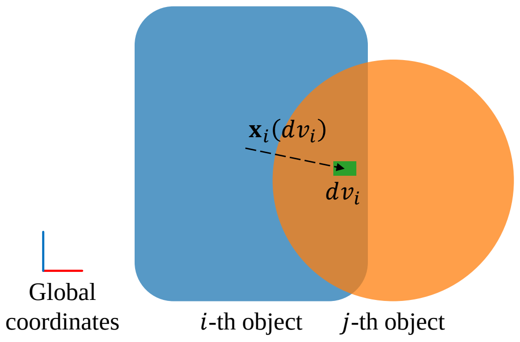
The penetration distance δ_p between a point $\mathbf{x}$ and a superquadric surface described by a superquadric parameters $\mathbf{s}$ is defined by the following descriptions; the penetration distance is either 1) the opposite of the superquadric distance between the point and the surface if the superquadric distance is less than or equal to 0, or 2) 0 if the superquadric distance is greater than 0. Therefore, mathematically, the penetration distance is defined as follow:

$$\delta_p(\mathbf{x}, \mathbf{s}) = \max(0, -\delta_s(\mathbf{x}, \mathbf{s})) \tag{3.1.1}$$

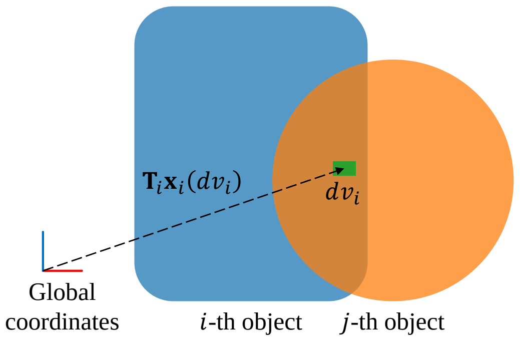
where δ_s denotes the superquadric distance defined in Equation 2.1.3.



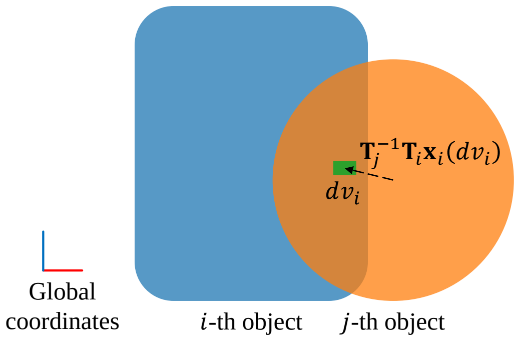
(a) Definition of the integration.



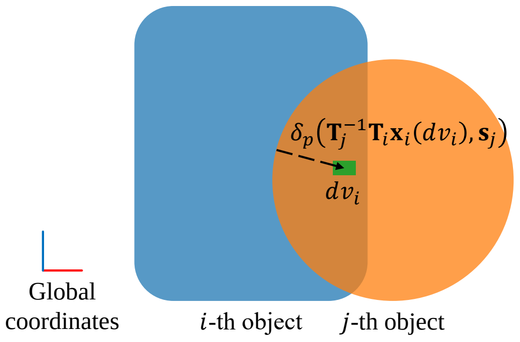
(b) i-th-object-centric position.



(c) Global position.



(d) j-th-object-centric position.


$\delta_p\left(\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{x}_i(dv_i), \mathbf{s}_j\right)$
dv_i
Global coordinates
i-th object
j-th object

(e) Penetration distance.

Figure 3.2: Integration for the penetration potential.

3.1.2 Integration

Figure 3.2 shows penetration between i-th and j-th objects in a top-down view. To evaluate the penetration potential of the i-th object relative to the j-th-object, the i-th-object is divided into the infinitesimal volumes and and the penetration distances between the j-th object are weighted as shown in Figure 3.2a.

To obtain the penetration distance of the i-th-object's infinitesimal volume between the j-th-object, following procedures are performed as shown in Figure 3.2b to 3.2e.

(i) Figure 3.2b: Define object-centric position of its center point $\mathbf{x}_i(dv_i)$ where dv_i is the infinitesimal volume of the i-th object.

(ii) Figure 3.2c: The i-th-object-centric position is transformed to the global position by $\mathbf{x}_i(dv_i) \to \mathbf{T}_i\mathbf{x}_i(dv_i)$.

(iii) Figure 3.2d: the global position is transformed to the j-th-object-centric position by $\mathbf{T}_i\mathbf{x}_i(dv_i) \to \mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{x}_i(dv_i)$.

(iv) Figure 3.2e: Using the Equation 3.1.1, the penetration distance between the center point of i-th object's infinitesimal volume and the j-th object's surface is calculated by $\delta_p(\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{x}_i(dv_i), \mathbf{s}_j)$.

Finally, The penetration potential V for the entire scene is defined by integrating the infinitesimal volume weighted by the penetration distance and summing over all object combinations as follows:

$$V = \sum_{i=1}^{N_o} \sum_{j \neq i} \int \delta_p(\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{x}_i(dv_i), \mathbf{s}_j)\, dv_i \tag{3.1.2}$$

3.2 Repulsion algorithm

Utilizing the recognized superquadric parameters and predicted transformed poses for all objects provided by SQPD-Net [1], the repulsion module reconstructs the scene and compute the penetration potential. In practical implementation, the integration over infinitesimal volumes is substituted by a summation over a voxel space to compute the penetration potential. We assume planar movement of objects in non-stacked situation as same with SQPD-Net [1], so object poses $\mathbf{T}_i$ $(i = 1, \ldots, N_o)$ are decomposed into xy coordinates $\mathbf{t}_{xy,i}$ and z-axis rotation angles $\theta_{z,i}$. They are adjusted via gradient descent of the potential as shown in Figure 3.3. A detailed description of the repulsion algorithm is provided in Algorithm 1. The terminating penetration distance is set as 0.5cm to prevent a long calculation time.

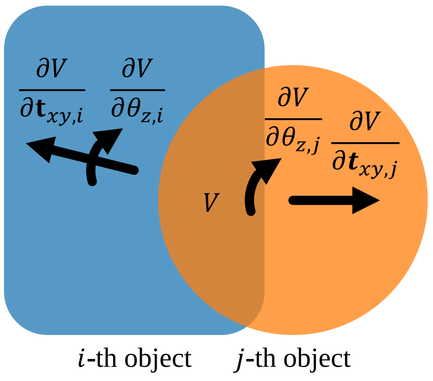

$\frac{\partial V}{\partial \mathbf{t}_{xy,i}}$
$\frac{\partial V}{\partial \theta_{z,i}}$
$\frac{\partial V}{\partial \theta_{z,j}}$
$\frac{\partial V}{\partial \mathbf{t}_{xy,j}}$
V
i-th object
j-th object

Figure 3.3: The gradient descent process as repulsion.

Algorithm 1 Repulsion algorithm

```
Require: Number of objects $N_o$
Require: Voxel size $d$
Require: Object's recognized superquadric parameters $\mathbf{s}_i$
Require: Object's $xy$ coordinates $\mathbf{t}_{xy,i}$, $i = 1, \ldots, N_o$
Require: Object's $z$-axis rotation angle $\theta_{z,i}$, $i = 1, \ldots, N_o$
Require: Terminating penetration distance $\delta_{p,eps}$
Require: Repulsion weight $\epsilon$
Require: Balancing parameter $\beta$
 1: Voxelize object-centrically each object
 2: Denote the number of voxels as $N_{v,i}, i = 1, \ldots, N_o$
 3: Obtain the object-centric position of each voxel $\mathbf{x}_{i,k}, i=1, \ldots, N_o, k=1, \ldots, N_{v,i}$
 4: while true do
 5:     $V \leftarrow 0$
 6:     $\delta_{p,max} \leftarrow 0$                         ▷ Maximum penetration distance
 7:     for $i = 1$ to $N_o$ do
 8:         for $j = 1$ to $N_o$ do
 9:             if $j \neq i$ then
10:                 for $k = 1$ to $N_{v,i}$ do
11:                     $\delta_p \leftarrow \delta_p(\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{x}_{i,k}, \mathbf{s}_j)$
12:                     $V \leftarrow V + \delta_p d^3$
13:                     if $\delta_p > \delta_{p,max}$ then
14:                         $\delta_{p,max} \leftarrow \delta_p$
15:                     end if
16:                 end for
```

Algorithm 1 Repulsion algorithm (continued)

```
17:             end if
18:         end for
19:     end for
20:     if $\delta_{p,max} > \delta_{p,eps}$ then
21:         for $i = 1$ to $N_o$ do
22:             $\mathbf{t}_{xy,i} \leftarrow \mathbf{t}_{xy,i} - (1+\beta)\epsilon\nabla_{\mathbf{t}_{xy,i}} V$
23:             $\theta_{z,i} \leftarrow \theta_{z,i} - (1-\beta)\epsilon\nabla_{\theta_{z,i}} V$
24:         end for
25:     else
26:         break
27:     end if
28: end while
```

3.3 Training

Since the repulsion module is designed to be trainable to find optimal balance between the repulsion of the objects' positions and orientations, balancing parameter β is multiplied in gradient descent procedure as line 22 and 23 in Algorithm 1. The balancing parameter β is initialized with 1e$-$10 to avoid numerical error. Following the termination of repulsion, the balancing parameter is updated by the loss function used in SQPD-Net [1]'s transformed pose prediction defined in Equation 2.5.9 which compares the ground-truth and adjusted after-action poses.

4 Experiments and Results

In this chapter, we provide datasets used to train and test the repulsion module, and experiments conducted to evaluate the performance of the repulsion module. Section 4.1 describes the simulation dataset and how to collect the real-world dataset, Section 4.2 describes a evaluation metrics used for dynamics accuracy, and Section 4.3 describes a metric used for evaluating penetration between objects. Section 4.4 explains the baseline used to compare the performance, and Section 4.5 explains the experimental results.

4.1 Datasets

4.1.1 Simulation dataset

The simulation dataset in SQPD-Net [1] is used to train the repulsion module, with training/validation/test data consisting of 12000, 1200, 1200 scenes. It is about how maximum four objects which consist of cylinder and boxes move according to the robot's push action. Each split is divided into four equal sets from one to four objects. Since we consider repulsion between objects, 300 test data when the number of objects is 1 are excluded, so total 900 test data are used to evaluation. One data is comprised with visual observation, e.g., point cloud before and after action, and action expressed in its starting point and direction. Traveling distance is fixed as 10cm in every actions. Additionally, to train the recognition and pose prediction modules in SQPD-Net [1], full point cloud of objects and ground-truth superquadric parameters and poses before action of objects are included.

4.1.2 Real-world dataset

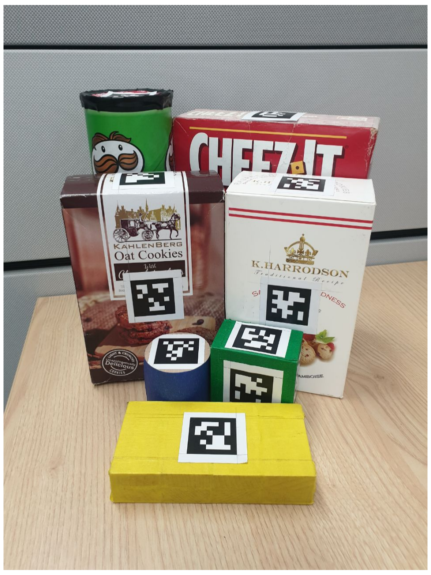
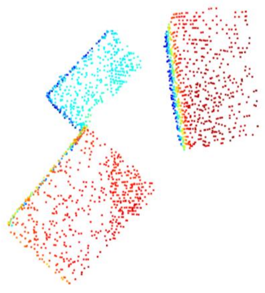
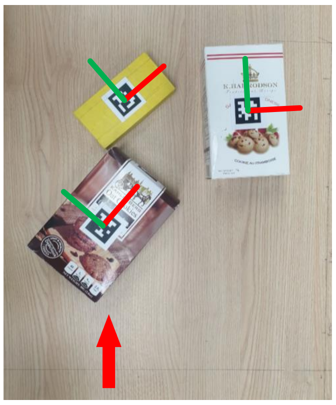


Figure 4.1: Objects used in real-world dataset.

A real-world test dataset is created to test the transfer of the repulsion module trained on simulation to real-world. Seven objects which consist of five boxes and two cylinders as shown in Figure 4.1 are used, and minimum two and maximum


(a) Point cloud data as visual observation.


(b) Push action and object poses before action.

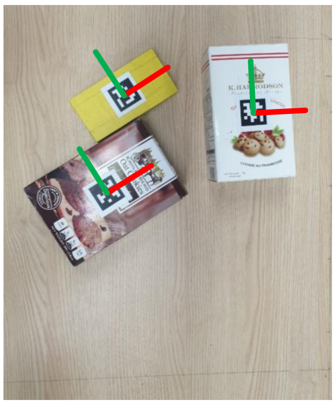

(c) Object poses after action.

Figure 4.2: Saved data in real-world dataset.

four objects are chosen at random and placed on the table randomly. Visual observation before action is saved as point cloud data as shown in Figure 4.2a. Red means that the point has a high z value, and blue means a low z value. Meanwhile, as same with when simulation data is generated, the robot randomly selects one object and randomly chooses an action direction among predefined octagonal directions. The starting point of the action is determined by providing an offset in the chosen direction from the center of the selected object. The amount of offset is determined by randomly selecting one of $\{0, 2, 4, 6\}$cm and adding the minimum of xy size parameters in object's superquadric parameters. For the selected action, if there is no collision after checking whether the robot collides with objects, the action is executed. As same with the simulation datasets, robot's traveling distance is 10 cm in every actions. To detect object poses before- and after-action, AprilTag [20] is used. Figure 4.2b and 4.2c show the detected object poses before and after action. Total 100 data are generated.

4.2 Evaluation Metrics for dynamics accuracy

To determine how well SQPD-Net [1] predicts the transformed poses of objects after the action, evaluation metrics for dynamics accuracy are required. SQPD-Net [1] is divided into R-SQPD-Net and GT-SQPD-Net according to the presence or absence of a pretrained segmentation and recognition module as explained in Section 2.2. Our repulsion module is tested with both types, and different evaluation metrics measuring dynamics accuracy are used for each type.

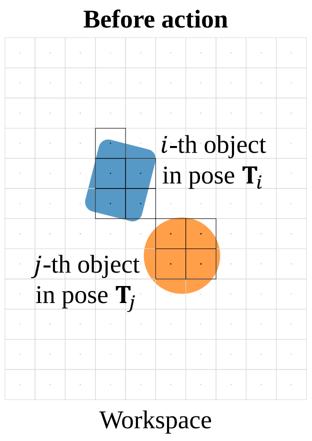
4.2.1 Position and orientation error

The GT-SQPD-Net does not recognize object shapes and poses but directly predicts the objects' transformed poses, so position and orientation error between ground-truth and prediction of transformed pose are used. Position error measures the difference between the ground-truth and predicted xy positions in Euclidean distance, and orientation error measures the difference of z-axis rotation angles in degree.

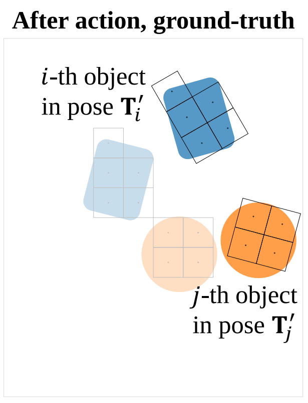
4.2.2 Flow error

In the R-SQPD-Net, if the segmentation of a scene point cloud is invalid, recognized objects may differ not be matched with ground-truth objects. Since it is impossible to compare ground-truth poses with predictions which are not matched in this case, flow error is employed instead. Following voxelizing the workspace, the flow error measures the difference between the ground-truth and predicted movement, e.g., flows of voxels inside objects in Euclidean distance. Figure 4.3 shows procedures to measure flow error in a top-down view.

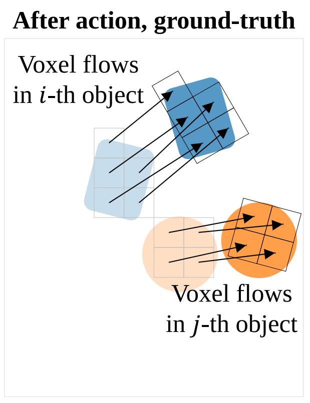
(i) Figure 4.3a: Following voxelizing the workspace, acquire voxels whose center



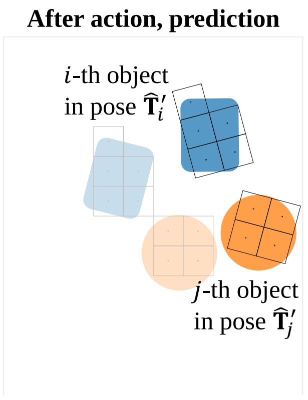
(a) Voxelization and acquisition of inside-object voxels at before-action.



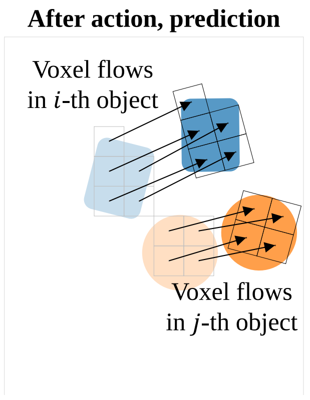
(b) Ground-truth voxel positions after action.



(c) Ground-truth voxel flows.


After action, prediction
i-th object
in pose $\widehat{\mathbf{T}}_i'$
j-th object
in pose $\widehat{\mathbf{T}}_j'$

(d) Predicted voxel positions after action.



(e) Predicted voxel flows.

Figure 4.3: Ground-truth and predicted voxel flows.

points are inside objects before action and note their center positions.

(ii) Figure 4.3b: Obtain ground-truth positions of the voxels after action using the ground-truth object poses after action.

(iii) Figure 4.3c: Obtain ground-truth flows of the voxels by subtracting the ground-truth positions after action from the positions before action.

(iv) Figure 4.3d, 4.3e: Obtain predicted positions and flows of the voxels in the same way with the ground-truth.

(v) Measure the errors between the ground-truth and predicted voxel flows in Euclidean distance, and average them to obtain the flow error.

4.3 Evaluation Metric for penetration

For measuring the amount of penetration between objects, the ratio of penetrated volume is used. For each object, it is defined as the ratio of penetrated volume per object volume, and every objects' penetration ratios are averaged to penetration ratio of the entire scene. Voxelization is used in practice to compute the volumes numerically. Thus, the penetration ratio for each object is numerically defined as the ratio of the number of penetrated voxels per the number of voxels inside object, and those of every objects are averaged.

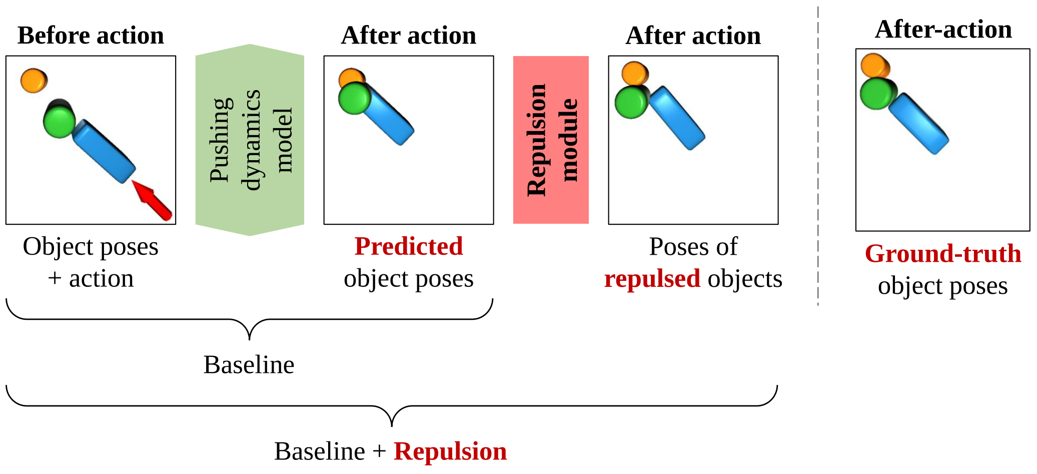


Figure 4.4: The overall structure of the baseline and repulsion module.

4.4 Baseline

Looking back at the overall structure before discussing the experimental results, the pushing dynamics model only is used as baseline. Therefore, as shown in Figure 4.4, performance of our repulsion module is measured by comparing the predicted object poses produced by the baseline and the repulsed objects' poses produced by adding our repulsion module to the baseline with the ground-truth object poses.

4.5 Experimental results

4.5.1 Results with GT-SQPD-Net on simulation dataset

Table 4.1 shows the evaluation metrics computed on simulation dataset with the GT-SQPD-Net. In every object quantities, the penetration ratio is significantly decreased when GT-SQPD-Net is combined with the repulsion module, and it shows

Objects #	**METHOD**	Position error (cm)	Orientation error (°)	Penetration ratio (%)
2	w/o repulsion	0.607	4.136	1.090
	with repulsion	**0.599**	**4.127**	**0.215**
3	w/o repulsion	0.499	**3.173**	1.306
	with repulsion	**0.489**	3.193	**0.193**
4	w/o repulsion	0.585	3.052	2.206
	with repulsion	**0.563**	3.052	**0.303**
Total	w/o repulsion	0.563	**3.453**	1.534
	with repulsion	**0.550**	3.457	**0.237**

Table 4.1: Evaluation metrics of the GT-SQPD-Net and repulsion module computed on simulation dataset.

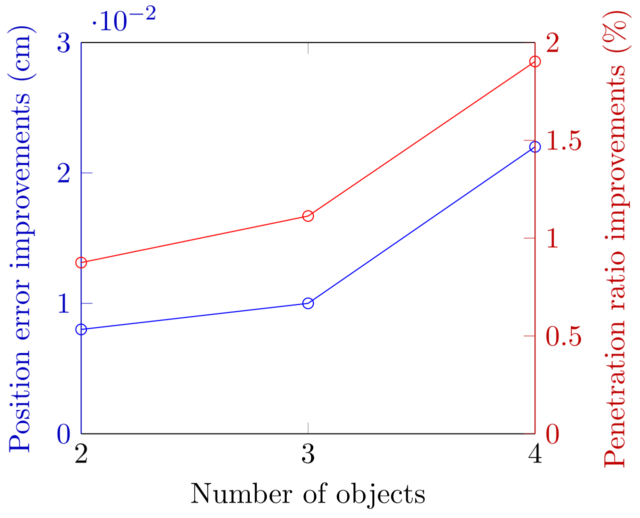


Figure 4.5: Improvement of the position error and penetration ratio according to the number of objects in GT-SQPD-Net.

that our repulsion module solves the penetration problem and GT-SQPD-Net with the repulsion module avoids penetration in predicting pushing dynamics. The position error is decreased in every object quantities as well when GT-SQPD-Net is combined with the repulsion module, and the differences in the orientation error are negligible. Thus, it shows that avoiding penetration by adding the repulsion module improves dynamics accuracy of GT-SQPD-Net. Furthermore, as shown in Figure 4.5, when the number of objects is increased, the position error and penetration ratio is improved when penetration is avoided, so it is critical to use the repulsion module when manipulated objects are numerous. Figure 4.6 shows the experimental results based on the number of objects. In every object quantities, the object poses are closer to the ground-truth if the repulsion is combined with the GT-SQPD-Net's prediction.

4.5.2 Results with R-SQPD-Net on simulation dataset

Table 4.2 shows experimental results with the R-SQPD-Net tested on the simulation dataset. The penetration ratio is considerably decreased when combined with the repulsion module, but the flow errors, which represent the dynamics accuracy, are increased negligibly for all object quantities. It is caused by the failure of R-SQPD-Net segmentation, not by the repulsion module.

Figure 4.7 shows how the segmentation influenced the repulsion and dynamics accuracy by altering success of the segmentation. As shown in the fourth row, if point cloud segmentation is failed in R-SQPD-Net, recognized objects before action are already penetrated. As the points from the red cylinder are segmented into red, yellow and blue objects, the three objects are incorrectly recognized. Furthermore, the red object is concealed by the yellow object. These misrecognized objects are already penetrated each other before predicting transformed poses. When

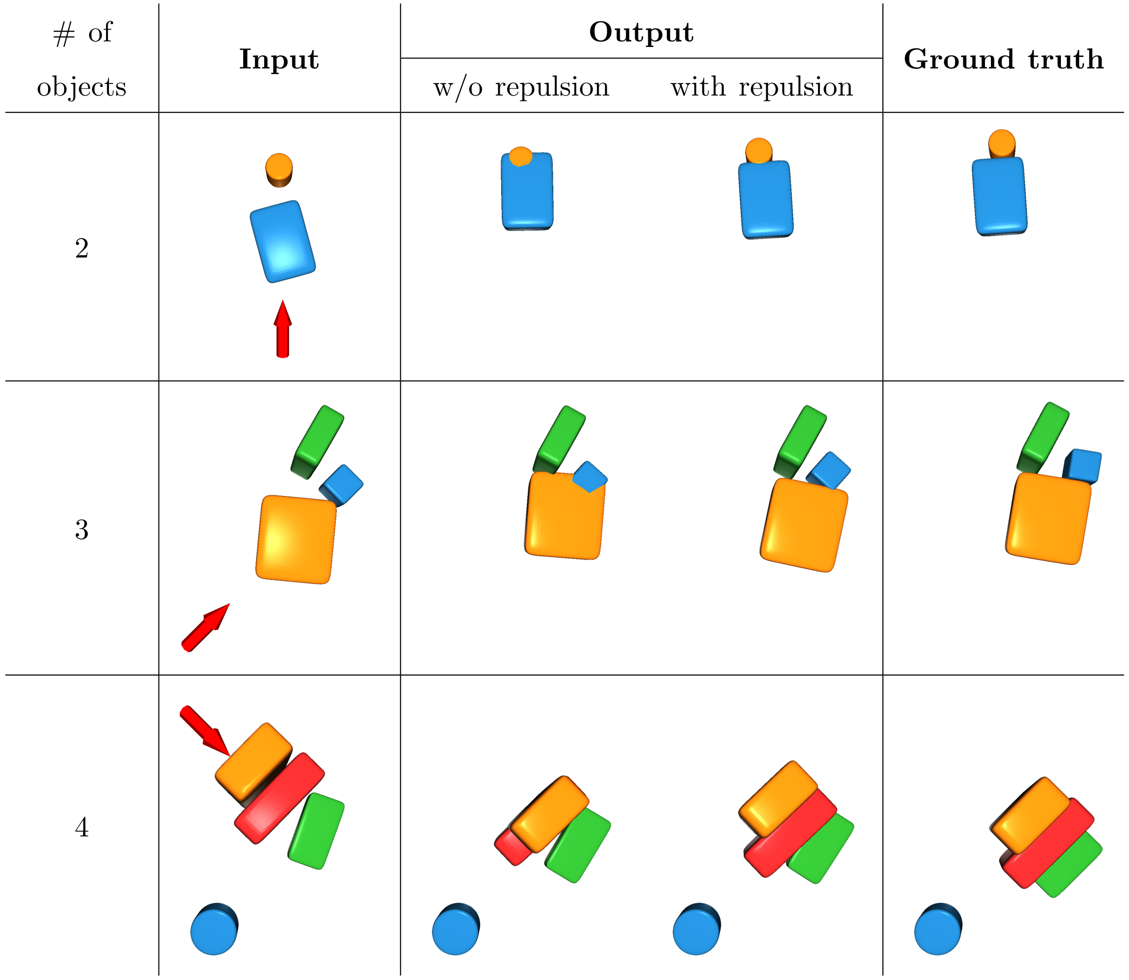


Figure 4.6: Object pose predictions of the GT-SQPD-net and repulsion module for simulation dataset.

# of objects	**METHOD**	Flow error (cm)	Penetration ratio (%)
2	w/o repulsion	**0.712**	3.228
	with repulsion	0.740	**0.569**
3	w/o repulsion	**0.610**	4.467
	with repulsion	0.679	**0.494**
4	w/o repulsion	**0.805**	5.730
	with repulsion	0.895	**0.536**
Total	w/o repulsion	**0.709**	4.475
	with repulsion	0.771	**0.533**

Table 4.2: Evaluation metrics of the R-SQPD-Net and repulsion module computed on simulation dataset.

predicting the transformed poses without the repulsion, poses that produce voxel flows close to the ground truth are somewhat predicted, but adding the repulsion induces the prediction to differ from the ground truth while avoiding penetration between objects. As a results, the repulsion module helps avoid the penetration, but decreases dynamics accuracy when point cloud segmentation and object recognition was already failed. If ground truth segmentation labels are used in the failure case as shown in the fifth row of Figure 4.7, R-SQPD-Net recognizes objects correctly and the repulsion makes the prediction of transformed pose close to the ground truth. If R-SQPD-Net segmented point cloud properly and recognized objects correctly as shown in the third row of Figure 4.7, the repulsion module improves dynamics accuracy by bringing the prediction close to the ground truth.

Table 4.3 shows experimental results when ground-truth segmentation labels

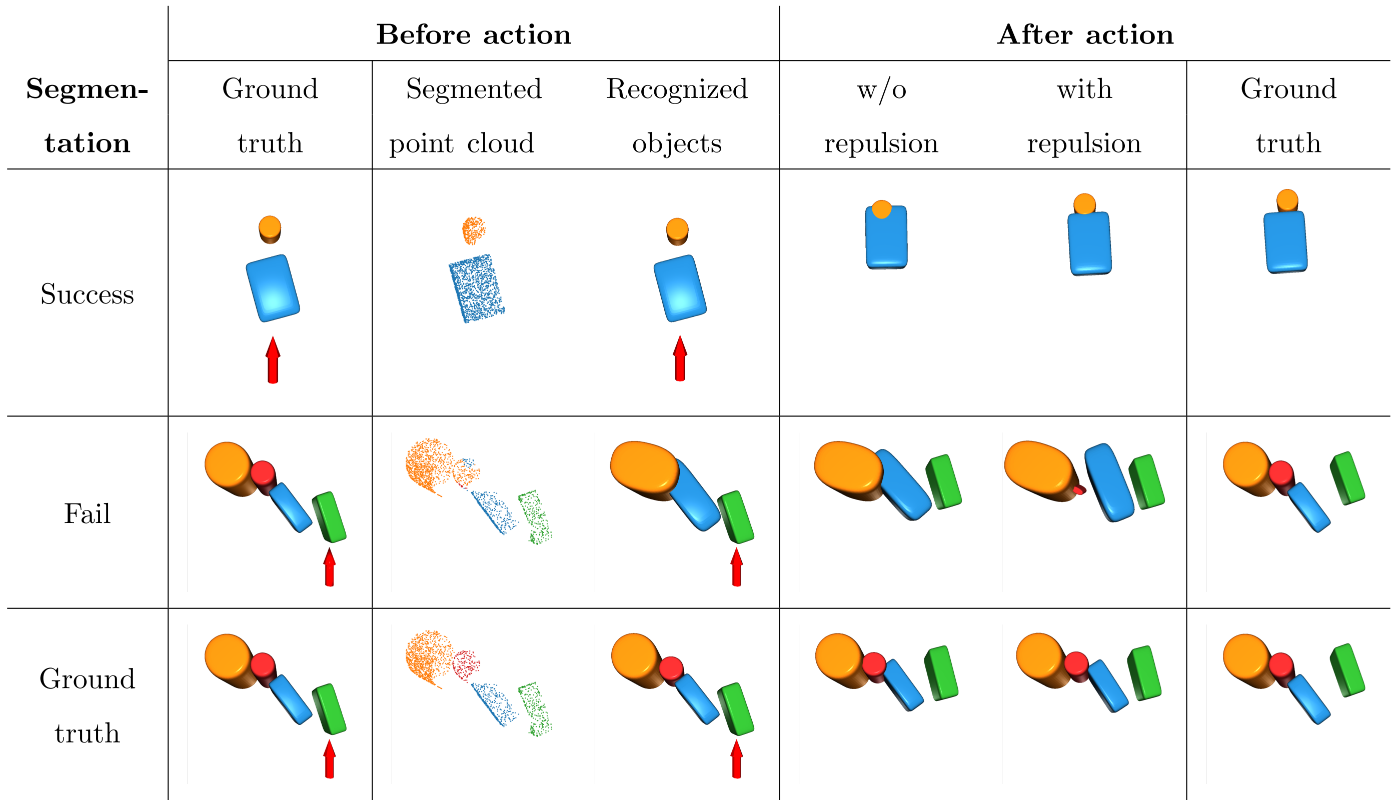


Figure 4.7: Point cloud segmentation, object recognition of the R-SQPD-Net and transformed pose predictions with and without repulsion module for simulation dataset.

is used in R-SQPD-Net. For all data, the flow error is decreased as the penetration ratio is decreased, so the dynamics accuracy is improved as the penetration is avoided, thanks to the repulsion.

# of objects	**METHOD**	Flow error (cm)	Penetration ratio (%)
2	w/o repulsion	0.710	1.301
	with repulsion	0.710	**0.267**
3	w/o repulsion	0.575	1.511
	with repulsion	**0.569**	**0.256**
4	w/o repulsion	**0.683**	2.406
	with repulsion	0.687	**0.365**
Total	w/o repulsion	0.656	1.739
	with repulsion	**0.655**	**0.296**

Table 4.3: Evaluation metrics of the R-SQPD-Net with ground-truth segmentation labels and repulsion module computed on simulation dataset.

4.5.3 Results with GT-SQPD-Net on real-world dataset

Table 4.4 shows experimental results tested on real-world dataset with the GT-SQPD-Net. The cases where the number of objects is two is excluded because there is no penetration in objects on the poses predicted by the GT-SQPD-Net. Although the dynamics accuracy is slightly decreased due to the issue of transferring between different domains, it demonstrates that deploying the dynamics model and repulsion module learned in simulation to real-world is feasible.

As same with the results on the simulation dataset, combining the GT-SQPD-Net with the repulsion module significantly decreases the penetration ratio across

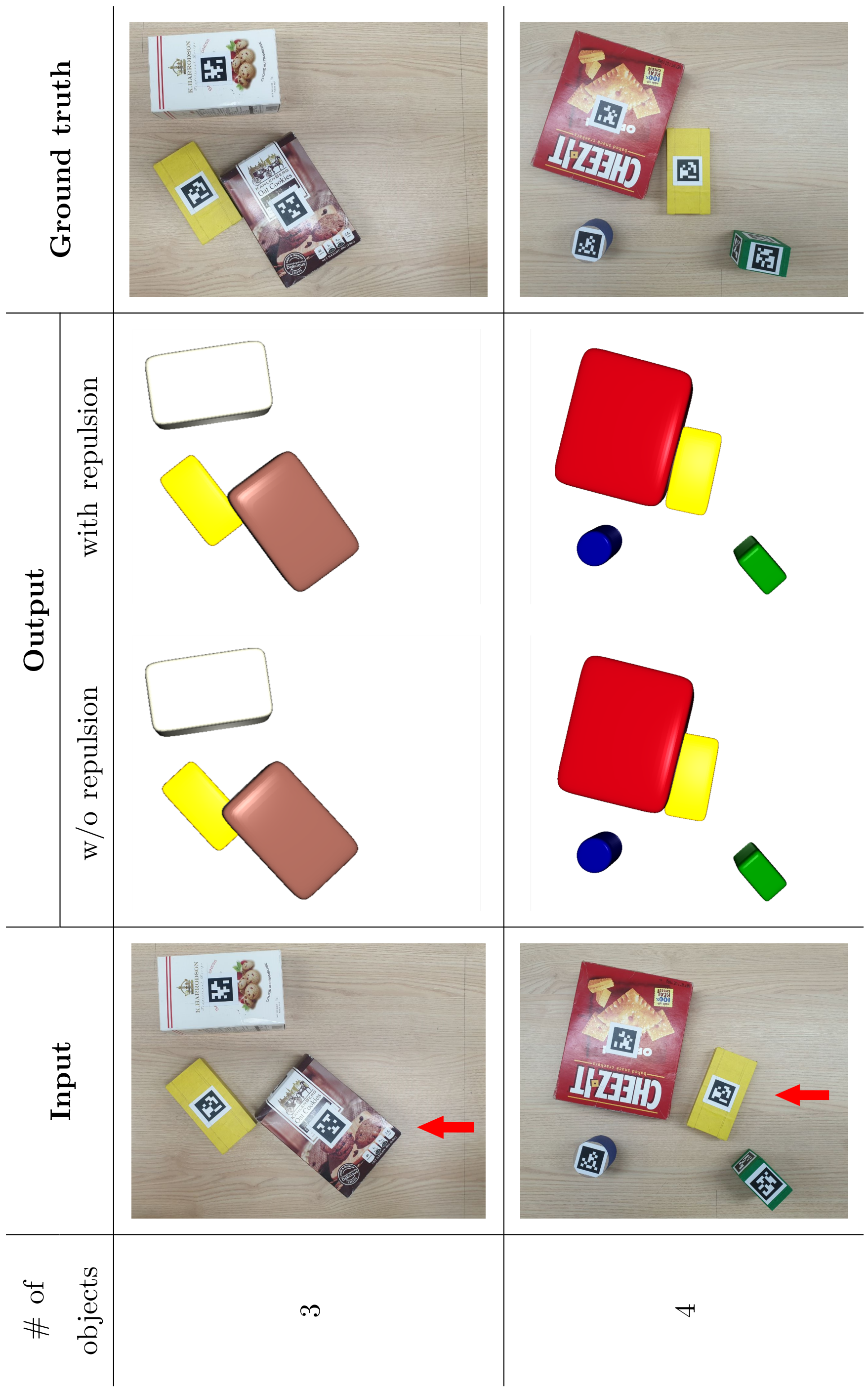


Figure 4.8: Object pose predictions of the GT-SQPD-net and repulsion module for real-world dataset.

# of objects	**METHOD**	Position error (cm)	Orientation error (°)	Penetration ratio (%)
3	w/o repulsion	0.902	3.277	0.248
	with repulsion	**0.900**	**3.106**	**0.034**
4	w/o repulsion	0.458	**2.328**	0.163
	with repulsion	**0.455**	2.340	**0.023**
Total	w/o repulsion	0.680	2.803	0.205
	with repulsion	**0.677**	**2.723**	**0.028**

Table 4.4: Evaluation metrics of the GT-SQPD-Net and repulsion module computed on real-world dataset.

every object quantities in real-world dataset. Furthermore, when GT-SQPD-Net is combined with the repulsion module, the position error is decreased in every object quantities, and the orientation error is generally decreased. As a result, even in real-world, adding the repulsion module to prevent the penetration improves the dynamics accuracy of GT-SQPD-Net.

The experimental results based on the number of objects are shown in Figure 4.8. When the repulsion is combined with the GT-SQPD-Net's prediction, the object poses are closer to the ground-truth in every object quantities.

4.5.4 Results with R-SQPD-Net on real-world dataset

The experimental results with the R-SQPD-Net tested on the real-world dataset are shows in Table 4.5. When the repulsion module is combined with the R-SQPD-Net, the penetration ratio is decreased but the flow errors are increased as same with the results, and the decrease in the ratio and the increase in the errors is

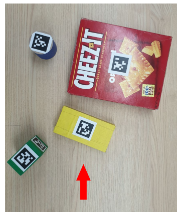
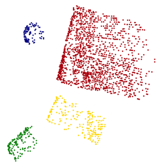
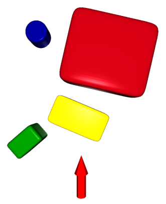
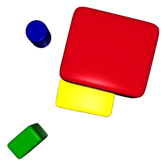
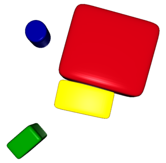
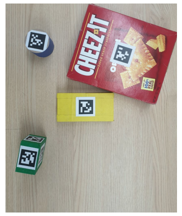
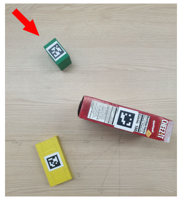
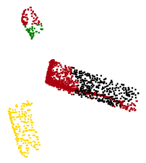
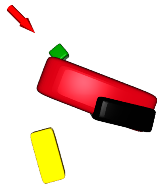
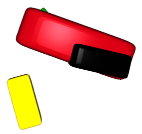
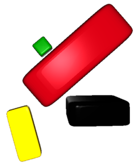
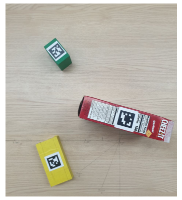
	Before action			After action		
Segmen-tation	Ground truth	Segmented point cloud	Recognized objects	w/o repulsion	with repulsion	Ground truth
Success						
Fail						

Figure 4.9: Point cloud segmentation, object recognition of the R-SQPD-Net and transformed pose predictions with and without repulsion module for real-world dataset.

# of objects	**METHOD**	Flow error (cm)	Penetration ratio (%)
3	w/o repulsion	**1.101**	11.708
	with repulsion	1.643	**0.609**
4	w/o repulsion	**0.591**	5.937
	with repulsion	0.919	**0.075**
Total	w/o repulsion	**0.846**	8.823
	with repulsion	1.281	**0.342**

Table 4.5: Evaluation metrics of the R-SQPD-Net and repulsion module computed on real-world dataset.

greater than in the simulation dataset.

Figure 4.9 shows that the degradation of dynamics accuracy is caused by the failure of R-SQPD-Net segmentation, as in the simulation dataset. As shown in the fourth row where segmentation fails, the points from the green box and the red box are segmented into green and red, red and black, respectively. The green and red objects, therefore, are incorrectly recognized and new black object is created. These objects are penetrated before the SQPD-Net predicts pose transformation, so objects in the predicted poses from the SQPD-Net are penetrated as well. In this case, the penetration avoidance of the repulsion module rather harms the dynamics accuracy. Since the real-world dataset has no ground-truth segmentation labels, the experiments using the ground-truth labels are not included. The repulsion module improves dynamics accuracy by moving SQPD-Net's prediction as close to the ground truth if R-SQPD-Net correctly segmented the point cloud and correctly recognized the objects, as shown in the third row of Figure 4.9.

5 Conclusion and Future Works

5.1 Conclusion

We introduce a repulsion module to avoid penetration between objects in poses predicted by pushing dynamics model, to realize learning realistic dynamics and improve dynamics accuracy. We conduct experiments to observe how the repulsion affects to the pushing dynamics model. The experiments and results show that the repulsion module successfully assists GT-SQPD-Net in avoiding penetration pushing dynamics learning. In addition to resolving the penetration problem, the repulsion module improves the dynamics accuracy of GT-SQPD-Net prediction.

However, when combined with R-SQPD-Net, the repulsion module solves penetration, but at the expense of dynamics accuracy. This issue is caused by a failure in the segmentation of the point cloud in R-SQPD-Net. Since point cloud segmentation fails in R-SQPD-Net, objects are incorrectly recognized, and the objects are already penetrated before action, as are the objects in predicted poses from SQPD-Net. In this case, the repulsion module's penetration avoidance degrades the dynamics accuracy. It is confirmed that the repulsion module can improve the dynamics accuracy if the point cloud segmentation is solved by demonstrating that the repulsion module improves the dynamics accuracy when ground-truth segmentation labels are used.

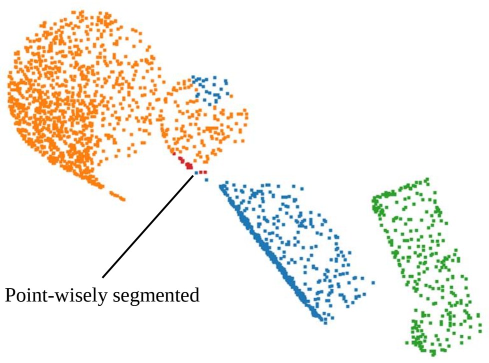
When using the pushing dynamics model in the real world, the ground-truth information about objects is not directly accessible, so taking and segmenting point clouds, and recognizing the objects are required. Thus, the R-SQPD-Net should be used in real world, and in order to use the repulsion module with the R-SQPD-Net, the segmentation performance must be improved.

5.2 Future works

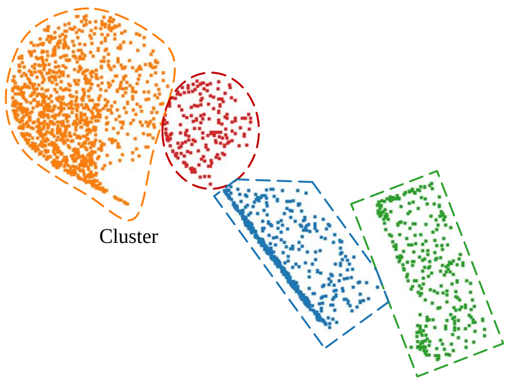
5.2.1 Point cloud segmentation

R-SQPD-Net's segmentation algorithm is conducted point-wisely as shown in Figure 5.1a, which is vulnerable to spread the segmentation labels between objects. However, since the object point clouds are generated on the observed surface of the objects, they have a characteristic to be clustered object-wisely. Therefore, a clustering-based segmentation method would better segment the object point cloud than the point-wise segmentation method as shown in Figure 5.1b.

FPCC [21] was constructed for instance segmentation of bin-picking scene, and its algorithm would help our object segmentation task by clustering method. A quick clustering algorithm is part of FPCC, along with the FPCC-Net network. The FPCC-Net includes a module that extracts features of each point and a module that infers the geometric center of each cluster. The clustering algorithm then clusters the points to the closest geometric center in feature space, and brings the



(a) Point-wisely segmented point cloud.



(b) Point cloud segmentation by clustering.

Figure 5.1: A comparison between point-wisely segmented and clustered point cloud.

clusters up to input space. Since the segmentation is conducted based on the clustering algorithm, the segmentation labels can be prevented from spreading between objects, and the segmentation performance can be improved when FPCC is used. The improved segmentation provided by FPCC would help the repulsion module in improving dynamics accuracy by avoiding penetration.

5.2.2 Pushing manipulation task

On the other hand, pushing manipulation tasks are required to be preformed further, which is helped by the improved dynamics provided by the repulsion module. Multi-step dynamics prediction would be one example of such manipulation tasks. Existing pushing dynamics models usually compare their performance on predicting single-step dynamics, not multi-step dynamics, where the previous prediction is used as the input for the next step.

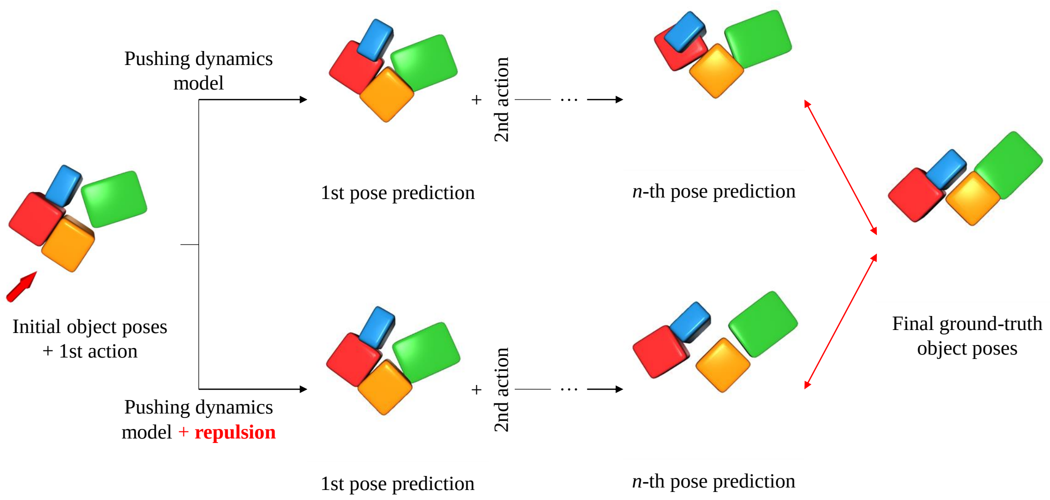


Figure 5.2: The multi-step dynamics prediction of the baseline and repulsion module.

Since the pushing dynamics model is trained to predict dynamics by using non-penetrated objects, if objects in the previous prediction are penetrated in multi-step dynamics prediction, the model, which uses the previous prediction as input and predicts dynamics in the next step, encounters an outlier that was not encountered during training. This situation degrades the performance of multi-step dynamics prediction because the pushing dynamics models, which are primarily made up of neural networks, do not predict correctly in outliers. Furthermore, the further the model predicts, the more errors accumulate and the greater the difference between the final predicted and ground-truth poses. This outlier problem can be resolved if repulsion is combined, because the repulsion module separates the penetrated objects in the previous prediction, and makes the scene inlier for the pushing dynamics models. Therefore, the repulsion module can help the multi-step dynamics prediction by maintaining the dynamics accuracy of the pushing dynamics models by making the outliers inlier through avoiding penetration. An experiments to perform the multi-step dynamics prediction should be performed further.

Bibliography

[1] Seungyeon Kim, Byeongdo Lim, Yonghyeon Lee, and Frank C Park. Se (2)-equivariant pushing dynamics models for tabletop object manipulations. In *6th Annual Conference on Robot Learning*.

[2] Michael Danielczuk, Jeffrey Mahler, Chris Correa, and Ken Goldberg. Linear push policies to increase grasp access for robot bin picking. In *2018 IEEE 14th international conference on automation science and engineering (CASE)*, pages 1249–1256. IEEE, 2018.

[3] Kechun Xu, Hongxiang Yu, Qianen Lai, Yue Wang, and Rong Xiong. Efficient learning of goal-oriented push-grasping synergy in clutter. *IEEE Robotics and Automation Letters*, 6(4):6337–6344, 2021.

[4] Marios Kiatos, Iason Sarantopoulos, Leonidas Koutras, Sotiris Malassiotis, and Zoe Doulgeri. Learning push-grasping in dense clutter. *IEEE Robotics and Automation Letters*, 7(4):8783–8790, 2022.

[5] Baichuan Huang, Shuai D Han, Jingjin Yu, and Abdeslam Boularias. Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement. *IEEE Robotics and Automation Letters*, 7(1):231–238, 2021.

[6] Haoran Song, Joshua A Haustein, Weihao Yuan, Kaiyu Hang, Michael Yu Wang, Danica Kragic, and Johannes A Stork. Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9433–9440. IEEE, 2020.

[7] Weihao Yuan, Kaiyu Hang, Danica Kragic, Michael Y Wang, and Johannes A Stork. End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer. *Robotics and Autonomous Systems*, 119:119–134, 2019.

[8] Yang Yang, Hengyue Liang, and Changhyun Choi. A deep learning approach to grasping the invisible. *IEEE Robotics and Automation Letters*, 5(2):2232–2239, 2020.

[9] Michael Danielczuk, Anelia Angelova, Vincent Vanhoucke, and Ken Goldberg. X-ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9577–9584. IEEE, 2020.

[10] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[11] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[12] Kevin M Lynch. The mechanics of fine manipulation by pushing. In *ICRA*, pages 2269–2276. Citeseer, 1992.

[13] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017.

[14] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor control. In *2018 IEEE*

International Conference on Robotics and Automation (ICRA), pages 3339–3346. IEEE, 2018.

[15] Zhenjia Xu, Zhanpeng He, Jiajun Wu, and Shuran Song. Learning 3d dynamic scene representations for robot manipulation. *arXiv preprint arXiv:2011.01968*, 2020.

[16] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019.

[17] Ari D Gross and Terrance E Boult. Error of fit measures for recovering parametric solids. In *ICCV*, 1988.

[18] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[19] Seungyeon Kim, Taegyun Ahn, Yonghyeon Lee, Jihwan Kim, Michael Yu Wang, and Frank C Park. Dsqnet: A deformable model-based supervised learning algorithm for grasping unknown occluded objects. *IEEE Transactions on Automation Science and Engineering*, 2022.

[20] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

[21] Yajun Xu, Shogo Arai, Diyi Liu, Fangzhou Lin, and Kazuhiro Kosuge. Fpcc: Fast point cloud clustering for instance segmentation. *arXiv e-prints*, pages arXiv–2012, 2020.

국문초록

로봇이 물체를 밀어서 조작하는 상황에서는, 로봇의 밀기 동작으로 인해 조작하는 물체들의 자세가 어떻게 변하는지 정확하게 예측하는 푸싱 역학 모델을 만드는 것이 중요하다. 본 논문에서, 우리는 그러한 역학 모델이 실제 물리학의 법칙을 알고있다면 더 정확한 예측을 할 수 있을 것이라 주장한다. 실제 세계에서 물체는 서로 관통하지 않는다는 물리적 법칙이 있지만, 기존의 푸싱 역학 모델들을 활용할 경우 예측된 자세에서 물체들의 관통을 방지할 수 없었다. 본 논문에서는 푸싱 역학 모델이 예측한 물체들의 자세를 입력으로 받아, 예측된 자세에서 물체들 간에 관통이 없도록 예측값을 조정하는 반발 모듈을 고안했다. 시뮬레이션 데이터를 통해 해당 반발 모듈을 훈련했고, 동일한 시뮬레이션 데이터 및 실제 로봇 동작을 통해 만든 실제 세계의 데이터로 해당 알고리즘의 성능을 테스트했다. 두 데이터가 보여주는 결과에 따르면, 우리의 반발 모듈이 물체 간의 관통을 효과적으로 방지하고 학습된 푸싱 역학의 정확도를 높일 수 있음을 보였다.

주요어: 푸싱 역학 학습, 관통 방지

학번: 2021-28590