Object-Level Dynamic Scene Reconstruction With Physical Plausibility From RGB-D Images

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Michael Felix Strecke
aus Engen

Tübingen
2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	11.10.2023
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Jörg-Dieter Stückler
2. Berichterstatter:	Prof. Dr. Andreas Geiger

Essentially, all models are wrong, but some are useful.

– George E. P. Box

Abstract

Humans have the remarkable ability to perceive and interact with objects in the world around them. They can easily segment objects from visual data and have an *intuitive* understanding of how physics influences objects. By contrast, robots are so far often constrained to tailored environments for a specific task, due to their inability to reconstruct a versatile and accurate scene representation. In this thesis, we combine RGB-D video data with background knowledge of real-world physics to develop such a representation for robots.

Our contributions can be separated into two main parts: a dynamic object tracking tool and optimization frameworks that allow for improving shape reconstructions based on physical plausibility. The dynamic object tracking tool "EM-Fusion" detects, segments, reconstructs, and tracks objects from RGB-D video data. We propose a probabilistic data association approach for attributing the image pixels to the different moving objects in the scene. This allows us to track and reconstruct moving objects and the background scene with state-of-the art accuracy and robustness towards occlusions.

We investigate two ways of further optimizing the reconstructed shapes of moving objects based on physical plausibility. The first of these, "Co-Section", includes physical plausibility by reasoning about the empty space around an object. We observe that no two objects can occupy the same space at the same time and that the depth images in the input video provide an estimate of observed empty space. Based on these observations, we propose intersection and hull constraints, which we combine with the observed surfaces in a global optimization approach. Compared to EM-Fusion, which only reconstructs the observed surface, Co-Section optimizes watertight shapes. These watertight shapes provide a rough estimate of unseen surfaces and could be useful as initialization for further refinement, *e.g.*, by interactive perception. In the second optimization approach, "DiffSDFSim", we reason about object shapes based on physically plausible object motion. We observe that object trajectories after collisions depend on the object's shape, and extend a differentiable physics simulation for optimizing object shapes together with other physical properties (*e.g.*, forces, masses, friction) based on the motion of the objects and their interactions. Our key contributions are using signed distance function models for representing shapes and a novel method for computing gradients that models the dependency of the time of contact on object shapes. We demonstrate that our approach recovers target shapes well by fitting to target trajectories and depth observations. Further, the ground-truth trajectories are recovered well in simulation using the resulting shape and physical properties. This enables predictions about the future motion of objects by physical simulation.

We anticipate that our contributions can be useful building blocks in the development of 3D environment perception for robots. The reconstruction of individual objects as in EM-Fusion is a key ingredient required for interactions with objects. Completed shapes as the ones provided by Co-Section provide useful cues for planning interactions like grasping of objects. Finally, the recovery of shape and other physical parameters using differentiable simulation as in DiffSDFSim allows simulating objects and thus predicting the effects of interactions. Future work might extend the presented works for interactive perception of dynamic environments by comparing these predictions with observed real-world interactions to further improve the reconstructions and physical parameter estimations.

Zusammenfassung

Menschen haben die bemerkenswerte Fähigkeit, Objekte in ihrer Umgebung wahrzunehmen und mit ihnen zu interagieren. Sie können ohne Anstrengung Objekte in visuellen Daten segmentieren und haben ein *intuitives* Verständnis davon, wie die Physik Objekte beeinflusst. Im Gegensatz dazu sind Roboter bisher oft auf aufgabenspezifisch zugeschnittene Umgebungen begrenzt, da sie keine vielseitige und genaue Szenenrepräsentation rekonstruieren können. In dieser Dissertation kombinieren wir RGB-D Videodaten mit Hintergrundwissen über die Physik der realen Welt, um eine solche Repräsentation für Roboter zu entwickeln.

Unsere Beiträge bestehen aus zwei Hauptbestandteilen: ein Werkzeug zur Verfolgung bewegter Objekte, und Optimierungsansätze, welche Rekonstruktionen der 3D Form basierend auf physikalischer Plausibilität verbessern können. Das Werkzeug zur Verfolgung bewegter Objekte, "EM-Fusion", detektiert, segmentiert, rekonstruiert und verfolgt Objekte in RGB-D Videodaten. Wir schlagen einen probabilistischen Datenassoziationsansatz vor, um die Pixel im Bild den unterschiedlichen bewegten Objekten in der Szene zuzuordnen. Dieser erlaubt uns, die Objekte und die Hintergrundszene mit Genauigkeit nach dem Stand der Technik und Robustheit gegenüber Verdeckungen zu verfolgen und zu rekonstruieren.

Wir erforschen zwei Ansätze, die rekonstruierte Form von bewegten Objekten basierend auf physikalischer Plausibilität weiter zu optimieren. Der erste hiervon, "Co-Section", schließt physikalische Plausibilität durch Argumentationen über den leeren Raum um ein Objekt herum ein. Wir stellen fest, dass keine zwei Objekte denselben Raum zur selben Zeit einnehmen können und dass die Tiefenbilder im Eingabevideo eine Schätzung von beobachtetem leeren Raum liefern. Basierend auf diesen Feststellungen schlagen wir Hüllen- und Überschneidungsbeschränkungen vor, welche wir mit den beobachteten Oberflächen in einem globalen Optimierungsansatz kombinieren. Verglichen mit EM-Fusion, welches nur die beobachteten Oberflächen rekonstruiert, optimiert Co-Section wasserdichte Objektformen. Diese wasserdichten Formen liefern eine grobe Schätzung über Oberflächen, die nicht direkt gesehen wurden, und können als Initialisierung für weitere Verbesserung, *z.B.* durch interaktive Wahrnehmung, dienen. Im zweiten Optimierungsansatz, "DiffSDFSim", argumentieren wir über Objektformen basierend auf physikalisch plausibler Objektbewegung. Wir stellen fest, dass Objekttrajektorien nach Kollisionen von der Objektform abhängen, und erweitern eine differenzierbare Physiksimulation, um die Objektformen gemeinsam mit anderen physikalischen Eigenschaften (*z.B.* Kräfte, Massen, Reibung) basierend auf der Bewegung der Objekte und ihrer Interaktionen zu optimieren. Unsere Hauptbeiträge sind die Verwendung vorzeichenbehafteter Distanzfunktionen zur Repräsentation von Objektformen, und eine neue Methode zur Berechnung von Gradienten, welche die Abhängigkeit des Kontaktzeitpunkts von der Objektform modelliert. Wir zeigen, dass unser Ansatz Referenzformen durch das Anpassen auf Referenztrajektorien und Tiefenmessungen gut rekonstruiert. Weiterhin werden die wahren Trajektorien in der Simulation mit den optimierten Formen und physikalischen Eigenschaften gut rekonstruiert, was Vorhersagen über zukünftige Bewegungen der Objekte durch die Physiksimulation zulässt.

Wir gehen davon aus, dass unsere Beiträge nützliche Bausteine in der Entwicklung von 3D Umgebungswahrnehmung für Roboter sein können. Die Rekonstruktion einzelner Objekte, wie in EM-Fusion, ist ein Schlüsselbaustein, welche für die Interaktion mit Objekten benötigt wird. Vervollständigte Formen, wie sie Co-Section bereitstellt, liefern nützliche Hinweise, um Interaktionen wie das Greifen von Objekten zu planen. Schließlich erlaubt die Schätzung von Form- und anderen physikalischen Parametern mittels differenzierbarer Physiksimulation, wie in DiffSDFSim, Objekte zu simulieren und damit die Effekte von Interaktionen vorherzusagen. Zukünftige Arbeiten könnten die präsentierten Ansätze zur interaktiven Wahrnehmung dynamischer Umgebungen erweitern, indem diese Vorhersagen mit beobachteten Interaktionen in der echten Welt verglichen werden, um die Rekonstruktionen und physikalischen Parameterschätzungen weiter zu verbessern.

Acknowledgments

First and foremost, I would like to thank my advisor Dr. Jörg Stückler for his guidance and support, and for providing the research environment without which this work would not have been possible. I would further like to express my gratitude to my thesis advisory committee Dr. Jörg Stückler, Prof. Dr. Andreas Geiger, and Michael J. Black, PhD, for taking the time for our yearly meetings and their insightful feedback. I want to thank my reviewers Dr. Jörg Stückler and Prof. Dr. Andreas Geiger for taking the time to review this thesis.

I want to further thank Haolong Li, Jan Achterhold, Jens Kreber, Markus Ring, Mikel Zhobro, and Rama Kandukuri for proofreading parts of this thesis.

I am grateful to all members of the Embodied Vision Group for interesting discussions about work and other topics. It was a pleasure working with you over the last years! I am especially thankful to Cathrin Elich, Jan Achterhold, Nathanael Bosch, and Vincent Stimper for many nice game nights and for being great friends.

As there is more to life than work, I would also like to thank all members of the *Musikverein Randegg e.V.*, the *Big Band of the University of Tübingen*, the *BrassBusters*, and *SuitUp!* for helping me to keep my work-life balance in check and providing the distractions that are sometimes necessary to recharge my creative batteries. Thank you especially to Tina Baumann, Fabian Roser, Jonas Malang, Manuel Herbst, and Theresa Störiko for many fun activities with and without music and for being great friends over the last years.

Finally, I would like to thank my parents and my siblings for their great support over the last years and in my life in general. Ihr seid die besten Eltern und Geschwister, die man sich vorstellen kann!

Michael Strecke
Tübingen, May 2023

Contents

List of Figures

List of Tables

Notation and Acronyms

Notation

Scalars	Regular lower case letters	$a, b, c, \alpha, \beta, \gamma$
Vectors	Bold lower case letters	$\mathbf{a}, \mathbf{b}, \mathbf{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$
Matrices	Bold upper case letters	$\mathbf{M}, \mathbf{A}, \boldsymbol{\Phi}$
Sets	Calligraphic upper case letters	$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{S}, \mathcal{K}$
Functions	Letters with parentheses for arguments	$f(\cdot), \phi(\cdot)$

Other symbols:

Number sets	$\mathbb{R}, \mathbb{N}, \mathbb{Z}$
Element i of vector $\mathbf{x}$	x_i
Element $\mathbf{x} = (i, j)^\top$ of matrix $\mathbf{A}$	$A_{\mathbf{x}} = A_{i,j}$
Vector or matrix transpose	$\mathbf{v}^\top, \mathbf{M}^\top$
Homogeneous coordinates of vector $\mathbf{x} \in \mathbb{R}^n$	$\bar{\mathbf{x}} = (x_1, \dots, x_n, 1)^\top \in \mathbb{R}^{n+1}$
Skew-symmetric matrix from vector $\mathbf{x}$	$\widehat{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^\wedge = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$
Twist matrix in $\mathbb{R}^{4\times4}$ for twist coordinate $\boldsymbol{\xi} \in \mathbb{R}^6$	$\widehat{\boldsymbol{\xi}} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix}^\wedge = \begin{pmatrix} \widehat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 1 \end{pmatrix}$
Gradient of a function f wrt. all parameters	∇f
Gradient of a function f wrt. parameter(s) $\mathbf{x}$	$\nabla_{\mathbf{x}} f$
L^p norm	$\lVert\cdot\rVert_p$
Euclidean norm (L^2 norm)	$\lVert\cdot\rVert_2, \lVert\cdot\rVert$

Fixed Symbols:

signed distance function (SDF), SDF measurement	$\phi, \check{\phi}$
Camera intrinsic matrix	$\mathbf{K}$
Projection function	π
Identity matrix in n dimensions	$\mathbb{I}_n$
Rotation matrix in SO(3), translation vector in $\mathbb{R}^3$	$\mathbf{R}, \mathbf{t}$
Transformation matrix in SE(3) with corresponding twist vector in $\mathfrak{se}(3)$	$\mathbf{T} = \mathbf{T}(\boldsymbol{\xi}) = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$
Angular velocity in $\mathfrak{so}(3)$ and linear velocity in $\mathbb{R}^3$	$\boldsymbol{\omega}, \mathbf{v}$
Velocity or twist coordinates in $\mathfrak{se}(3)$	$\boldsymbol{\xi} = (\boldsymbol{\omega}^\top, \mathbf{v}^\top)^\top$
Energy and loss functions	$E(\cdot)$
Vector/Matrix of all 0s or all 1s	$\mathbf{0}, \mathbf{1}$

Acronyms

AR	augmented reality
AT(E)	absolute trajectory (error)
CNN	convolutional neural network
CPU	central processing unit
EM	expectation maximization
FJ	Fast Jacobi
GAN	generative adversarial network
GPU	graphics processing unit
ICP	iterative closest points
IMLS	implicit moving least squares
IRLS	iteratively reweighted least squares
IMU	intertial measurement unit
LCP	linear complementarity problem
R-CNN	regions with CNN features
RGB	red, green, and blue (used for color images)
RGB-D	RGB and depth (color and depth)
RMSE	root mean squared error
RP(E)	relative pose (error)
SfM	structure from motion
SLAM	simultaneous localization and mapping
(T)SDF	(truncated) signed distance function
SSD	smooth signed distance surface reconstruction
VR	virtual reality

Introduction 1

We humans have the remarkable ability to perceive our environment from visual input. Our perception allows us to safely navigate through diverse environments, interact with arbitrary objects and estimate the motion of other agents or moving objects. While most animals have this ability, we should not underestimate that a large part of the brain in humans and primates is devoted to processing visual information[1]. To build autonomous agents (*e.g.*, household robots, self-driving cars) that can safely act in their environment, we need to equip them with similar perception capabilities.

1: Up to 50% for macaques and estimated 20-30% in humans (Sheth and Young 2016).

In this thesis, we focus on the topic of 3D scene reconstruction for autonomous agents, *i.e.*, implementing methods for perceiving the 3D world from images. As images are only 2D projections of the 3D world, this problem is generally ill-posed. However, if we consider *image sequences* (*i.e.*, *videos*) with a moving camera, we can observe changes of pixels that originate from the camera motion. A key effect we can observe is *motion parallax*, *i.e.*, that objects in the distance appear to move slower than those close to us[2]. This observation has inspired research in the computer vision community on 3D reconstruction from image sequences for many years (Hartley and Zisserman 2004; Szeliski 2011).

2: Imagine looking out of the window when riding a train. The power line posts close to the rails seem to move much faster than *e.g.*, trees in the distance.

(Hartley and Zisserman 2004): *Multiple View Geometry in Computer Vision*
(Szeliski 2011): *Computer Vision - Algorithms and Applications*

Most previous work on 3D reconstruction has made the assumption that the camera or the autonomous agent is the only thing moving in the scene. This simplifies the problem as all pixel changes can be attributed to the motion of the camera. However, most of the time, this assumption does not hold in the real world. The most interesting environments are *dynamic*, *i.e.*, changing over time, as people or other agents move and manipulate objects. As it is seemingly effortlessly possible for humans and animals to react to changes in the world around them, a useful autonomous agent should be able to do the same. The agent should be able to interact with people or other autonomous agents who move around themselves and manipulate objects. The core focus of this thesis thus lies in the reconstruction of *dynamic scenes*, *i.e.*, scenes with *moving objects*. In contrast to static scene reconstruction, the problem complexity now increases as pixel changes can be associated to several independent motions. To reconstruct dynamic scenes, we thus need to *segment objects*[3], so that we can associate different image pixels to different object motions.

3: *i.e.*, find out which pixel belongs to which object

With multiple objects in a scene, we further observe that not all object configurations are *physically plausible*. We know from experience that objects usually do not float in the air or that no two objects can occupy the same 3D space at the same time. If we see, *e.g.*, a cup standing on a table we would be surprised if it was not in contact with the table or penetrating the table. When objects are in motion, we further have an expectation about how *physical interactions*[4] between objects affect their motion trajectories. When seeing an object sliding on another surface or colliding with a wall, we can get some estimate about the material properties, giving us a coarse estimate of how it would feel to interact with the object. While this knowledge from experience about the physics of the real world is often not very accurate, we expect that *physical plausibility*

4: *e.g.*, collisions, friction

in general can help to improve the dynamic scene reconstruction. The second focus of this thesis is thus concerned with reasoning about the physical plausibility of dynamic scenes, which most previous approaches did not consider.

In summary, this thesis addresses the following three challenges, which humans are able to solve seemingly effortlessly:

1. Building a model of the 3D world from 2D images.
2. Segmenting objects, *i.e.*, telling which 2D points in the image correspond to different objects.
3. Incorporating knowledge about physical plausibility of the world into the reconstruction.

We will now introduce these challenges in more detail.

1.1 3D Reconstruction from 2D images

Challenge 1 has been a central topic in computer vision for many years. While the problem is ill-posed from a single image, finding projections of the same points in multiple views, *e.g.*, from a video, allows reconstructing the relative motion between the cameras together with the 3D locations of the points (Hartley and Zisserman 2004). The problem can usually only be solved up to an unknown scale factor. If we already know the absolute distance between camera poses from which the images were taken, metric reconstruction is possible. We can further see that *e.g.*, camera motion parallel to the image plane restricts the search space for correspondences to lines in the image parallel to the camera displacement. These observations have led to the development of stereo cameras, in which two identical cameras are fixed at a known horizontal displacement, allowing for easily finding feature correspondences in horizontal lines and computing absolute depth values for these pixels. Some manufacturers went one step further and directly implemented the depth computation algorithm efficiently on the camera, leading to the development of *RGB-D* (*i.e.*, RGB[5] color and depth) cameras, which can be accurately calibrated to provide both color and metric depth for each pixel (see Section 2.3 for more details on this type of data).

5: R, G and B here denote the red, green, and blue color channels, respectively. These are the base color used to mix all other colors in the additive color model used in digital displays or cameras. The human eye also has sensor cells sensitive to these colors.

Reconstructing 3D geometry from several views, or *structure from motion* (SfM) has started with early approaches like (Tomasi and Kanade 1992), which found feature points in the 2D images and identified their corresponding points in the other images to reconstruct camera motion and the 3D locations of the feature points. Pollefeys et al. (1999) extend the approach by first computing camera motion and metric geometry from feature correspondences and then filling in model gaps by dense depth estimation. During the first decade of this century, advances in these approaches and improvements in computing resources have led to city-scale reconstructions from arbitrary image collections (Agarwal, Furukawa, et al. 2011; Agarwal, Snavely, et al. 2009).

While the problem of SfM is generally concerned with map reconstruction from image collections (not necessarily image sequences) and runtime often is not crucial[6], it is sometimes desirable to build the map incrementally with close to real-time performance and localize the camera *at the same time*[7]. This problem is known as *visual simultaneous localization*

6: The approaches (Agarwal, Furukawa, et al. 2011; Agarwal, Snavely, et al. 2009) took one day on a compute cluster.

7: *e.g.*, to achieve close to real-time in robotics applications *and mapping* (vSLAM), but as we always work with visual data in this thesis, we will just use *simultaneous localization and mapping* (SLAM) from now on[8]. Several approaches have tackled this problem with monocular cameras (Davison et al. 2007; Mur-Artal, Montiel, et al. 2015), but other camera setups like stereo or RGB-D cameras have shown to improve performance (*e.g.*, Mur-Artal and Tardós 2017).

8: SLAM in general considers different sensors like cameras, LiDAR, or others.

While these approaches mainly reconstruct point clouds, another line of research has led to approaches for building *dense volumetric models* in online SLAM frameworks (Bylow et al. 2013; Kerl et al. 2013; Newcombe et al. 2011), making use of the broad availability of RGB-D cameras from the early 2010s. We will also build dense volumetric models for each object and for the background scene in this thesis. In Chapter 3, we will build upon the works by Newcombe et al. (2011) for mapping[9] the object and background models and by Bylow et al. (2013) for tracking[10] the object and camera poses. Our work extends over these approaches by introducing a novel method for segmenting moving objects in dynamic scenes.

9: *i. e.*, fusing depth measurements from multiple frames into consistent models

10: *i. e.*, estimating the camera motion between frames

1.2 Object Segmentation for Object-Level Scene Reconstruction

More recently, object-level SLAM approaches were explored in static (McCormac et al. 2018) and dynamic (Rünz and Agapito 2017; Rünz, Buffier, et al. 2018; B. Xu et al. 2019) environments, adding the requirement to address challenge 2 and segment objects in images. One way to address the segmentation problem in images is *instance segmentation*, which has seen tremendous progress over the last decades due to advances in deep learning. The segmentation problem is in this setting often combined with *classifying*[11] the segments, which is then called *semantic instance segmentation*[12]. A seminal work in this semantic instance segmentation is Mask R-CNN (He et al. 2017). It builds on early works for identifying and classifying regions of interest (Girshick et al. 2014) and several approaches to improve the efficiency (Girshick 2015; Ren et al. 2017). While the previous approaches mainly identify and classify bounding boxes of possible objects, Mask R-CNN (He et al. 2017) adds a module for pixel-levels segmentation.

11: *i. e.*, identifying the type of object

12: Not to be confused with *semantic segmentation*, which just classifies image pixels without segmenting individual instances.

Approaches like Mask R-CNN (He et al. 2017) can also be used to address the segmentation problem in static and dynamic environments based on learned object features. Several of the aforementioned works (McCormac et al. 2018; Rünz, Buffier, et al. 2018; B. Xu et al. 2019) demonstrated this use-case and we will also use Mask R-CNN in Chapter 3 to segment possibly moving objects in images. In dynamic environments, the motion of different object provides additional geometric cues for segmenting them (Rünz and Agapito 2017) and we will combine such cues with semantic segmentation for computational efficiency.

Similar to (Rünz and Agapito 2017; Rünz, Buffier, et al. 2018; B. Xu et al. 2019), we jointly address challenges 1 and 2[13] in Chapter 3. We present EM-Fusion (Strecke and Stueckler 2019), a method for reconstructing the 3D geometry of the static background and several independently moving objects from a moving camera while simultaneously tracking camera and

13: *i. e.*, 3D reconstruction and object segmentation

(Strecke and Stueckler 2019): *EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association* object motion. As mentioned before, we address the 3D reconstruction problem from challenge 1 by fusing depth information from several frames in *truncated signed distance function* models (TSDF models; see also Section 2.2) for the background and each moving object in an approach similar to (Curless and Levoy 1996; Newcombe et al. 2011). We further track the camera and object motion by aligning the depth measurements from the RGB-D images with the TSDF models as proposed by Bylow et al. (2013).

As we track and map multiple models, we need to address challenge 2 for identifying which pixel's information should be used to update which object. We approach this goal by combining semantic segmentation (He et al. 2017) with a principled model for motion segmentation. Based on the *pose*[14] of models from the previous frame, we compute an association likelihood for each pixel to each object model. We then use this likelihood as a soft weight for adjusting the influence of the respective pixels when updating the poses and the TSDF map of the different models. This approach is similar to the well-known expectation maximization algorithm (EM; Bishop 2007). The association likelihood formulates an *expectation* which pixel in the next frame is most likely associated with which object (E-Step) and the update of model poses and geometry to *maximizes* the likelihood given this association and the data from the next frame (M-Step). We demonstrate that our combination of semantic segmentation and geometry-based association likelihoods can track multiple moving objects with state-of-the-art accuracy and robustness.

14: *i. e.*, position and orientation

1.3 Physical Plausibility in Dynamic Scene Reconstruction

1.3.1 Dense Reconstruction from Point Clouds with Plausibility Constraints

(Strecke and Stueckler 2020): *Where Does It End? - Reasoning About Hidden Surfaces by Object Intersection Constraints*

In Chapter 4 (Strecke and Stueckler 2020), we observe that the geometry reconstructions recovered from EM-Fusion in Chapter 3 only cover observed object surfaces. These reconstructions are not physically plausible as they are single-sided surfaces "floating" in the air, something we would be surprised to see when we look at objects from different sides. In robotics applications, closed watertight shapes are desirable to enable *e.g.*, grasping of objects. We thus extend the incremental online mapping approach from Chapter 3 to retrieve optimized watertight object models.

Previous work has addressed the issue of completing the object shape using data-driven learned priors on observable 3D shapes (Dai et al. 2018; Firman et al. 2016; Song et al. 2017; Yang et al. 2019). However, in the general setting, the 3D training data required for these approaches is unavailable or difficult to obtain. We thus follow an orthogonal approach and employ physical background knowledge to address the task of object completion.

Our approach does not require 3D training data and attempts to complete object models using only information present in the observed scene, including depth measurements and reasoning about the physical plausibility of the object configurations in every frame. Transforming oriented point clouds computed from the depth maps[15] to the object coordinate systems with the object and camera poses recovered in Chapter 3 yields a registered collection of points and surface normals. This oriented point cloud is an unordered sampling of the observed surface. Several approaches exist for recovering the surface as an *implicit function*[16] (Calakli and Taubin 2011; Kazhdan, Bolitho, et al. 2006; Kazhdan and Hoppe 2013) have been developed. The TSDF models we build in Chapter 3 are one type of such implicit functions, but optimizing the models with an approach as the ones mentioned here provides more accurate models at the cost of higher computational complexity. Schroers et al. (2014) provide a taxonomy and mathematical formulation for generalizing several of these approaches.

15: see Section 2.3

16: *i. e.*, a function with 3D space as domain in which the surface is encoded *implicitly* as a level-set, see Section 2.2 for the representation we use in this thesis.

(Schroers et al. 2014): *A Variational Taxonomy for Surface Reconstruction from Oriented Points*

So far, these approaches only consider the observed surfaces of the objects. As we track and reconstruct multiple objects in Chapter 3, we can impose additional *physical plausibility* priors on the reconstructions. One such prior is that no two objects *intersect*[17] at any point in time. A second prior we use in Chapter 4 is based on the geometry of the camera and the depth measurements. When we see a 3D point in a pixel, we can reason that the space on the ray between that point and the camera is empty. By this argument, we can construct a *hull* as the complement of this known empty space. Schroers et al. (2014) demonstrated that a hull like this can help to improve the surface reconstructions.

17: *i. e.*, share the same 3D point

We propose an intersection constraint based on our first observation and integrate it together with the hull constraint in the Hessian-IMLS formulation by Schroers et al. (2014) to optimize SDF models for all objects and the background. Our experiments demonstrate improved object shape completeness over the results by EM-Fusion (Chapter 3; Strecke and Stueckler 2019) purely based on physical plausibility without the requirement for 3D training data.

1.3.2 Shape Optimization via Differentiable Physics

In Chapter 5 (Strecke and Stueckler 2021), we use recent advances in differentiable simulation and parametric shape representations to optimize the object shape and physical parameters. Our key observation is that the shape of an object affects its motion after collisions with other objects[18]. If we now observe such a trajectory with a collision, we can reason about *physically plausible* shapes based on the *motion* of the objects.

(Strecke and Stueckler 2021): *DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes*

18: For example, a ball thrown towards a wall with fixed initial center position and velocity will collide earlier or later with the wall depending on its radius.

Many methods for physical simulation have been developed in the last decades in the computer graphics and mechanical engineering communities (Bender, Erleben, et al. 2013). These methods work well for simulating the physical behavior of objects of known shape with known physical parameters. More recently, the *inverse problem*, *i. e.*, the estimation of physical parameters that result in a given trajectory, has attracted interest in the computer vision and robotics communities and led to the development of differentiable physics engines (Geilinger et al. 2020; Hu, Anderson, et al. 2020; Krishna Murthy et al. 2021). Estimating physical parameters like friction can enable the transfer of real-world properties to simulated environments, possibly enabling reasoning about future interactions with the world. One example for differentiable simulations is the work by Avila Belbute-Peres et al. (2018). While it demonstrated the optimization of physical parameters, it is limited to relatively simple object primitives in 2D scenes. We thus extend this work to 3D environments and use signed distance function (SDF; see Section 2.2) models for object shapes. This enables the simulation of more complex shapes and the computation of gradients with respect to shape parameters.

(Avila Belbute-Peres et al. 2018): *End-to-End Differentiable Physics for Learning and Control*

While general shape optimization without any prior 3D knowledge on the considered shapes as in Chapters 3 and 4 might not be feasible for this approach, recent advances in neural implicit modeling (Mescheder et al. 2019; Park et al. 2019) have shown that families of shapes can be represented in low dimensional parameter spaces and optimized from incomplete point clouds. Further, surface extraction from these models can be made differentiable (Remelli et al. 2020), allowing the optimization of shape parameters for aerodynamics or rendered silhouettes by employing differentiable rendering (Kato et al. 2018). This inspires us to employ these parametric SDF models in our differentiable physics simulation and optimize for shape parameters through the simulation.

Our experiments demonstrate that we can accurately recover the shape of objects with given simulated reference trajectories or depth measurements. We further see that simulating with the optimized parameters accurately recovers the reference trajectories. Thus, we anticipate that the simulation model will be useful for predicting interactions[19] with the scene given the optimized parameters.

19: *i.e.*, application of forces

1.4 Contributions

In summary, the contributions of this thesis are the following:

- We address challenges 1 and 2[20] in EM-Fusion (Strecke and Stueckler 2019), an approach for tracking and mapping dynamic objects together with background geometry and camera motion in Chapter 3. Our approach models motion segmentation as a probabilistic data association problem and is robust to occlusions.
- We present a first approach for addressing challenge 3[21], Co-Section (Strecke and Stueckler 2020), which optimizes watertight shapes based on physical plausibility priors in Chapter 4. Our method can optimize completed watertight models based on physical plausibility constraints without the need for learned shape priors.
- In an orthogonal approach for addressing challenge 3, we present DiffSDFSim (Strecke and Stueckler 2021) for optimizing object shape and physical properties to match target observations in Chapter 5. The optimized results allow for re-generating the target trajectories in simulation, potentially enabling predictions about future scene states or real-to-sim transfer.

20: *i.e.*, 3D reconstruction and object segmentation

(Strecke and Stueckler 2019): *EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association*

21: *i.e.*, incorporating physical plausibility

(Strecke and Stueckler 2020): *Where Does It End? - Reasoning About Hidden Surfaces by Object Intersection Constraints*

(Strecke and Stueckler 2021): *DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes*

1.5 Publications

Large parts of this thesis were published in peer-reviewed conference proceedings. Each of the chapters mentioned before (Chapters 3 to 5) is based on one of the following publications.

1. **Strecke, Michael** and Joerg Stueckler (2019). 'EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00596, Chapter 3.
2. **Strecke, Michael** and Joerg Stueckler (2020). 'Where Does It End? - Reasoning About Hidden Surfaces by Object Intersection Constraints'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr42600.2020.00961, Chapter 4.
3. **Strecke, Michael** and Joerg Stueckler (2021). 'DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes'. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. DOI: 10.1109/3dv53792.2021.00020, Chapter 5.

The following work was performed during my PhD, but does not form a part of this thesis:

1. Kandukuri, Rama Krishna**, Michael Strecke**, and Joerg Stueckler (2024). 'Physics-Based Rigid Body Object Tracking and Friction Filtering From RGB-D Videos'. In: *International Conference on 3D Vision (3DV)*. accepted, preprint arXiv: 2309.15703. DOI: 10.1109/3DV62453.2024.00111.

1.6 Open-Source Software Releases

We provide open-source software releases for EM-Fusion[22] (Strecke and Stueckler 2019) and Co-Section[23] (Strecke and Stueckler 2020). The releases allow other researchers to reproduce our results and build upon them in future research.

22: https://github.com/EmbodiedVision/emfusion

23: https://github.com/EmbodiedVision/cosection

Background 2

This chapter introduces theoretical background required for all following chapters of this thesis. Preliminaries only required in a single chapter will be introduced as separate sections within the respective chapters.

2.1 Rigid Body Motion

In this thesis, we will consider the motion of *rigid bodies*, *i.e.* the *poses*, consisting of *rotation* and *translation* of multiple objects at different points in time. This section will introduce the notation used to represent this kind of motion in this thesis. The derivations and definitions in this section follow (Ma et al. 2004, Section 2.3 and 2.4), to which we refer for further details.

(Ma et al. 2004): *An Invitation to 3-D Vision*

2.1.1 Purely Rotational Motion

We describe the orientation of an object by the *rotation*, a linear transformation $\mathbf{R}$ acting on all points on the object:

$$\mathbf{x}^W = \mathbf{R}^{OW}\mathbf{x}^O, \tag{2.1}$$

where the superscripts ^{W} and ^{O} denote the point $\mathbf{x}$ on the object in world and object frame, respectively and the rotation $\mathbf{R}^{OW}$ maps from object to world frame. Whenever clear from the context, we will omit the respective superscripts to avoid notational clutter. The space of matrices in $\mathbb{R}^{3\times3}$ realizing a rotation is called the *special orthogonal group* in 3 dimensions:

$$\mathrm{SO}(3) := \left\{\mathbf{R} \in \mathbb{R}^{3\times3} \,\middle|\, \mathbf{R}^\top\mathbf{R} = \mathbb{I}_3, \det(\mathbf{R}) = 1\right\}, \tag{2.2}$$

where $\mathbb{I}_n$ denotes the identity matrix in n dimensions. While a matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ has 9 entries, the constraint in Equation (2.2) limits the degrees of freedom of rotation matrices to 3. We can derive a minimal representation by modeling any rotation as part of a trajectory, *i.e.*, a time-dependent mapping

$$\begin{aligned}\mathbf{R} : \mathbb{R} &\to \mathrm{SO}(3),\\ t &\mapsto \mathbf{R}(t),\end{aligned} \tag{2.3}$$

which at any point in time t must satisfy the condition

$$\mathbf{R}(t)\mathbf{R}(t)^\top = \mathbb{I}_3. \tag{2.4}$$

Differentiating Equation (2.4) yields

$$\dot{\mathbf{R}}(t)\mathbf{R}^\top(t) + \mathbf{R}(t)\dot{\mathbf{R}}^\top(t) = 0 \implies \dot{\mathbf{R}}(t)\mathbf{R}^\top(t) = -\left(\dot{\mathbf{R}}(t)\mathbf{R}^\top(t)\right)^\top. \tag{2.5}$$

Thus, $\dot{\mathbf{R}}(t)\mathbf{R}^\top(t)$ is a *skew-symmetric* matrix and there exists a vector $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ such that (Ma et al. 2004)

$$\widehat{\omega}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^\top(t), \tag{2.6}$$

where the skew-symmetric matrix $\widehat{\boldsymbol{\omega}} \in \mathbb{R}^{3\times3}$ for a vector $\boldsymbol{\omega} \in \mathbb{R}^3$ is defined as

The matrix in Equation (2.7) can also be used to write the cross-product as a matrix multiplication, *i.e.*, $\boldsymbol{\omega} \times \mathbf{x} = \widehat{\boldsymbol{\omega}}\mathbf{x}$ with $\boldsymbol{\omega} \in \mathbb{R}^3, \mathbf{x} \in \mathbb{R}^3$.

$$\widehat{\boldsymbol{\omega}} := \begin{pmatrix}\omega_1\\\omega_2\\\omega_3\end{pmatrix}^{\wedge} := \begin{pmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{pmatrix} \text{ with } \boldsymbol{\omega} = \begin{pmatrix}\omega_1\\\omega_2\\\omega_3\end{pmatrix}. \tag{2.7}$$

In Equation (2.7), we implicitly defined the operator $\wedge$ for computing a skew-symmetric matrix $\widehat{\boldsymbol{\omega}}$ from a vector $\boldsymbol{\omega} \in \mathbb{R}^3$. We further define the inverse $\vee$ of this operator by

$$\widehat{\boldsymbol{\omega}}^{\vee} := \begin{pmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{pmatrix}^{\vee} := \begin{pmatrix}\omega_1\\\omega_2\\\omega_3\end{pmatrix} = \boldsymbol{\omega}. \tag{2.8}$$

Multiplying Equation (2.6) by $\mathbf{R}(t)$ now yields

$$\dot{\mathbf{R}}(t) = \widehat{\boldsymbol{\omega}}(t)\mathbf{R}(t), \tag{2.9}$$

where we can see that for $\mathbf{R}(t_0) = \mathbb{I}_3$ we have $\dot{\mathbf{R}}(t_0) = \widehat{\boldsymbol{\omega}}(t_0)$, *i.e.* $\widehat{\boldsymbol{\omega}}$ and in turn the vector $\boldsymbol{\omega}$ locally represents the *angular velocity*[1]. Thus, the skew-symmetric matrix $\widehat{\boldsymbol{\omega}}$ further gives a first-order Taylor approximation to the rotation trajectory:

1: *i.e.*, the rate of change of the orientation around identity

$$\mathbf{R}(t_0 + \Delta t) \approx \mathbb{I}_3 + \widehat{\boldsymbol{\omega}}(t_0)\Delta t. \tag{2.10}$$

We will denote the space of skew-symmetric matrices in 3 dimensions by

$$\mathfrak{so}(3) := \left\{\widehat{\boldsymbol{\omega}} \in \mathbb{R}^{3\times3} \;\middle|\; \boldsymbol{\omega} \in \mathbb{R}^3\right\}. \tag{2.11}$$

Using the approximation in Equation (2.10), we can derive a mapping from $\mathfrak{so}(3)$ to SO(3), called the *exponential map* (see (Ma et al. 2004) for details):

$$\begin{aligned}\exp : \mathfrak{so}(3) &\longrightarrow \mathrm{SO}(3)\\ \widehat{\boldsymbol{\omega}} &\longmapsto e^{\widehat{\boldsymbol{\omega}}},\end{aligned} \tag{2.12}$$

where $e^{\widehat{\boldsymbol{\omega}}}$ is the matrix exponential

$$e^{\widehat{\boldsymbol{\omega}}} = \sum_{i=0}^{\infty} \frac{\widehat{\boldsymbol{\omega}}^i}{i!}, \tag{2.13}$$

which for skew-symmetric matrices $\widehat{\boldsymbol{\omega}} \in \mathfrak{so}(3)$ can be simplified to *Rodrigues' formula for rotation matrices*:

$$e^{\widehat{\boldsymbol{\omega}}} = \mathbb{I}_3 + \frac{\widehat{\boldsymbol{\omega}}}{\|\boldsymbol{\omega}\|}\sin(\|\boldsymbol{\omega}\|) + \frac{\widehat{\boldsymbol{\omega}}^2}{\|\boldsymbol{\omega}\|^2}(1 - \cos(\|\boldsymbol{\omega}\|)). \tag{2.14}$$

The mapping in Equation (2.12) has a physical interpretation: the resulting rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ is a rotation around the axis $\boldsymbol{\omega}$ by $\theta = \|\boldsymbol{\omega}\|$ radians. In this interpretation, the *axis-angle-vector* $\boldsymbol{\omega}$ in Equation (2.12) is not the angular velocity anymore, but rather the integrated constant unit velocity $\boldsymbol{\omega}'$ over some time interval Δt, *i.e.*, $\boldsymbol{\omega} = \Delta t\boldsymbol{\omega}'$.

While the mapping in Equation (2.12) is not unique due to the periodic nature of rotations (for $\|\boldsymbol{\omega}\| = 1$, we have $\exp(2\pi k\widehat{\boldsymbol{\omega}}) = \mathbb{I}_3$ for all $k \in \mathbb{Z}$), we can still always find a $\boldsymbol{\omega} \in \mathbb{R}^3$ such that $\mathbf{R} = \exp(\widehat{\boldsymbol{\omega}})$ for $\mathbf{R} \in \mathrm{SO}(3)$, which we call the *logarithm map* of SO(3):

$$\begin{aligned} \log : \mathrm{SO}(3) &\longrightarrow \mathfrak{so}(3) \\ \mathbf{R} &\mapsto \begin{cases} \widehat{\boldsymbol{\omega}}\theta, & \text{if } \mathbf{R} \neq \mathbb{I}_3 \\ \mathbf{0}, & \text{otherwise,} \end{cases} \end{aligned} \tag{2.15}$$

where

$$\theta = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R}) - 1}{2}\right), \quad \boldsymbol{\omega} = \frac{1}{2\sin(\theta)}\begin{pmatrix} R_{3,2} - R_{2,3} \\ R_{1,3} - R_{3,1} \\ R_{2,1} - R_{1,2} \end{pmatrix} \tag{2.16}$$

and $R_{i,j}$ denotes the entry in the i-th row and j-th column of the matrix $\mathbf{R}$.

2.1.2 Full Rigid-Body Motion

In addition to rotational motion explained in Subsection 2.1.1, rigid bodies can undergo *translational motion*:

$$\mathbf{x}^W = \mathbf{R}^{OW}\mathbf{x}^O + \mathbf{t}^{OW}, \tag{2.17}$$

where the superscripts are the same as in Equation (2.1) and $\mathbf{t}^{OW}$ denotes the translation offset from object to world space (*i.e.* location of the object origin in world space).

While Equation (2.1) is a linear operation, Equation (2.17) is *affine*. For convenience, we convert Equation (2.17) to a linear operation by introducing *homogeneous coordinates* for vectors $\mathbf{x} \in \mathbb{R}^n$:

$$\bar{\mathbf{x}} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}. \tag{2.18}$$

This allows to define the space of transformation matrices as the *special Euclidean group* in 3 dimensions:

$$\mathrm{SE}(3) := \left\{ \mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \middle| \; \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times 4}. \tag{2.19}$$

We can now rewrite Equation (2.17) as

$$\bar{\mathbf{x}}^W = \mathbf{T}^{OW}\bar{\mathbf{x}}^O, \tag{2.20}$$

with $\mathbf{T}^{OW} \in \mathrm{SE}(3)$ composed of $\mathbf{R}^{OW}$ and $\mathbf{t}^{OW}$ from Equation (2.17). We will omit the overline operator "$\bar{\cdot}$" for homogeneous coordinates in this thesis whenever the correct dimension of the vector is clear from the context.

Similar as before for rotation matrices, looking at trajectories

$$\begin{aligned} \mathbf{T} : \mathbb{R} &\longrightarrow \mathrm{SE}(3), \\ t &\mapsto \mathbf{T}(t) \end{aligned} \tag{2.21}$$

we can look at the structure of

$$\dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t) = \begin{pmatrix} \dot{\mathbf{R}}(t)\mathbf{R}^\top(t) & \dot{\mathbf{t}}(t) - \dot{\mathbf{R}}(t)\mathbf{R}^\top(t)\mathbf{t}(t) \\ \mathbf{0} & 0 \end{pmatrix} \in \mathbb{R}^{4\times4}. \tag{2.22}$$

We can now write the top-left 3×3 sub-matrix as a skew-symmetric matrix $\widehat{\boldsymbol{\omega}}(t)$ (see Equation (2.6)) and define a vector $\mathbf{v}(t) = \dot{\mathbf{t}}(t) - \widehat{\boldsymbol{\omega}}(t)\mathbf{t}(t)$. Using $\boldsymbol{\omega}$ and $\mathbf{v}$, we can then define the *twist*

$$\widehat{\boldsymbol{\xi}}(t) = \dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t) = \begin{pmatrix} \widehat{\boldsymbol{\omega}}(t) & \mathbf{v}(t) \\ \mathbf{0} & 0 \end{pmatrix} \in \mathbb{R}^{4\times4}, \tag{2.23}$$

which similar to $\widehat{\boldsymbol{\omega}}$ for rotations defines a local *tangent* on SE(3). The space of all twists is defined as

$$\mathfrak{se}(3) := \left\{ \widehat{\boldsymbol{\xi}} = \begin{pmatrix} \widehat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 0 \end{pmatrix} \,\middle|\, \widehat{\boldsymbol{\omega}} \in \mathfrak{so}(3), \mathbf{v} \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times4} \tag{2.24}$$

and we define operators $\vee$ and $\wedge$ to convert between the twist $\widehat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ and the *twist coordinates* $\boldsymbol{\xi} \in \mathbb{R}^6$:

$$\begin{pmatrix} \widehat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 0 \end{pmatrix}^\vee := \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix} \in \mathbb{R}^6, \quad \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix}^\wedge := \begin{pmatrix} \widehat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 0 \end{pmatrix} \in \mathfrak{se}(3). \tag{2.25}$$

The vector $\boldsymbol{\omega}$ can be interpreted as the *angular velocity* (as before for rotations) and $\mathbf{v}$ as the *linear velocity* for the trajectory. As tangent vectors, twists can again be seen as the first order Taylor approximation around $\mathbf{T}(t)$ and by similar arguments as for rotations before, the *exponential and logarithm maps* between SE(3) and $\mathfrak{se}(3)$ can be defined:

$$\begin{aligned} \exp : \mathfrak{se}(3) &\longrightarrow \mathrm{SE}(3) & \log : \mathrm{SE}(3) &\longrightarrow \mathfrak{se}(3) \\ \widehat{\boldsymbol{\xi}} &\longmapsto e^{\widehat{\boldsymbol{\xi}}}, & \mathbf{T} &\longmapsto \log(\mathbf{T}). \end{aligned} \tag{2.26}$$

We refer to (Ma et al. 2004, Section 2.4) for more details on the derivation. In this thesis we will use $\mathbf{T}(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ as a shorthand for the matrix $\mathbf{T} = \exp\left(\widehat{\boldsymbol{\xi}}\right) \in \mathrm{SE}(3)$. Similar to the case for rotation matrices, Equation (2.26) acts on twist coordinates $\boldsymbol{\xi}$, which are not velocities but the integral of some unit velocity $\boldsymbol{\xi}'$ over some time interval Δt, *i.e.*, $\boldsymbol{\xi} = \Delta t \boldsymbol{\xi}'$.

2.2 Signed Distance Functions (SDFs)

Throughout this thesis, we represent the geometry of objects or the background scene as *signed distance functions* (SDFs). In contrast to inherently discretized explicit geometry representations like point clouds or triangular meshes, SDFs represent the surface $\mathcal{S}$ of an object or the background scene *continuously* and *implicitly* as the zero-level set of the SDF. Formally, we define the signed distance function as follows.

Definition 2.2.1 Signed Distance Function. *Let*

$$\mathcal{S} \subset \left\{ (\mathbf{p}, \mathbf{n}) \,\middle|\, \mathbf{p} \in \mathbb{R}^d, \mathbf{n} \in \mathbb{R}^d, \|\mathbf{n}\| = 1 \right\} \subset \mathbb{R}^d \times \mathbb{R}^d \tag{2.27}$$

be an oriented surface in d-dimensional space (where $\mathbf{n}$ *denotes the outward unit normal). The signed distance function* ϕ *representing* $\mathcal{S}$ *is the mapping*

$$\begin{aligned} \phi : \mathbb{R}^d &\to \mathbb{R} \\ \mathbf{p} &\mapsto \operatorname{sgn}\left(\mathbf{n}^{*\top}(\mathbf{p}-\mathbf{p}^*)\right)\|\mathbf{p}-\mathbf{p}^*\|\,, \end{aligned} \tag{2.28}$$

where $(\mathbf{p}^*, \mathbf{n}^*) = \arg\min_{(\mathbf{q},\mathbf{n})\in\mathcal{S}} \|\mathbf{p}-\mathbf{q}\|$ *and*

$$\begin{aligned} \operatorname{sgn} : \mathbb{R} &\to \{-1, 1\} \\ x &\mapsto \begin{cases} -1, & \textit{if } x < 0, \\ 1, & \textit{otherwise} \end{cases} \end{aligned} \tag{2.29}$$

is the sign function.

Intuitively, the SDF $\phi(\mathbf{p})$ gives the distance to the closest point on the surface $\mathcal{S}$, with the sign indicating whether $\mathbf{p}$ is inside or outside the surface. As mentioned before, the surface $\mathcal{S}$ can be recovered from the SDF ϕ as its zero-level set

$$\mathcal{S} = \left\{(\mathbf{p}, \mathbf{n}) \,\middle|\, \phi(\mathbf{p}) = 0, \mathbf{n} = \nabla\phi(\mathbf{p})\right\}. \tag{2.30}$$

Figure 2.1 illustrates an example of an SDF in 2D. The surface of the bowl is represented by the zero isosurface (white) of the SDF, while the inside of the object contains negative SDF values (blue) and the area outside the object positive ones (red). Several other isosurfaces are shown as lines with equal distance to the surface.

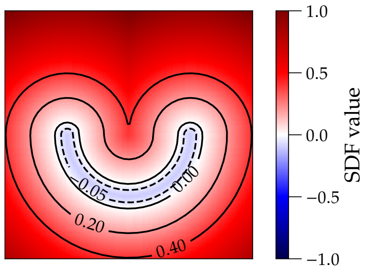


Figure 2.1: Plot of an SDF in 2D for a bowl shape. The surface is represented by the zero level set (white), blue regions are inside the object and red ones are outside.

In practice, one can often express the SDF ϕ in Equation (2.28) without resorting to the explicit point-normal representation of $\mathcal{S}$ (see *e.g.* Subsection 2.2.1). While it is usually not possible to *explicitly* express $\mathcal{S}$ without discretization or sampling, this allows to represent $\mathcal{S}$ *implicitly* and *continuously* via the SDF ϕ. If needed, a discretized approximation of $\mathcal{S}$ can be extracted from ϕ as we will explain in Subsection 2.2.4.

In addition to representing the surface, SDFs carry additional information (*i.e.*, the distance to the closest surface) for off-surface points. This representation has thus been used in robotic path planning with collision avoidance (Oleynikova et al. 2017; Pan et al. 2022) or for contact detection in physical simulation (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020).

In this thesis, we will use SDFs in different representations. In Chapters 3 and 4, we will represent objects and the background as volumetric grids (see Subsection 2.2.2) and in Chapter 3 we will additionally truncate large distances (see Subsection 2.2.3). We will further use continuous parametric models (see Subsection 2.2.1) in Chapter 5.

2.2.1 Parametric Signed Distance Functions / Shape Spaces

The formulation from Definition 2.2.1 can be extended for an additional parameter $\mathbf{z} \in \mathbb{R}^n$ to allow representing families of shapes, which we also call *shape spaces*:

$$\begin{aligned} \phi : \mathbb{R}^d \times \mathbb{R}^n &\to \mathbb{R} \\ (\mathbf{p}, \mathbf{z}) &\mapsto \phi(\mathbf{p}, \mathbf{z}). \end{aligned} \tag{2.31}$$

For some primitive shapes, this parameterized SDF can be derived analytically. For a solid sphere of radius $r \in \mathbb{R}^{\geq 0} \subset \mathbb{R}$ and center at the origin for example, the SDF is given as

$$\phi_\circ(\mathbf{p}, r) = \|\mathbf{p}\|_2 - r. \tag{2.32}$$

Several other primitive shapes like cuboids, cylinders or capsules have compact analytical formulas. A good overview over these SDFs and tutorials on the derivation of some of them can be found on the homepage of Inigo Quilez[2]. Even the non-convex 2D bowl in Figure 2.1 can be expressed by a parametric SDF describing an arc[3] for points $\mathbf{p} = (p_x, p_y)^\top$ and parameters $\mathbf{z} = (\theta, r_a, r_b)^\top$, where θ denotes the aperture (*i.e.* half the angle covered by the arc), r_a denotes the radius of the arc, and r_b is the inner radius (*i.e.* half the thickness of the arc):

2: https://iquilezles.org/articles/distfunctions/

3: https://iquilezles.org/articles/distfunctions2d/

$$\phi_\cup(\mathbf{p}, \mathbf{z}) = \begin{cases} \left\| \begin{pmatrix} |p_x| - \sin(\theta) \cdot r_a \\ p_y - \cos(\theta) \cdot r_a \end{pmatrix} \right\|, & \text{if } \cos(\theta) \cdot p_x > \sin(\theta) \cdot p_y, \\ \big| \|\mathbf{p}\| - r_a \big| - r_b, & \text{otherwise.} \end{cases} \tag{2.33}$$

Two main advantages of these parametric SDFs are a *continuous* query domain (as opposed to the discretized grid we will explain in Subsection 2.2.2) and *differentiability* for both the query points and the shape parameters. However, modeling complex shapes requires combinations of primitives, which can be tedious manually and sometimes does not give the exact SDF values away from the surface.

More recently, the problem of modeling complex shapes was addressed by several works which use neural networks to represent SDFs (Park et al. 2019) or other implicit surface representations like occupancy (Mescheder et al. 2019). In DeepSDF (Park et al. 2019), similar to the formula for the sphere in Equation (2.32), a network with trained parameters $\boldsymbol{\theta} \in \mathbb{R}^m$ can represent a family of shapes parameterized by latent codes $\mathbf{z} \in \mathbb{R}^n$. We denote an SDF represented by such a neural network by $\phi_{\boldsymbol{\theta}}$. Note that formally the network parameters $\boldsymbol{\theta}$ are also function parameters, but we denote them using a subscript as they are fixed during inference. Park et al. (2019) formulate training of this network as an "auto-decoder". They assume a zero-mean multivariate Gaussian distribution over the latent codes $\{\mathbf{z}_i\}_{i=1}^N$ for N shapes with spherical covariance $\sigma^2 \mathbb{I}_n$[4] and minimize the negative log-likelihood over latent codes $\{\mathbf{z}_i\}_{i=1}^N$ and network parameters $\boldsymbol{\theta}$ given K sampled points $\{\mathbf{p}_j\}_{j=1}^K$ with given ground truth SDF values $\{s_j\}_{j=1}^K$ per shape:

(Park et al. 2019): *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*

(Mescheder et al. 2019): *Occupancy Networks: Learning 3D Reconstruction in Function Space*

4: where $\mathbb{I}_n$ denotes the $n \times n$ identity matrix

$$\left(\boldsymbol{\theta}^*, \{\mathbf{z}_i^*\}_{i=1}^N\right) = \underset{\boldsymbol{\theta}, \{\mathbf{z}_i\}_{i=1}^N}{\arg\min} \sum_{i=1}^N \left(\sum_{j=1}^K E\left(\phi_{\boldsymbol{\theta}}\left(\mathbf{p}_j, \mathbf{z}_i\right), s_j\right) + \frac{1}{\sigma^2} \|\mathbf{z}_i\|^2 \right). \tag{2.34}$$

Here, E is a loss function penalizing the deviation between predicted and ground-truth SDF values. An L^2-loss would for example assume

Gaussian noise on the SDF values. Park et al. (2019) used a clamped L^1-loss[5] in their experiments as it allowed to concentrate network capacity close to the surface:

5: similar to the TSDF formulation in Subsection 2.2.3

$$E(\phi_{\boldsymbol{\theta}}(\mathbf{p},\mathbf{z}),s) = \left|\operatorname{clamp}\left(\phi_{\boldsymbol{\theta}}(\mathbf{p},\mathbf{z}),\delta\right) - \operatorname{clamp}(s,\delta)\right|, \tag{2.35}$$

where $\operatorname{clamp}(s,\mu) := \min(\mu, \max(-\mu, s))$.

This formulation relies on supervision for the training point samples the SDF is computed for. Computing SDF values from meshes can be challenging, especially far from the surface. A follow-up work (Gropp et al. 2020) thus included implicit geometric regularization in the training loss. The key observation is that the gradient of the SDF should satisfy the *Eikonal constraint*[6]:

(Gropp et al. 2020): *Implicit Geometric Regularization for Learning Shapes*

6: The intuition behind this constraint is that moving along the surface normals (which match the SDF gradients as mentioned in Equation (2.30)), the distance to the surface increases exactly by the distance traveled.

$$\forall \mathbf{p} \in \mathbb{R}^d : \left\|\nabla\phi(\mathbf{p})\right\| = 1. \tag{2.36}$$

Based on this, Gropp et al. (2020) proposed training the network based on point samples with or without normal information on the surface and enforce the correct off-surface SDF values using the constraint Equation (2.36):

$$\underset{\boldsymbol{\theta},\{\mathbf{z}_i\}_{i=1}^N}{\arg\min} \sum_{i=1}^{N} \left(E_{\mathcal{S}_i}(\boldsymbol{\theta},\mathbf{z}_i) + \lambda \mathbb{E}_{\mathbf{p}}\left(\left\|\nabla_{\mathbf{p}}\phi_{\boldsymbol{\theta}}(\mathbf{p},\mathbf{z}_i)\right\| - 1\right)\right), \tag{2.37}$$

where

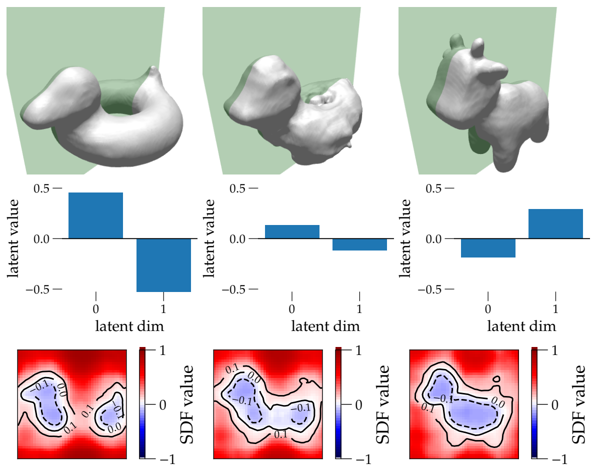
$$E_{\mathcal{S}_i}(\boldsymbol{\theta},\mathbf{z}_i) = \frac{1}{|\mathcal{S}_i|} \sum_{(\mathbf{p},\mathbf{n})\in\mathcal{S}_i} \left(\left|\phi_{\boldsymbol{\theta}}(\mathbf{p},\mathbf{z}_i)\right| + \tau \left\|\nabla_{\mathbf{p}}\phi_{\boldsymbol{\theta}}(\mathbf{p},\mathbf{z}_i) - \mathbf{n}\right\|\right). \tag{2.38}$$

The loss in Equation (2.38) encourages the SDF to vanish on the surface samples. The parameter τ allows disabling the second term in Equation (2.38) if normal data is absent. In this thesis, we always consider surfaces with normal data (as specified in Equation (2.27)), thus we always set $\tau = 1$. The Eikonal constraint from Equation (2.36) is incorporated in the second term in Equation (2.37) as expectation over a distribution of query point samples. An example of how different cuts through the SDF look for these variants of DeepSDF is given in Figure 2.2. While the reconstructed surfaces for the two training shapes look very similar, training with the Eikonal loss in Equation (2.37) yields smoother surface reconstructions and a smoother interpolation between the two shapes. Furthermore, cuts through the SDFs exhibit some artifacts for off-surface SDFs when trained with the original DeepSDF loss in Equation (2.34) (see Figure 2.2a), while they are closer to the true distances when trained with the loss in Equation (2.37) (see Figure 2.2b).

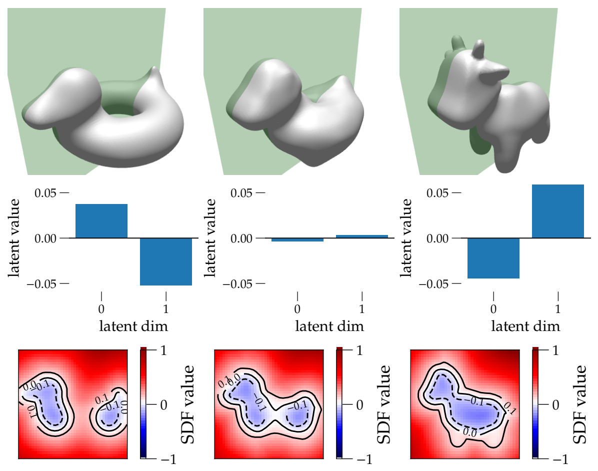
2.2.2 SDFs as Volumetric Grids

For practical reasons, SDFs are often used as *discretized* volumetric grids, *i.e.* the SDF ϕ is evaluated at regular spaced grid points and the SDF values at these points are then stored in a 3D array. Mathematically, a cuboid-shaped domain $\Omega = [x_1, x_2] \times [y_1, y_2] \times [z_1, z_2] \subset \mathbb{R}^3$ is chosen and discretized at resolution m_x, m_y and m_z for the different coordinate axes. The points for evaluating the SDF are placed at *voxel*[7] locations

7: *i.e.* "volume element"



(a) Trained with SDF samples as in (Park et al. 2019).



(b) Trained with the Eikonal loss from (Gropp et al. 2020).

Figure 2.2: Learned shape spaces with and without Eikonal loss. Top: renderings of the reconstructions of the two training shapes and an interpolation. Center: The corresponding latent codes. Bottom: SDF cuts for the green plane in the top row. One can see that training purely based on SDF samples with the loss in Equation (2.34) does not yield true distances for off-surface points (a). In (b), training with the Eikonal loss from Equation (2.37), distances for off-surface samples are closer to true distances and the interpolation between the shapes looks smoother.

$$\mathbf{v}_{i,j,k} = \begin{pmatrix} x_1 + \frac{i}{m_x - 1}(x_2 - x_1) \\ y_1 + \frac{j}{m_y - 1}(y_2 - y_1) \\ z_1 + \frac{k}{m_z - 1}(z_2 - z_1) \end{pmatrix}, \tag{2.39}$$

for indices $i \in \{0, \dots, m_x - 1\}$, $j \in \{0, \dots, m_y - 1\}$ and $k \in \{0, \dots, m_z - 1\}$ and the corresponding SDF values are stored in a 3D tensor $\mathbf{\Phi}$ as

$$\Phi_{i,j,k} = \phi\left(\mathbf{v}_{i,j,k}\right). \tag{2.40}$$

The indices i, j, k in Equations (2.39) and (2.40) can be linearized as

$$\ell = \left(i + m_x j + m_x m_y k\right) \in \{0, \dots, M - 1\}, \tag{2.41}$$

with $M = m_x m_y m_z$, allowing to write $\mathbf{\Phi}$ as a vector $\boldsymbol{\phi} \in \mathbb{R}^M$.

We can now evaluate the discretized SDF at a continuous location $\mathbf{x} = (x_x, x_y, x_z)^\top \in \Omega$ by *trilinear interpolation*. To do this, we first compute the index coordinates of $\mathbf{x}$:

$$\check{\mathbf{x}} = \begin{pmatrix} \check{x}_x \\ \check{x}_y \\ \check{x}_z \end{pmatrix} = \begin{pmatrix} \frac{x_x - x_1}{x_2 - x_1}(m_x - 1) \\ \frac{x_y - y_1}{y_2 - y_1}(m_y - 1) \\ \frac{x_z - z_1}{z_2 - z_1}(m_z - 1) \end{pmatrix}. \tag{2.42}$$

The SDF value at $\mathbf{x}$ is approximated by trilinear interpolation from the 8 neighboring voxels by first computing interpolation coefficients

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{pmatrix} = \check{\mathbf{x}} - \lfloor \check{\mathbf{x}} \rfloor, \tag{2.43}$$

where $\lfloor \cdot \rfloor$ denotes the floor operation, which is applied element-wise in Equation (2.43). The vector $\boldsymbol{\alpha}$ contains the decimal part of $\check{\mathbf{x}}$, which we use as interpolation factor in each dimension when computing the interpolated SDF value:

$$\begin{aligned}
\Phi^*_{\check{x}_y^-,\check{x}_z^-} &= (1-\alpha_x)\Phi_{\check{x}_x^-,\check{x}_y^-,\check{x}_z^-} + \alpha_x\Phi_{\check{x}_x^+,\check{x}_y^-,\check{x}_z^-}\\
\Phi^*_{\check{x}_y^-,\check{x}_z^+} &= (1-\alpha_x)\Phi_{\check{x}_x^-,\check{x}_y^-,\check{x}_z^+} + \alpha_x\Phi_{\check{x}_x^+,\check{x}_y^-,\check{x}_z^+}\\
\Phi^*_{\check{x}_y^+,\check{x}_z^-} &= (1-\alpha_x)\Phi_{\check{x}_x^-,\check{x}_y^+,\check{x}_z^-} + \alpha_x\Phi_{\check{x}_x^+,\check{x}_y^+,\check{x}_z^-}\\
\Phi^*_{\check{x}_y^+,\check{x}_z^+} &= (1-\alpha_x)\Phi_{\check{x}_x^+,\check{x}_y^+,\check{x}_z^+} + \alpha_x\Phi_{\check{x}_x^+,\check{x}_y^+,\check{x}_z^+}
\end{aligned} \tag{2.44}$$

$$\begin{aligned}
\Phi^*_{\check{x}_z^-} &= (1-\alpha_y)\Phi^*_{\check{x}_y^-,\check{x}_z^-} + \alpha_y\Phi^*_{\check{x}_y^+,\check{x}_z^-}\\
\Phi^*_{\check{x}_z^+} &= (1-\alpha_y)\Phi^*_{\check{x}_y^-,\check{x}_z^+} + \alpha_y\Phi^*_{\check{x}_y^+,\check{x}_z^+}
\end{aligned} \tag{2.45}$$

$$\Phi(\mathbf{x}) \approx (1-\alpha_z)\Phi^*_{\check{x}_z^-} + \alpha_z\Phi^*_{\check{x}_z^+}, \tag{2.46}$$

where the notation $s^- = \lfloor s \rfloor$ and $s^+ = \lceil s \rceil$ is used to denote the lower and higher integer indices in the discrete grid. Note that if x_i, x_j and x_k are integers, $\boldsymbol{\alpha} = \mathbf{0}$ and this interpolation is the same as just querying the grid at $\phi(\mathbf{x}) = \Phi_{\check{x}_x,\check{x}_y,\check{x}_z}$.

Works like (Curless and Levoy 1996) or (Newcombe et al. 2011) fuse depth information from several frames in these volumetric grids. If required, gradients are usually computed using finite differences on these grids, but recent works have demonstrated advantages of directly storing gradient information in the grid (Sommer et al. 2022).

(Curless and Levoy 1996): *A volumetric method for building complex models from range images*

(Newcombe et al. 2011): *KinectFusion: Real-time dense surface mapping and tracking*

2.2.3 Truncated Signed Distance Functions (TSDFs)

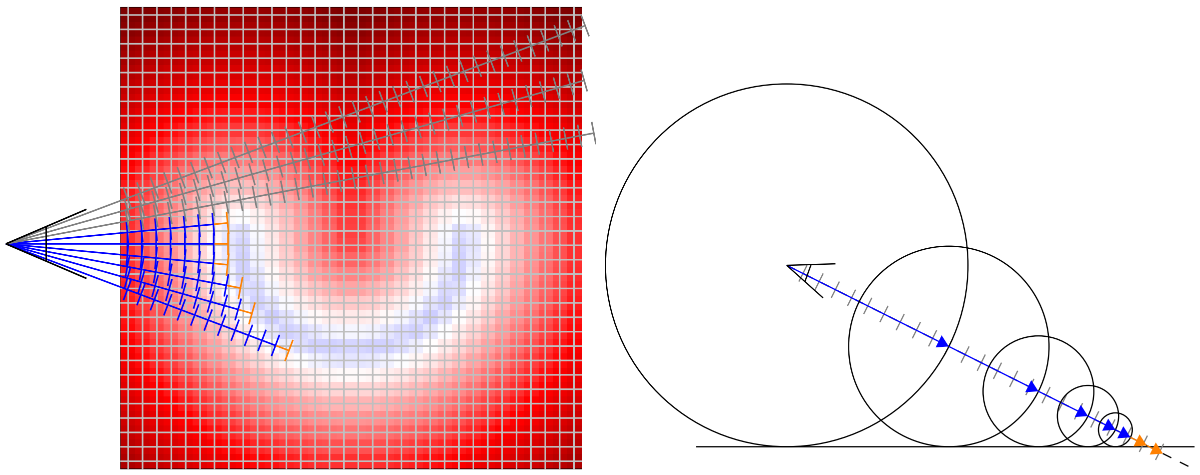
When fusing depth information in volumetric grids as in (Curless and Levoy 1996) or (Newcombe et al. 2011), one often uses a variant of the SDF called the *truncated signed distance function* (TSDF), which we denote by $\overline{\phi}$. This variant truncates the SDF at some specified threshold μ, *i.e.* all points $\mathbf{p}$ for which $|\phi(\mathbf{p})| > \mu$ get the truncated value of μ:

$$\overline{\phi}(\mathbf{p}) = \operatorname{clamp}(\phi(\mathbf{p}), \mu), \tag{2.47}$$

where $\operatorname{clamp}(s, \mu) := \min(\mu, \max(-\mu, s))$. In practice, the truncation threshold μ is chosen small enough to prevent interference between measurements from different sides of the observed surface, but large enough so that averaging multiple views (see Subsection 3.3.3) can remove noise from individual depth measurements (Curless and Levoy 1996; Newcombe 2012).

2.2.4 Explicit Surface Extraction

For some applications, like visualization of the geometry, extracting the explicit surface (*i.e.* points on the zero level set) is required. In this section, we will present two approaches that can be used for either rendering an oriented point cloud from a given camera pose or to extract a full triangle mesh from the SDF volume.



(a) Raycasting the bowl shape from Figure 2.1. **(b)** Sphere tracing can speed up the ray casting process.

Figure 2.3: (a) Ray casting. Starting from the center of projection of the camera, one ray for every pixel is traversed in steps (the ticks on the rays) until it crosses the isosurface (orange step). Rays not crossing the surface (gray) will not generate points. (b) Sphere tracing. One way to speed up the ray casting process is to take adaptive step sizes according to the SDF value from the last sample (blue arrows) and only switching to fixed step sizes (orange arrows) close to the surface. This yields far fewer steps than directly using the same small fixed step size (gray ticks).

Ray Casting. If only the oriented point cloud from a single given camera pose is required (*e.g.* to render the geometry from a single view), ray casting can be performed to find the isosurface of the SDF (Parker et al. 1998). The idea of this approach is to march along a ray for each pixel from the camera center through the domain of the SDF until the ray intersects the desired isosurface (see Figure 2.3a).

(Parker et al. 1998): *Interactive ray tracing for isosurface rendering*

The ray $\mathbf{r}(\mathbf{u}, t) = t\mathbf{K}^{-1}\pi^{-1}(\bar{\mathbf{u}})$ for a pixel $\mathbf{u} \in \Omega$, where Ω denotes the image pixel domain, is typically traversed in fixed step sizes Δt (after computing the step t_{min} entering the SDF volume), evaluating the SDF at every step (ticks on the rays in Figure 2.3a). Once the SDF values cross the isosurface (orange steps in Figure 2.3a) between steps $t^+(\mathbf{u})$ and $t^+(\mathbf{u}) + \Delta t$:

$$t^+(\mathbf{u}) = \min_{k \in \mathcal{K}(\mathbf{u})} t_{\text{min}} + k\Delta t, \tag{2.48}$$

where

$$\mathcal{K}(\mathbf{u}) = \Big\{k \in \mathbb{N} \;\Big|\; \phi\left(\mathbf{r}\left(\mathbf{u}, t_{\text{min}} + k\Delta t\right)\right) > 0, \\ \phi\left(\mathbf{r}\left(\mathbf{u}, t_{\text{min}} + (k+1)\Delta t\right)\right) \leq 0\Big\} \tag{2.49}$$

is the set of steps crossing the zero-isosurface for pixel $\mathbf{u}$, the location of the surface point is known to be between the points $\mathbf{r}(\mathbf{u}, t^+(\mathbf{u}))$ and $\mathbf{r}(\mathbf{u}, t^+(\mathbf{u}) + \Delta t)$. While the exact location can be computed by solving a ray/triangle intersection (Parker et al. 1998), Newcombe et al. (2011) propose to approximate the step $t^*(\mathbf{u})$ intersecting the isosurface between $t^+(\mathbf{u})$ and $t^+(\mathbf{u}) + \Delta t$:

$$t^*(\mathbf{u}) = t^+(\mathbf{u}) - \frac{\Delta t \phi\left(\mathbf{r}\left(\mathbf{u}, t^+(\mathbf{u})\right)\right)}{\phi\left(\mathbf{r}\left(\mathbf{u}, t^+(\mathbf{u}) + \Delta t\right)\right) - \phi\left(\mathbf{r}\left(\mathbf{u}, t^+(\mathbf{u})\right)\right)}. \tag{2.50}$$

The surface point and normal generated from the raycast are then given by

$$\mathbf{p}(\mathbf{u}) = \mathbf{r}\left(\mathbf{u}, t^*(\mathbf{u})\right) \quad \text{and} \quad \mathbf{n}(\mathbf{u}) = \nabla_{\mathbf{p}}\phi\left(\mathbf{p}\left(\mathbf{u}\right)\right). \tag{2.51}$$

Rays that leave the volume without crossing the zero-isosurface ($t(\mathbf{u}) > t_{\max}$; gray rays in Figure 2.3a) or that cross the isosurface from the wrong side[8] will not generate a surface measurement. We thus set $t^*(\mathbf{u}) = \infty$ for these rays and can define the rendered surface mask as

8: *i.e.*, $\phi(\mathbf{r}(\mathbf{u}, t)) < 0$ and $\phi(\mathbf{r}(\mathbf{u}, t + \Delta t)) > 0$, which can happen after incremental mapping as explained in Subsection 3.3.3

$$\begin{aligned} m : \Omega &\to \{0, 1\} \\ \mathbf{u} &\mapsto \begin{cases} 1, & \text{if } t^*(\mathbf{u}) < \infty \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{2.52}$$

A fixed step size Δt has to be chosen small enough so that even for thin surfaces the rays will encounter negative SDF values with high probability[9]. The process can be sped up by choosing adaptive step sizes as large as $\phi(r(\mathbf{u}, t))$ from the last step as long as the distance to the surface $\phi(r(\mathbf{u}, t))$ is more than some threshold, which is often referred to as *sphere tracing* (Hart 1996), see Figure 2.3b[10]. For TSDF volumes (see Subsection 2.2.3), by a similar argument, efficient ray casting can choose the step size as large as the truncation value initially by reasoning that at least one nun-truncated value must appear before crossing the surface (Newcombe et al. 2011).

9: *i.e.*, close to the thickness of the thinnest surface to be reconstructed

(Hart 1996): *Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces*

10: The threshold is required to actually *cross* the surface at some point.

Once the 3D point on the surface is found, the SDF allows for computing the surface normal as the gradient of the SDF at that location (see Equation (2.30)). This gradient can either be computed analytically in the case of parametric SDFs (Subsection 2.2.1) or via finite differences for volumetric grids (Subsection 2.2.2).

Marching Cubes. For applications like evaluating surface accuracy, physics simulation, or visualization, the full object surface is required. We can extract the object surface as triangular meshes from SDFs using the Marching Cubes algorithm (Lorensen and H. E. Cline 1987) after computing the SDF values on a regular grid (see Subsection 2.2.2). Triangular meshes $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ consist of a set of vertices $\mathcal{V} = \{\mathbf{v}_i\}_{i=0}^{N} \subset \mathbb{R}^3$ and a set of faces $\mathcal{F} \subset \{0, \ldots, N\}^3$ describing the connectivity between the vertices as vertex indices belonging to a single triangle. The idea behind this algorithm is that signed distance values on a regular grid classify grid nodes as *inside* or *outside* the surface. From this classification one can identify edges in the grid on which vertices lie and argue about connectivity between these vertices.

(Lorensen and H. E. Cline 1987): *Marching cubes: A high resolution 3D surface construction algorithm*

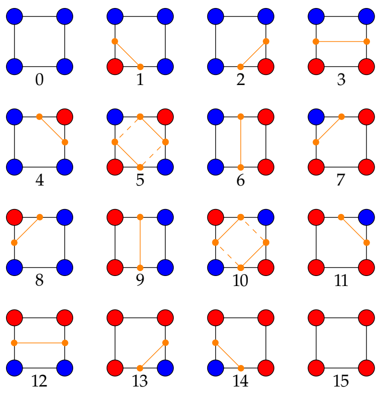
Figure 2.4a illustrates the corner configurations in the 2D case and the edges they generate[11]. Note that there are two corner configurations (cases 5 and 10) can generate two possible edge configurations each and require disambiguation[12]. An example for the zero-isosurface extracted from the bowl SDF from Figure 2.1 is shown in Figure 2.4b.

11: In 2D, the connections between vertices are not triangles but edges containing only 2 vertices.

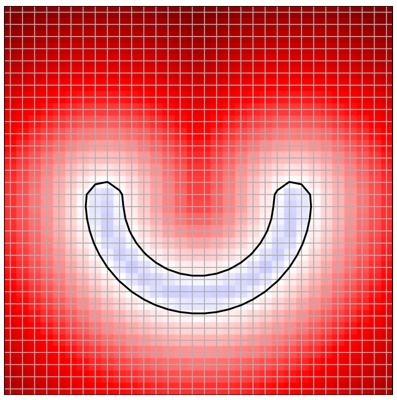
12: *e.g.* by computing the SDF value in the center of the square or always connecting positive or negative regions as illustrated by the connected lines in Figure 2.4a

In three dimensions, the number of classification cases increases, but the general concept remains the same[13]. By classifying the corners of 3D cubes one can compute which edges of the cube intersect the zero level set of the SDF. Depending on which set of edges intersects the isosurface triangles can be created to separate corners *inside* and *outside* the surface. The vertex locations can then be refined by interpolating the location on the edge according to the SDF values at its ends. Similar to ray casting one can also compute surface normals from the gradient of the SDF, which allow for smooth shading when rendering the reconstructed surface.

13: Although the number of ambiguous cases increases and more care needs to be taken to avoid holes in the reconstruction (Newman and Yi 2006)



(a) Square classification cases for marching squares.



(b) Marching squares applied to the bowl from Figure 2.1.

Figure 2.4: Marching squares. (a) The 16 square classification cases for the marching squares algorithm in 2D. (b) Zero-isosurface for the bowl in Figure 2.1 as extracted by the marching squares algorithm.

Parallel Computation. Both ray casting and the marching cubes algorithm can be accelerated using modern GPU hardware. For ray casting, each image pixel can be trivially processed in parallel. In marching cubes, both the initial cube classification and the creation of triangles can be performed in parallel after allocating the necessary memory buffers. We implemented both methods for surface extraction using CUDA[14] in the course of the projects contained in this thesis.

14: `https://developer.nvidia.com/cuda-toolkit`

Differentiable Surfaces. For some applications it is desirable to compute gradients $\frac{\partial E}{\partial \mathbf{z}}$ of some loss E formulated as a sum over losses on the extracted surface points $\mathbf{v}$ with respect to the shape parameters $\mathbf{z}$ of the underlying SDF (Subsection 2.2.1). Remelli et al. (2020) proposed computing this gradient via the chain rule as

(Remelli et al. 2020): *MeshSDF: Differentiable Iso-Surface Extraction*

$$\frac{\partial E}{\partial \mathbf{z}} = \sum_{\mathbf{v}\in\mathcal{V}} \frac{\partial E}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \phi} \frac{\partial \phi}{\partial \mathbf{z}}, \tag{2.53}$$

where all parts except for $\frac{\partial \mathbf{v}}{\partial \phi}$ are known if E is differentiable with respect to $\mathbf{v}$ and the SDF ϕ is differentiable with respect to the shape parameters $\mathbf{z}$ as in Subsection 2.2.1.

They further reason by infinitesimal perturbance $\Delta\phi$ of the surface ϕ that the gradient $\frac{\partial \mathbf{v}}{\partial \phi}$ is given by the negative surface normal at $\mathbf{v}$, which is the same as the negative gradient of the SDF (see Equation (2.30)):

$$\frac{\partial \mathbf{v}}{\partial \phi}(\mathbf{v}) = -\mathbf{n}(\mathbf{v}) = -\nabla\phi(\mathbf{v}). \tag{2.54}$$

This allows to compute the gradient $\frac{\partial E}{\partial \mathbf{z}}$ as

$$\frac{\partial E}{\partial \mathbf{z}} = \sum_{\mathbf{v}\in\mathcal{V}} -\frac{\partial E}{\partial \mathbf{v}} \nabla_{\mathbf{v}}\phi(\mathbf{v},\mathbf{z})\nabla_{\mathbf{z}}\phi(\mathbf{v},\mathbf{z}). \tag{2.55}$$

2.3 Depth Image Capture and Point Cloud Computation

Throughout this thesis, we will work with videos and images captured by *RGB-D cameras*. These cameras acquire depth images[15] in addition to color and have become more and more accessible in the last decades. Newcombe (2012) provides a good overview of the advent of this technology since the early 2000s, from early research prototypes (Rusinkiewicz et al. 2002) to commodity hardware like the Mircosoft Kinect or Asus Xtion cameras (both based on the PrimeSense sensor) in the early 2010s. In this thesis, we will use data sets that were recorded with these cameras (Rünz and Agapito 2017; Sturm et al. 2012). We also record our own real-world data using Intel RealSense cameras (Keselman et al. 2017). In short, these cameras use a known set of calibrated stereo cameras with efficient correspondence search to estimate depth from multiple view geometry (Hartley and Zisserman 2004). To overcome the absence of features in uniformly colored surfaces, they are often extended with *active sensing* by, *e.g.*, projecting a pattern in the infrared spectrum into the scene. The PrimeSense sensor uses only one infrared camera and models the projector as a "virtual camera". Correspondence search then reduces to finding features from the known projected pattern in the camera image as in (Rusinkiewicz et al. 2002). This method might fail in the presence of strong external lighting[16], as the projected pattern might be too weak to be detected. Thus, the more recent Intel RealSense product line comes with stereo infrared cameras and an additional infrared projector (Keselman et al. 2017). An alternative active sensing technology for depth cameras is time-of-flight (ToF), *i.e.*, measuring the time between sending a light pulse and recording it with a camera. Newer consumer-grade cameras like the second-generation Microsoft Kinect or the Azure Kinect rely on this technology for depth estimation (Sarbolandi et al. 2015).

15: *i.e.* pixel-wise distance to the camera

(Newcombe 2012): *Dense visual SLAM*

(Rünz and Agapito 2017): *Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects*

(Sturm et al. 2012): *A benchmark for the evaluation of RGB-D SLAM systems*

(Hartley and Zisserman 2004): *Multiple View Geometry in Computer Vision*

16: *e.g.*, in outdoor scenarios

Given the pixel-wise depth and the calibrated camera intrinsic matrix

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.56}$$

the 3D points corresponding to each pixel can be recovered by inverting the camera projection. To see how this is done, we first explain how points in 3D are projected to image pixels (Hartley and Zisserman 2004). Typically, we assume a pinhole camera model in which the matrix **K** in Equation (2.56) relates camera and pixel coordinates. The camera coordinate frame is chosen so that the camera looks along the positive z axis and x and y axes are aligned with the image axes. We can then define the function π for projecting the point $\mathbf{p} = (p_x, p_y, p_z)^\top$ in camera coordinates is projected to pixel coordinates $\mathbf{u} = (u_x, u_y)^\top$:

$$\mathbf{u} = \pi(\mathbf{p}) = \left(\frac{\mathbf{Kp}}{p_z}\right)_{1:2} = \left(\begin{pmatrix} \frac{f_x p_x + c_x p_z}{p_z} \\ \frac{f_y p_y + c_y p_z}{p_z} \\ \frac{p_z}{p_z} \end{pmatrix}\right)_{1:2} = \begin{pmatrix} f_x \frac{p_x}{p_z} + c_x \\ f_y \frac{p_y}{p_z} + c_y \end{pmatrix}, \tag{2.57}$$

where the subscript $\mathbf{x}_{1:2}$ denotes taking only the first two dimensions of the vector $\mathbf{x}$. From Equation (2.57) we can see how the entries of the camera intrinsic matrix $\mathbf{K}$ in Equation (2.56) affect the projection: f_x and f_y scale the normalized x and y coordinates from world to pixel units and c_x and c_y represent the offset of the center of projection from the origin of the pixel coordinate space.

As the positive z-axis of the camera coordinate frame is aligned with the viewing axis, the z-coordinate p_z of $\mathbf{p}$ is the distance of $\mathbf{p}$ to the camera in that direction, *i.e., the projective distance* of $\mathbf{p}$. We define the depth of $\mathbf{p}$ as this distance and denote the *depth map* of a surface $\mathcal{S} \subset \mathbb{R}^3$ as:

$$\begin{aligned} d : \Omega &\to \mathbb{R} \\ \mathbf{u} &\mapsto \begin{cases} \min_{\mathbf{p} \in \mathcal{P}_\mathbf{u}} p_z, & \text{if } \mathcal{P}_\mathbf{u} \neq \emptyset \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{2.58}$$

where $\mathcal{P}_\mathbf{u} = \{\mathbf{p} \in \mathcal{S} \mid \pi(\mathbf{p}) = \mathbf{u}\}$. The depth map d thus maps each pixels $\mathbf{u} \in \Omega \subset \mathbb{R}^2$ to the depth of the closest point $\mathbf{p} \in \mathcal{S}$ that projects to it or to 0 if there is no point projecting to that pixel[17].

17: This mapping to 0 for missing measurements allows for easily filtering them, as the unprojected points below stay at the origin for these pixels.

Given a depth measurement $z = d(\mathbf{u})$ for pixel coordinates $\mathbf{u} = (u_x, u_y)^\top$, we can invert Equation (2.57) as follows to compute the 3D point $\mathbf{p}$:

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \pi^{-1}(\mathbf{u}, z) = z\mathbf{K}^{-1}\overline{\mathbf{u}} = z\mathbf{K}^{-1} \begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix} = z \begin{pmatrix} \frac{u_x - c_x}{f_x} \\ \frac{u_y - c_y}{f_y} \\ 1 \end{pmatrix}, \tag{2.59}$$

where $\overline{\mathbf{u}} = (u_x, u_y, 1)^\top$ are the homogeneous coordinates for $\mathbf{u}$ (see Equation (2.18)). Applying the "unprojection" operation in Equation (2.59) to all pixels produces a point cloud in camera coordinates.

We can further compute surface normals from this point cloud as the vector orthogonal to two tangent vectors to the surface (Besl and Jain 1986). These tangent vectors can be computed as the gradients of the point cloud in x and y direction

$$\nabla_x \mathbf{p} = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial z}{\partial x} \end{pmatrix} \quad \text{and} \quad \nabla_y \mathbf{p} = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial z}{\partial y} \end{pmatrix}, \tag{2.60}$$

yielding the surface normal as

$$\mathbf{n} = \frac{\nabla_x \mathbf{p} \times \nabla_y \mathbf{p}}{\left\| \nabla_x \mathbf{p} \times \nabla_y \mathbf{p} \right\|} = \frac{\left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1 \right)^\top}{\left\| \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1 \right)^\top \right\|}. \tag{2.61}$$

EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association 3

The contents of this chapter are based on the peer-reviewed conference publication

©2019 IEEE. Reprinted, with permission, from **Strecke, Michael** and Joerg Stueckler (2019). 'EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00596, (Strecke and Stueckler 2019),

with the following co-author contributions:

	Ideas	**Experiments**	**Analysis**	**Writing**
Michael Strecke	60%	100%	80%	65%
Jörg Stückler	40%	0%	20%	35%

This chapter contains tables and according descriptions that were originally part of the supplementary material of the conference publication.

Compared to the conference publication, this chapter contains more detailed preliminaries and unified notation with the rest of the thesis.

3.1 Introduction

In a first step towards physically plausible reconstruction of dynamic scenes, we develop a method that uses RGB-D images (Section 2.3) as input and reconstructs the 3D objects as signed distance function (SDF; Section 2.2) models in this chapter. We further track the motion of the camera and the object models in SE(3) (Section 2.1).

As mentioned before, RGB-D cameras are popular devices for dense visual 3D scene acquisition (Bylow et al. 2013; Kerl et al. 2013; Newcombe et al. 2011). Most of these approaches to simultaneous localization and mapping (SLAM) with RGB-D cameras only map the static part of the environment and localize the camera within this map. However, in many applications of robotics and augmented reality (AR), agents[1] interact with the environment and hence the environment state is *dynamic*. While some approaches filter dynamic objects (Keller et al. 2013; Scona et al. 2018) as outliers from the measurements, we are interested in reconstructing and tracking them in this thesis. This potentially to enables interactions of autonomous agents with observed dynamic objects or using the reconstruction of dynamic objects in AR applications.

1: *i.e.*, humans or other robotic agents

In this chapter, we propose a novel approach to dynamic SLAM that maps and tracks objects in the scene. We detect objects by semantic instance segmentation of the images and subsequently perform tracking and mapping of the static background and the objects. In previous approaches (Rünz and Agapito 2017; Rünz, Buffier, et al. 2018; B. Xu et al. 2019), data association of measurements to objects is either solved through image-based instance segmentation or by ray casting in the maps. We propose to determine the unknown association of pixels to objects in a probabilistic expectation maximization (EM; Bishop 2007) formulation which estimates

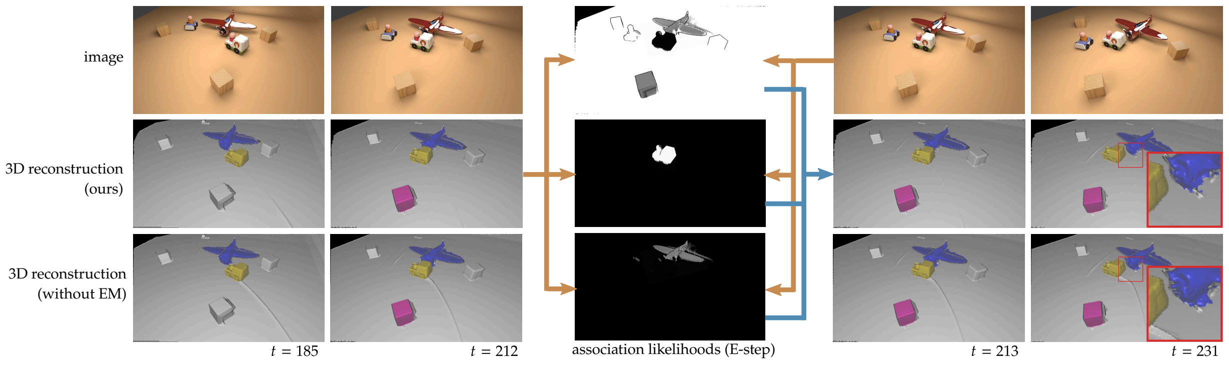


Figure 3.1: Dynamic object-level SLAM with probabilistic data association. We infer the association likelihood of pixels with objects in an expectation-maximization framework. The probabilistic data association improves accuracy and robustness of tracking and mapping. It implicitly handles occlusions. The E-step (orange arrows) estimates the association likelihoods based on the data likelihood of the current image given the latest object maps and poses. In the M-step (blue arrows) poses and map are updated with the measurements according to the association likelihoods. Association likelihoods are visualized for the background (top), the train (middle) and the airplane (bottom). The moving truck occludes the table and the airplane which is well recovered by the association likelihoods. Without association likelihoods, artifacts are integrated into the map due to wrong data association.

the soft association likelihood from the likelihood of the measurements in our map representation. The probabilistic association uses the distance of depth measurements to object surfaces as additional geometric cue and implicitly handles occlusions for object segmentation, tracking, and mapping (see Figure 3.1). We represent the object maps by volumetric SDFs and augment the maximum likelihood integration of the SDF from depths to incorporate their association likelihood. The probabilistic data association facilitates the direct alignment of the depth maps with the SDF object maps for object tracking. This avoids projective data association through ray casting which is needed for the iterative closest point (ICP) algorithm used in other approaches. In our experiments, we evaluate our approach on several datasets and demonstrate superior performance over the state-of-the-art methods. Our results demonstrate that proper probabilistic treatment of data associations is a key ingredient to robust object-level SLAM in dynamic scenes. In summary, we make the following contributions in our work,

- We propose a probabilistic EM formulation for dynamic object-level SLAM that naturally leads to data association and occlusion handling strategies.
- Based on our EM formulation, we approach multi-object tracking as direct alignment of RGB-D images with SDF object representations and evaluate this tracking approach for dense dynamic SLAM.
- Our approach achieves state-of-the-art performance on several datasets for dynamic object-level SLAM.
- We provide the source code of our approach to allow other researchers to reproduce our results and build upon them in future research[2] .

2: `https://github.com/EmbodiedVision/emfusion`

3.2 Related Work

Static SLAM. Simultaneous localization and mapping (SLAM) with RGB-D sensors has seen tremendous progress quickly after the sensors have become broadly available on the market. KinectFusion (Newcombe et al. 2011) is a prominent approach that incrementally tracks the camera motion and maps the environment densely in volumetric signed distance function (SDF) grids. Several other RGB-D SLAM approaches have been proposed that differ in tracking methods such as ICP (Newcombe et al. 2011), direct image alignment (Kerl et al. 2013) or SDF alignment (Bylow et al. 2013), and map representations such as surfels (Keller et al. 2013) or keyframes (Kerl et al. 2013). Extensive research has gone into scaling the approaches to large environments (Nießner et al. 2013; Whelan, Kaess, et al. 2012) or supporting loop-closing (Kerl et al. 2013; Whelan, Leutenegger, et al. 2015) to reduce drift. Some approaches also consider the creation of object-level maps (McCormac et al. 2018; Salas-Moreno et al. 2013), but assume the objects to remain static. Other approaches consider dynamic environments, but only reconstruct the static part and treat dynamic parts as outliers (Keller et al. 2013; Scona et al. 2018; Whelan, Kaess, et al. 2012; Whelan, Leutenegger, et al. 2015).

Dynamic SLAM. Research on tracking and reconstruction of articulated objects such as human body parts (Taylor et al. 2016; Tzionas and Gall 2016) or robots (Cifuentes et al. 2017; Schmidt et al. 2015) is related to dynamic SLAM. Recently, some RGB-D SLAM methods have been proposed that represent and track moving rigid objects. An early approach extends key frame-based RGB-D SLAM to object-level dynamic SLAM (Stückler and Behnke 2013). The approach segments moving objects between RGB-D frames (Stückler and Behnke 2015) and builds a key frame pose graph for associated motion segments in the key frames. Co-Fusion (Rünz and Agapito 2017) extends surfel[3]-based representations for moving objects. It combines geometric and motion segmentation to detect moving objects. Tracking camera motion with respect to the scene background and the objects is based on ICP (Besl and McKay 1992) alignment using geometry and color cues. MaskFusion (Rünz, Buffier, et al. 2018) does not use motion segmentation but fuses geometric and deep-learning based instance segmentation (Mask R-CNN; He et al. 2017). MID-Fusion (B. Xu et al. 2019) follows a similar approach, but represents the 3D map in volumetric SDFs using octrees. We also represent objects in using SDFs but formulate tracking using efficient but accurate direct SDF alignment (Bylow et al. 2013) and further propose novel strategies for handling occlusions and disocclusions.

3: *i.e.*, surface element, a point with associated normal, typically represented as a fixed-size disk

3.3 Preliminaries

3.3.1 Image Preprocessing and Projection

We apply a bilateral filter on the raw depth images to smooth depth quantization artifacts. From the filtered depth maps $\mathbf{z}_t = d_t(\mathbf{u})$ (see Equation (2.58)) we compute 3D point coordinates $\mathbf{p} = \pi^{-1}(\mathbf{u}, z_{t,\mathbf{u}}) \in \mathbb{R}^3$ at each pixel $\mathbf{u} \in \Omega \subset \mathbb{R}^2$, where we define $\pi^{-1}(\mathbf{u}, z_{\mathbf{u}}) := z_{\mathbf{u}}\, \mathbf{K}^{-1} \left(u_x, u_y, 1\right)^\top$ and $\mathbf{K}$ is the camera intrinsic matrix of the calibrated pinhole camera (see Section 2.3).

3.3.2 Map Representation

We represent background and objects maps by volumetric SDFs. The SDF $\phi(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ yields the signed distance of a point $\mathbf{p}$ to the closest surface represented by the SDF (see Section 2.2). The object surface is determined by the zero level-set $\left\{\mathbf{p} \in \mathbb{R}^3 \,\middle|\, \phi(\mathbf{p}) = 0\right\}$ of the SDF. We implement the volumetric SDF through discretization in a 3D grid of voxels. The SDF value at a point within the grid is found through trilinear interpolation (see Subsection 2.2.2). We maintain several SDF volumes: one background volume (resolution 512^3) and several smaller SDF volumes, one for each detected object (initialized with a size of 64^3 and resized as needed, see Subsection 3.3.5).

3.3.3 Depth Image Fusion

(Curless and Levoy 1996): *A volumetric method for building complex models from range images*

Curless and Levoy (1996) present an approach for fusing measurements from multiple calibrated depth frames efficiently in a consistent volumetric model. Their formulation computes the SDF ϕ on a discrete grid as the weighted average of multiple depth images $\{d_t\}_{t=1}^{T}$:

$$\phi(\mathbf{p}) = \frac{1}{\sum_{t=1}^{T} w_t(\mathbf{p})} \sum_{t=1}^{T} w_t(\mathbf{p}) \check{\phi}_t(\mathbf{p}), \tag{3.1}$$

where w_t is a weight describing the confidence in measurement d_t and the projective SDF measurement $\check{\phi}_t$ for depth map d_t is defined as

$$\check{\phi}_t(\mathbf{p}) = d_t\left(\mathbf{u}_\mathbf{p}\right) - \frac{1}{\lambda_\mathbf{u}} \left\|\mathbf{p} - \mathbf{t}_t\right\|. \tag{3.2}$$

In Equation (3.2), $\mathbf{u}_\mathbf{p} = \pi\left(\mathbf{T}_t^{-1}\overline{\mathbf{p}}\right)$, where π denotes the perspective projection of the point $\mathbf{p} \in \mathbb{R}^3$ from 3D camera to 2D pixel coordinates (see Section 2.3) and

$$\lambda_\mathbf{u} = \left\| \left(\frac{u_x - c_x}{f_x}, \frac{u_y - c_y}{f_y}, 1 \right)^\top \right\|, \tag{3.3}$$

where u_x and u_y are the pixel coordinates in $\mathbf{u}_\mathbf{p}$ and f_x, f_y, c_x and c_y are entries in the intrinsic matrix $\mathbf{K}$ (see Equation (2.56)). It allows scaling the distance between the point and the camera center to the projective distance along the viewing axis, as can be seen from Equation (2.59). The transformation

$$\mathbf{T}_t = \begin{pmatrix} \mathbf{R}_t & \mathbf{t}_t \\ \mathbf{0} & 1 \end{pmatrix} \in \mathrm{SE}(3), \tag{3.4}$$

with $\mathbf{R}_t \in \mathrm{SO}(3)$ and $\mathbf{t}_t \in \mathbb{R}^3$ is the camera transformation of frame t (see Section 2.1) and is used to transform the point $\mathbf{p}$ to camera coordinates before projecting it to image pixels. Equation (3.2) first computes the pixel $\mathbf{u}_\mathbf{p} = \pi\left(\mathbf{T}_t^{-1}\overline{\mathbf{p}}\right)$ into which the point $\mathbf{p}$ projects under the camera transformation $\mathbf{T}_t$. It then computes the projective signed distance along the ray of sight by comparing the measured depth d_t at the pixel $\mathbf{u}_\mathbf{p}$ with the actual projective distance between $\mathbf{p}$ and the camera center $\mathbf{t}_t$. For points in front of the measured surface, $d_t\left(\mathbf{u}_\mathbf{p}\right)$ will be larger than $\frac{1}{\lambda_\mathbf{u}}\left\|\mathbf{p} - \mathbf{t}_t\right\|$, yielding positive values for $\check{\phi}_t(\mathbf{p})$, while points behind the surface (*i.e.*, *inside* objects) will get negative values for $\check{\phi}_t(\mathbf{p})$, consistent with the notion in Section 2.2. Equation (3.1) is equivalent to the least squares fit of the surface given the measurements (Curless and Levoy 1996) and corresponds to a maximum likelihood estimate (Newcombe 2012). In scenarios like the ones considered in this thesis, we are often interested in *online* mapping. Thus, we do not want to wait for all measurement frames to build the map, but allow for *incremental* mapping. By using an additional volume W to collect the accumulated weights w_t, Equation (3.1) can be reformulated to compute the estimates ϕ_t and W_t from the previous estimates ϕ_{t-1} and W_{t-1} and the new measurements $\check{\phi}_t$ and w_t incrementally:

$$\begin{aligned} \phi_t(\mathbf{p}) &= \frac{W_{t-1}(\mathbf{p})\phi_{t-1}(\mathbf{p}) + w_t(\mathbf{p})\check{\phi}_t(\mathbf{p})}{W_{t-1}(\mathbf{p}) + w_t(\mathbf{p})} \\ W_t(\mathbf{p}) &= W_{t-1}(\mathbf{p}) + w_t(\mathbf{p}). \end{aligned} \tag{3.5}$$

Because the depth measurement does not contain information about the area behind the surface or its "thickness", the SDF ϕ is typically replaced with the truncated version $\overline{\phi}$ (see Subsection 2.2.3). The truncation parameter μ in Equation (2.47) is chosen small enough to prevent interference of conflicting measurements when surfaces are seen from different sides but large enough to allow noise removal by the weighted average fusion (Curless and Levoy 1996; Newcombe 2012). The weights w_t are adjusted so that $w_t(\mathbf{p}) = 0$ if $\check{\phi}(\mathbf{p}) < -\mu$.

We will further cap the fused weights W at a maximum value $\overline{W}$ to allow the SDF volume to slowly adapt to measurements inconsistent with the present map by setting

$$W_t(\mathbf{p}) = \min\left(W_{t-1}(\mathbf{p}) + w_t(\mathbf{p}), \overline{W}\right) \tag{3.6}$$

in Equation (3.5). While we use the truncated SDF in this chapter, we still denote it by ϕ instead of $\overline{\phi}$ to avoid notational clutter.

The cumulative integration in Equation (3.5) can be parallelized efficiently on modern GPU hardware (Newcombe 2012; Newcombe et al. 2011). We use this formulation in this chapter for building volumetric models for individual moving objects and the static background scene.

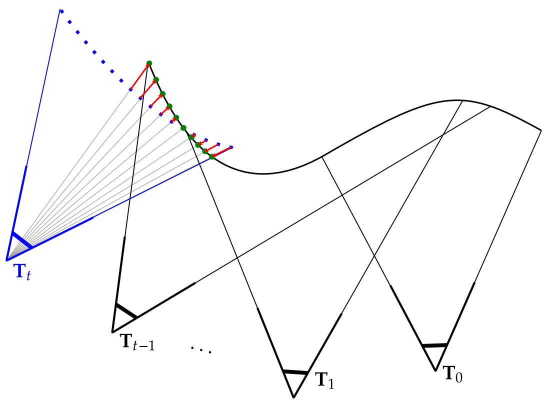
3.3.4 Dense Volumetric RGB-D SLAM

Newcombe et al. (2011) combined the range image fusion presented in Subsection 3.3.3 with frame-to-model tracking for real-time incremental RGB-D SLAM. Given the camera pose $\mathbf{T}_{i-1}$ and TSDF model ϕ_{i-1} from the previous frame, they first extract the surface from the model using ray casting (see Subsection 2.2.4) and then set up an optimization problem to align the oriented model point cloud from the previous frame $\mathcal{S}_{t-1}$ with the point cloud $\check{\mathcal{S}}_t$ computed from the current depth map (see Section 2.3). They use the projective point to plane distance

(Newcombe et al. 2011): *KinectFusion: Real-time dense surface mapping and tracking*

$$E(\mathbf{T}_t) = \sum_{\mathbf{u}\in\Omega} \left\| \left(\mathbf{T}_t\check{\mathbf{p}}(\mathbf{u}) - \mathbf{p}(\check{\mathbf{u}})\right)^\top \mathbf{n}(\check{\mathbf{u}}) \right\|, \tag{3.7}$$

where Ω denotes the depth image's pixel domain, $\check{\mathbf{p}}(\mathbf{u})$ is the point computed from depth map pixel $\mathbf{u}$ according to Equation (2.59) and



(a) ICP alignment **(b)** SDF alignment

Figure 3.2: ICP and SDF alignment for finding the pose $\mathbf{T}_t$ for a new frame. (a) In ICP alignment (Newcombe et al. 2011), each point from the depth map (blue) is matched to a rendered point from the current pose estimate (green). (b) In SDF alignment (Bylow et al. 2013; Canelhas et al. 2013), the alignment objective is directly aligning the points from the depth map with the isosurface.

$\mathbf{p}(\check{\mathbf{u}})$ and $\mathbf{n}(\check{\mathbf{u}})$ are the model point and normal computed by raycasting (Subsection 2.2.4) for the pixel $\check{\mathbf{u}} = \pi\left(\widetilde{\mathbf{T}}^t_{t-1}\check{\mathbf{p}}(\mathbf{u})\right)$, where $\widetilde{\mathbf{T}}^t_{t-1}$ denotes the incremental transformation between frame $t-1$ and t such that $\mathbf{T}_t = \widetilde{\mathbf{T}}^t_{t-1}\mathbf{T}_{t-1}$. Equation (3.7) matches points between the model and the captured depth map that project into the same pixel (see Figure 3.2a).

(Bylow et al. 2013): *Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions*
(Canelhas et al. 2013): *SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images*

We follow Bylow et al. (2013) and Canelhas et al. (2013) and use direct SDF alignment for tracking camera and object poses (see Figure 3.2b). The optimization objective encourages the points reconstructed from the depth map to project to the zero-isosurface of the SDF and is defined as

$$E\left(\mathbf{T}_t\right) = \sum_{\mathbf{u}\in\Omega} \phi_{t-1}\left(\mathbf{T}_t\check{\mathbf{p}}(\mathbf{u})\right)^2, \tag{3.8}$$

where ϕ_{t-1} is the SDF of the reconstructed background model after frame $t-1$. We can optimize Equation (3.8) using methods for solving least-squares problems. We will use a more general formulation including weights for individual pixels:

$$E\left(\mathbf{T}_t\right) = \sum_{\mathbf{u}\in\Omega} w_{\mathbf{u}}\phi_{t-1}\left(\mathbf{T}_t\check{\mathbf{p}}\left(\mathbf{u}\right)\right)^2, \tag{3.9}$$

where we now use the index i for indexing image pixels. Equation (3.9) can be optimized by iterative reweighed least squares algorithms, and we employ the Levenberg-Marquardt method (Levenberg 1944; Marquardt 1963) for optimizing it.

(Levenberg 1944): *A method for the solution of certain non-linear problems in least squares*
(Marquardt 1963): *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*

3.3.5 Instance Detection and Segmentation

The previously explained method by Newcombe et al. (2011) only builds one global SDF model for the scene and tracks the camera pose relative to it. This limits the method to *static* environments. In this thesis however, we consider *dynamic* scenes in which we want to acquire models and motion trajectories for *individual objects*. We build upon Fusion++ (McCormac et al. 2018) for fusing separate SDF models for each moving object. Fusion++ was designed for robust tracking in static scenes by building object-level models and using them in a pose graph to track the camera poses. We

(McCormac et al. 2018): *Fusion++: Volumetric Object-Level SLAM* propose a different formulation for dynamic object tracking, but follow McCormac et al. (2018) for object model initialization and foreground mask fusion. For detecting possibly moving objects, we segment the input images using Mask R-CNN (He et al. 2017), which we execute on every 30th frame for computational efficiency. If detections are available for a frame, we try to match the detected masks with existing objects and initialize new models or update existing ones as described below.

(He et al. 2017): *Mask R-CNN*

Initialization. For initialization, objects are detected using a semantic instance segmentation method like Mask R-CNN (He et al. 2017), generating M semantic segmentation masks $\{\check{m}^i\}_{i=1}^M$ which are matched with existing volumes as explained below. Unmatched segmentations are denoted by $\check{m}^o$ with $o \in \{N+1, \dots\}$ for N existing objects. We use them to compute initialization point clouds from the depth map $\mathbf{z}$

$$\check{\mathscr{P}}^o = \left\{ \check{\mathbf{p}} = \pi^{-1}(\mathbf{u}, \mathbf{z}_\mathbf{u}) \;\middle|\; \mathbf{u} \in \check{\mathscr{M}}^o \right\}, \tag{3.10}$$

where $\check{\mathscr{M}}^o = \{\mathbf{u} \in \Omega \mid \check{m}^o(\mathbf{u}) = 1\}$. As RGB-only segmentation masks might include pixels that are not on the object and include potentially far-away points, we follow McCormac et al. (2018), who propose to use the component-wise 10th and 90th percentiles $\check{\mathbf{p}}_{10}^o$ and $\check{\mathbf{p}}_{90}^o$

$$\begin{aligned} \left(\check{p}_{10}^o\right)_i &= \min\left\{ \check{p}_i \;\middle|\; \check{\mathbf{p}} \in \check{\mathscr{P}}^o, \frac{\left|\left\{\check{\mathbf{p}}' \in \check{\mathscr{P}}^o \;\middle|\; \check{p}'_i \le \check{p}_i\right\}\right|}{\left|\check{\mathscr{P}}^o\right|} \le 0.1 \right\}, \\ \left(\check{p}_{90}^o\right)_i &= \min\left\{ \check{p}_i \;\middle|\; \check{\mathbf{p}} \in \check{\mathscr{P}}^o, \frac{\left|\left\{\check{\mathbf{p}}' \in \check{\mathscr{P}}^o \;\middle|\; \check{p}'_i \ge \check{p}_i\right\}\right|}{\left|\check{\mathscr{P}}^o\right|} \ge 0.9 \right\}, \end{aligned} \tag{3.11}$$

to determine the object size s^o and position $\mathbf{t}^o$ as

$$s^o = p \left\|\check{\mathbf{p}}_{90}^o - \check{\mathbf{p}}_{10}^o\right\|_\infty \quad \text{and} \quad \mathbf{t}^o = \frac{\check{\mathbf{p}}_{90}^o + \check{\mathbf{p}}_{10}^o}{2}, \tag{3.12}$$

where $\|\mathbf{x}\|_\infty = \max_{i \in \{1,\dots,d\}} x_i$ is the maximum norm of a vector $\mathbf{x} \in \mathbb{R}^d$ and p is a padding factor, which we set to 2.0 to account for erosion by the heuristic or parts of the surfaces that become visible in later frames. The TSDF volume is then initialized at position $\mathbf{t}^o$ as a cube with size s^o and a fixed resolution r^o of 64 in each dimension, yielding a voxel size of $v^o = \frac{s^o}{r^o}$. To account for potentially inaccurate masks, all depth measurements projecting into the volume are integrated using the formulas in Equation (3.5). The new volume is only initialized if its center $\mathbf{t}^o$ is within 5 m from the camera and the volumetric IoU with any other volume is lower than 0.5.

Volumetric Foreground Mask Fusion. In addition to the integration weight volume W[4], foreground and background weight volumes F and B are maintained for each object. The weights for voxel $\mathbf{v}$ are updated with the foreground probabilities $p_{\text{fg}}^o(\mathbf{u}_\mathbf{v} \mid \check{m}^o) = \check{m}^o(\mathbf{u}_\mathbf{v})$ from the masks matched with the same object, where $\mathbf{u}_\mathbf{v} = \pi\left(\left(\mathbf{T}_t^o\right)^{-1} \bar{\mathbf{v}}\right)$ with camera-to-

4: see Equation (3.5)

object transformation $\mathbf{T}_t^o$ is the pixel into which the voxel $\mathbf{v}$ projects:

$$\begin{aligned} F_t^o(\mathbf{v}) &= F_{t-1}^o + p_{\text{fg}}^o\left(\mathbf{u}_\mathbf{v} \mid \check{m}_t^o\right) \\ B_t^o(\mathbf{v}) &= B_{t-1}^o + \left(1 - p_{\text{fg}}^o\left(\mathbf{u}_\mathbf{v} \mid \check{m}_t^o\right)\right). \end{aligned} \tag{3.13}$$

The fused foreground probability for object o is then computed as

$$p_{\text{fg}}(\mathbf{v} \mid o) = \frac{F^o(\mathbf{v})}{F^o(\mathbf{v}) + B^o(\mathbf{v})}. \tag{3.14}$$

A point $\mathbf{v}$ in the object volume is considered foreground if $p_{\text{fg}}(\mathbf{v} \mid o) > 0.5$. When ray casting the objects, this foreground condition is added to the set $\mathcal{K}$ in Equation (2.49). The ray cast is performed in all object volumes and the ray with the shortest length (*i.e.*, the object in the foreground in case of occlusions) is taken to generate object masks and rendered points. For rays that do not hit objects or where the object surface is more than 5 cm behind the background surface, we render the background volume following McCormac et al. (2018).

Matching Existing Objects. When M semantic segmentation masks $\{\check{m}^i\}_{i=1}^M$ are generated by Mask R-CNN (He et al. 2017) for a new frame, they need to be matched to the N existing object model IDs $o \in \{1, \ldots N\}$. As the masks $\{\check{m}^i\}_{i=1}^M$ are in pixel space, we perform the matching in this space. We adapt the set $\mathcal{K}$ in Equation (2.49) to include the volumetric foreground probability (see above in Equation (3.14)) and ray cast[5] all SDFs ϕ^o, $o \in \{0, \ldots, N\}$, taking for each pixel the first visible surface across all volumes. We then compute object masks $\{m^o\}_{o=1}^N$ for the N object models currently present in the scene using Equation (2.52).

5: see Equation (2.48)

We determine the association of the segmentation masks computed from existing models with the detected segments by the intersection-over-union (IoU) measure:

$$\text{IoU}\left(\check{\mathcal{M}}^i, \mathcal{M}^o\right) = \frac{\left|\check{\mathcal{M}}^i \cap \mathcal{M}^o\right|}{\left|\check{\mathcal{M}}^i \cup \mathcal{M}^o\right|}, \tag{3.15}$$

where $\check{\mathcal{M}}^i = \left\{\mathbf{u} \in \Omega \,\middle|\, \check{m}^i(\mathbf{u}) = 1\right\}$ and $\mathcal{M}^o = \{\mathbf{u} \in \Omega \mid m^o(\mathbf{u}) = 1\}$ are the sets of pixels inside the detected and rendered masks, respectively, and $|\cdot|$ denotes the number of elements in a set. Mask $\check{m}^i$ is then matched to the object o^* with the largest overlap:

$$o_i^* = \underset{o \in \{1,\ldots,M\}}{\arg\max}\; \text{IoU}\left(\check{\mathcal{M}}^i, \mathcal{M}^o\right), \tag{3.16}$$

if $\text{IoU}\left(\check{\mathcal{M}}^i, \mathcal{M}^{o^*}\right)$ is larger than a matching threshold (0.2 in our experiments). For matched masks, we denote the segmentation mask matched with object o^* with $\check{m}^{o^*}$. Unmatched masks $\check{m}^i$ are used for initializing new objects as explained above and will be assigned with new object indices $o \in \{N+1, \ldots\}$. If points from new detections matched with an existing model fall outside the existing volume o^*, the volume is resized. We keep the voxel size v^{o^*} fixed and determine an increased r^{o^*} required to fit the new detection. To achieve this, we first compute the minimum size $\overline{r}^{o^*}$ that fits the new detection, then compute r^{o^*} as the smallest larger even resolution and finally compute the new volume size s^{o^*} from the resolution and the voxel size:

$$\overline{r}^{o^*} = \left\lceil \frac{p \left\| \mathbf{p}_{90}^{o^*} - \mathbf{p}_{10}^{o^*} \right\|_\infty}{v^{o^*}} \right\rceil, \quad r^{o^*} = \left\lfloor \frac{\overline{r}^{o^*} + 1}{2} \right\rfloor \cdot 2, \quad s^{o^*} = r^{o^*} v^{o^*}, \tag{3.17}$$

where $\mathbf{p}_{90}^{o^*}$ and $\mathbf{p}_{10}^{o^*}$ are computed as in Equation (3.11), replacing $\check{\mathscr{P}}^o$ by

$$\mathscr{P}^{o^*} = \left\{ \check{\mathbf{p}} = \pi^{-1}(\mathbf{u}, \mathbf{z_u}) \,\middle|\, \mathbf{u} \in \check{\mathscr{M}}^{o^*} \cup \mathscr{M}^{o^*} \right\}. \tag{3.18}$$

We then shift $\mathbf{t}^{o^*}$ by a multiple of v^{o^*} so that it is still in the center of the volume:

$$\mathbf{t}^{o^*} = \left\lfloor \frac{\frac{\mathbf{p}_{90}^{o^*} + \mathbf{p}_{10}^{o^*}}{2}}{v^{o^*}} \right\rfloor v^{o^*}. \tag{3.19}$$

Existence Probability. Since Mask R-CNN can deliver false detections, we follow (McCormac et al. 2018) and maintain an existence probability

$$p_{\text{ex}}(i) = \frac{E(i)}{E(i) + N(i)}, \tag{3.20}$$

similar to Equation (3.14), where for each frame with a Mask R-CNN segmentation available $E(i)$ is incremented if the object is matched to a segment and otherwise $N(i)$ is incremented. We delete objects where $p_{\text{ex}}(i) < 0.1$.

Semantic Labels. As Mask R-CNN is a *semantic* instance segmentation approach, it provides a probability distribution $p(l^o \mid \mathbf{i}_t)$ over class labels l^o for the matched object o in each RGB frame $\mathbf{i}_t, t \in \{1, \ldots, T\}$. We follow McCormac et al. (2018) and average these probabilities over all frames:

$$p(l_T^o \mid \mathbf{i}_1, \ldots, \mathbf{i}_T) = \frac{1}{T} \sum_{t=1}^{T} p(l^o \mid \mathbf{i}_t). \tag{3.21}$$

3.4 Method

Our dynamic SLAM approach performs incremental tracking and mapping of objects and the static background. We propose a probabilistic formulation for tracking and mapping of multiple objects which naturally leads to a principled method for data association and occlusion handling. We represent the 3D shape of objects and background in volumetric SDF representations which we estimate from depth images. New object instances are initially detected and segmented using a deep learning approach to appearance-based semantic instance segmentation (Mask R-CNN; He et al. 2017).

3.4.1 Probabilistic Dynamic Tracking and Mapping

We formulate SLAM as maximum likelihood estimation of the camera trajectory and the map from visual observations $\mathbf{z}_t = (z_{t,\mathbf{u}})_{\mathbf{u}\in\Omega}$, the depth images in the image domain Ω. The map is composed of separate TSDF volumes $\boldsymbol{\phi} := \left\{\phi^o\right\}_{o=0}^{N}$ for the background (ϕ^0) and N objects. In each camera frame at time t, we track the camera pose with regard to the objects and background with distinct poses $\mathbf{T}_t := \left\{\mathbf{T}_t^o\right\}_{o=0}^{N}$, $\mathbf{T}_t^o \in \mathrm{SE}(3)$. We choose incremental tracking and mapping in which we optimize the joint posterior likelihood of the map and the camera poses in the current frame, given all images so far,

$$\arg\max_{\boldsymbol{\phi},\mathbf{T}_t} p\left(\boldsymbol{\phi}, \mathbf{T}_t \mid \mathbf{z}_{1:t}\right) = \arg\max_{\boldsymbol{\phi},\mathbf{T}_t} p\left(\mathbf{z}_t \mid \boldsymbol{\phi}, \mathbf{T}_t\right) \, p\left(\boldsymbol{\phi} \mid \mathbf{z}_{1:t-1}\right) \, p\left(\mathbf{T}_t\right). \tag{3.22}$$

We optimize the posterior separately first for the camera pose, then for the map. By causality, each pixel measurement can only be attributed to one of the objects or the background, such that we also need to find the association of each pixel $\mathbf{u}$ in the image domain Ω to one of the objects. This association is a latent variable $\mathbf{c}_t = (c_{t,\mathbf{u}})_{\mathbf{u}\in\Omega} \in \mathscr{C}$ in our probabilistic model which we infer during the tracking and mapping. The set $\mathscr{C} = \{0, \ldots, N\}^{|\Omega|}$ is the set of all pixel-level association vectors.

3.4.2 Expectation Maximization Framework

(Bishop 2007): *Pattern recognition and machine learning, 5th Edition*

Expectation-maximization (EM; Bishop 2007) is a natural framework for our problem of finding the latent data association with the map and camera pose estimates. In EM, we treat the map and camera poses as parameters $\boldsymbol{\theta}_t = (\boldsymbol{\phi}, \mathbf{T}_t)$ to be optimized. In the E-step, we recover a variational approximation of the association likelihood for each pixel $\mathbf{u} \in \Omega$ given the current parameter estimate $\boldsymbol{\theta}_{t-1}$ from the previous EM iteration,

$$q(\mathbf{c}_t) \leftarrow \arg\max_{q(\mathbf{c}_t)} \sum_{\mathbf{c}_t\in\mathscr{C}} q(\mathbf{c}_t) \ln p(\mathbf{z}_t, \mathbf{c}_t \mid \boldsymbol{\theta}_{t-1}). \tag{3.23}$$

The maximum is achieved for $q(\mathbf{c}_t) = p(\mathbf{c}_t \mid \mathbf{z}_t, \boldsymbol{\theta}_{t-1})$. For the M-step, we maximize the expected log posterior under the approximate association likelihood

$$\boldsymbol{\theta}_t \leftarrow \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{c}_t\in\mathscr{C}} q(\mathbf{c}_t) \ln p(\mathbf{z}_t, \mathbf{c}_t \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}). \tag{3.24}$$

Note that $p(\boldsymbol{\theta}) = p(\boldsymbol{\phi} \mid \mathbf{z}_{1:t-1})\, p(\mathbf{T})$ for $\boldsymbol{\theta} = (\boldsymbol{\phi}, \mathbf{T})$.

In our case the E-step can be performed by evaluating

$$p(\mathbf{c}_t \mid \mathbf{z}_t, \boldsymbol{\theta}_{t-1}) = \frac{p(\mathbf{z}_t \mid \mathbf{c}_t, \boldsymbol{\theta}_{t-1}) p(\mathbf{c}_t \mid \boldsymbol{\theta}_{t-1})}{\sum_{\mathbf{c}_t'\in\mathscr{C}} p(\mathbf{z}_t \mid \mathbf{c}_t', \boldsymbol{\theta}_{t-1}) p(\mathbf{c}_t' \mid \boldsymbol{\theta}_{t-1})}. \tag{3.25}$$

Since we treat data and association likelihood stochastically independent between pixels, the association likelihood can be determined for each pixel individually. Assuming uniform prior association likelihood, we arrive at

$$p(\mathbf{c}_t \mid \mathbf{z}_t, \boldsymbol{\theta}_{t-1}) = \frac{p(\mathbf{z}_t \mid \mathbf{c}_t, \boldsymbol{\theta}_{t-1})}{\sum_{\mathbf{c}_t'\in\mathscr{C}} p(\mathbf{z}_t \mid \mathbf{c}_t', \boldsymbol{\theta}_{t-1})}. \tag{3.26}$$

The M-step is solved individually per object by taking into account the association likelihood of the pixels to the objects. We optimize first for the camera poses in the previous map and then integrate the measurement into the map using the new pose estimates. In the following, we detail the steps in our pipeline that implement the EM algorithm.

3.4.3 Data Association (E-Step)

We associate the pixels $\mathbf{u}$ in the current frame according to Equation (3.26), modeling the probability

$$p(\mathbf{c}_t \mid \mathbf{z}_t, \boldsymbol{\theta}) = \prod_{\mathbf{u}\in\Omega} p(c_{t,\mathbf{u}} \mid z_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1}) \tag{3.27}$$

independently for each pixel. Let $\overline{\mathbf{p}}_i := \mathbf{T}^i_{t-1}\, \overline{\pi^{-1}}(\mathbf{u}, z_{t,\mathbf{u}})$ be the local point coordinate of pixel $\mathbf{u}$ in the coordinate frame of object i, where we denote $\overline{\mathbf{p}} := (\mathbf{p}^\top, 1)^\top$. We model the data likelihood of a pixel that falls inside the map volume of object $c_{t,\mathbf{u}} \in \{0, \dots, N\}$ with a mixture distribution,

$$\begin{aligned} p(z_{t,\mathbf{u}} \mid c_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1}) = \alpha \frac{1}{2\sigma} \exp\left(- \frac{|\phi^{c_{t,\mathbf{u}}}_{t-1}(\mathbf{p}_{c_{t,\mathbf{u}}})|}{\sigma} \right) p_{fg}(\mathbf{p}_{c_{t,\mathbf{u}}} \mid c_t) \\ + (1-\alpha)\, p_{\mathcal{U}}(\mathbf{p}_{c_{t,\mathbf{u}}}), \end{aligned} \tag{3.28}$$

where $\phi^{c_{t,\mathbf{u}}}$ is the SDF of object $c_{t,\mathbf{u}}$. The mixture is composed of a Laplace distribution which explains the measurement within the object, and a uniform component $p_{\mathcal{U}}$ that models outlier measurements and objects that are not yet detected and missing in the multi-object map. If the pixel is not within the map volume of object $c_{t,\mathbf{u}}$, we set its data likelihood to zero for this object. Equation (3.28) gives the likelihood for each model individually. We normalize this likelihood over all models to arrive at the final pixel association likelihood:

$$p(c_{t,\mathbf{u}} \mid z_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1}) = \frac{p(z_{t,\mathbf{u}} \mid c_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1})}{\sum_{c'_{t,\mathbf{u}}=0}^{N} p(z_{t,\mathbf{u}} \mid c'_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1})}. \tag{3.29}$$

Occlusions are implicitly handled by our data association approach. If an object is occluded by another object in the map, the association likelihood will be higher within the occluding object. This results in a lower weight for the measurements in the occluded object for tracking and map integration. Figure 3.3 illustrates such a case for a clock which is moved upwards along a wall.

3.4.4 Tracking (M-Step)

Most existing approaches to dynamic multi-object SLAM employ a variant of the iterative closest points algorithm (ICP; Besl and McKay 1992) for tracking the camera pose. This requires that a point cloud is extracted from the existing TSDF volume and associations are found between this point cloud and the depth image. A typical approach with SDF map representations is to apply raycasting[6] to determine the zero-crossings along the line-of-sight of the pixels. The point clouds from the ray cast are

6: see Subsection 2.2.4

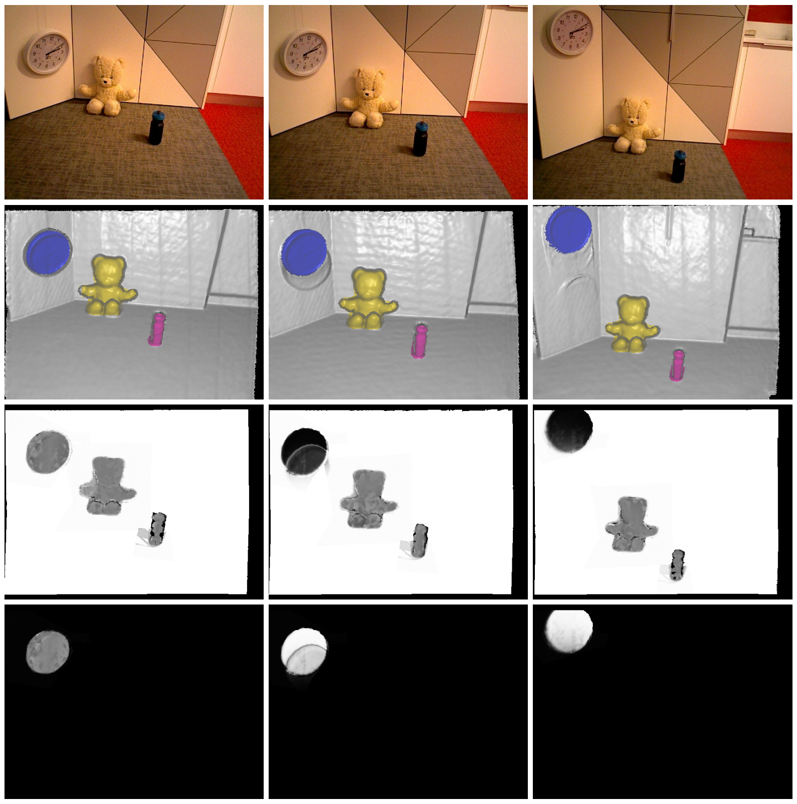


Figure 3.3: Pixel association likelihood. The E-step of our EM method determines the association likelihood (black: 0, white: 1) for the background (third row) and all objects (fourth row: clock). The association likelihood is determined from the data likelihood of the pixels in all objects given the current pose and map estimates (second row, object segments overlaid by color). Before the clock starts to move, the association weight is equally distributed between the background and the clock model. While the clock moves upwards, the background above the clock becomes occluded and the clock measurements are stronger associated with the object map than with the background.

then aligned with the point clouds from the depth maps using non-linear least squares techniques. In this approach, depth measurements are associated projectively with the zero-level surface.

We instead follow the approach in (Bylow et al. 2013) and associate the depth measurements with the closest point on the surface as shown in Figure 3.2b. This is achieved by minimizing the signed distance of the measured points to the surface which is directly given by the SDF function at the points. The main advantage of this strategy is that pixels are associated with the correct part of the implicit surface using only one trilinear interpolation lookup per pixel in each iteration of the algorithm. In ICP, the projective association is only performed once and requires several lookups per pixel until a surface is found.

For the M-step in Equation (3.24) we need to update camera and object poses, as well as the background and object maps. In a first step, we estimate the camera and object poses with regard to an SDF volume by minimizing

$$E(\mathbf{T}_t^c) = \frac{1}{2} \sum_{\mathbf{u} \in \boldsymbol{\Omega}} q(c_{\mathbf{u}}) \left| \phi_{t-1}^c \left(\mathbf{T}_t^c \, \overline{\mathbf{p}}(\mathbf{u}) \right) \right|_\delta , \tag{3.30}$$

where $\mathbf{p}(\mathbf{u}) := \pi^{-1}(\mathbf{u}, z_{t,\mathbf{u}})$ and $q(c_{\mathbf{u}}) = p(c_{t,\mathbf{u}} \mid z_{t,\mathbf{u}}, \boldsymbol{\theta}_{t-1})$ is the association likelihood of pixel $\mathbf{u}$ for the object/background with index c as computed in Equation (3.29). We use the Huber norm with threshold δ to achieve robustness with regard to outliers.

We optimize Equation (3.30) using the iteratively reweighted non-linear least squares (IRLS) algorithm. Since the camera and object poses are in SE(3), we optimize Equation (3.30) by reformulating it with a local parameterization $\boldsymbol{\xi}_t^c = \log\left(\mathbf{T}_t^c\right)^\vee \in \mathfrak{se}(3)$ using the Lie algebra $\mathfrak{se}(3)$ (see

Section 2.1). To this end, we apply local increments $\boldsymbol{\delta\xi} \in \mathfrak{se}(3)$ to the current solution for $\boldsymbol{\xi}$ in each iteration which we linearize at $\boldsymbol{\delta\xi} = \mathbf{0}$. For convenience, we write $\mathbf{T}(\boldsymbol{\xi}) = \exp\left(\hat{\boldsymbol{\xi}}\right) \in \mathrm{SE}(3)$ (see Section 2.1). Consequently, Equation (3.30) becomes

$$E\left(\boldsymbol{\delta\xi}^c\right) = \frac{1}{2}\sum_{\mathbf{u}\in\boldsymbol{\Omega}} q\left(c_{\mathbf{u}}\right) w_{\mathbf{u}} \left(\phi^c_{t-1}\left(\mathbf{T}\left(\boldsymbol{\xi}^c_t\right)\mathbf{T}\left(\boldsymbol{\delta\xi}^c\right)\overline{\mathbf{p}}\left(\mathbf{u}\right)\right)\right)^2, \tag{3.31}$$

with weights

$$w_{\mathbf{u}} = \min\left(1, \frac{\delta}{\left|\phi^c_{t-1}\left(\mathbf{T}\left(\boldsymbol{\xi}^c_t\right)\mathbf{T}\left(\boldsymbol{\delta\xi}^c\right)\overline{\mathbf{p}}\left(\mathbf{u}\right)\right)\right|}\right) \frac{W(\pi^{-1}(\mathbf{u}, z_{t,\mathbf{u}}))}{\max_{\mathbf{u}'\in\boldsymbol{\Omega}} W(\pi^{-1}(\mathbf{u}', z_{t,\mathbf{u}'}))}, \tag{3.32}$$

where the first term implements the Huber norm in Equation (3.31) and the second term is the map confidence. $W(\mathbf{p})$ is the accumulated integration weight (see Subsection 3.4.5), which quantifies how certain we are about a surface estimate in the model. This robustifies the tracking when large objects enter the frame from the image boundary.

As Equation (3.31) is of the same form as Equation (3.9), we optimize it using the Levenberg-Marquardt method (Levenberg 1944; Marquardt 1963) following Bylow et al. (2013). The pose estimate $\boldsymbol{\xi}^c_t$ in Equation (3.31) is initialized with $\boldsymbol{\xi}^c_{t-1}$ and updated by $\boldsymbol{\delta\xi}^c$ in every iteration of the algorithm. This tracking optimization is first run on the background TSDF to estimate the updated camera pose before recomputing the association probabilities and running the same algorithm on each object TSDF for updating the individual object poses.

(Levenberg 1944): *A method for the solution of certain non-linear problems in least squares*
(Marquardt 1963): *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*

Figure 3.4 illustrates the effectiveness of using the association likelihood for tracking. We compare our approach with just using the foreground probabilities without geometric cues by replacing $q(c_{\mathbf{u}})$ with $p_{fg}(\mathbf{p}_{c_{\mathbf{u}}} \mid c_{\mathbf{u}})$ in Equation (3.31). While the foreground probability also provides a segmentation cue, it is not sufficient for robust tracking due to the inaccurate instance segmentations by Mask R-CNN. The segmentation by the foreground probability (last column) has a high weight even on non-object pixels like the wall in the first and the floor in the second scene. By contrast, our association likelihood (third column) and as a result our tracking weights (fourth column) are lower in these regions as they are already explained well by the background model. Consequently, these pixels have lower influence on the energy function in Equation (3.31), allowing the hand to move back and the horse to tilt in rows two and four, respectively.

3.4.5 Mapping (M-Step)

Once the new camera poses $\mathbf{T}_t$ have been estimated, we implement the second part of the M-step (Equation (3.24)) by integrating the depth maps into the background and object volumes. Following (Curless and Levoy 1996), we find the SDF as the maximum likelihood surface fit to

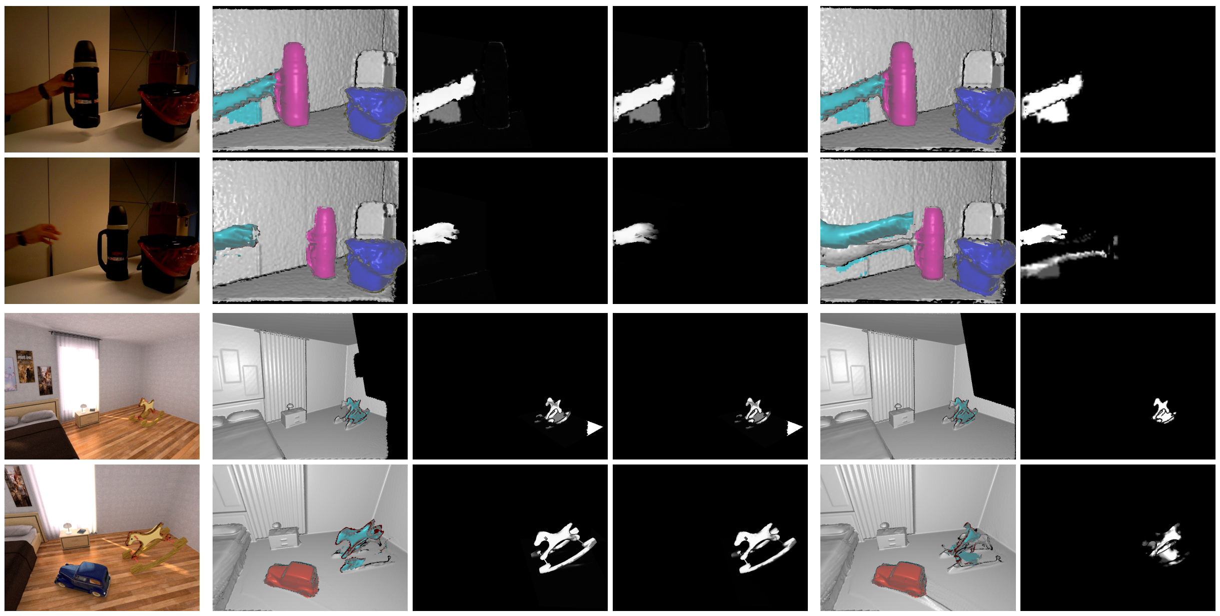


Figure 3.4: Tracking with association likelihoods. Probabilistic data association helps to overcome inaccuracies of the instance segmentation with geometric cues and makes the tracking more robust. From left to right: RGB images, our 3D reconstruction with reprojected object segmentation, association likelihood for the hand/horse object, our total pixel weights for tracking for the hand/horse object, 3D reconstruction with foreground probability instead of the association likelihood, total tracking weights with foreground probability instead of association likelihood. Our tracking weights (column four) are lower on non-object pixels in rows one and three than the foreground probability (last column). Thus, the hand can move back, and the horse can tilt in our approach (rows two and four).

the depth images using the recursive integration akin Equation (3.5):

$$\begin{aligned}\phi_t^o(\mathbf{v}) &\leftarrow \frac{W_{t-1}^o(\mathbf{v})\phi_{t-1}^o(\mathbf{v}) + q(c_{\mathbf{u}})\,\check{\phi}_t^{\,o}(\mathbf{v})}{W_{t-1}(\mathbf{v}) + q(c_{\mathbf{u}})},\\ W_t^o(\mathbf{v}) &\leftarrow \min\left(W_{t-1}(\mathbf{v}) + q(c_{\mathbf{u}}), \overline{W}\right),\end{aligned} \tag{3.33}$$

where $\check{\phi}_t^{\,o}(\mathbf{v})$ is the measured depth difference of the voxel towards the integrated depth image $\mathbf{z}_t$ as defined in Equation (3.2). For implementing the M-step in Equation (3.24), we incorporate the association likelihood $q(c_{\mathbf{u}})$ of the pixel $\mathbf{u} = \pi\left(\left(\mathbf{T}_t^o\right)^{-1}\overline{\mathbf{v}}\right)$ which passes through the voxel for computing the update weight. The cap on $W(\mathbf{v})$ prevents the model from becoming overconfident in the SDF estimate and allows for faster adaptation in case of inaccurate or missing segmentations of dynamic objects (see Subsection 3.3.3). Non-moving objects are initially integrated in the background map as well until the moving object map fits the measurements better. We consider backtracing these insertions too costly. One could reweigh the accumulated weight $W(\mathbf{v})$ with the combined IRLS weight and map confidence $w_{\mathbf{u}}$ from Equation (3.32) for faster map updates, which increases drift though:

$$W(\mathbf{v}) \leftarrow q(c_{\mathbf{u}})\,w_{\mathbf{u}}\,W(\mathbf{v}) + (1 - q(c_{\mathbf{u}}))W(\mathbf{v}). \tag{3.34}$$

3.5 Experiments

We evaluate the performance of our method qualitatively and quantitatively on datasets containing dynamic scenes published with (Rünz and Agapito 2017) and the benchmark (Sturm et al. 2012). We employ the Mask R-CNN implementation of Abdulla (2017). In our experiments, the truncation distance μ in Equation (2.47) is chosen to be 10 times the voxel size for each TSDF volume and the parameter δ in Equation (3.30) is twice the voxel size. In Equation (3.28), we set $\sigma = 0.02$, $\alpha = 0.8$, and $p_{\mathcal{U}}(\mathbf{p}_{c_t}) = 1.0$. Mask R-CNN detections are only accepted if they are large enough (at least 40×40 pixels) and objects are classified as invisible (tracking and mapping unreliable) and deleted if their projected mask area within a region of 20 pixels from the image boundary is below this threshold. To avoid cluttering the scene with large volumes containing static objects for which Mask R-CNN usually generates very inaccurate masks, we exclude a list of these object classes (*e.g.*, tables, beds, refrigerators, etc.) from the detections used for instantiating new object volumes. While one could implement a sliding window version for the background TSDF (Whelan, Kaess, et al. 2012), we found that in our experiments a volume size of 5.12 m with the camera positioned at the center of one of the sides of the volume usually worked well. The only exception from this strategy is the scene *Room4*, where we increased the volume size to 7.68 m and moved the initial camera pose further inside the volume to keep the scene within the volume boundaries.

(Rünz and Agapito 2017): *Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects*

(Sturm et al. 2012): *A benchmark for the evaluation of RGB-D SLAM systems*

(Abdulla 2017): *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*

3.5.1 Quantitative Evaluation

Evaluation Measures. We evaluate our approach using the absolute trajectory (AT) and relative pose (RP) evaluation measures as proposed by Sturm et al. (2012). For both error measures we compute the root mean squared error (RMSE). We use the toolkit provided by Sturm et al. (2012)[7] for computing these evaluation measures. For the absolute trajectory error, the tool first aligns the estimated and ground truth trajectories as they are both in arbitrary coordinate frames. The absolute trajectory root mean squared error (AT-RMSE) between the aligned estimated and ground-truth trajectories, $\mathcal{T}^e = \left\{\mathbf{T}_t^e\right\}_{t=1}^T$ and $\mathcal{T}^g = \left\{\mathbf{T}_t^g\right\}_{t=1}^T$, respectively, is then defined as

(Sturm et al. 2012): *A benchmark for the evaluation of RGB-D SLAM systems*

7: `https://cvg.cit.tum.de/data/datasets/rgbd-dataset/tools`

$$\text{AT-RMSE}\left(\mathcal{T}^e, \mathcal{T}^g\right) = \sqrt{\frac{1}{n}\sum_{t=1}^{T}\left\|\bar{\mathbf{t}}_t\right\|^2}, \tag{3.35}$$

where $\bar{\mathbf{t}}_t$ is the translation part of the difference transformation

$$\overline{\mathbf{T}}_t = \left(\mathbf{T}_t^g\right)^{-1}\mathbf{T}_t^e. \tag{3.36}$$

The relative pose error compares the relative pose increments in both trajectories in a window of size Δt:

$$\overline{\mathbf{T}}_t = \left(\left(\mathbf{T}_t^g\right)^{-1}\mathbf{T}_{t+\Delta t}^g\right)^{-1}\left(\left(\mathbf{T}_t^e\right)^{-1}\mathbf{T}_{t+\Delta t}^e\right). \tag{3.37}$$

The relative pose root mean squared error (RP-RMSE) is then computed as the average RMSE over this difference in each window for all possible window sizes $\Delta t \in \{1, \ldots, T\}$:

$$\text{RP-RMSE}\left(\mathcal{T}^e, \mathcal{T}^g\right) = \frac{1}{T} \sum_{\Delta t=1}^{T} \sqrt{\frac{1}{T-\Delta t} \sum_{t=1}^{T-\Delta t} \left\|\bar{\mathbf{t}}_t\right\|^2}, \tag{3.38}$$

where $\bar{\mathbf{t}}_t$ is the translation part of $\overline{\mathbf{T}}_t$ in Equation (3.36). The relative pose error can alternatively also be computed on the rotation:

$$\text{RP-RMSE}\left(\mathcal{T}^e, \mathcal{T}^g\right) = \frac{1}{T} \sum_{\Delta t=1}^{T} \sqrt{\frac{1}{T-\Delta t} \sum_{t=1}^{T-\Delta t} \left\|\log\left(\overline{\mathbf{R}}_t\right)^\vee\right\|^2}, \tag{3.39}$$

where $\overline{\mathbf{R}}_t$ is the rotation part of $\overline{\mathbf{T}}_t$ and log computes the logarithm map in SO(3)[8]. The result of the logarithm map is then converted to the axis-angle vector using the $\vee$ operator[9], and the norm gives the angle between the estimated and ground-truth rotations.

8: see Equation (2.15)

9: see Equation (2.8)

Tracking of Dynamic Objects. We perform quantitative evaluation of dynamic object tracking on the synthetic scenes provided by Rünz and Agapito (2017) in Co-Fusion. As the Mask R-CNN model we use (Abdulla 2017) is trained on the COCO dataset (Lin et al. 2014), our method is theoretically limited to detect object classes present in this training set. Remarkably, although the classes of many objects present in the scenes are not contained in the COCO dataset, Mask R-CNN manages to generate detections of most of the moving objects. We compare our method to Kintinuous (KT; Whelan, Kaess, et al. 2012), ElasticFusion (EF; Whelan, Leutenegger, et al. 2015), Co-Fusion (CF; Rünz and Agapito 2017), and MaskFusion (MF; Rünz, Buffier, et al. 2018). KT and EF are static SLAM systems that treat dynamic objects as outliers. CF uses geometric and motion segmentation for dynamic objects, while MF combines geometric segmentation with Mask R-CNN based instance segmentation. For the publicly available implementation of MF we adjusted the minimum number of pixels required for instantiating a new object model to work well on the sequences. We used the same threshold as in our approach, but MF still failed to instantiate an object instance for the rocking horse in the *Room4* scene.

(Rünz and Agapito 2017): *Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects*

(Whelan, Kaess, et al. 2012): *Kintinuous: Spatially Extended KinectFusion*

(Whelan, Leutenegger, et al. 2015): *ElasticFusion: Dense SLAM Without A Pose Graph*

(Rünz and Agapito 2017): *Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects*

(Rünz, Buffier, et al. 2018): *MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects*

The results of our evaluation are shown in Table 3.1. One can see that our method achieves competitive results. Especially for the dynamic objects, our method outperforms the competing dynamic object-level SLAM approaches CF and MF. The large camera tracking error with respect to the static background (Static Bg) for MF in the *ToyCar3* scene is caused by a very late detection of one of the moving cars, causing significant drift at the beginning of the trajectory. This shows that the ICP tracking without a robust norm used in MF is sensitive to missing detections. Our robust tracking using direct SDF alignment and the Huber norm, however, manages to keep the trajectory error low.

Robust Camera Tracking. Similar to experiments performed in MaskFusion (Rünz, Buffier, et al. 2018) and MID-Fusion (B. Xu et al. 2019), we can use Mask R-CNN detections with certain labels (*e.g.*, *person*) to exclude these labels from the reconstruction and tracking. In our approach, the association likelihoods already prevent parts of depth

		KT	EF	CF	MF	**Ours**
ToyCar3	Static Bg	**0.10**	0.59	0.61	20.60	0.95
	Car1	-	-	7.78	1.53	**0.77**
	Car2	-	-	1.44	0.58	**0.18**
Room4	Static Bg	**0.16**	1.22	0.93	1.41	1.37
	Airship	-	-	0.91/ 1.01	13.62/ 2.29/ 3.46	**0.56**/ 1.41/ 0.75
	Car	-	-	**0.29**	2.66	2.10
	Horse	-	-	5.80	-	**3.57**

Table 3.1: Absolute trajectory (AT) RMSE (in cm) of estimated trajectories for the synthetic sequences from Co-Fusion (Rünz and Agapito 2017). The Airship trajectory is split into multiple parts due to separate geometric segments and detections with too little overlap for assignment. Our method achieves competitive results with a static SLAM system (EF) for the static background and outperforms other dynamic SLAM approaches (CF, MF) on the objects.

maps projecting into foreground parts of object volumes from being integrated into the background volume used for camera tracking. We thus maintain object volumes for detected people but do not render them during raycasting for visualization[10]. The association likelihood then tends to associate even non-rigidly moving people to the object volumes rather than the background, enabling us to robustly track the camera with respect to the background.

10: Using the fused class probability from Equation (3.21) for classification.

We compare our method to five state-of-the-art dynamic SLAM approaches in Table 3.2. Two of these, joint visual odometry and scene flow (VO-SF; Jaimez et al. 2017), and StaticFusion (SF; Scona et al. 2018) were designed for reconstructing the static background while ignoring dynamic parts. The remaining ones, CF (Rünz and Agapito 2017), MF (Rünz, Buffier, et al. 2018), MID-Fusion (MID-F; B. Xu et al. 2019) were designed for multi-object reconstruction. The latter two of these methods, like our approach, use Mask R-CNN (He et al. 2017) detections for instantiating objects. One can see that our method achieves competitive results in most cases, especially compared to MF (Rünz, Buffier, et al. 2018) and MID-F (B. Xu et al. 2019). Like all these methods, our method might fail if large undetected objects cover the major part of the image. Our results demonstrate that the combination of robust tracking and our data association strategy improves robustness on these sequences. The table rows are ordered approximately by scene difficulty, so the latter rows exhibit large dynamic parts with heavy occlusions. *f3s* abbreviates *freiburg3_sitting* while *f3w* stands for *freiburg3_walking*. MID-F did not report RP-RMSE and thus is not shown in Tables 3.2b and 3.2c.

(Jaimez et al. 2017): *Fast odometry and scene flow from RGB-D cameras based on geometric clustering*

(Scona et al. 2018): *StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments*

(Rünz and Agapito 2017): *Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects*

(Rünz, Buffier, et al. 2018): *MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects*

(B. Xu et al. 2019): *MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM*

We further compare to MF (Rünz, Buffier, et al. 2018) on the scene *f3_long_office_household* of the benchmark (Sturm et al. 2012). By exporting the relative trajectory of the teddy bear and the camera, we can compare the object trajectory to the ground truth camera trajectory as was done in (Rünz, Buffier, et al. 2018). While we achieve slightly worse results on the teddy bear trajectory (3.5 cm, while MF achieved 2.2cm), our camera trajectory is more accurate (5.0 cm compared to 8.9 cm for MF). Note that while MF improved their camera trajectory with respect to the background to 7.2 cm AT-RMSE when not tracking the teddy bear, we do not expect a notable change for this case in our approach since the teddy is implicitly reconstructed with partial association likelihood in the background and would be disassociated and removed from it if it started moving.

In Table 3.3, we perform an ablation study to evaluate the contributions of different parts of our method on the synthetic scene *Room4*. Since most objects only observe minor changes in their local topology (the Airship moving freely in the air, the car driving on the ground), and

Table 3.2: Comparison of robust camera tracking with respect to the static background in dynamic scenes for different methods. Our approach provides state-of-the-art results and outperforms previous methods in the majority of sequences.

	VO-SF	SF	CF	MF	MID-F	**Ours**
f3s static	2.9	1.3	1.1	2.1	1.0	**0.9**
f3s xyz	11.1	4.0	**2.7**	3.1	6.2	3.7
f3s halfsphere	18.0	4.0	3.6	5.2	**3.1**	3.2
f3w static	32.7	**1.4**	55.1	3.5	2.3	**1.4**
f3w xyz	87.4	12.7	69.6	10.4	6.8	**6.6**
f3w halfsphere	73.9	39.1	80.3	10.6	**3.8**	5.1

(a) Absolute trajectory (AT) RMSE (in cm)

	VO-SF	CF	SF	MF	**Ours**
f3s static	2.4	1.1	1.1	1.7	**0.9**
f3s xyz	5.7	2.7	2.8	4.6	**2.6**
f3s halfsphere	7.5	**3.0**	**3.0**	4.1	**3.0**
f3w static	10.1	22.4	1.3	3.9	**1.2**
f3w xyz	27.7	32.9	12.1	9.7	**6.0**
f3w halfsphere	33.5	40.0	20.7	9.3	**5.1**

(b) Translational relative pose (RP) RMSE (cm/s)

	VO-SF	CF	SF	MF	**Ours**
f3s static	0.71	0.44	**0.43**	0.54	0.44
f3s xyz	1.44	1.00	0.92	1.25	**0.90**
f3s halfsphere	2.98	1.92	2.11	2.07	**1.82**
f3w static	1.68	4.01	**0.38**	0.76	0.40
f3w xyz	5.11	5.55	2.66	2.00	**1.57**
f3w halfsphere	6.69	13.02	5.04	3.35	**1.96**

(c) Rotational relative pose (RP) RMSE (deg/s)

Table 3.3: Ablation study on the synthetic scene *Room4*. We compare AT-RMSE for our approach to not using association likelihoods, and to not using map confidence weights for tracking.

		w/o assoc.	w/o map conf.	Ours
Room4	Static Bg	1.42	**1.37**	**1.37**
	Airship	**0.49**/ **1.13**/ 1.24	0.73/ 1.47/ **0.75**	0.56/ 1.41/ **0.75**
	Car	**2.01**	2.11	2.10
	Horse	9.12	8.38	**3.57**

there are no large objects moving into view from the edge of the image, the effects of not using association likelihoods or map confidence weights for tracking are numerically negligible for most objects. However, the rocking horse is subject to topology changes in its surrounding since wall and floor intersect the volume at different angles. We observe a significant improvement for this object in Table 3.3.

Computational Performance. While our implementation is not yet tuned for runtime performance (*e.g.*, parallel processing of objects), the average runtime per frame on the CF-datasets (Rünz and Agapito 2017) ranges from 106 ms to 257 ms on an Nvidia GTX 1080 Ti GPU with 11GB of memory and an Intel Xeon Silver 4112 CPU with 4 cores and 2.6 GHz.

In Table 3.4, we report the average runtime per frame on the dynamic sequences from Co-Fusion (Rünz and Agapito 2017). One can clearly see that the runtime for detection frames is much higher than on the frames where only tracking and mapping with our probabilistic data association is run. The overhead in detection frames amounts to more than just the inference runtime of Mask R-CNN (which has been reported around 200 ms/frame or 5 Hz (Rünz, Buffier, et al. 2018)), since we also allocate new object volumes in these frames or match and update existing

	det. frames	non-det. frames	Overall
ToyCar3	789	241	259
Room4	561	124	139
PlaceItems	611	89	107
TeddyHandover	772	142	164
SlidingClock	759	188	208

Table 3.4: Average runtime per frame in ms on the dynamic sequences of the Co-Fusion data set (Rünz and Agapito 2017). The runtime on detection frames includes not only Mask R-CNN inference but also creation and updates of object maps.

Table 3.5: Ablation study with varying detection rates. (a) Average trajectory (AT) RMSE when running detections every 1, 15, 30, or 60 frames. (b) Trajectory coverage for these setups. (c) Number of non-moving objects detected in these cases.

	Det. rate	1	15	30	60
ToyCar3	Static Bg	0.94	0.94	0.95	0.96
	Car1	0.80	0.83	0.77	0.85
	Car2	0.13	0.17	0.18	-
Room4	Static Bg	1.35	1.37	1.37	1.40
	Airship	0.34/ 3.87/ 0.75	0.56/ 1.35/ 0.75	0.56/ 1.41/ 0.75	0.56/ 1.43/ 0.75
	Car	2.34	2.13	2.10	2.10
	Horse	2.19	3.57	3.57	-

(a) AT-RMSE (in cm)

		1	15	30	60
ToyCar3	Car1	82.6%	81.1%	81.1%	62.3%
	Car2	21.4%	6.5%	6.5%	0%
Room4	Airship	7.4%/ 11.1%/ 10.9%	9.2%/ 6.1%/ 5.7%	9.2%/ 6.3%/ 5.7%	9.2%/ 6.3%/ 5.7%
	Car	12.0%	9.7%	7.9%	7.9%
	Horse	19.5%	18.5%	18.5%	0%

(b) Trajectory coverage

	1	15	30	60
ToyCar3	6	2	2	1
Room4	10	1	0	0

(c) Static objects

ones. Note that this implementation is not yet tuned for computational efficiency. Note further, that the *ToyCar3* data set has a resolution of 960×540 pixels while all other datasets are captured at a resolution of 640×480 pixels. We can thus partly attribute the higher runtime in *ToyCar3* to the higher amount of data that needs to be processed per frame.

Varying Detection Rates. As Table 3.4 indicates significantly higher runtime requirements for detection frames, one possibility for improving the overall computation time of our approach is lowering the detection rate for Mask R-CNN. Table 3.5 shows an ablation study of how varying the detection rate for Mask R-CNN affects trajectory accuracy, trajectory coverage, and the number of detected non-moving objects. The coverage is computed as the percentage of frames in the sequence for which our approach maintained a model for the object. Note that by this measure, 100% cannot be achieved for most objects since they are not visible in all frames. Note further, that in *ToyCar3*, the static airplane is always detected and instantiated as an object. Thus, there is always at least one non-moving object detected in this scene (see Table 3.5c).

The clear and intuitive tendency is that trajectory coverage improves with increased detection rate, but this also creates more spurious detections instantiating objects. If the detection frequency is too low, some objects might be missed. This happens for the second car in *ToyCar3* and the horse in *Room4* when we run the detection only every 60 frames (s. Tables 3.5a and 3.5b).

Interestingly, while a larger trajectory coverage can induce higher AT-RMSE (since more frames can deviate from the ground truth), we do not observe this as a clear tendency. For most objects, the AT-RMSE remains at a very similar level. In some cases, such as for Car2 in *ToyCar3*, the AT-RMSE even improves with increased trajectory coverage.

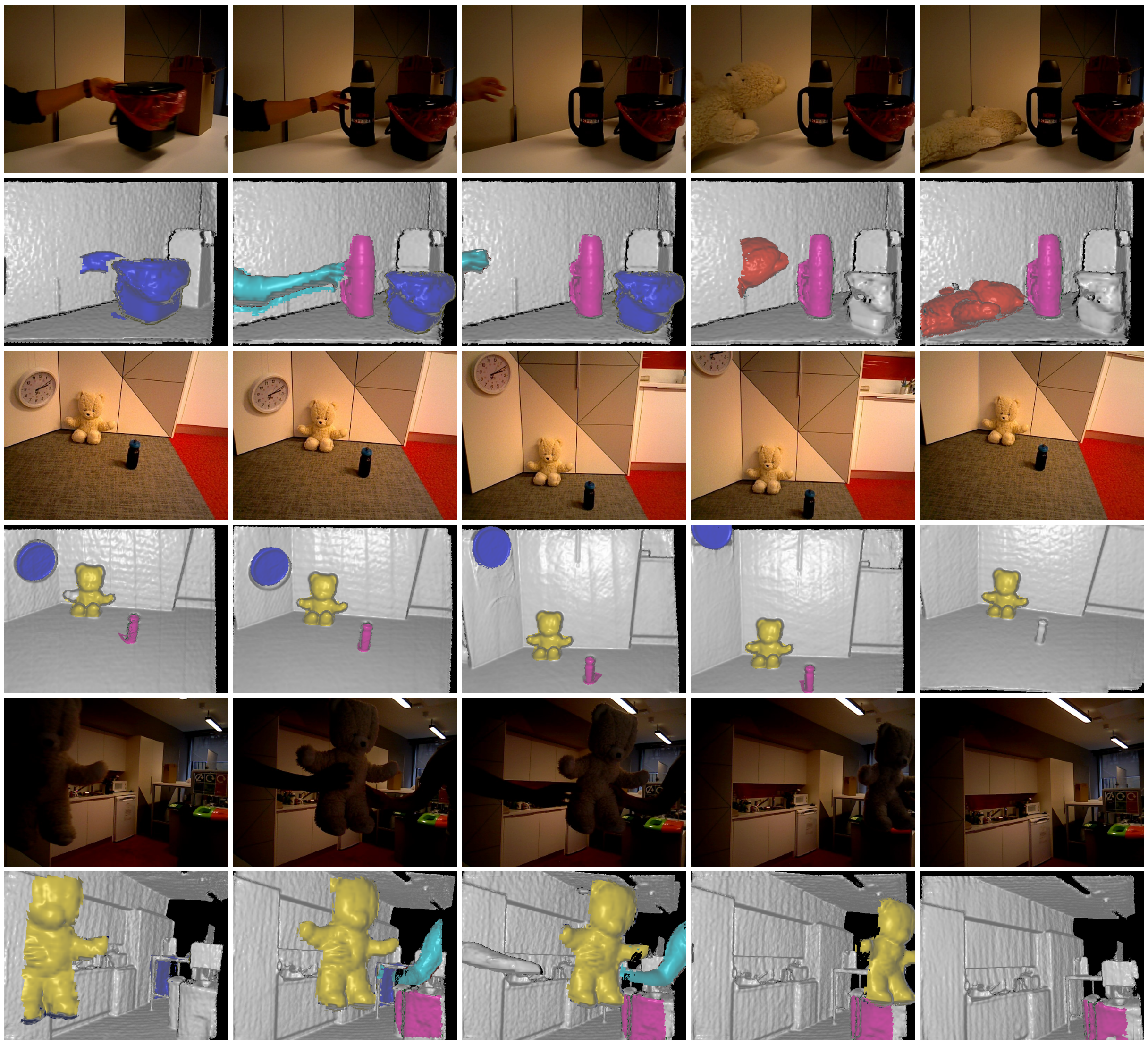


Figure 3.5: Qualitative evaluation on the real-world datasets published with Co-Fusion (Rünz and Agapito 2017). We demonstrate that we can handle fast movement (the second and the third image of the first sequence are only 25 frames apart), as well as objects with relative weak geometric cues, such as the clock in the second scene. Note that the left arm handing over the teddy is not detected in the last scene. While it initially is integrated into the background it is quickly overwritten by actual background depth soon after it moves out of view.

3.5.2 Qualitative Evaluation

Figure 3.5 shows a qualitative evaluation on the real-world datasets published with Co-Fusion (Rünz and Agapito 2017). One can see that we manage to reconstruct dynamic and static objects in these scenes if they are detected by Mask R-CNN. Note that some of the objects, like the trashcan in the first sequence, are not contained in the set of classes that Mask R-CNN is trained on. Thus, after the initial detection the trashcan is not detected anymore and deleted because of a low existence probability p_{ex} (Equation (3.20)). The bottle and the clock in the second sequence are deleted after they are classified as "not visible" because they move out of view between the fourth and the last displayed frame and the number of pixels in view is too low.

We show how the incremental integration of foreground probabilities into object volumes improves the object masks in Figure 3.6. Note that

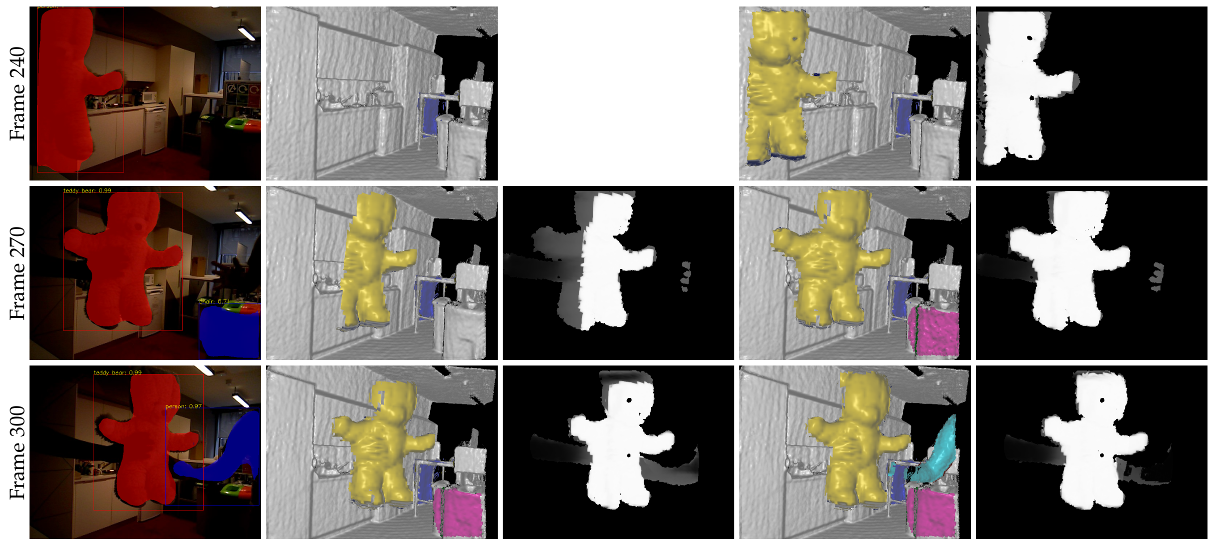


Figure 3.6: Incremental mask integration. From left to right: Masked RGB Frame, model output and association likelihood for teddy before mask integration, model output and association likelihoods after mask integration. One can see that the association likelihoods provide a soft geometric segmentation for the moving geometry inside the volume of the teddy object. It gets stronger for the pixels that actually belong to the object once Mask R-CNN confirms those pixels to belong to that object. Note that the teddy bear is first detected in frame 240 and thus does not have association likelihoods in this frame yet.

the association likelihoods are already non-zero for the parts which geometrically better match the object than the far-away background and thus the mask integration makes an already integrated volumetric 3D model for these regions visible. Finally, for a qualitative evaluation of the effect of the association likelihood, we refer to Figures 3.1 and 3.4, where moving objects leave a visible trace because their depth values are integrated into the background if just using the foreground probability as segmentation cue. Figure 3.4 further shows that the association likelihood helps to improve the tracking quality by including geometric cues if Mask R-CNN segmentations do not fit the actual object shape.

3.6 Conclusion

In this chapter, we propose a novel probabilistic formulation for dynamic object-level SLAM with RGB-D cameras. We infer the latent data association of pixels to the objects in the map concurrently with the maximum likelihood estimates of camera poses and maps. The maps are represented as volumetric signed distance functions. For tracking, our probabilistic formulation facilitates direct alignment of depth images with the SDF representation. Our results demonstrate that proper probabilistic treatment of data associations is a key ingredient to robust tracking and mapping in dynamic scenes. To the best of our knowledge, our approach is the first that considers EM for dynamic object-level SLAM with RGB-D cameras.

Note that our approach treats the detected objects models always as dynamic. While our experiments have shown that their poses are stable in most settings for static objects, in future work, an additional classification into static and dynamic objects might be developed to prevent drifting of static objects and to refine the camera pose by tracking it relative to the static object volumes. This might prove beneficial since the object volumes usually exhibit a higher relative resolution. We further plan to integrate information from the RGB image for tracking to further increase the accuracy and robustness of the method in planar surfaces in future work. Furthermore, more efficient data structures and global graph optimization are interesting directions to further scale our approach. Finally, we plan to investigate how our approach could be used on mobile manipulation platforms for the interactive perception of objects.

Where Does It End? – Reasoning About Hidden Surfaces by Object Intersection Constraints 4

The contents of this chapter are based on the peer-reviewed conference publication

with the following co-author contributions:

	Ideas	Experiments	Analysis	Writing
Michael Strecke	80%	100%	80%	65%
Jörg Stückler	20%	0%	20%	35%

Compared to the conference publication, this chapter contains more detailed preliminaries and unified notation with the rest of the thesis.

4.1 Introduction

In the previous chapter, we tracked and reconstructed object models in TSDF volumes in an online fashion. While the previous approach only reconstructed *visible surfaces* of the objects, we will now investigate how priors on the *physical plausibility* of dynamic scene configurations[1] can help to infer missing information. We observe that humans have the remarkable ability to infer missing information using assumptions on the physical plausibility of scenes. For instance, if we see objects lying on a table, we immediately conclude that the shape of the objects is constrained by the table surface. We would be surprised if an object continues within the table. Our approach uses this concept to get a coarse estimate about the shape on the hidden side of the objects[2].

1: *i.e.*, the geometries and relative poses of the objects at different points in time

2: *i.e.*, the side facing away from the camera

In this chapter we want to build a model that uses physical plausibility constraints to answer the question: "How far does the object extend to the background if we have a physically plausible scene configuration in every frame?"[3]. We propose Co-Section, a novel approach to incorporate such constraints into a bottom-up 3D scene reconstruction method. Our method tracks and maps moving objects in the scenes. However, in dynamic scenes[4], we can use more than just the image measurements to reason about the object shapes. Physical plausibility dictates that any point in 3D is only occupied by at most a single object at any point in time. Objects thus cannot penetrate each other and the geometry of each object is constrained not to occupy the space already occupied by other objects moving close to it or touching it. In contrast to a variety of learning-based approaches for shape completion (Dai et al. 2018; Firman et al. 2016; Song et al. 2017; Yang et al. 2019), we incorporate such physical plausibility as intersection constraints in an energy-minimization framework.

3: Or, in short "Where does it (the object) end?"

4: *i.e.*, scene with moving objects

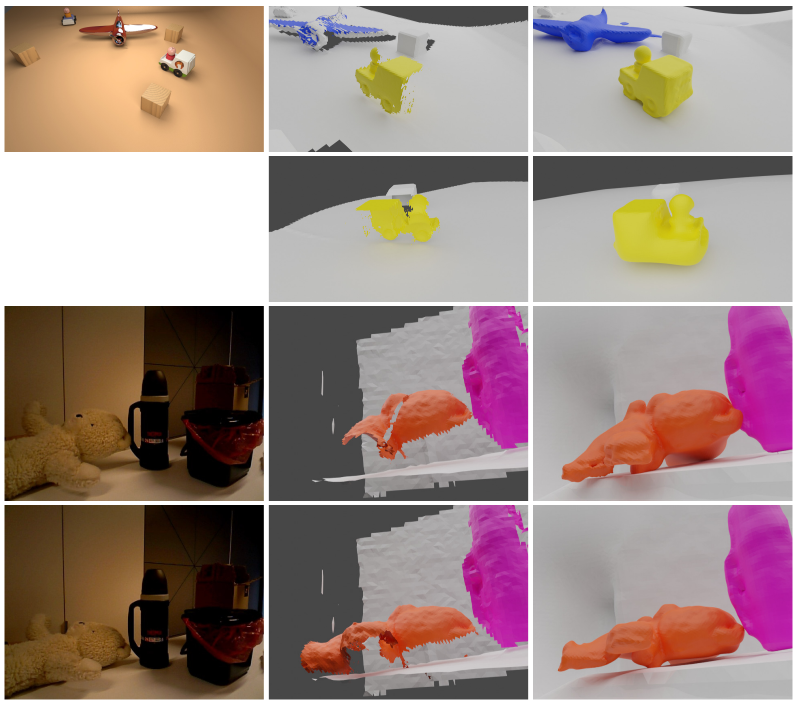


Figure 4.1: We propose a novel energy minimization approach to 3D reconstruction in dynamic scenes. Our approach uses a dynamic object-level SLAM method as front-end (such as EM-Fusion; Strecke and Stueckler 2019). Compared to the TSDF object maps of the front-end, our approach can leverage intersection constraints between objects and visibility constraints to complete object shapes from scenes published with Co-Fusion (Rünz and Agapito 2017) in a physically plausible way. The top two rows show the same time step in the scene *ToyCar3* from different viewpoints. The bottom rows show different time steps from the *PlaceItems* scene. Note that the shape reconstructed by Co-Section in the first frame would penetrate the table with the changed object pose in the second frame. The shape of the teddy is thus adapted to satisfy the intersection constraint.

Our formulation optimizes the implicit surfaces of the objects as signed distance functions from oriented points. The oriented point measurements are computed from the depth maps (see Section 2.3) and transformed to the object models with the camera and object poses provided by a dynamic object-level SLAM front-end. The dynamic SLAM front-end EM-Fusion (see Chapter 3; Strecke and Stueckler 2019) detects, segments, tracks and 3D reconstructs the objects in local volumetric signed distance function (SDF) representations. Our energy minimization back-end applies a method for implicit surface reconstruction from registered point clouds (Schroers et al. 2014) to optimize the object maps. We incorporate intersection and hull constraints to complete the object shapes. In Figure 4.1, we can see that the truck can be closed plausibly by our optimization (top two rows) and that reconstructions by our approach improve as more information becomes available. In the third row, the optimized reconstruction of the teddy is not yet well-constrained and the teddy model would penetrate the table with the changed pose in the last row. Thus, our approach uses the more constraining object pose in the last row and adapts the teddy geometry to avoid this penetration.

We demonstrate our approach on real and synthetic dynamic scene datasets. We also assess our shape completion quantitatively using the ground-truth 3D models on the synthetic scenes.

In summary, our contributions are

- We include intersection constraints for physically plausible shape completion in dynamic scenes. This allows for inferring hidden surfaces of objects which can be useful for robotics or VR/AR applications. To the best of our knowledge, we are the first to include such constraints for object-level 3D reconstruction in dynamic scenes.

- We demonstrate efficient optimization by initializing new object volumes coarse-to-fine and optimizing the volumes incrementally when new measurements are available.
- We assess our approach qualitatively and quantitatively on dynamic SLAM datasets. Our evaluation can serve as a baseline for future research in this area.
- We provide the source code of our approach to allow other researchers to reproduce our results and build upon them in future research[5] .

5: `https://github.com/EmbodiedVision/cosection`

4.2 Related Work

4.2.1 Dense Surface Reconstruction

Several approaches reconstruct 3D surfaces densely from point clouds (Calakli and Taubin 2011; Kazhdan, Bolitho, et al. 2006; Schroers et al. 2014; Ummenhofer and Brox 2015), depth images (Curless and Levoy 1996) or RGB images from multiple view points (Cremers and Kolev 2011). The methods typically consider the data as belonging to a single surface and do not reason about the surfaces of individual objects.

Dense Reconstruction from Point Clouds. Moving least squares surface reconstruction (Alexa et al. 2003) fits a point set representing the surface to oriented point samples. Local parameterizations such as planar surface patches are assumed and the points are fit using moving least squares. Smooth signed distance surface reconstruction (SSD; Calakli and Taubin 2011) recovers a volumetric implicit surface representation from the oriented points using a variational energy minimization formulation. The energy includes data terms favoring the points lying on the zero level set of the optimized implicit function and aligning its gradients with the surface normals. A regularization term keeps the Hessian[6], of the implicit function small. While we also use this regularization term, we use a data term that measures the distance of a voxel towards a point along the normal. In Poisson surface reconstruction (Kazhdan, Bolitho, et al. 2006), a smoothed indicator function is estimated whose gradient follows the surface normal field approximated with the oriented points. The implicit function is recovered using a least squares approximation of the Poisson equation. Implicit moving least squares formulations recover an implicit surface function at a set of points or voxel centers using a quadratic penalty on the deviation from local point samples along their normals. Shen et al. (2004) estimate a fit of linear function coefficients to point samples. Our formulation extends the Hessian-IMLS formulation by Schroers et al. (2014) which combines a quadratic data penalty term similar to (Shen et al. 2004) with the L^2-Hessian regularizer as in (Calakli and Taubin 2011) to estimate the signed distance function on a grid. The approach by Ummenhofer and Brox (2015) uses L^1 norms on the data terms of SSD and a total variation regularizer on the gradient of the vector field[7] which affords a primal-dual optimization scheme. We favor the L^2-Hessian regularization of the implicit function over total variation regularization of its vector field, since we are interested in recovering

6: *i.e.*, the second-order derivatives

7: *i.e.*, keeping the norm of the gradient small smooth implicit surface functions without discontinuities in order to complete object shapes watertight.

Dense Reconstruction from RGB-D Images. Several approaches fuse depth images which can be recovered using multi-view stereo or active sensing principles such as structured light or time-of-flight (see Section 2.3). A popular approach for incremental fusion of depth maps into volumetric signed distance functions is the seminal approach by Curless and Levoy (1996)[8], which we used in Chapter 3 for building object models. Cremers and Kolev (2011) optimize the signed distance function directly from intensity[9] images using multi-view stereo and silhouette constraints. These dense reconstruction methods are restricted to static scenes and do not incorporate intersection constraints between objects like our method.

8: see Subsection 3.3.3

9: *i.e.*, gray scale

4.2.2 Dense Object-Level 3D Reconstruction in Dynamic Environments

Several methods have been proposed that track and map moving objects with RGB-D cameras (Hachiuma et al. 2019; Rünz and Agapito 2017; Rünz, Buffier, et al. 2018; B. Xu et al. 2019) and (Strecke and Stueckler 2019, Chapter 3). However, none of the methods consider mutual constraints between the objects for inferring hidden surfaces at their intersections. Some works also consider interactions between objects, *e.g.*, between human bodies and the static environment (Hassan et al. 2019), or hands and objects (Hasson et al. 2019). However, these methods require learned parametric deformation models of bodies and hands. They furthermore rely on deep learning for pose and shape regression from images.

4.2.3 Shape Completion

Various learning-based approaches to completing volumetric reconstructions obtained with depth cameras have been proposed (Dai et al. 2018; Firman et al. 2016; Song et al. 2017; Yang et al. 2019). Firman et al. (2016) train random forests to map depth image patches to local 3D voxel representations. Song et al. (2017) complete a voxel representation of single depth images. They train a 3D convolutional neural network (CNN) end-to-end on a synthetic 3D scene dataset. ScanComplete (Dai et al. 2018) applies 3D CNNs in a hierarchical fashion to complete and semantically label voxel grids of point clouds fusing multiple views. Yang et al. (2019) propose an approach based on Generative Adversarial Networks (GANs) to predict completed shapes from depth images. X-Section (Nicastro et al. 2019) predicts the end-point of an object along a ray which can be used with volumetric SDF fusion (Curless and Levoy 1996, Subsection 3.3.3) to obtain completed shapes.

Our energy minimization approach includes physical plausibility constraints which do not need to be learned from data. It thus provides an orthogonal approach to these learning-based approaches.

4.3 Preliminaries

We extend the *incremental* online mapping from Chapter 3 to retrieve *globally optimal* models in an offline optimization approach. Schroers et al. (2014) proposed a formulation generalizing previous optimization-based surface reconstruction methods (*e.g.*, Calakli and Taubin 2011; Kazhdan, Bolitho, et al. 2006) based on the *calculus of variations*.

We want to find an SDF ϕ^* which is optimal given oriented point measurements accumulated over several frames from depth maps[10]. Generally, this problem can be phrased as minimizing an *energy functional* $E : \mathcal{V} \to \mathbb{R}$, where $\mathcal{V} = \left\{\phi \mid \phi : \Omega \to \mathbb{R}\right\}$ is the space of admissible SDF functions defined on a domain $\Omega \subset \mathbb{R}^3$:

10: see Section 2.3

$$\phi^* = \underset{\phi \in \mathcal{V}}{\arg\min}\, E(\phi). \tag{4.1}$$

We call E a *functional* because its argument is itself a function in our case. As mentioned before, optimizing Equation (4.1) in the general sense is often done using methods from the calculus of variations, which poses the necessary condition for minimizers of Equation (4.1) as

$$\frac{\partial E}{\partial \phi}(\phi) = 0, \tag{4.2}$$

where we call the derivative $\frac{\partial E}{\partial \phi}$ the first *variation* of E, which is induced by infinitesimal changes[11] in the *function* ϕ:

11: Very similar to the definition of the derivatives of functions.

$$\frac{\partial E}{\partial \phi}(\phi) = \lim_{h \to 0} \frac{E(\phi + h\eta) - E(\phi)}{h}, \tag{4.3}$$

where $\eta : \Omega \to \mathbb{R}$ is an "increment function" (Bolza 1904; Rindler 2018). Variational calculus and its applications in optimization problems are the topics of several textbooks (Bolza 1904; Boyd and Vandenberghe 2004; Rindler 2018; Ralph Tyrell Rockafellar 1970; Ralph Tyrell Rockafellar and Wets 1998), to which we refer for more details[12].

12: These details include more restrictions on the function spaces of ϕ, η and E. As a thorough introduction is beyond the scope of this thesis, we refer the interested reader to (Rindler 2018)

Schroers et al. (2014) propose finding ϕ^* by setting E in Equation (4.1) to the energy functional

(Schroers et al. 2014): *A Variational Taxonomy for Surface Reconstruction from Oriented Points*

$$E(\phi) = \sum_{k=0}^{K} \alpha_k D_k(\phi) + \alpha S(\phi), \tag{4.4}$$

consisting of a smoothness functional $S(\phi)$ and a data term for the k-th derivative and N oriented point measurements $\widetilde{\mathcal{S}} = \{(\mathbf{p}_i, \mathbf{n}_i)\}_{i=1}^{N}$:

$$D_k(\phi) = \sum_{i=1}^{N} \int_\Omega \frac{w_{i,k}}{d_k} \left\| \mathcal{D}^{(k)} \left(\phi - \phi_{\mathbf{p}_i}\right) \right\|^2 \mathrm{d}\mathbf{x}, \tag{4.5}$$

where we omit the argument $\mathbf{x}$ to the functions $w_{i,k}$, d_k, ϕ and $\phi_{\mathbf{p}_i}$ in Equation (4.5) to avoid notational clutter. In Equation (4.5),

$$\phi_{\mathbf{p}_i}(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_i)^\top \mathbf{n}_i \tag{4.6}$$

is the point-to-plane distance from $\mathbf{x}$ to point sample i and $\mathcal{D}^{(k)}$ is the operator implementing the k-th derivative in all dimensions. The function

$w_{i,k}$ is a weight adjusting the influence of point sample i and can be chosen differently for every order of derivative k. Finally, d_k accounts for possible weight normalization. One example for the weighting function would be $w_{i,k}(\mathbf{x}) = w_\sigma(\|\mathbf{x} - \mathbf{p}_i\|)$ with

$$w_\sigma(s) = \begin{cases} 1, & \text{if } |s| \leq \sigma, \\ 0, & \text{otherwise,} \end{cases} \tag{4.7}$$

for constraining the influence of point sample i to the σ-ball or

$$w_\sigma(s) = \exp\left(-\left(\frac{s}{\sigma}\right)^2\right), \tag{4.8}$$

for a smooth decaying Gaussian weight.

This energy formulation generalizes other popular surface reconstruction methods, *e.g.*, for $K = 1$, the formulation for Poisson Surface Reconstruction (Kazhdan, Bolitho, et al. 2006; Kazhdan and Hoppe 2013) can be derived. We now consider the cases $K = 0$ and $K = 2$ in more detail, as they provide the basis for the formulation used in our approach. Implicit Moving Least Squares (IMLS; Shen et al. 2004) results from Equation (4.4) for $K = 0$, $\alpha = 0$ and $d_0 = 1$:

(Kazhdan, Bolitho, et al. 2006): *Poisson Surface Reconstruction*
(Kazhdan and Hoppe 2013): *Screened poisson surface reconstruction*

(Shen et al. 2004): *Interpolating and approximating implicit surfaces from polygon soup*

$$E(\phi) = \sum_{i=1}^{N} \int_\Omega w_{i,0} \left(\phi - \phi_{\mathbf{p}_i}\right)^2 \mathrm{d}\mathbf{x}, \tag{4.9}$$

which has the analytic solution (Schroers et al. 2014)

$$\phi^*(\mathbf{x}) = \frac{\sum_{i=0}^{N} w_{i,0}(\mathbf{x}) \phi_{\mathbf{p}_i}(\mathbf{x})}{\sum_{i=0}^{N} w_{i,0}(\mathbf{x})}. \tag{4.10}$$

The Smooth Signed Distance Surface Reconstruction (SSD; Calakli and Taubin 2011) formulation can be recovered in the higher-order framework by setting $K = 2$ and $\alpha = 0$. Further, $w_{i,0}$ and $w_{i,1}$ should be set to Dirac distributions for creating point-wise constraints and $w_{i,2} = \frac{1}{N}$ as a global constraint. The normalization factor d_k is chosen to automatically scale the data term by all N weight functions:

(Calakli and Taubin 2011): *SSD: Smooth Signed Distance Surface Reconstruction*

$$d_k = \sum_{i=1}^{N} \int_\Omega w_{i,k}(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{4.11}$$

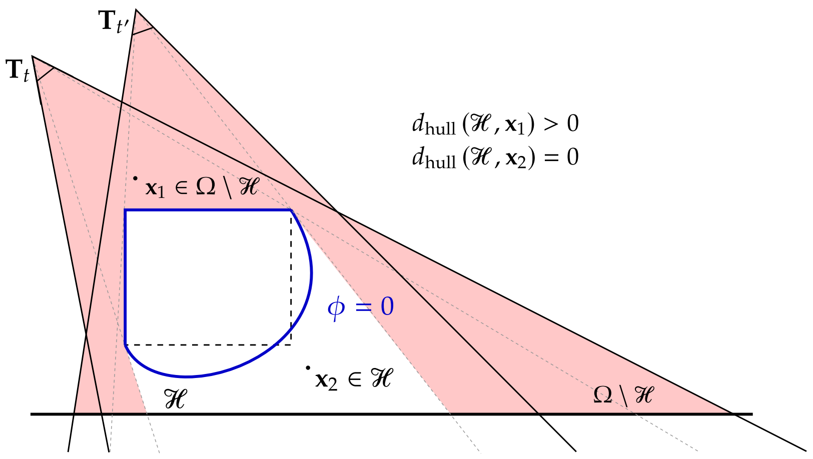
Plugging this into Equation (4.4) yields:

$$E(\phi) = \frac{\alpha_0}{N} \sum_{i=1}^{N} \left\|\phi\left(\mathbf{p}_i\right)\right\|^2 + \frac{\alpha_1}{N} \sum_{i=1}^{N} \left\|\nabla\phi\left(\mathbf{p}_i\right) - \mathbf{n}_i\right\| + \frac{\alpha_2}{N} \int_\Omega \left\|\mathbf{H}\phi(\mathbf{x})\right\|_F^2 \mathrm{d}\mathbf{x}, \tag{4.12}$$

where $\mathbf{H}$ denotes the *Hessian* operator[13]. Equation (4.12) follows from Equation (4.4) by recalling that $\phi_{\mathbf{p}_i}(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_i)^\top \mathbf{n}_i$ and thus $\nabla\phi_{\mathbf{p}_i}(\mathbf{x}) = \mathbf{n}_i$ and $\mathbf{H}\phi_{\mathbf{p}_i}(\mathbf{x}) = 0$.

13: *i.e.*, $\mathbf{H}\phi(\mathbf{x})$ is a matrix containing all second-order derivatives of ϕ at $\mathbf{x}$

Schroers et al. (2014) propose the following combination of the first-order data term from Equation (4.9) with the Hessian regularizer in



$\mathbf{T}_{t'}$
$\mathbf{T}_t$
$d_{\text{hull}}(\mathscr{H}, \mathbf{x}_1) > 0$
$d_{\text{hull}}(\mathscr{H}, \mathbf{x}_2) = 0$
$\mathbf{x}_1 \in \Omega \setminus \mathscr{H}$
$\phi = 0$
$\mathbf{x}_2 \in \mathscr{H}$
$\mathscr{H}$
$\Omega \setminus \mathscr{H}$

Figure 4.2: The hull constraint effectively limits the reconstructed surface within the observed free-space.

Equation (4.12):

$$E_{\text{Hessian-IMLS}}(\phi) = \underbrace{\sum_{i=1}^{N} \int_{\Omega} w_i \left(\phi - \phi_{\mathbf{p}_i}\right)^2 \mathrm{d}\mathbf{x}}_{=:E_{\text{data}}(\phi)} + \alpha \underbrace{\int_{\Omega} \left\|\mathbf{H}\phi\right\|_F^2 \mathrm{d}\mathbf{x}}_{=:E_{\text{reg}}(\phi)}, \tag{4.13}$$

with $w_i = w_\sigma(\|\mathbf{x} - \mathbf{p}_i\|)$, where w_σ is the Gaussian weight from Equation (4.8). They argue that this is the simplest choice within the higher order framework combining the simple point-to-plane distance from IMLS with appropriate regularization. The Hessian regularizer encourages the gradients of the reconstructed SDF to vary smoothly on Ω. This, together with the point-to-plane distance from Equation (4.6), which encourages the SDF to follow the unit surface normals, poses a prior on the SDF similar to the Eikonal constraint from Equation (2.36)[14]. This combination of data term and regularization can propagate the distance information from the point samples even to points far away in the SDF volume, as for those points the influence w_i of the data term E_{data} is low and the regularizer E_{reg} encourages the SDF values to increase along the surface normals.

14: *i.e.*, the gradients should have unit norm close to the surface and should stay almost constant away from the surface, thus staying close to the unit norm

Schroers et al. (2014) further propose an optional hull constraint

$$E_{\text{hull}}(\phi) = \beta \int_{\Omega \setminus \mathscr{H}} \max\left\{0, d(\mathscr{H}, \mathbf{x}) - \phi(\mathbf{x})\right\}^2 \mathrm{d}\mathbf{x}, \tag{4.14}$$

encouraging the surface to lie within a hull $\mathscr{H} \subset \Omega$ with, where

$$d(\mathscr{H}, \mathbf{x}) = \min_{\mathbf{p} \in \mathscr{H}} \|\mathbf{x} - \mathbf{p}\| \tag{4.15}$$

denotes the distance of the point $\mathbf{x}$ to the hull $\mathscr{H}$. A hull can often be estimated, *e.g.*, as the visual hull from the silhouette of the point cloud in different views. Essentially, this constrains the reconstructed implicit surface to a reasonable area and proves beneficial in their experiments (Schroers et al. 2014). We illustrate the hull constraint in Figure 4.2 and explain how we estimate the hull in our approach in Subsection 4.4.2.

We reconstruct SDFs on discrete grids (see Subsection 2.2.2), *i.e.*, we optimize the SDF values $\phi_{i,j,k} = \phi\left(\mathbf{v}_{i,j,k}\right)$ as in Equation (2.40) directly. Following Schroers et al. (2014), we thus optimize the discretized version of Equation (4.13):

$$E(\boldsymbol{\phi}) = \sum_{i=1}^{N} \left\| \mathbf{W}_i^{\frac{1}{2}} \left(\boldsymbol{\phi} - \boldsymbol{\phi}_{\mathbf{p}_i} \right) \right\|^2 + \alpha \sum_{j=1}^{M} \sum_{\gamma \in V^2} \left(\mathbf{D}_\gamma \boldsymbol{\phi} \right)_j^2 , \tag{4.16}$$

where we collect the weight functions w_i in diagonal matrices $\mathbf{W}_i$[15]. We further denote the discretized versions of ϕ and $\phi_{\mathbf{p}_i}$ by $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_{\mathbf{p}_i} \in \mathbb{R}^M$, respectively (see Subsection 2.2.2)[16]. The matrices $\mathbf{D}_\gamma$ implement the finite difference kernels for the second order derivatives in directions $\gamma \in V^2$ with $V = \{x, y, z\}$. The hull constraint in Equation (4.14) remains mostly the same, just transforming the integral into a sum for the indices for which the corresponding grid point is outside the hull.

15: As $\mathbf{W}_i$ is diagonal, $\mathbf{W}_i^{\frac{1}{2}}$ is just the element-wise square root.

16: We use linearized indices as in Equation (2.41) here.

In this setting, because Equation (4.13) is convex in ϕ, the necessary condition from Equation (4.2) is also sufficient for finding a minimizer (Schroers et al. 2014). Setting the derivative of Equation (4.16) to zero yields

$$\left(\sum_{i=1}^{N} \mathbf{W}_i + \alpha \sum_{\gamma \in V^2} \mathbf{D}_\gamma^\top \mathbf{D}_\gamma \right) \boldsymbol{\phi} = \sum_{i=1}^{N} \mathbf{W}_i \boldsymbol{\phi}_{\mathbf{p}_i} . \tag{4.17}$$

If the hull constraint is active, we further need the derivative of Equation (4.14), which can be expressed component-wise in its discrete form as

$$2 \beta H(d_j)\, H(d_j - \phi_j) \, (\phi_j - d_j) , \tag{4.18}$$

where $\mathbf{d}$ denotes the discrete version of $d(\mathcal{H}, \mathbf{x})$ and

$$\begin{aligned} H : \mathbb{R} &\longrightarrow \{0, 1\} \\ x &\longmapsto \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \end{aligned} \tag{4.19}$$

is the Heaviside step function. Note that ϕ_j here denotes index j of the *discrete vector* $\boldsymbol{\phi}$, in contrast to the time index t for the *function* ϕ in Subsection 3.3.3.

The linear system of equations in Equation (4.17), possibly with the added term from Equation (4.18) for the corresponding indices, can be solved by the Fast Jacobi (FJ) solver by Weickert et al. (2015).

4.4 Method

In our method, we use the previously explained optimization framework to extend our object-level dynamic SLAM method EM-Fusion from Chapter 3. As mentioned before, our goal is to improve the object-level 3D reconstruction by including physical plausibility constraints such as the hull constraint Equation (4.14) and a novel intersection constraint, which we will explain in this section. The key idea behind the intersection constraint is that dynamic objects reveal some information about their maximum extend in the viewing direction if they pass in front of another known surface. We use this information to infer where an object ends in directions that have not been directly observed.

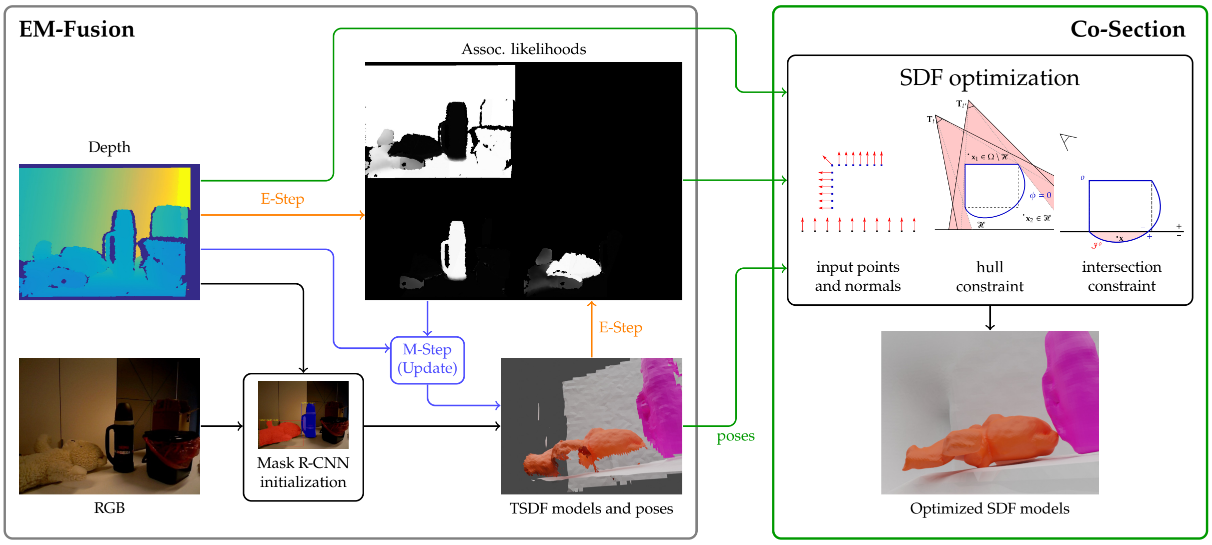


Figure 4.3: Our approach uses input depth maps and the estimated object and camera poses as well as the pixel-wise object association likelihoods from EM-Fusion (Strecke and Stueckler 2019) in a global optimization framework to retrieve optimized SDF models that take physical plausibility constraints into account. EM-Fusion uses Mask R-CNN (He et al. 2017) segmentations to initialize objects and subsequently tracks and maps them in an EM-like framework by estimating geometric association likelihoods from existing models and input depth maps.

4.4.1 Object Tracking and Data Association

For implementing our optimization method, we need to associate depth measurements to object models and recover the relative poses between the camera and the objects in each frame. As we have explained in detail in Chapter 3 and visualized in the left part of Figure 4.3, EM-Fusion (Strecke and Stueckler 2019) can provide this information[17].

(Strecke and Stueckler 2019): *EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association*

17: Note that one could also use other dynamic SLAM methods (Hachiuma et al. 2019; Rünz and Agapito 2017; Rünz, Buffier, et al. 2018; B. Xu et al. 2019) for this purpose.

While EM-Fusion also reconstructs dense 3D models for individual objects as truncated signed distance fields (TSDFs), we will show that these models can be improved by relaxing computation time constraints and performing *offline optimization* instead of *online mapping*. We find the signed distance field (SDF) representation especially useful for the goal of including intersection constraints since it contains distance information to the closest surfaces. Additionally, it gives information whether a 3D point is inside or outside an object by negative or positive values, respectively.

In addition to including the physical plausibility constraints mentioned before, we want to address another issue with the weighted average integration from Chapter 3 (Curless and Levoy 1996; Newcombe et al. 2011). Although this approach has been proven to yield a maximum likelihood estimate of the surface (Curless and Levoy 1996; Newcombe 2012; see Subsection 3.3.3), this method cannot map thin structures and edges of objects well if they are viewed from different sides. Choosing the truncation threshold for both conditions mentioned in Subsection 2.2.3[18] is not always possible as we illustrate in Figure 4.4. The black line indicates the surface, the dashed black lines show the truncation distance, and the light blue/orange frustum show the space around the surface observed by the cameras. The lines of sight, along which the projective signed distance is measured, are indicated by dotted lines. The point **x** is within the truncation threshold (indicated by dashed lines) from both camera poses $\mathbf{T}_t$ and $\mathbf{T}_{t'}$. It will thus receive conflicting integration measurements as its measured SDF value[19] is positive for $\mathbf{T}_t$ and negative for $\mathbf{T}_{t'}$[20].

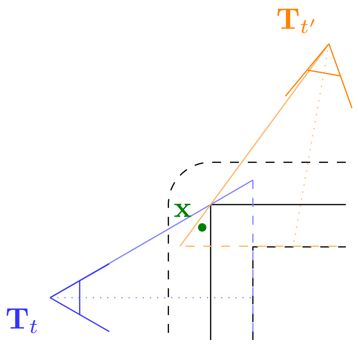


Figure 4.4: Weighted average TSDF integration can lead to conflicting measurements.

18: *i. e.*, small enough to prevent interference between measurements from different sides and large enough to multiple views to remove noise.

19: see Equation (3.2)

20: *i. e.*, $\check{\phi}_t(\mathbf{x}) > 0$ and $\check{\phi}_{t'}(\mathbf{x}) < 0$

We address this issue by collecting the depth maps as raw surface measurements and employing the optimization framework explained in Section 4.3 to retrieved globally optimized models. We collect the depth maps at equidistant time steps and compute oriented point clouds (consisting of points $\mathbf{p}_i$ and their corresponding normals $\mathbf{n}_i$) from them given the poses estimated by EM-Fusion. These oriented point clouds are then used in the data term in Equation (4.13).

4.4.2 Global SDF Optimization

Energy Formulation. As explained before in Section 4.3, we adopt the Hessian-IMLS energy formulation by Schroers et al. (2014). Our energy formulation uses point measurements associated with object models by EM-Fusion, our implementation of the hull constraint for dynamic scenes, and our novel intersection constraint explained below. The overall energy that we optimize is the sum of three parts:

$$E(\phi) = E_{\text{Hessian-IMLS}}(\phi) + E_{\text{hull}}(\phi) + E_{\text{inter}}(\phi), \tag{4.20}$$

where $E_{\text{Hessian-IMLS}}$ and E_{hull} are defined in Equations (4.13) and (4.14), respectively.

We compute the weight

$$w_i^o(\mathbf{x}) = w_\sigma(\|\mathbf{x} - \mathbf{p}_i\|) \cdot a^o(\mathbf{p}_i) \tag{4.21}$$

in Equation (4.13) as the product of the Gaussian weight in Equation (4.8) and the association likelihood $a^o(\mathbf{p}_i)$ of point $\mathbf{p}_i = \pi^{-1}(\mathbf{u}, d_t(\mathbf{u}))^{21}$ to object o from EM-Fusion, *i.e.*,

21: see Section 2.3

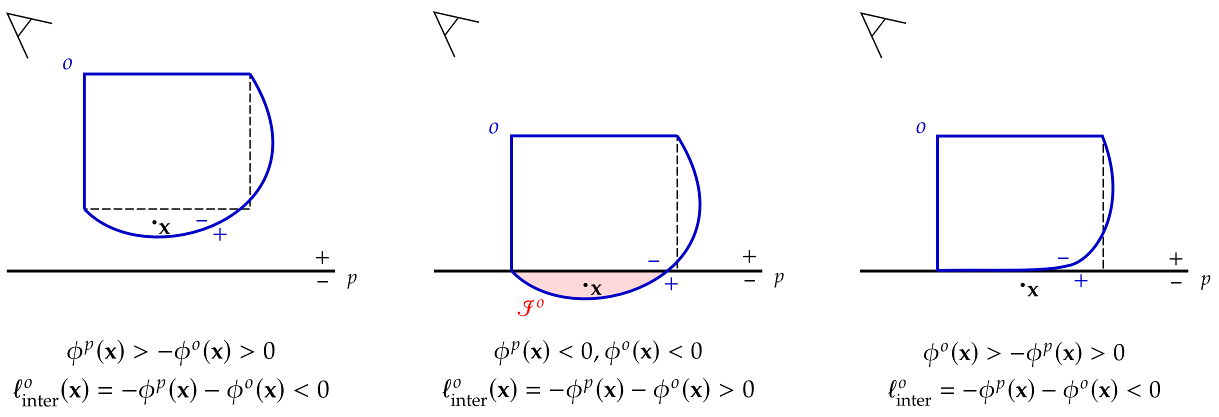
$$a^o(\mathbf{p}_i) = q(c_{t,\mathbf{u}}) \tag{4.22}$$

for $c_{t,\mathbf{u}} = o$. We set σ equal to the voxel size in our experiments and cap the weights when the distance to the measurement is larger than 3σ.

Hull Constraint. As mentioned in Section 4.3, we can compute a hull constraint as proposed by Schroers et al. (2014). Since the data we collect is not just the raw oriented point cloud, but depth measurements over time, we implement the hull $\mathcal{H}$ to consist of all unseen parts of the scene (hidden behind seen surfaces or outside the field of view). Furthermore, to avoid the potentially expensive exact computation of $d_{\text{hull}}(\mathcal{H}, \mathbf{x})$, we use the voxel size as a lower bound for all voxels $\mathbf{v} \in \Omega \setminus \mathcal{H}$. In practice our implementation records $d_{\text{hull}}(\mathcal{H}, \mathbf{v})$ as

$$d_{\text{hull}}(\mathcal{H}, \mathbf{v}) = \begin{cases} v_o, & \text{if } \check{\phi}_t^o(\mathbf{v}) > v_o \text{ for any } t, \\ 0, & \text{otherwise,} \end{cases} \tag{4.23}$$

where $\check{\phi}_t^o(\mathbf{v})$ is the projective SDF measurement for object o in frame t from EM-Fusion defined in Equation (3.2) and v_o is the voxel size of the considered volume. We effectively use the projective SDF measurements from EM-Fusion in a space carving approach to record a mask for the empty space $\Omega \setminus \mathcal{H}$.



o
$\mathbf{x}$
$-$ $+$
$+$ $-$ p
$\phi^p(\mathbf{x}) > -\phi^o(\mathbf{x}) > 0$
$\ell^o_{\text{inter}}(\mathbf{x}) = -\phi^p(\mathbf{x}) - \phi^o(\mathbf{x}) < 0$
o
$\mathbf{x}$
$-$ $+$
$\mathcal{I}^o$
$+$ $-$ p
$\phi^p(\mathbf{x}) < 0, \phi^o(\mathbf{x}) < 0$
$\ell^o_{\text{inter}}(\mathbf{x}) = -\phi^p(\mathbf{x}) - \phi^o(\mathbf{x}) > 0$
o
$\mathbf{x}$
$-$ $+$
$+$ $-$ p
$\phi^o(\mathbf{x}) > -\phi^p(\mathbf{x}) > 0$
$\ell^o_{\text{inter}} = -\phi^p(\mathbf{x}) - \phi^o(\mathbf{x}) < 0$

Figure 4.5: Our intersection constraint is based on a measure of interpenetration distance d_{inter}. It penalizes signed distance functions that interpenetrate each other. Left: interpenetration distance for two objects that do not intersect each others. Middle: case of two intersecting objects. Right: Distance after resolved interpenetration.

Intersection Constraint. We propose to implement an intersection constraint for multiple objects with a similar form like the hull constraint in Equation (4.14). For each object o in the set of objects $\mathbb{O}$, we define the intersection distance d_{inter} based on the observations of multiple (moving) objects in several time steps t:

$$d^o_{\text{inter}}(\mathbf{x}) = \max\left\{\max_{t\in T, p\in\mathbb{O}}\left\{-\phi^p(\mathbf{x}_t)\right\}, 0\right\}. \tag{4.24}$$

Here, $\mathbf{x}_t = (\mathbf{T}^p_t)^{-1}\mathbf{T}^o_t\mathbf{x}$ denotes the point $\mathbf{x}$ transformed from the coordinate system of object o to object p at time step t using the poses $\mathbf{T}^o_t$ and $\mathbf{T}^p_t$ of o and p at that time step. Intuitively, this distance measures the maximum penetration of a point $\mathbf{x}$ of object o in object p over all time steps.

Using this distance, we define the energy term

$$E_{\text{inter}}\left(\phi^o\right) = \beta_{\text{inter}}\int_{\mathcal{I}^o}\max\Big\{0, \underbrace{d^o_{\text{inter}} - \phi^o}_{=:\ell^o_{\text{inter}}}\Big\}^2 \mathrm{d}\mathbf{x}, \tag{4.25}$$

where $\mathcal{I}^o = \left\{\mathbf{x}\in\Omega \,\middle|\, d^o_{\text{inter}}(\mathbf{x}) > 0\right\}$. This term favors SDFs that do not interpenetrate each other. Figure 4.5 illustrates the intersection constraint and the value of ℓ^o_{inter} in several situations.

4.4.3 Implementation Details

Optimization. Since all parts of the energy are convex and differentiable in ϕ, we follow Schroers et al. (2014) and implement the optimization with the Fast Jacobi algorithm proposed by Weickert et al. (2015) as mentioned in Section 4.3. We integrate our approach with EM-Fusion and compute the intersection constraint using relative object poses in every frame. At a lower temporal resolution[22], we insert key frames in which we record the point clouds from the depth maps and compute the hull constraint. The optimization is then run after every key frame insertion.

(Schroers et al. 2014): *A Variational Taxonomy for Surface Reconstruction from Oriented Points*

(Weickert et al. 2015): *Cyclic Schemes for PDE-Based Image Analysis*

22: In our experiments every tenth frame

Since in our case the data measurements are sparse compared to the overall volume size, optimization of the background volume on the finest resolution of 256^3 only converges slowly for the first run. We thus propose to initialize each volume with a coarse-to-fine optimization by optimizing only $E_{\text{Hessian-IMLS}}$ from Equation (4.13) first on a resolution of 32^3 and subsequently up-sampling the result by splitting each voxel into 8 and using this as initialization for the next level until the target resolution is reached. Even without interpolation, this strategy yields reasonably fast and accurate results. For the finest level, we optimize Equation (4.20) and include the hull and intersection constraints. The coarse-to-fine scheme is only applied for the first optimization[23] of each volume. In our experiments, the previous optimization result can adapt reasonably fast to new measurements in later optimization runs. Furthermore, we found the application of a coarse-to-fine scheme in later time steps to actually slow down the optimization in some cases. This is caused by removing fine details at lower resolutions which were already recovered in the previous optimization run.

23: *i.e.*, the optimization after the first key frame

Wrong Data and Inaccurate Detections. The detection of dynamic objects in (Strecke and Stueckler 2019) relies on Mask R-CNN (He et al. 2017). The detection is carried out at low frame rate. This might lead to dynamic objects being missed when they move into view. Consequently, point associations with the background or other object models are incorporated which are difficult to remove later. Since EM-Fusion is mainly designed to track rigid objects, it is fair to assume that space that was seen unoccupied before in a model will not be occupied by the same object in the future. We thus do not allow association of points in this space with the respective model and set $\phi_{\mathbf{p}_i}(\mathbf{x}) = w_i(\mathbf{x}) = 0$ for $\mathbf{x} \in \Omega \setminus \mathcal{H}$. Note that this way, the hull $\mathcal{H}$ will only get smaller over time.

(Strecke and Stueckler 2019): *EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association*

Furthermore, EM-Fusion yields balanced association likelihoods for the background model and objects that remain static from the first view. Including these points in both the object and the background model during our optimization would yield conflicting data terms for the intersection constraint. To avoid this, we use the volumetric foreground probability estimated by EM-Fusion and remove all point measurements in the foreground region of an object model from all other models. While this strategy helps with handling static objects, it might cause dents when models touch each others since it might also remove correct point measurements from the other model. This strategy might not be needed in scenes where no static objects are detected by EM-Fusion.

Ordering the Optimization of Different Volumes. The intersection constraint as formulated in Equation (4.24) introduces a chicken-and-egg-problem. The optimization for object o depends on the optimization results for all other models while those results depend on the optimization result of o. To resolve this, we observe that planar surfaces (as they are common, *e.g.*, for walls and floors in man-made environments) can be completed reasonably well using the Hessian norm prior in Equation (4.17). Thus, to compute the intersection constraint in Equation (4.24), we use the optimized SDF of the background model and approximate

ϕ^p in the case of object models using the point measurements, *i.e.*,

$$\phi^p(\mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} w_i^p(\mathbf{x}) \phi_{\mathbf{p}_i}^p(\mathbf{x}). \tag{4.26}$$

4.5 Experiments

We evaluate our approach in qualitative and quantitative experiments on the data set provided with Co-Fusion (Rünz and Agapito 2017). We use two real and one synthetic scene in which up to three objects move independently. The qualitative results show visually plausible shape completions by our approach, which we further evaluate quantitatively in the synthetic scene using the ground-truth meshes of the objects. In an ablation study, we analyze the effectiveness of the individual components of our energy minimization approach.

We chose the parameters for weighting the different parts of the energy empirically as $\alpha = 0.005$, $\beta_{\text{hull}} = \beta_{\text{inter}} = 0.001$, and choose a cycle length of 20 for the Fast Jacobi optimizer (Weickert et al. 2015). We used a single set of parameters throughout our experiments. For visualization and evaluation of the reconstructed geometry, we compute the triangle mesh from the SDF using the marching cubes algorithm (Subsection 2.2.4, Lorensen and H. E. Cline 1987).

(Lorensen and H. E. Cline 1987): *Marching cubes: A high resolution 3D surface construction algorithm*

4.5.1 Qualitative Results

We provide qualitative results of our approach in Figure 4.6. The first three examples show different viewpoints of the same time step, while the last one (placement of teddy) illustrates how the shape changes over time. The TSDF reconstructions estimated by EM-Fusion can only take the visible part of the object into account. When using our global optimization approach without hull and intersection constraints, the object surface is continued as smooth as possible towards the borders of the object map. The hull constraint wraps the surfaces such that they are limited by the observed free space. However, the surface can freely intersect with the background surface under the object. When including the intersection constraint, this penetration is avoided. However, without the hull constraint, the gradient introduced by the intersection constraint might be continued in regions outside the actual objects, causing unwanted zero-crossings (see column 4 in Figure 4.6). In combination with the hull constraint, these unwanted zero-crossings are removed by penalizing negative SDF values in observed free space. Thus, the object shape is closed and approximates the actual object shape well.

4.5.2 Quantitative Results

Evaluation Metrics. We evaluate shape reconstruction between the reconstructed mesh $\mathcal{M}_{\text{rec}} = (\mathcal{V}_{\text{rec}}, \mathcal{F}_{\text{rec}})$ and the ground-truth mesh $\mathcal{M}_{\text{gt}} = (\mathcal{V}_{\text{gt}}, \mathcal{F}_{\text{gt}})$ using the measures and tools suggested in (Stutz and Geiger 2018)[24]. We sample 10,000 points on the mesh reconstructed from the SDFs and the ground-truth mesh uniformly and compute two evaluation

24: `https://github.com/davidstutz/mesh-evaluation`

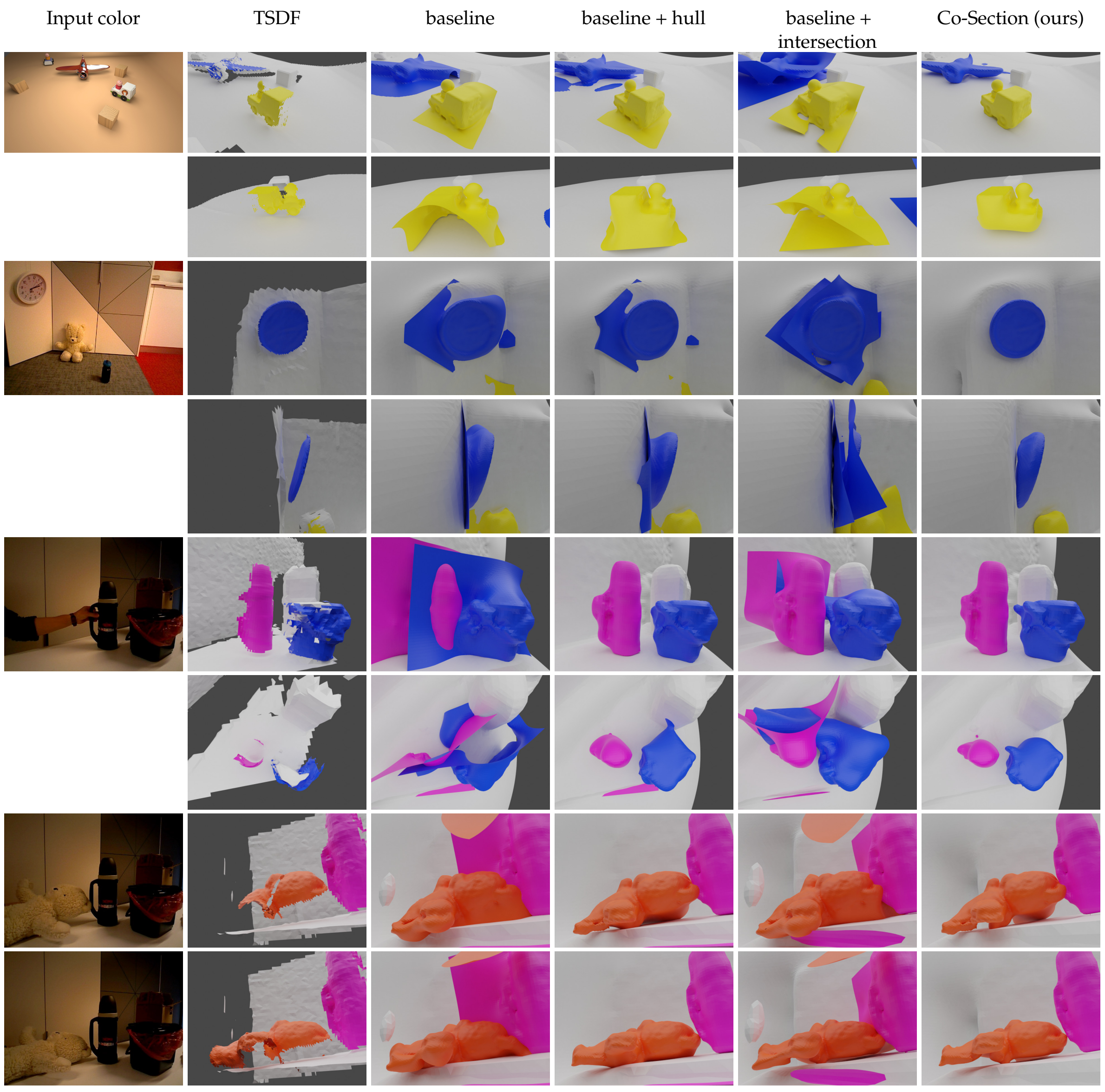


Figure 4.6: Qualitative object shape reconstruction results on Co-Fusion sequences. From top to bottom: truck (*ToyCar3*), clock (*SlidingClock*), bottle and trash can (*PlaceItems*), and teddy (*PlaceItems*). The TSDF shape only includes the observed part of the objects. While the hull constraint limits the surface in the observed free space, the intersection constraint closes the object shape at intersections with other shapes. Our full approach recovers closed surfaces which approximate the actual object shapes. Note that we do not show "person" objects such as the arm in the bottle and trash can example.

metrics. The first of these, *accuracy*, is the average distance of each sample point in the reconstructed mesh to the ground-truth mesh:

$$\mathrm{Acc}\left(\mathcal{P}_{\mathrm{rec}}, \mathcal{M}_{\mathrm{gt}}\right) = \frac{1}{|\mathcal{P}_{\mathrm{rec}}|} \sum_{\mathbf{p} \in \mathcal{P}_{\mathrm{rec}}} \min_{\mathbf{f} \in \mathcal{F}_{\mathrm{gt}}} d^{\triangle}(\mathbf{p}, \mathbf{f}), \tag{4.27}$$

where $\mathcal{P}_{\mathrm{rec}}$ is the set of sampled points from $\mathcal{V}_{\mathrm{rec}}$ and $d^{\triangle}(\mathbf{p}, \mathbf{f})$ is the point-to-triangle distance of point $\mathbf{p}$ and triangle face $\mathbf{f}$. The second, *completeness*, is the average distance of each sample from the ground-truth mesh to the reconstructed mesh:

$$\mathrm{Comp}\left(\mathcal{P}_{\mathrm{gt}}, \mathcal{M}_{\mathrm{ref}}\right) = \frac{1}{|\mathcal{P}_{\mathrm{gt}}|} \sum_{\mathbf{p} \in \mathcal{P}_{\mathrm{gt}}} \min_{\mathbf{f} \in \mathcal{F}_{\mathrm{rec}}} d^{\triangle}(\mathbf{p}, \mathbf{f}), \tag{4.28}$$

Table 4.1: Accuracy and completeness (lower is better) on the Co-Fusion *ToyCar3* sequence for different variants of our method. Best in bold. Our full approach is clearly superior in completeness and also performs well in accuracy despite the fact that it "guesses" parts of the object surface from the constraints.

	Truck		Car		Airplane		Average	
Method	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
TSDF	0.0095	0.0376	**0.0020**	0.0277	**0.0365**	0.0281	**0.0160**	0.0311
baseline	0.0334	0.0328	0.0155	0.0196	0.1569	0.0229	0.0686	0.0251
baseline + hull	0.0234	0.0239	0.0118	0.0123	0.1510	0.0175	0.0621	0.0179
baseline + intersection	0.0326	0.0116	0.0163	0.0109	0.1732	0.0203	0.0740	0.0143
Co-Section	**0.0088**	**0.0110**	0.0052	**0.0102**	0.0872	**0.0162**	0.0337	**0.0125**

where $\mathcal{P}_{\text{gt}}$ is the point set sampled from $\mathcal{V}_{\text{gt}}$. Unfortunately, for this evaluation the alignment of the ground-truth mesh is not provided by the data set. Thus, we manually align scale and pose of the ground-truth meshes with our output meshes by selecting point correspondences on the meshes. Note that we only need to align each object once for each optimized map (using the full approach), since all reconstructions are based on the EM-Fusion result and share the same pose estimate.

Results. Table 4.1 lists accuracy and completeness for the objects on the synthetic *ToyCar3* sequence of the Co-Fusion data set. Our full optimization approach strongly improves the completeness of the shape towards the TSDF map. For two of three objects, accuracy is reduced towards the TSDF map. However, this is expected since our method in-paints new object parts which are not modeled by the TSDF. Still, the accuracy of our full approach is better than any variant of the optimization method. We also observe this trend for the average over the three objects in Table 4.1. Note that the larger airplane model results in overall larger error measures (especially for accuracy) and thus has a higher impact on the average. We observe that while the hull constraint tends to improve accuracy, the intersection constraint tends to improve completeness. We conclude that both the hull and intersection constraints are important to achieve the performance of the full model. In Figure 4.7 we show accuracy and completeness per mesh vertex (point-to-triangle distance). For accuracy, this visualization highlights large distances for estimated surfaces that are far from the ground truth. For completeness, parts of the ground truth model that were not well recovered in the reconstruction yield high distances. The color map is computed based on the maximum accuracy or completeness (whichever is larger) for each object.

Our method relies on accurate pose estimates by the underlying dynamic SLAM method. In case of inaccurate pose estimates, hull and intersection volumes might become inaccurate as well, leading to inaccuracies in the surface reconstruction. While we have not observed this problem in our experiments, a stronger weighting of the data term E_{data} in Equation (4.13) could alleviate this problem to some extent.

Computation Times. The intersection constraint can be computed for several voxels in parallel on a GPU. In our experiments this computation takes under 10 ms on average per volume. Similarly, the data measurements and the hull constraint can be computed fast for all the voxels on parallel hardware. The optimization itself is more time-consuming. For the object volumes in our experiments, our implementation of our variant

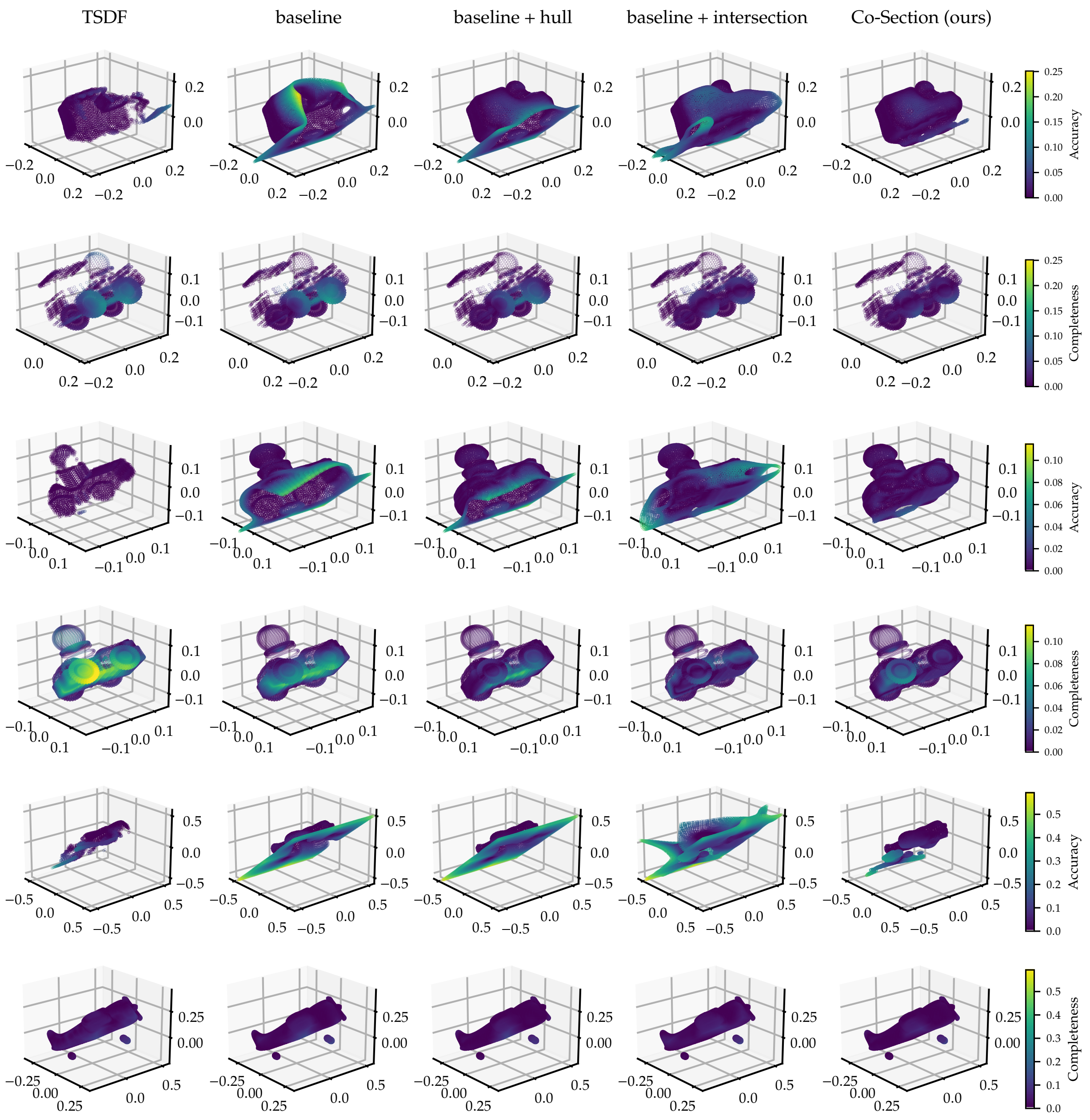


Figure 4.7: Accuracy (top row per object) and completeness (bottom row per object) of object shape reconstruction results on the Co-Fusion *ToyCar3* sequence. Color indicates distance of mesh vertices to the other mesh. From top to bottom: truck, car, airplane. Our full approach is clearly superior to all other variants in accuracy as well as completeness.

of the Hessian-IMLS (Schroers et al. 2014) achieves computation times of several seconds (average 0.98 s, peak 19.8 s). The peak runtime typically occurs when a mostly empty volume needs to be filled from sparse measurements after initialization. We consider further parallelization and improvement of the runtime efficiency of our approach as future work. Schroers et al. (2014) report computation times between 1 s and 4 s for volumes with resolutions up to 400^3 (our resolution is $64^3/256^3$ for objects/background, respectively).

4.6 Conclusion

In this chapter we propose a novel energy minimization approach for object shape completion in dynamic scenes. We incorporate hull and intersection constraints between objects into a formulation which optimizes for the implicit surface in a volumetric representation. The data terms of our method are obtained with a dynamic object-level SLAM front-end which detects, segments, tracks and maps the objects in local TSDF volumes. In our experiments, we demonstrate that our formulation can achieve object shape completion which is physically plausible. We analyze the contributions of the constraints for accuracy and completeness of object shape reconstruction. Our approach improves completeness over the TSDF reconstruction and can achieve high accuracy even if parts of the object are unobserved.

In future work, we would like to investigate the incorporation of further regularization constraints to achieve improved scene reconstruction. Currently, our method is not real-time capable. Optimization of a volume takes from under a second to several seconds. This can still be interesting for back-end optimization in a parallel thread with the front-end. Devising methods for increasing the run-time efficiency is an interesting direction for future research.

DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes

5

The contents of this chapter are based on the peer-reviewed conference publication

©2021 IEEE. Reprinted, with permission, from **Strecke, Michael** and Joerg Stueckler (2021). 'DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes'. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. DOI: `10.1109/3dv53792.2021.00020`, (Strecke and Stueckler 2021),

with the following co-author contributions:

	Ideas	Experiments	Analysis	Writing
Michael Strecke	50%	100%	80%	65%
Jörg Stückler	50%	0%	20%	35%

This chapter contains tables, figures and according descriptions that were originally part of the supplementary material of the conference publication.

Compared to the conference publication, this chapter contains more detailed preliminaries and unified notation with the rest of the thesis.

5.1 Introduction

So far, we only used the 3D information available in the observed scene and simple plausibility reasoning in the recorded scenes to reconstruct and track dynamic objects. We will now combine a stronger prior on the present shapes with a more sophisticated physical model for shape estimation. For combining shape estimation with physical plausibility, we now take an orthogonal approach to Chapter 4. Previously, we assumed fixed given object trajectories and optimized object geometries for physical plausibility by reasoning about empty space that is not occupied by an object. We now want to improve the geometry based on *plausible object motion* in this chapter.

Towards this goal, we model object motion using a differentiable physics simulation. Differentiable physics simulations have recently attracted interest in the computer vision and robotics communities as they allow for the identification of physical parameters like friction by comparing trajectories with estimated parameters to observation trajectories (Geilinger et al. 2020; Hu, Anderson, et al. 2020; Krishna Murthy et al. 2021). The estimated parameters can then be used, *e.g.*, for real-to-sim transfer[1] (Geilinger et al. 2020), which has the potential to allow autonomous agents to reason about interactions with the environment.

1: *i.e.*, replicating real scenes in simulation

However, existing approaches have frequently been limited to objects with primitive shapes or shapes that are known in advance. They are thus not directly applicable in our case, where we want to optimize the shape as part of the physical properties from coarse estimates. To address this issue, we develop a framework capable of simulating objects of complex shapes and optimizing the shape and other physical parameters, such as mass and friction, of these objects (see Figure 5.1). We represent shapes

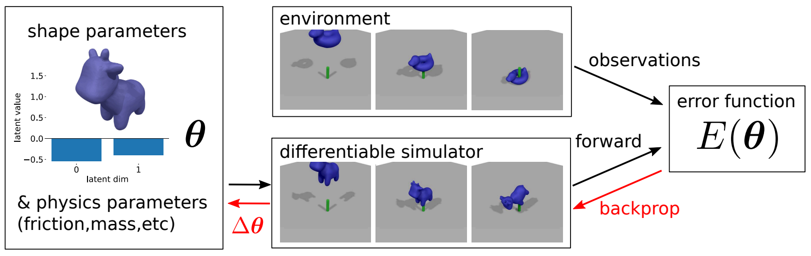


Figure 5.1: Shape and physics parameter optimization through differentiable physics simulation. We represent arbitrary watertight shapes by signed distance functions (SDF) and optimize the parameters through the physics-based dynamics to fit trajectories and depth image observations.

using signed distance functions (SDFs)[2], implicit shape representations that allow for low-dimensional description and detection of collisions for complex shapes. We propose a new differential through the physical dynamics with contacts that allows us to optimize shape parameters.

2: see Section 2.2

While optimizing for general shapes might not be feasible, recent advances in neural implicit models were able to show shape optimization in a low-dimensional latent space using partial depth observations (Park et al. 2019) or rendered silhouettes via differentiable mesh extraction (Remelli et al. 2020) and differentiable rendering (Kato et al. 2018). We thus use learned shape spaces like these and primitive parametric shape models (see Subsection 2.2.1) as a stronger prior on the present shapes.

Our approach is based on a class of constraint velocity-based dynamics simulation methods (Stewart and Trinkle 1996) which lead to linear complementarity problems (LCPs). We build upon the work by Avila Belbute-Peres et al. (2018), who proposed an approach for differentiating the LCPs at the solution to make the simulation differentiable. We represent shapes using signed distance functions (SDFs, see Section 2.2), which represent the object shape as the zero-level set of the signed distance of 3D points to the surface. The sign of the distance defines whether the queried point is inside or outside the object. In our approach, we assume that the SDF is differentiable for the coordinates of the 3D points and shape parameters[3]. We devise methods for differentiating time of contact and mass-inertia tensors which enables gradient-based shape optimization. In experiments, we demonstrate that our approach can be used in optimization approaches which identify physical parameters such as shape, forces, mass and friction from sample trajectories and depth image observations.

3: see Subsection 2.2.1

In summary, the contributions of our work are:

- We propose a novel differentiable physics simulation approach which supports contact handling and differentiable mass-inertia tensor calculation for arbitrary watertight shapes represented as SDFs.
- We develop a novel formulation of differentiable time of contact which enables gradient-based shape optimization through collision constraints.
- We demonstrate that our approach makes shape optimization and system identification feasible for shapes modeled by SDFs from object trajectories and depth image observations in several synthetic scenarios and a real RGB-D image sequence.

5.2 Related Work

Physics Simulation. Over the last decades, a large body of methods for physical simulation has been developed in the computer graphics and mechanical engineering communities (Bender, Erleben, et al. 2013). Physics-based simulation methods can be distinguished as following different paradigms: time stepping or event-driven impulse-based.

Time stepping methods formulate the dynamics by Newton-Euler equations and add equality and inequality constraints to model joints, contacts and collisions. The methods can be phrased as position-based (Müller et al. 2007), velocity-based (Anitescu and Potra 1997; Stewart and Trinkle 1996), or acceleration-based (Baraff 1996) optimization problems. Position-based methods often solve for collision and joint constraints using Gauss-Seidel and fast projection methods. Friction and impulse-conservation laws have to be included in a post-processing step which augment the velocities of the objects. While the methods have been demonstrated to yield visibly plausible results, they are typically difficult to tune for accuracy. Position-based methods can also be extended to simulate deformable objects and fluids by solving collision and deformation constraints between particles. Recently, a differentiable position-based approach has been proposed (Macklin, Erleben, Müller, Chentanez, Jeschke, and T.-Y. Kim 2020). Acceleration-based approaches estimate constraint forces through solving complementarity problems. They can suffer from indeterminacy for Coulomb friction which can be solved by reformulating the optimization problem with contact impulses, leading to a velocity-based formulation (Stewart and Trinkle 1996). In this chapter, we also follow a constraint-based velocity-based approach as proposed in (Anitescu and Potra 1997). Avila Belbute-Peres et al. (2018) make this formulation differentiable at the solution using the OptNet approach (Amos and Kolter 2017). More recently, the method has also been extended to increase efficiency for many objects and mesh-based collision detection (Qiao et al. 2020). Geilinger et al. (2020) propose a different approach for differentiable simulation. As hard constraints for frictional contact are expensive to solve exactly and difficult to differentiate, they propose a mollified contact model which can be applied for rigid and deformable objects. Their approach allows them to tune for a trade-off between accuracy and smoothness of the objective landscapes. Differently, we represent shapes using signed distance functions in a differentiable way and derive a novel time of contact differential which allows for shape optimization from collision constraints.

Another line of research are event-driven impulse-based methods (Bender and Schmitt 2006; Mirtich and Canny 1995; Weinstein et al. 2006) which have been introduced in the seminal work of (Mirtich and Canny 1995). Impulse-based methods iteratively update the velocities of the rigid bodies at the events of contacts until all joint and collision constraints are satisfied.

Other related simulation methods are finite element (Terzopoulos and Fleischer 1988) or meshless methods such as (Hu, Liu, et al. 2019; Sulsky et al. 1995) which are used to simulate deformables and fluids. In contrast to our LCP-based rigid-body physics formulation, these methods cannot model strictly rigid objects or hard collision constraints.

Physical System Identification. Physical system identification from observations has recently attracted attention in the machine learning and computer vision communities. Early approaches use non-differentiable physics engines (Wu, Lu, et al. 2017; Wu, Yildirim, et al. 2015) such as Bullet (Coumans 2010) or integrate specific physical laws for each scenario (Wu, Lim, et al. 2016). Lidec et al. (2021) propose a variant of the staggered projections method to identify Coulomb friction coefficients from objects observed in video via markers. Weiss et al. (2020) estimate material properties of deformables by matching a differentiable deformable simulation to point cloud observations. In (Kandukuri, Achterhold, et al. 2020) a differentiable physics simulation based on (Avila Belbute-Peres et al. 2018) is embedded as layer into a deep neural network which infers the physical state from images and predicts the next states. Krishna Murthy et al. (2021) proposed gradSim, a framework that combines differentiable simulation and differentiable rendering for system identification from video and visual control. The method uses penalty based resolution of contacts for rigid body modeling. Yet, these approaches lack a physical model which supports differentiation for arbitrary watertight shapes like ours.

Parametric Shape Optimization. In recent years, learning based implicit models were proposed for representing families of shapes (Mescheder et al. 2019; Park et al. 2019). Once these models are trained, their decoder models can be interpreted as differentiable parametric shape models (see Subsection 2.2.1). Park et al. (2019) demonstrated that the latent code representing the shape can be optimized to fit depth measurements. Extending the model with differentiable mesh extraction, Remelli et al. (2020) further demonstrated optimization of the shape latent codes *e.g.*, for aerodynamics or to match a given silhouette via differentiable rendering (Kato et al. 2018). In this chapter, we also optimize shape parameters of parametric SDF models. Different to the mentioned methods, we combine the parametric models with a differentiable simulator and fit the shape to match a simulated motion with known parameters.

5.3 Background

We base our formulation on a differentiable velocity-based constraint-based time stepping method. We extend the approach with differentiable SDF shape representations, inertia tensors and time of contact.

5.3.1 Velocity-Based Time Stepping Dynamics

Constraint-based time stepping methods formulate the dynamics as solving the Newton-Euler equations with equality and inequality constraints to model joints, collisions and frition (Anitescu and Potra 1997; Avila Belbute-Peres et al. 2018). The Newton-Euler equations relate wrenches (*i.e.*, torques and forces) acting on the objects in the scene with their motion, *i.e.*,

$$\mathbf{f} = \mathbf{M}\ddot{\mathbf{x}} + \text{Coriolis forces.} \tag{5.1}$$

We denote wrenches by a time dependent mapping $\mathbf{f} : [0, \infty) \longrightarrow \mathbb{R}^{6N}$. The N objects in the scene are described by their mass-inertia matrices $\mathbf{M} \in \mathbb{R}^{6N \times 6N}$ and their poses (positions and orientations) $\mathbf{x} \in \mathrm{SE}(3)^N$, where SE(3) is the special Euclidean group. The block-diagonal matrix $\mathbf{M}$ has the form

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_N \end{pmatrix}, \text{ where } \mathbf{M}_i = \begin{pmatrix} \mathbf{I}_i & \mathbf{0} \\ \mathbf{0} & m_i \mathbb{I}_3 \end{pmatrix} \in \mathbb{R}^{6\times 6}, \tag{5.2}$$

and $\mathbf{I}_i$ denotes the shape-specific inertia tensor, m_i the mass of the object and $\mathbb{I}_3$ the 3×3 identity matrix. We represent the velocities $\dot{\mathbf{x}}_i = \boldsymbol{\xi}_i = \left(\boldsymbol{\omega}_i^\top, \mathbf{v}_i^\top\right)^\top$ of object i by twist coordinates[4], which stack rotational and linear velocities $\boldsymbol{\omega}_i : [0, \infty) \longrightarrow \mathbb{R}^3, \mathbf{v}_i : [0, \infty) \longrightarrow \mathbb{R}^3$, respectively (M. B. Cline 2002).

4: see Subsection 2.1.2

To simulate joints, collisions and friction, corresponding constraints are included. When including friction, the solution of acceleration-based dynamics equations can become indeterminate (M. B. Cline 2002). Hence, acceleration is discretized as

$$\ddot{\mathbf{x}} = \dot{\boldsymbol{\xi}} = \frac{\boldsymbol{\xi}_{t+h} - \boldsymbol{\xi}_t}{h}, \tag{5.3}$$

where $\boldsymbol{\xi}_{t+h}$ and $\boldsymbol{\xi}_t$ are the velocities in successive time steps at times $t + h$ and t, and h is the time step size. Plugging Equation (5.3) into Equation (5.1) (including the Coriolis forces in $\mathbf{f}_{\text{ext}}$), multiplying both sides by h and adding $\mathbf{M}\boldsymbol{\xi}_t$ yields

$$\mathbf{M}\boldsymbol{\xi}_{t+h} = \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_{\text{ext}}. \tag{5.4}$$

Joints between objects impose equality constraints $g_e(\mathbf{x}) = \mathbf{0}$ on their poses and restrict degrees of freedom in their motion. The velocity constraints $\dot{g}_e(\mathbf{x}) = \mathbf{J}_e\boldsymbol{\xi} = \mathbf{0}$ are obtained by computing the derivative of the pose constraints, relating velocities $\boldsymbol{\xi}$ to the derivative with the corresponding Jacobian $\mathbf{J}_e$. In another interpretation, the rows of the Jacobian $\mathbf{J}_e$ are the basis vectors of the forces required to keep the constraint satisfied. Solving for the constraint forces thus reduces to solving for Lagrange multipliers $\boldsymbol{\lambda}_e$, which determine the magnitude of the constraint force. Adding the constraint forces and the velocity constraints to Equation (5.4), we get

$$\begin{aligned} \mathbf{M}\boldsymbol{\xi}_{t+h} &= \mathbf{M}\boldsymbol{\xi}_t + h\,\overbrace{\mathbf{f}_{\text{ext}} + \mathbf{J}_e^\top \boldsymbol{\lambda}_e}^{\text{forces combining external and constraint forces}} \\ \mathbf{J}_e\boldsymbol{\xi}_{t+h} &= \mathbf{0}, \end{aligned} \tag{5.5}$$

which we can write in matrix form as

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}_e^\top \\ \mathbf{J}_e & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_{t+h} \\ \boldsymbol{\lambda}_e \end{pmatrix} = \begin{pmatrix} \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_{\text{ext}} \\ \mathbf{0} \end{pmatrix}. \tag{5.6}$$

Collisions give rise to inequality constraints $g_c(\mathbf{x}) \geq 0$ in the poses which prevent objects from interpenetrating each others. In our 3D simulation, the collision pose constraint function is $g_c(\mathbf{x}) = \mathbf{n}^\top \left(\mathbf{p}_i^w - \mathbf{p}_j^w\right) - \epsilon$, where $\mathbf{p}_i^w := \mathbf{x}_i + \mathbf{p}_i^i = \mathbf{x}_i + \mathbf{R}_i\mathbf{p}_i$ is the contact point on object i in the world frame (indicated by the superscript w), $\mathbf{n}$ is the contact normal in world frame, and $\mathbf{R}_i \in \mathrm{SO}(3)$ is the rotation of the object frames relative to the world frame. As indicated by the superscript i, the contact point $\mathbf{p}_i^i$ is given in relative coordinates of body i. We find a velocity constraint through time differentiation of the pose constraint as (M. B. Cline 2002):

$$\dot{g}_c(\mathbf{x}) = \mathbf{J}_c \boldsymbol{\xi} \geq \mathbf{0}, \tag{5.7}$$

where $\mathbf{J}_c$ is the Jacobian of the contact pose constraint function, which follows from

$$\begin{aligned}
\dot{g}(\mathbf{x}) &= \mathbf{n}^\top (\dot{\mathbf{p}}_i^w - \dot{\mathbf{p}}_j^w) && (5.8)\\
&= \mathbf{n}^\top \left(\left(\mathbf{v}_i + \boldsymbol{\omega}_i \times \mathbf{p}_i^i \right) - \left(\mathbf{v}_j + \boldsymbol{\omega}_\mathrm{j} \times \mathbf{p}_j^j \right) \right) && (5.9)\\
&= \underbrace{\begin{pmatrix} (\mathbf{p}_i^i \times \mathbf{n})^\top & \mathbf{n}^\top \end{pmatrix}}_{\mathbf{J}_i} \underbrace{\begin{pmatrix} \boldsymbol{\omega}_i \\ \mathbf{v}_i \end{pmatrix}}_{\xi_i} + \underbrace{\begin{pmatrix} -\left(\mathbf{p}_j^j \times \mathbf{n}\right)^\top & -\mathbf{n}^\top \end{pmatrix}}_{\mathbf{J}_j} \underbrace{\begin{pmatrix} \boldsymbol{\omega}_j \\ \mathbf{v}_j \end{pmatrix}}_{\xi_j} && (5.10)\\
&= \underbrace{\begin{pmatrix} \mathbf{J}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_j \end{pmatrix}}_{\mathbf{J}_c} \underbrace{\begin{pmatrix} \boldsymbol{\xi}_i \\ \boldsymbol{\xi}_j \end{pmatrix}}_{\xi} = \mathbf{J}_c \boldsymbol{\xi}. && (5.11)
\end{aligned}$$

The constraint in Equation (5.7) only prevents penetration between objects. We will consider collisions with *restitution* in this work, *i.e.*, after collision the velocity along the contact normal should be inverted and scaled with the *restitution coefficient* k. Incorporating this condition into Equation (5.7) yields:

$$\mathbf{J}_c \boldsymbol{\xi}_{t+h} \geq -k \mathbf{J}_c \boldsymbol{\xi}_t =: -\mathbf{c}. \tag{5.12}$$

Similar as before, the Jacobian $\mathbf{J}_c$ forms a basis for the corresponding constraint forces, and we can reduce finding the constraint forces to finding their magnitudes in the Lagrange multiplier vector $\boldsymbol{\lambda}_c$, *i.e.*, $\mathbf{J}_c^\top \boldsymbol{\lambda}$ approximates the *contact force* at the solution. The collision constraint Equation (5.12) can be rewritten as

$$\mathbf{a} = \mathbf{J}_c \boldsymbol{\xi}_{t+h} + \mathbf{c} \geq \mathbf{0}, \tag{5.13}$$

interpreting $\mathbf{a}$ as the acceleration along the contact normal. The constraint in Equation (5.13) ensures the acceleration $\mathbf{a}$ causes the objects to move apart. Furthermore, the contact force magnitude $\boldsymbol{\lambda}_c$ must push the objects apart, *i.e.*, $\boldsymbol{\lambda}_c \geq \mathbf{0}$ and component-wise either $\mathbf{a}$ or $\boldsymbol{\lambda}_c$ must be zero (see (M. B. Cline 2002) for details), leading to $\mathbf{a}^\top \boldsymbol{\lambda}_c = \mathbf{0}$. We can now include these constraints in Equation (5.6), including $\mathbf{a}$ as slack variables and $\boldsymbol{\lambda}_c$ as Lagrange multipliers (similar to $\boldsymbol{\lambda}_e$), leading to complementarity constraints (Boyd and Vandenberghe 2004):

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{a} \end{pmatrix} + \begin{pmatrix} \mathbf{M} & -\mathbf{J}_e^\top & -\mathbf{J}_c^\top \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_c & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_{t+h} \\ \boldsymbol{\lambda}_e \\ \boldsymbol{\lambda}_c \end{pmatrix} = \begin{pmatrix} \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_\mathrm{ext} \\ \mathbf{0} \\ -\mathbf{c} \end{pmatrix}, \quad \text{subject to } \mathbf{a} \geq \mathbf{0}, \boldsymbol{\lambda}_c \geq \mathbf{0}, \mathbf{a}^\top \boldsymbol{\lambda}_c = \mathbf{0}. \tag{5.14}$$

Friction also leads to inequality constraints as in the Coulomb friction model, the magnitude of the frictional force must be smaller than μ times the magnitude of the normal force $\left\|\mathbf{f}_f\right\| \leq \mu \left\|\mathbf{f}_c\right\|$, where μ is the *coefficient of friction*. This constraint limits the *contact force* (*i.e.*, the combined collision and friction forces) to lie inside a spherical cone, which we approximate with a polyhedral cone using 8 directions $\mathbf{d}_1, \dots, \mathbf{d}_8 \in \mathbb{R}^3$ with unit norm and equal angular spacing on the tangential plane. In general, the number of directions can be chosen arbitrarily[5], and use 8 directions as a trade-off between efficiency and accuracy and determine the directions as

5: as long as for any direction $\mathbf{d}$, the opposite direction $-\mathbf{d}$ is also included (M. B. Cline 2002)

$$\begin{aligned}
\mathbf{d}_1 &= \frac{\mathbf{n}^\perp}{\|\mathbf{n}^\perp\|_2}, & \mathbf{d}_2 &= \frac{\mathbf{d}_1 \times \mathbf{n}}{\|\mathbf{d}_1 \times \mathbf{n}\|_2}, \\
\mathbf{d}_3 &= \frac{\mathbf{d}_1 + \mathbf{d}_2}{\|\mathbf{d}_1 + \mathbf{d}_2\|_2}, & \mathbf{d}_4 &= \frac{\mathbf{d}_3 \times \mathbf{n}}{\|\mathbf{d}_3 \times \mathbf{n}\|_2}, \\
\mathbf{d}_{4+i} &= -\mathbf{d}_i, i \in \{1, \dots, 4\},
\end{aligned} \tag{5.15}$$

where $\mathbf{p}^\perp = \mathbf{p} \times \mathbf{u}_{j^*}$ with $j^* = \arg\min_{j \in \{0,1,2\}} |p_j|$ and $\mathbf{u}_j \in \mathbb{R}^3$ is the j-th unit vector. This set of directions allows us to compute the friction Jacobian $\mathbf{J}_f$ similar to $\mathbf{J}_c$ (Equation (5.11)), now replacing the contact normal direction $\mathbf{n}$ with directions $\mathbf{d}_1, \dots, \mathbf{d}_8$ along the tangential contact surface.

The set of contact forces lying inside this polyhedral cone is then given by

$$\left\{\mathbf{J}_{c_i}^\top \lambda_{c_i} + \mathbf{J}_{f_i}^\top \boldsymbol{\lambda}_{f_i} \;\middle|\; \lambda_{c_i} \geq 0, \boldsymbol{\lambda}_{f_i} \geq \mathbf{0}, \mathbf{e}_i^\top \boldsymbol{\lambda}_{f_i} \leq \mu_i \lambda_{c_i}\right\}, \tag{5.16}$$

where i denotes the contact index, $\mathbf{e}_i = \mathbf{1} \in \mathbb{R}^8$, λ_{c_i} is the entry of $\boldsymbol{\lambda}_c$ corresponding to contact i, $\boldsymbol{\lambda}_{f_i} \in \mathbb{R}^8$ are the Lagrange multipliers for contact i in all friction directions and μ_i is the corresponding coefficient of friction. Intuitively, the conditions in Equation (5.16) mean that the force magnitudes λ_{c_i} and $\boldsymbol{\lambda}_{f_i}$ must be positive and the sum of friction force magnitudes $\mathbf{e}_i^\top \boldsymbol{\lambda}_{f_i}$ must be at most μ_i times the collision force magnitude λ_{c_i}.

This leads to two complementarity constraints (M. B. Cline 2002):

$$\begin{aligned}
\gamma_i \mathbf{e}_i + \mathbf{J}_{f_i} \boldsymbol{\xi}_{t+h} \geq 0 \quad &\text{complementary to} \quad \boldsymbol{\lambda}_{f_i} \geq \mathbf{0}, \\
\mu_i \lambda_{c_i} - \mathbf{e}_i^\top \boldsymbol{\lambda}_{f_i} \geq 0 \quad &\text{complementary to} \quad \gamma_i \geq 0,
\end{aligned} \tag{5.17}$$

which limit the friction impulse $\boldsymbol{\lambda}_{f_i}$ to $\mathbf{0}$ if $\lambda_{c_i} = 0$ (as otherwise $\mu_i \lambda_{c_i} - \mathbf{e}_i^\top \boldsymbol{\lambda}_{f_i} < 0$) and tie the two constraint together via γ_i if the normal impulse λ_{c_i} is positive. In the latter case, without tangential motion ($\mathbf{J}_{f_i} \boldsymbol{\xi}_{t+h} = \mathbf{0}$), we have $\gamma_i = 0$, allowing a cumulated friction force ($\mathbf{e}_i^\top \boldsymbol{\lambda}_{f_i}$) smaller than $\mu_i \lambda_{c_i}$, and with tangential motion γ_i will be positive, forcing the friction impulse to the maximal possible magnitude. Stacking the constraint force magnitudes $\boldsymbol{\lambda}_{f_i}$ in a vector $\boldsymbol{\lambda}_f \in \mathbb{R}^{8n}$, γ_i in Lagrange multipliers $\boldsymbol{\gamma} \in \mathbb{R}^n$, the coefficients of friction in a diagonal matrix $\boldsymbol{\mu} \in \mathbb{R}^{n \times n}$, and the vectors $\mathbf{e}_i$ in a block-diagonal matrix $\mathbf{E} \in \mathbb{R}^{8n \times n}$ for n contact points

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_n \end{pmatrix} \quad \text{and} \quad \mathbf{E} = \begin{pmatrix} \mathbf{e}_1 & & \\ & \ddots & \\ & & \mathbf{e}_n \end{pmatrix}, \tag{5.18}$$

lets us rewrite Equation (5.17) as $\mathbf{J}_f \boldsymbol{\xi}_{t+h} + \mathbf{E}\boldsymbol{\gamma} \geq 0$ and $\boldsymbol{\mu}\boldsymbol{\lambda}_c \geq \mathbf{E}^\top \boldsymbol{\lambda}_f$. Similar to contacts and joint constraints, the friction forces are given by $\mathbf{J}_f^\top \boldsymbol{\lambda}_f$. Further, adding slack variables $\boldsymbol{\sigma}, \boldsymbol{\zeta}$ corresponding to $\boldsymbol{\lambda}_f$ and $\boldsymbol{\gamma}$, we can add the constraints to Equation (5.14) to get the constrained dynamics model as the following linear complementarity problem (LCP):

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{a} \\ -\boldsymbol{\sigma} \\ -\boldsymbol{\zeta} \end{pmatrix} + \begin{pmatrix} \mathbf{M} & -\mathbf{J}_e^\top & -\mathbf{J}_c^\top & -\mathbf{J}_f^\top & \mathbf{0} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_c & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\mu} & -\mathbf{E}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_{t+h} \\ \boldsymbol{\lambda}_e \\ \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_{ext} \\ \mathbf{0} \\ -\mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{5.19}$$

$$\text{subject to} \begin{pmatrix} \mathbf{a} \\ \boldsymbol{\sigma} \\ \boldsymbol{\zeta} \end{pmatrix} \geq \mathbf{0}, \begin{pmatrix} \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \end{pmatrix} \geq \mathbf{0}, \begin{pmatrix} \mathbf{a} \\ \boldsymbol{\sigma} \\ \boldsymbol{\zeta} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \end{pmatrix} = 0.$$

The LCP is solved using a primal-dual algorithm as described in (Boyd and Vandenberghe 2004; Mattingley and Boyd 2012) and can be differentiated for the input states, forces and physical parameters at the solution (Avila Belbute-Peres et al. 2018) as we will explain in Subsection 5.3.2.

5.3.2 Differentiable Time-Stepping Dynamics

(Amos and Kolter 2017): *OptNet: Differentiable Optimization as a Layer in Neural Networks*

(Avila Belbute-Peres et al. 2018): *End-to-End Differentiable Physics for Learning and Control*

Amos and Kolter (2017) proposed an approach for computing derivatives of quadratic programs (QPs) at their solution, effectively enabling the computation of gradients through the optimization problem. Avila Belbute-Peres et al. (2018) applied the similar derivations to the LCP in Equation (5.19) for differentiating through the physics simulation.

In a first step, Avila Belbute-Peres et al. (2018) reorder Equation (5.19) to get a block-diagonal matrix structure by moving the second row and column to the last row/column:

$$\begin{pmatrix} \mathbf{0} \\ -\mathbf{a} \\ -\boldsymbol{\sigma} \\ -\boldsymbol{\zeta} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{M} & -\mathbf{J}_c^\top & -\mathbf{J}_f^\top & \mathbf{0} & -\mathbf{J}_e^\top \\ \mathbf{J}_c & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\mu} & -\mathbf{E}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_{t+h} \\ \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \\ \boldsymbol{\lambda}_e \end{pmatrix} = \begin{pmatrix} \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_{ext} \\ -\mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{5.20}$$

They then multiply the equation by -1 on both sides, applying the sign change to $\boldsymbol{\xi}_{t+h}$ instead of the first column of the matrix in the matrix-vector product and to the matrix columns for the rest:

$$\begin{pmatrix} \mathbf{0} \\ \mathbf{a} \\ \boldsymbol{\sigma} \\ \boldsymbol{\zeta} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{M} & \mathbf{J}_c^\top & \mathbf{J}_f^\top & \mathbf{0} & \mathbf{J}_e^\top \\ \mathbf{J}_c & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & -\mathbf{E} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\mu} & \mathbf{E}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} -\boldsymbol{\xi}_{t+h} \\ \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \\ \boldsymbol{\lambda}_e \end{pmatrix} = \begin{pmatrix} -\mathbf{M}\boldsymbol{\xi}_t - h\mathbf{f}_{ext} \\ \mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{5.21}$$

Finally, they identify the following blocks in Equation (5.21):

$$\begin{aligned} &\mathbf{x} := -\boldsymbol{\xi}_{t+h} && \mathbf{q} := \mathbf{M}\boldsymbol{\xi}_t + h\mathbf{f}_{\text{ext}} && \mathbf{s} := \begin{pmatrix} \mathbf{a} \\ \boldsymbol{\sigma} \\ \boldsymbol{\zeta} \end{pmatrix} && \\ &\mathbf{y} := \boldsymbol{\lambda}_e && \mathbf{A} := \mathbf{J}_e && && \mathbf{F} := \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{E} \\ -\boldsymbol{\mu} & \mathbf{E}^\top & \mathbf{0} \end{pmatrix}, \\ &\mathbf{z} := \begin{pmatrix} \boldsymbol{\lambda}_c \\ \boldsymbol{\lambda}_f \\ \boldsymbol{\gamma} \end{pmatrix} && \mathbf{G} := \begin{pmatrix} \mathbf{J}_c \\ \mathbf{J}_f \\ \mathbf{0} \end{pmatrix} && \mathbf{m} := \begin{pmatrix} \mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} && \end{aligned} \tag{5.22}$$

allowing to rewrite the equation as

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{s}\\ \mathbf{0}\end{pmatrix} + \begin{pmatrix}\mathbf{M} & \mathbf{G}^\top & \mathbf{A}^\top\\ \mathbf{G} & \mathbf{F} & \mathbf{0}\\ \mathbf{A} & \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{z}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}-\mathbf{q}\\ \mathbf{m}\\ \mathbf{0}\end{pmatrix}, \quad \text{subject to } \mathbf{s} \geq \mathbf{0}, \mathbf{z} \geq \mathbf{0}, \mathbf{s}^\top\mathbf{z} = 0. \tag{5.23}$$

From this formulation, they followed Amos and Kolter (2017) and applied matrix differential calculus (Magnus and Neudecker 1988) to compute the matrix differentials of the equations resulting from the system in Equation (5.23) at the solution $\left(\mathbf{x}^{*\top}, \mathbf{z}^{*\top}, \mathbf{y}^{*\top}\right)^\top$:

$$\begin{pmatrix}\mathbf{M} & \mathbf{G}^\top & \mathbf{A}^\top\\ \operatorname{diag}(\mathbf{z}^*)\,\mathbf{G} & \operatorname{diag}(\mathbf{G}\mathbf{x}^* + \mathbf{F}\mathbf{z}^* - \mathbf{m}) + \mathbf{F} & \mathbf{0}\\ \mathbf{A} & \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathrm{d}\mathbf{x}\\ \mathrm{d}\mathbf{z}\\ \mathrm{d}\mathbf{y}\end{pmatrix} = \begin{pmatrix}-\,\mathrm{d}\mathbf{M}\mathbf{x}^* - \mathrm{d}\mathbf{A}^\top\mathbf{y}^* - \mathrm{d}\mathbf{G}^\top\mathbf{z}^* - \mathrm{d}\mathbf{q}\\ -\operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{G}\mathbf{x}^* - \operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{F}\mathbf{z}^* + \operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{m}\\ -\,\mathrm{d}\mathbf{A}\mathbf{x}^*\end{pmatrix}. \tag{5.24}$$

Given the upstream gradient $\frac{\partial E}{\partial \mathbf{x}^*}$ relating some loss function E to the resulting velocity from solving Equation (5.23) $\mathbf{x}^* = -\xi^*_{t+h}$, the gradients for the input parameters can be computed by the chain rule. Avila Belbute-Peres et al. (2018) do this by first defining

$$\begin{pmatrix}\mathbf{d}_\mathbf{x}\\ \mathbf{d}_\mathbf{z}\\ \mathbf{d}_\mathbf{y}\end{pmatrix} = \begin{pmatrix}\mathbf{M} & \mathbf{G}^\top & \mathbf{A}^\top\\ \operatorname{diag}(\mathbf{z}^*)\,\mathbf{G} & \operatorname{diag}(\mathbf{G}\mathbf{x}^* + \mathbf{F}\mathbf{z}^* - \mathbf{m}) + \mathbf{F} & \mathbf{0}\\ \mathbf{A} & \mathbf{0} & \mathbf{0}\end{pmatrix}^{-\top}\begin{pmatrix}\left(\frac{\partial E}{\partial \mathbf{x}^*}\right)^\top\\ \mathbf{0}\\ \mathbf{0}\end{pmatrix} \tag{5.25}$$

and multiplying Equation (5.24) with its transpose yielding

$$\frac{\partial E}{\partial \mathbf{x}^*}\,\mathrm{d}\mathbf{x} = \begin{pmatrix}\mathbf{d}_\mathbf{x}\\ \mathbf{d}_\mathbf{z}\\ \mathbf{d}_\mathbf{y}\end{pmatrix}^\top\begin{pmatrix}-\,\mathrm{d}\mathbf{M}\mathbf{x}^* - \mathrm{d}\mathbf{A}^\top\mathbf{y}^* - \mathrm{d}\mathbf{G}^\top\mathbf{z}^* - \mathrm{d}\mathbf{q}\\ -\operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{G}\mathbf{x}^* - \operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{F}\mathbf{z}^* + \operatorname{diag}(\mathbf{z}^*)\,\mathrm{d}\mathbf{m}\\ -\,\mathrm{d}\mathbf{A}\mathbf{x}^*\end{pmatrix}, \tag{5.26}$$

from which the following derivatives for the components defined in Equation (5.22) can be derived:

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{q}} &= -\mathbf{d}_\mathbf{x} & \frac{\partial E}{\partial \mathbf{M}} &= -\frac{1}{2}\left(\mathbf{d}_\mathbf{x}\mathbf{x}^\top + \mathbf{x}\mathbf{d}_\mathbf{x}^\top\right)\\
\frac{\partial E}{\partial \mathbf{m}} &= \operatorname{diag}(\mathbf{z}^*)\,\mathbf{d}_\mathbf{z} & \frac{\partial E}{\partial \mathbf{G}} &= -\operatorname{diag}(\mathbf{z}^*)\,\mathbf{d}_\mathbf{z}\mathbf{x}^\top + \mathbf{z}\mathbf{d}_\mathbf{x}^\top\\
\frac{\partial E}{\partial \mathbf{A}} &= -\mathbf{d}_\mathbf{y}\mathbf{x}^\top - \mathbf{y}\mathbf{d}_\mathbf{x}^\top & \frac{\partial E}{\partial \mathbf{F}} &= -\operatorname{diag}(\mathbf{z}^*)\,\mathbf{d}_\mathbf{z}\mathbf{z}^\top.
\end{aligned} \tag{5.27}$$

After computing these derivatives, the derivatives on the original parameters (*e.g.*, initial velocity, friction, restitution) can be computed component-wise from the blocks in which they appear in Equation (5.22).

Dependency of the LCP Solution Derivative on Contact Points. The derivatives of the physics simulation in Equation (5.27) are found at the solution of the LCP in Equation (5.19) (Avila Belbute-Peres et al. 2018). The contact points appear in the contact and friction Jacobians in a cross product with the contact normal which is then multiplied with the rotational velocity in a dot product (see Equation (5.11)). Direct dependency on linear velocity is not incorporated in the velocity constraint as the linear velocity is only multiplied with the direction of the contact normal in Equation (5.11). Moreover, the LCP does not solve for the time step h through the collision constraint, but assumes it constant. Thus, no implicit dependency of the time step on other states and parameters is modeled. We include the dependencies of the contact points on linear velocities and of the time step size on the shape with our time of contact differential based on the contact position constraint.

5.3.3 Differentiable SDF Shape Representation

We represent shapes by signed distance functions (SDFs) (see Section 2.2). More specifically, we use parametric shape models, which are differentiable for the shape parameters (see Subsection 2.2.1). To represent complex shapes, we train neural network representations (Gropp et al. 2020; Park et al. 2019). We further use the SDF directly for contact detection (see Subsection 5.3.4). Thus, it is important that the distance is accurate even for off-surface points. To achieve this for the learned parametric models, we use the loss proposed by Gropp et al. (2020) (see Subsection 2.2.1 for details). For contact detection and visualization, we need to extract explicit meshes from the SDF (see Subsection 2.2.4; Lorensen and H. E. Cline 1987) and we make them differentiable with respect to the shape parameters by following the approach proposed by Remelli et al. (2020) (see Subsection 2.2.1 for details).

5.3.4 Contact Detection Between SDFs

We detect contacts and estimate contact points and normals from the SDFs of both objects based on the approach of (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020). The approach determines contact points between a triangular mesh and an SDF by finding the point with the lowest SDF value for each triangle[6]. We first transform the object SDF representations into their differentiable mesh representations (see Subsection 2.2.4). The culling and starting point strategies of (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020) determine an initial set of mesh triangles to consider for contact detection. This involves determining for each triangle with vertices $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, if the signed distance $\phi_j(\mathbf{c})$ at the triangle's centroid position $\mathbf{c}$ in the other object is below the radius $\max_{\mathbf{v}\in\mathcal{V}} \|\mathbf{v}-\mathbf{c}\|_2$ of the triangle:

6: *i. e.*, the points with lowest distance between the objects

$$\phi_j(\mathbf{c}) < \max_{\mathbf{v}\in\mathcal{V}} \|\mathbf{v}-\mathbf{c}\|_2 \,, \text{with } \mathbf{c} = \frac{1}{3}\sum_{\mathbf{v}\in\mathcal{V}} \mathbf{v}. \tag{5.28}$$

The contact point for such a triangle is found iteratively from the triangle vertex position $\mathbf{p}_0$ with the smallest SDF value in the other object using the Frank-Wolfe algorithm (Frank and Wolfe 1956) based approach in (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020).

In each iteration k, the algorithm determines a point $\mathbf{s}_k$ that minimizes

$$\mathbf{s}_k = \arg\min_{\mathbf{s}\in\mathcal{T}} \mathbf{s}^\top \nabla\phi(\mathbf{p}_k), \tag{5.29}$$

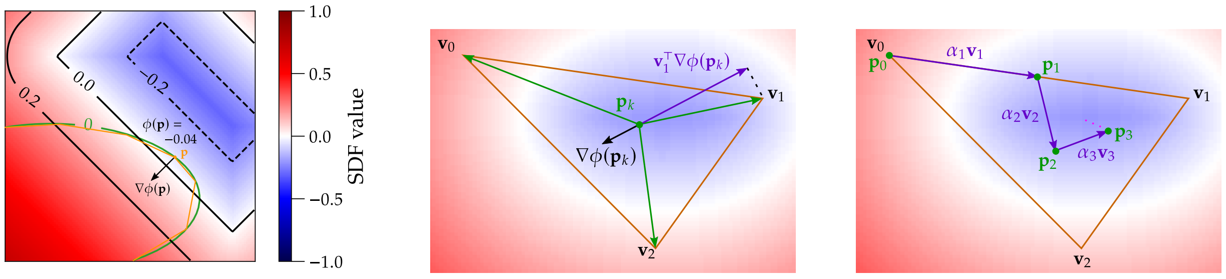


Figure 5.2: Collision detection with SDFs. Left: we extract a differentiable mesh (orange) from an SDF (ellipse, green) and find contact points $\mathbf{p}$ (points of maximum penetration) and normals $\nabla\phi(\mathbf{p})$ (direction towards closest point on penetrated surface) on the mesh faces towards a second SDF ϕ (box, white). The differentiable mesh allows for propagating gradients from the contact points and normals onto the SDF shape parametrization. Middle: We localize contact points using a Frank-Wolfe algorithm which iteratively select the vertex $\mathbf{v}_i$, which projected on the signed distance gradient $\mathbf{v}_i^\top \nabla\phi(\mathbf{p}_k)$ provides the best improvement. Right: The solution of the Frank-Wolfe algorithm can be written as a linear combination of the vertex positions. The found contact point is differentiable for the underlying SDF through the vertex positions.

where $\mathcal{T} \subset \mathbb{R}^3$ is the surface spanned by the triangle vertices. The minimum in Equation (5.29) provides the point in the triangle for which the projection onto the SDF gradient yields the best improvement. As $\mathcal{T}$ is convex, the minimum of Equation (5.29) is achieved for one of the triangle vertices $\mathbf{v} \in \mathcal{V}$ (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020):

$$\mathbf{s}_k = \arg\min_{\mathbf{v}\in\mathcal{V}} \mathbf{v}^\top \nabla\phi(\mathbf{p}_k) \tag{5.30}$$

The contact point is updated according to

$$\mathbf{p}_{k+1} = (1-\alpha_k)\mathbf{p}_k + \alpha_k \mathbf{s}_k, \tag{5.31}$$

where $\alpha_k = \frac{2}{k+2}$. We iterate the algorithm until the criterion

$$\left|(\mathbf{p}_k - \mathbf{s}_k)^\top \nabla\phi(\mathbf{p}_k)\right| < \tau \tag{5.32}$$

converges below a threshold τ or a maximum iteration count is reached. This process is illustrated in Figure 5.2.

5.4 Method

We now detail our simulation framework and novel contributions to include a differentiable SDF shape representation and differentiable time of contact into a velocity- and constraint-based time-stepping 3D simulation method. Our overall goal is to develop a differentiable model $\mathbf{x}_t = g(\mathbf{x}_{t-1}, \boldsymbol{\theta}, \mathbf{f}_{\text{ext}}, h)$ for physics simulation which can be used for predicting the next scene state $\mathbf{x}_t$ (object poses) based on physics parameters $\boldsymbol{\theta}$ (including shape parameters $\mathbf{z}$), initial states (object poses and velocities) $\mathbf{x}_{t-1}$, external wrench $\mathbf{f}_{\text{ext}}$, and a simulation time step h. In our approach, the function g is given by the solution of the LCP in Equation (5.19). The property of differentiability of g for its inputs allows us to formulate optimization problems to fit simulated to target trajectories by adjusting parameters $\boldsymbol{\theta}$. For instance,

$$E(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=0}^{N-1} \|g(\mathbf{x}_i, \boldsymbol{\theta}, \mathbf{f}_{\text{ext},i}, h_i) - \check{\mathbf{x}}_{i+1}\|_2^2 \tag{5.33}$$

fits a target trajectory $\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_N$ of duration T with appropriate time discretization $t_i = t_0 + \sum_{j=0}^{i-1} h_j$ where $h_i \geq 0$ and $\sum_{i=0}^{N-1} h_i = T$.

5.4.1 Differentiable Inertia Tensors

The mass-inertia matrix $\mathbf{M}$ stacks angular inertia tensor $\mathbf{I}$ and object mass m for the angular and linear parts of the Newton-Euler equations. We determine the angular inertia tensor of the object from the shape represented implicitly in an SDF. The tensor $\mathbf{I}$ is found efficiently from a triangular surface mesh using (Mirtich 1996) which we implement using differentiable operations. We extract the mesh from the SDF using the marching cubes algorithm (Lorensen and H. E. Cline 1987). Using (Remelli et al. 2020) (see Subsection 2.2.4), we differentiate the inertia tensor for the underlying SDF and subsequently the shape encoding $\mathbf{z}$ through the mesh,

$$\frac{\mathrm{d}\mathbf{I}}{\mathrm{d}\mathbf{z}} = \sum_{\mathbf{v} \in \mathcal{V}} \frac{\partial \mathbf{I}}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \phi} \frac{\partial \phi}{\partial \mathbf{z}}, \tag{5.34}$$

with mesh vertices $\mathcal{V}$.

5.4.2 Differentiable Contact Modeling

Our differentiable physics simulation supports friction and collision contact constraints. Both constraints are formulated in terms of contacts with associated contact points $\mathbf{p}_i, \mathbf{p}_j$ relative to the reference frames of both objects i and j, contact normal $\mathbf{n}_{i/j}$ and penetration distance $d_{i/j}$ (see Subsection 5.3.1).

(Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020): *Local Optimization for Robust Signed Distance Field Collision*

(Frank and Wolfe 1956): *An algorithm for quadratic programming*

Differentiable Contact Points and Normals. We extend the approach in (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020) (see Subsection 5.3.4) to develop efficient differentiable contact detection for SDF shapes. The contact detection algorithm provides the contact point $\mathbf{p}^*$ on a mesh triangle through Frank-Wolfe iterations (Frank and Wolfe 1956). The solution $\mathbf{p}^*$ of the algorithm achieved by iterating Equation (5.31) is a linear combination

$$\mathbf{p}^* = \sum_{l=1}^{3} w_l \mathbf{v}_l \tag{5.35}$$

of the triangle vertices, where

$$w_{l,k+1} = (1 - \alpha_k) w_{l,k} + \begin{cases} \alpha_k, & \text{if } \mathbf{s}_k = \mathbf{v}_l \\ 0, & \text{otherwise,} \end{cases} \tag{5.36}$$

since $\mathbf{s}_k \in \mathcal{V}$ for all k as illustrated in Figure 5.2.

We directly differentiate the resulting linear combination of the Frank-Wolfe algorithm, and find the derivative of this linear combination for the underlying SDF using the mesh to SDF differential in Equation (2.55). This also enables us to propagate gradients through the contact points onto the shape encoding $\mathbf{z}$ of SDF shape spaces to optimize the object shapes for physical plausibility in contact situations.

Finally, we find the contact normal and penetration distance directly from the object SDFs. The normal is given by the SDF gradient at the contact point

$$\mathbf{n}_i = \pm \frac{\nabla\phi_{j/i}(\mathbf{p}_i)}{\left\|\nabla\phi_{j/i}(\mathbf{p}_i)\right\|_2} \tag{5.37}$$

in either the other or the own object, while the penetration is the signed distance value $d_i = \phi_j(\mathbf{p}_i)$. We choose the contact normal from the object with the smallest mean curvature of the SDF at the contact point $\mathbf{p}_i$. The corresponding contact point on the other object j is detected at $\mathbf{p}_j = \mathbf{p}_i - \phi_j(\mathbf{p}_i)\nabla\phi_j(\mathbf{p}_i)$.

Reducing the Number of Contact Points. Since the runtime complexity for solving the LCP increases with the number of contact points, we determine redundant contact points by clustering the points according to normal similarity and reducing the set to the points on its convex hull. This does not change the contact and friction constraints as shown by the following theorem:

Theorem 5.4.1 *Let C be a set of contact points between two bodies i and j sharing a common surface normal. Let further* $\mathscr{C} = \{(\mathbf{p}_i^1, \mathbf{p}_j^1), \ldots, (\mathbf{p}_i^n, \mathbf{p}_j^n)\}$ *be the convex hull of C and let the contact constraint*

$$\mathbf{J}_c\xi_{t+h} \geq -k\mathbf{J}_c\xi_t = -\mathbf{c} \tag{5.38}$$

be satisfied for all contact points in $\mathscr{C}$. *Then the contact constraint Equation* (5.38) *is also satisfied for all points of C.*

Proof. We start by expressing the contact Jacobians in their full form:

$$\mathbf{J}_c = \begin{pmatrix} \mathbf{J}_i & 0 \\ 0 & \mathbf{J}_j \end{pmatrix} \tag{5.39}$$

where, as in Equation (5.10),

$$\mathbf{J}_i = \left((\mathbf{p}_i \times \mathbf{n})^\top \quad \mathbf{n}^\top\right), \quad \mathbf{J}_j = -\left((\mathbf{p}_j \times \mathbf{n})^\top \quad \mathbf{n}^\top\right). \tag{5.40}$$

By definition of the convex hull, we can express any point that lies inside the convex hull as a linear combination of the hull points:

$$\mathbf{p}_i = \sum_{k=1}^{n} a_k \mathbf{p}_i^k, \text{ where } \sum_{k=1}^{n} a_k = 1; \forall k : a_k \geq 0. \tag{5.41}$$

Now, we apply these properties to show that the contact constraint is automatically satisfied for all contact points in C if it is satisfied for all points in $\mathscr{C}$:

$$\mathbf{J}_c \boldsymbol{\xi}_{t+h} \tag{5.42}$$

$$= \left((\mathbf{p}_i \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t+h} - \left((\mathbf{p}_j \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t+h} \tag{5.43}$$

$$= \left((\textstyle\sum_{l=1}^n a_l \mathbf{p}_i^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t+h} - \left((\textstyle\sum_{l=1}^n a_l \mathbf{p}_j^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t+h} \tag{5.44}$$

$$= \sum_{l=1}^n a_l \left(\left((\mathbf{p}_i^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t+h} - \left((\mathbf{p}_j^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t+h}\right) \tag{5.45}$$

$$\geq \sum_{l=1}^n a_l \left(-k \left(\left((\mathbf{p}_i^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t} - \left((\mathbf{p}_j^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t}\right)\right) \tag{5.46}$$

$$= -k \left(\left((\textstyle\sum_{l=1}^n a_l \mathbf{p}_i^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t} - \left((\textstyle\sum_{l=1}^n a_l \mathbf{p}_j^l \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t}\right) \tag{5.47}$$

$$= -k \left(\left((\mathbf{p}_i \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_i^{t} - \left((\mathbf{p}_j \times \mathbf{n})^\top \quad \mathbf{n}^\top\right) \xi_j^{t}\right) \tag{5.48}$$

$$= -\mathbf{c} \tag{5.49}$$

In Equations (5.44) and (5.48), we applied the definition of the convex hull (Equation (5.41)). Equations (5.45) and (5.47) follow from the distributive law and the fact that

$$\mathbf{n} = \sum_{l=1}^n a_l \mathbf{n} \text{ if } \sum_{l=1}^n a_l = 1. \tag{5.50}$$

In Equation (5.46), we applied the constraint inequality for the individual hull points. We have thus shown that satisfying the contact constraint Equation (5.38) for all points in $\mathscr{C}$ also satisfies it for all points in C. □

The proof for the friction Jacobians follows analogous to the proof of Theorem 5.4.1 by replacing the contact normals with the friction directions.

5.4.3 Differentiable Time of Contact

The LCP contains the contact points in the contact and friction constraint Jacobians by which the LCP solution can also be differentiated for the contact points and hence for the shape parameters of the objects. Due to the time discretization and time derivatives of the position constraint, there is no direct dependency of the contact points on the linear velocity of the bodies as explained in Subsection 5.3.2. We observe, that the shape and the induced collisions influence the time of contact: the larger the shape in the direction of motion, the earlier the contact (see Figure 5.3). Hence, changes in object shape induce changes in time of contact. We propose an approach to model this dependency of the simulation on the shape parameters.

The LCP from Subsection 5.3.1 assumes the time step size h constant and does not model its dependency on other parameters of the simulation. The time step is found through a separate optimization process which determines the largest step size $h \leq H$ until the next collision with maximum step size H. The dependency of h on the shape parameters is implicitly defined by the solution of the optimization problem. It

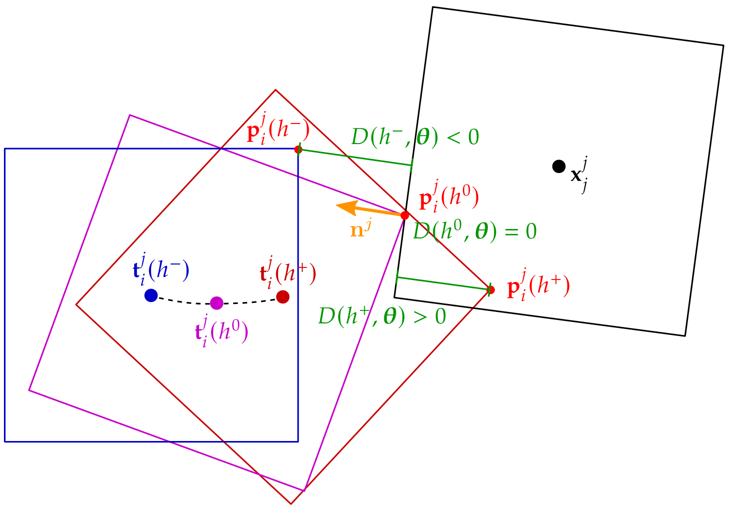


Figure 5.3: Differentiable time of contact. We differentiate the time of contact h for the contact point to enable shape optimization through collisions. The constraint $D(h,\boldsymbol{\theta})=0$ requires the distance of the contact point $\mathbf{p}_i^j$ of object i to its corresponding contact point of object j along the contact normal direction $\mathbf{n}^j$ to be zero. Shape, physical states and parameters which D depends on are summarized in $\boldsymbol{\theta}$. The constraint defines an implicit relationship between h and $\boldsymbol{\theta}$. The distance is defined with body frame of object j as reference frame (denoted by superscript j on points and vectors). In this frame, object i and its contact point move along trajectories $\mathbf{t}_i^j(h)$ and $\mathbf{p}_i^j(h)$, respectively. We use implicit differentiation to determine $\partial h/\partial\boldsymbol{\theta}$.

determines the time step for which corresponding points of contact $\mathbf{p}_i, \mathbf{p}_j$ on both objects i, j coincide, *i.e.*,

$$D\left(h,\mathbf{p}_i^j(h,\boldsymbol{\theta}(h)),\mathbf{p}_j^j,\boldsymbol{\theta}(h)\right)=\mathbf{n}^{j\top}\left(\mathbf{p}_j^j-\mathbf{p}_i^j(h,\boldsymbol{\theta}(h))\right)=0. \qquad (5.51)$$

The distance D also depends on the contact normal $\mathbf{n}$ and variables $\boldsymbol{\theta}(h)$, such as the object shape parameters, poses, and velocities at the time of collision. We express the distance in the body frame of object j (indicated by the superscript j) so that the contact point on object j and the contact normal remain constant, while the contact point on object i depends on time h via the pose $\mathbf{T}$[7], velocity $\mathbf{v}$ and acceleration $\mathbf{a}$ of the objects,

7: consisting of rotation $\mathbf{R}$ and translation $\mathbf{t}$, see Subsection 2.1.2

$$\mathbf{p}_i^j(h,\boldsymbol{\theta}(h))=(\exp(\widehat{\boldsymbol{\omega}}_j h)\mathbf{R}_j)^\top\Big(\exp(\widehat{\boldsymbol{\omega}}_i h)\mathbf{p}_i^I+\mathbf{t}_i + \mathbf{v}_i h+\tfrac{1}{2}\mathbf{a}_i h^2-\big(\mathbf{t}_j+\mathbf{v}_j h+\tfrac{1}{2}\mathbf{a}_j h^2\big)\Big) \qquad (5.52)$$

and $\mathbf{p}_j^j=\exp(\widehat{\boldsymbol{\omega}}_j h)\mathbf{R}_j^\top\mathbf{p}_j^J$, where $\mathbf{R}_j$ and $\mathbf{t}_j$ are the rotation and translation of object in the world frame, respectively. The operator $\widehat{\cdot}$ maps twist coordinates $\boldsymbol{\omega}$ to $\mathfrak{so}(3)$ Lie algebra elements and exp is the exponential map of SO(3)[8]. By the constraint in Equation (5.51), the time step $h(\mathbf{p}_i,\mathbf{p}_j,\boldsymbol{\theta})$ implicitly becomes a function of the contact points, the contact normal, and the simulation state and parameters (see Figure 5.3).

8: see Subsection 2.1.1

When optimizing system identification objectives such as Equation (5.33) using gradient-based methods, we require the derivative of the time of contact h for its parameters. For instance, differentiating the objective in Equation (5.33) yields a Jacobian depending on three derivative terms

$$\frac{\mathrm{d}E(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}=\sum_{i=0}^{N-1}\left(g\left(\mathbf{x}_i,\boldsymbol{\theta},\mathbf{f}_{\mathrm{ext},i},h_i(\boldsymbol{\theta})\right)-\widehat{\mathbf{x}}_{i+1}\right) \left(\frac{\partial g}{\partial\boldsymbol{\theta}}+\frac{\partial g}{\partial\mathbf{x}_i}\frac{\partial\mathbf{x}_i}{\partial\boldsymbol{\theta}}+\frac{\partial g}{\partial h_i}\frac{\partial h_i}{\partial\boldsymbol{\theta}}\right). \qquad (5.53)$$

The first term $\frac{\partial g}{\partial\boldsymbol{\theta}}$ is obtained by differentiating the LCP at its solution for the physics parameters. The second term $\frac{\partial g}{\partial\mathbf{x}_i}\frac{\partial\mathbf{x}_i}{\partial\boldsymbol{\theta}}$ is the derivative of the LCP for the input state $\frac{\partial g}{\partial\mathbf{x}_i}$ multiplied with the derivative of g for the parameters from the previous time step as $\mathbf{x}_i = g(\mathbf{x}_{i-1}, \boldsymbol{\theta}, \mathbf{f}_{\text{ext},i-1}, h_{i-1}(\boldsymbol{\theta}))$. For the third term, we require the time of contact derivative $\frac{\partial h_i}{\partial \boldsymbol{\theta}}$.

In practice, in each simulation step, the step size h is either chosen as a maximum time step H or the time until the first contact occurs. To this end, the simulation step size h is iteratively halved until all penetrations between objects are below a contact threshold distance ϵ. This yields the step size $h = h^0 - h^-$ until the approximate time of contact h^0 in Figure 5.3 and a set of contact points $\mathbf{p}_i^k, \mathbf{p}_j^k$ with contact normals $\mathbf{n}^k$ determined as the surface normal on one of the objects with $k \in \{1, \dots, K\}$. The constraints form an over-determined set of equations,

$$\mathbf{D}(h^0, \boldsymbol{\theta}) = \begin{pmatrix} D_1(h^0, \boldsymbol{\theta}) \\ \vdots \\ D_K(h^0, \boldsymbol{\theta}) \end{pmatrix} = \mathbf{0}, \tag{5.54}$$

where $\boldsymbol{\theta}$ subsumes the parameters h^0 depends on and $D_k(h^0, \boldsymbol{\theta}) = 0$ is the time of contact constraint for the k-th contact point. From this set of equations, we find the time of contact derivative through implicit differentiation,

$$\frac{\mathrm{d}\mathbf{D}(h, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathbf{D}(h, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{D}(h, \boldsymbol{\theta})}{\partial h} \frac{\partial h}{\partial \boldsymbol{\theta}} = \mathbf{0}, \tag{5.55}$$

which gives

$$\frac{\partial h}{\partial \boldsymbol{\theta}} = -\frac{\partial \mathbf{D}(h, \boldsymbol{\theta})}{\partial h}^{+} \frac{\partial \mathbf{D}(h, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{5.56}$$

By applying the Moore-Penrose pseudo-inverse $\frac{\partial \mathbf{D}(h,\boldsymbol{\theta})}{\partial h}^{+}$ we find the least squares fit to the over-determined set of equations. The time of contact derivative effectively allows for optimizing the shape of objects through the contact points so that collisions occur earlier or later and the trajectories of the objects are adapted. Components k only contribute meaningfully if the bodies move towards each other according to the relative velocity, *i.e.*, if $\frac{\partial D_k(h,\boldsymbol{\theta})}{\partial h} \geq 0$. Otherwise, the component is excluded from $\mathbf{D}$.

The actually simulated step size h until the time of contact h^0 might be smaller than the target simulation step size $H = h^+ - h^-$ (see Figure 5.3). Thus, both the time step before and after the collision depend on the time of contact, and we compute gradients for both $h(\boldsymbol{\theta}) = h^0 - h^-$ and $h'(\boldsymbol{\theta}) = h^+ - h^0$.

We distinguish contacts with impact and resting contacts and determine the time of contact differential only for contacts with impact. Contacts with impact are those which are newly found between two objects in a time step (*i.e.*, the previous time step did not have contacts between these objects). Resting contacts are contacts in successive time steps after the first contact time step between two objects.

5.5 Experiments

We evaluate and demonstrate our method in several physical system identification scenarios which involve inference on shape, friction coef-

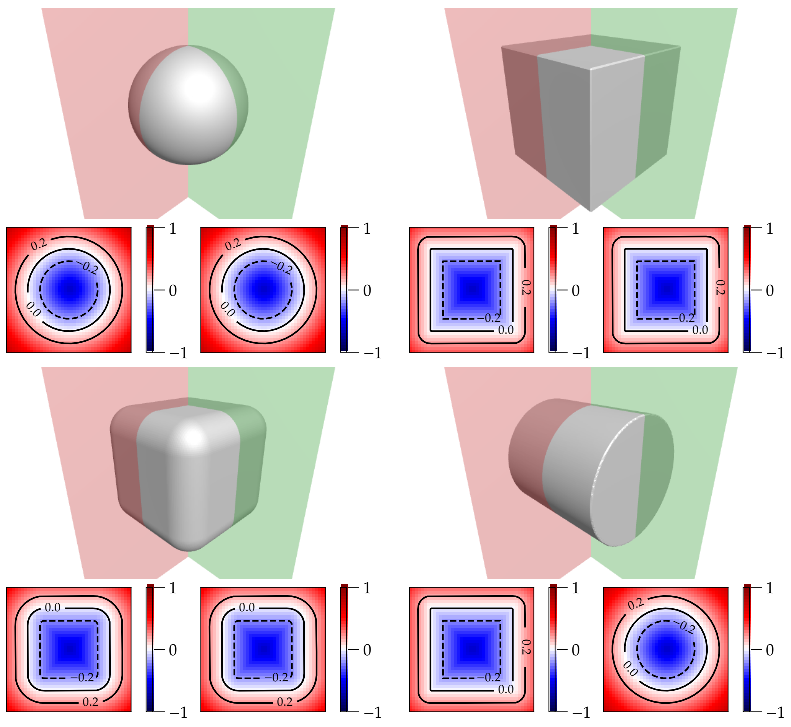


Figure 5.4: Example shapes for the sphere, box, rounded box and cylinder shape spaces from left to right. Mesh renderings are shown on the top and the bottom rows illustrate cuts through the respective SDFs along the yz- (left; green in the rendering) and xy-planes (right; red in the rendering). Note that for all rendered but the cylinder the two cuts are identical. Note further, that the rounded box SDF is basically a box SDF for a smaller box with the 0-level set shifted further outside.

ficient, mass or forces. The observations are generated using the LCP physics engine with ground truth parameters with a time step size of $H = \frac{1}{30}$. We evaluate the accuracy of our novel approach quantitatively and provide an ablation study.

5.5.1 Shape Spaces

Primitive Shapes. For shape primitives, *i.e.*, spheres, boxes and cylinders, we compute the SDF analytically. In Figure 5.4, we show example objects as rendered meshes as well as cuts through the SDFs for the primitive shapes used in the experiments.

Learned Shape Spaces. In Figure 5.5 we show the generated meshes for the two training shapes “bob” and “spot” together with their two-dimensional encodings in their learned shape space. We also show the “mean shape” for this shape space, *i.e.*, the result of interpolating half-way between bob and spot. This shape space has latent size 2 and the DeepSDF (Park et al. 2019) auto-decoder has 8 layers with a hidden dimension of 128.

Figures 5.6 to 5.8 show the generated meshes for the training shapes of the can, camera and mug classes, respectively. These shape spaces are trained with a latent size of 4 and the DeepSDF (Park et al. 2019) auto-decoder has 8 layers with a hidden dimension of 256.

All learned shape spaces were trained using the implicit geometric regularization losses from (Gropp et al. 2020) as explained in Subsection 2.2.1, to encourage the network to learn actual distances to the surface.

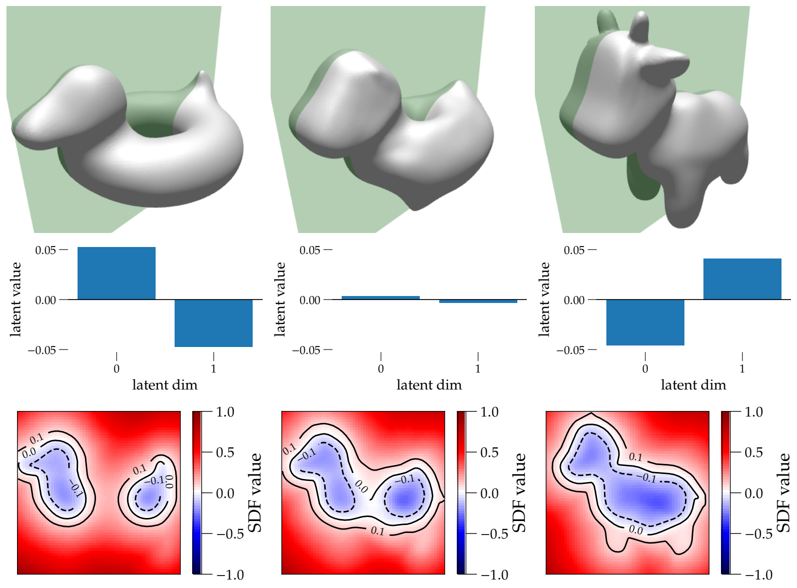


Figure 5.5: SDF shape representation. Shapes (top) represented in an examplary two-dimenional SDF shape embedding with corresponding latent codes (middle) and cuts through the SDF along the xy-plane (bottom; the plane is indicated in green in the rendering on the top).

5.5.2 Evaluation Metrics

Shape Identification (Subsection 5.5.3). The shape accuracy in the bouncing objects and shape from inertia experiments is evaluated by the symmetric Chamfer distance between the mesh estimate $\mathcal{M}_e = \{\mathcal{V}_e, \mathcal{F}_e\}$ and the target mesh $\mathcal{M}_t = \{\mathcal{V}_t, \mathcal{F}_t\}$. The meshes consist of sets of vertices $\mathcal{V}$ and faces $\mathcal{F}$ and are extracted from the SDF using the marching cubes algorithm (Lorensen and H. E. Cline 1987, see Subsection 2.2.4). The symmetric Chamfer distance is defined as

$$\mathrm{CD}(\mathcal{V}_e, \mathcal{V}_t) = \frac{1}{|\mathcal{V}_e|} \sum_{\mathbf{v}_e \in \mathcal{V}_e} \min_{\mathbf{v}_t \in \mathcal{V}_t} \|\mathbf{v}_e - \mathbf{v}_t\|^2 + \frac{1}{|\mathcal{V}_t|} \sum_{\mathbf{v}_t \in \mathcal{V}_t} \min_{\mathbf{v}_e \in \mathcal{V}_e} \|\mathbf{v}_t - \mathbf{v}_e\|^2 , \tag{5.57}$$

where $\|\cdot\|$ denotes the Euclidean norm and $|\mathcal{V}|$ the number of elements in $\mathcal{V}$.

Friction and Mass Identification, Force Optimization (Subsection 5.5.4). Friction coefficient and mass are evaluated by the absolute difference between the estimated and target values. The force vector is evaluated by the Euclidean norm between the estimated and target force vectors.

Fitting to Depth Image Observations (Subsection 5.5.5). The position error is the Euclidean norm between the locations of the center of the estimated and target objects. The rotation error is measured as the relative angle between the estimated and target rotation of the object. The size error is the absolute difference between the estimated and target objects radius for the sphere and edge length for the cube.

5.5.3 Shape Identification

The following experiments analyze the accuracy of our method for estimating shape from collisions and inertia in several scenarios such as

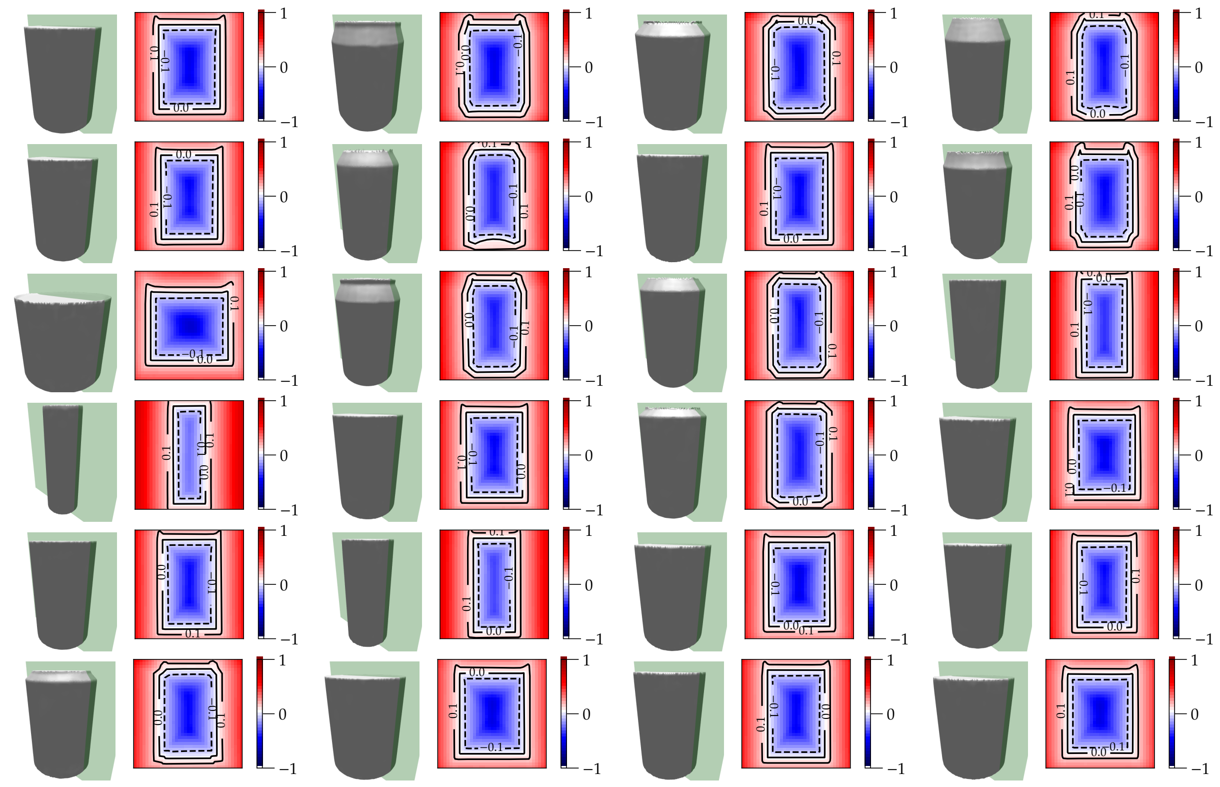


Figure 5.6: Renderings (left) and cuts through the xy-plane (right; the plane is indicated in green in the rendering) for the 24 can objects in their learned shapespace. Each object is represented by a four-dimensional latent code.

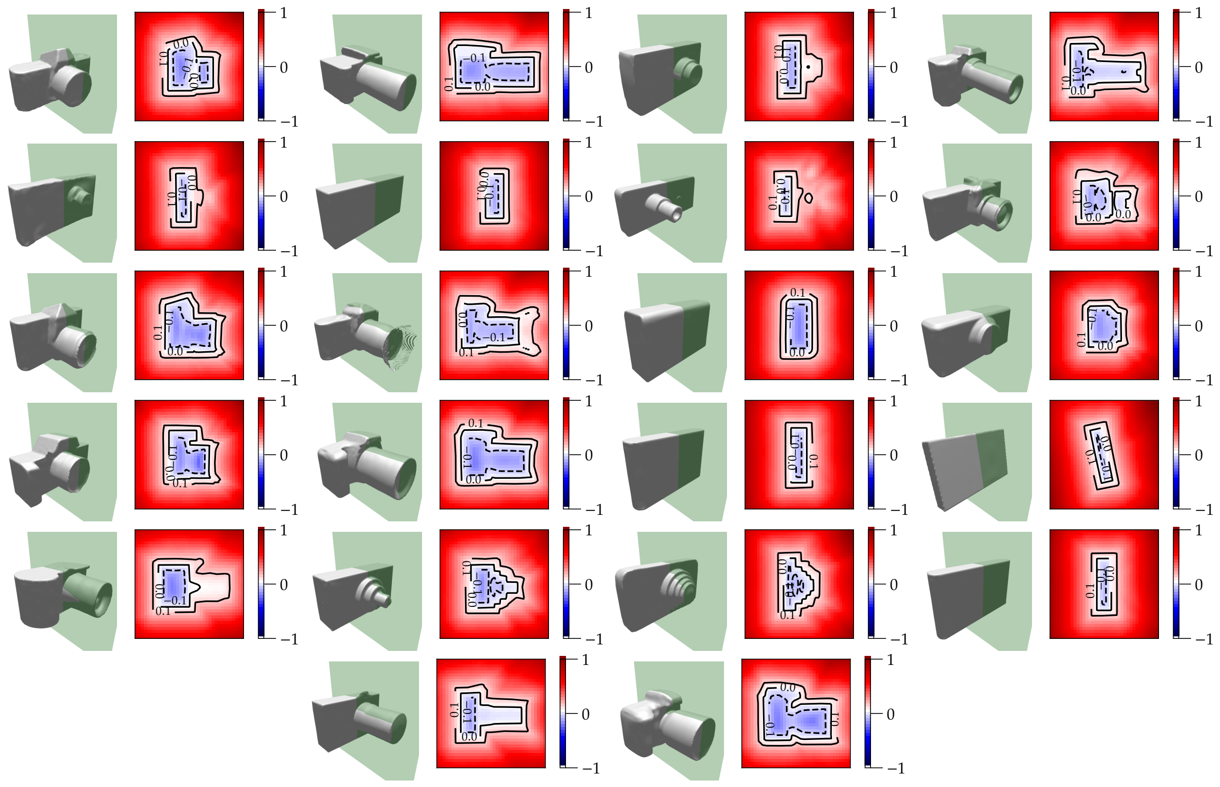


Figure 5.7: Renderings (left) and cuts through the xy-plane (right; the plane is indicated in green in the rendering) for the 22 camera objects in their learned shapespace. Each object is represented by a four-dimensional latent code.

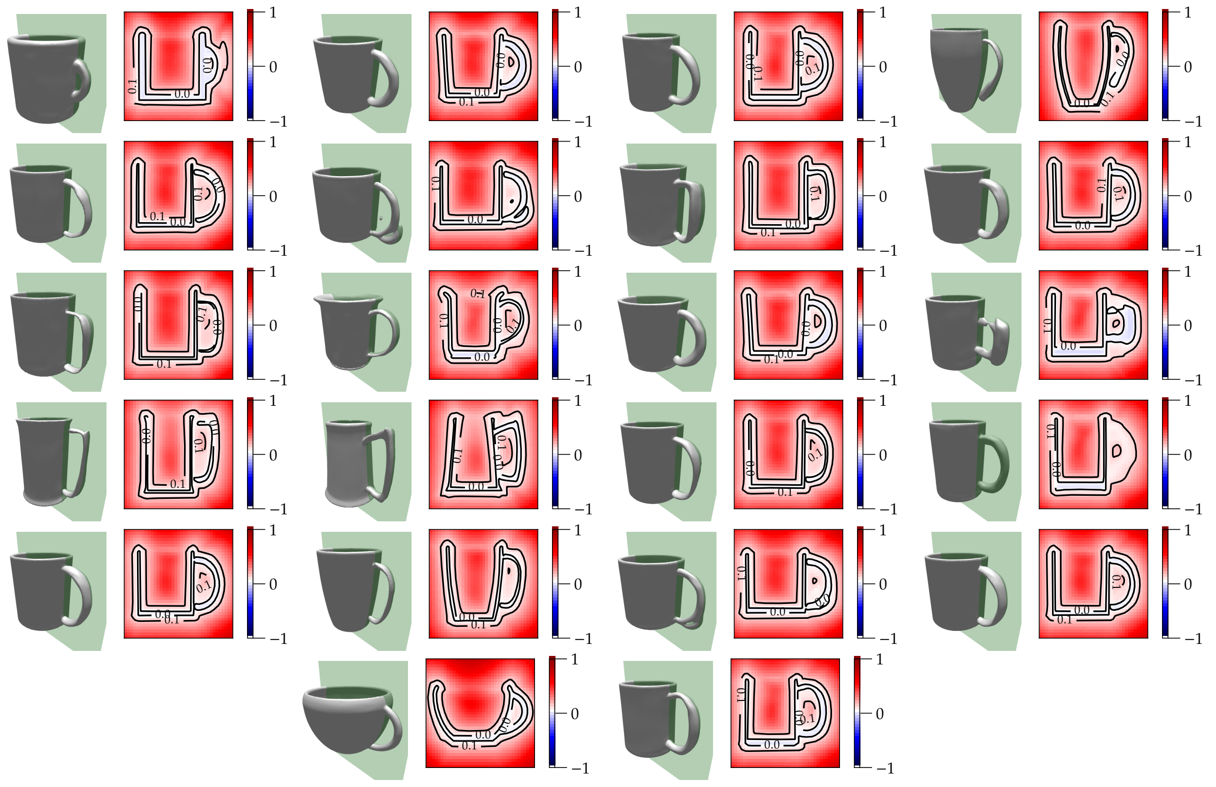


Figure 5.8: Renderings (left) and cuts through the xy-plane (right; the plane is indicated in green in the rendering) for the 22 mug objects in their learned shapespace. Each object is represented by a four-dimensional latent code.

bouncing shapes, topological shape changes, and shape from inertia for rotating objects.

Bouncing Objects. We evaluate our approach in several scenarios in which we let objects collide with a wall and possibly the floor. The objects start with a known initial horizontal velocity towards the wall and the collision yields different trajectories depending on the shape parameters $\boldsymbol{\theta}$. The optimization objective in these experiments is the mean-squared position error over the entire trajectory in all time steps $t \in \{1, \ldots, T\}$ plus an optional regularization term on the shape parameters:

$$E(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} \left\| \mathbf{t}(\boldsymbol{\theta})_e^t - \mathbf{t}(\boldsymbol{\theta})_g^t \right\|^2 + \lambda \left\| \boldsymbol{\theta} \right\|^2 , \qquad (5.58)$$

where $\mathbf{t}(\boldsymbol{\theta})_e^t$ and $\mathbf{t}(\boldsymbol{\theta})_g^t$ denote the center location of the estimated and goal spheres at time t, respectively and the parameter λ scales the contribution of the regularization. In all bouncing object experiments, the friction and restitution coefficients are set to 0.25 and 0.5, respectively. For numerical evaluation, we run take 50 runs with random initialization and targets in each setting (with/without gravity and with/without time of contact differential).

We evaluate our time of contact differential for estimating the radius of sphere that bounces against a wall and the floor (see Figures 5.9 and 5.10). We generate 50 scenes with randomly sampled sphere radii between 0.4 and 2.0 with an initial velocity of 5 towards the wall in two scenarios with gravity enabled and disabled. We optimize the radius

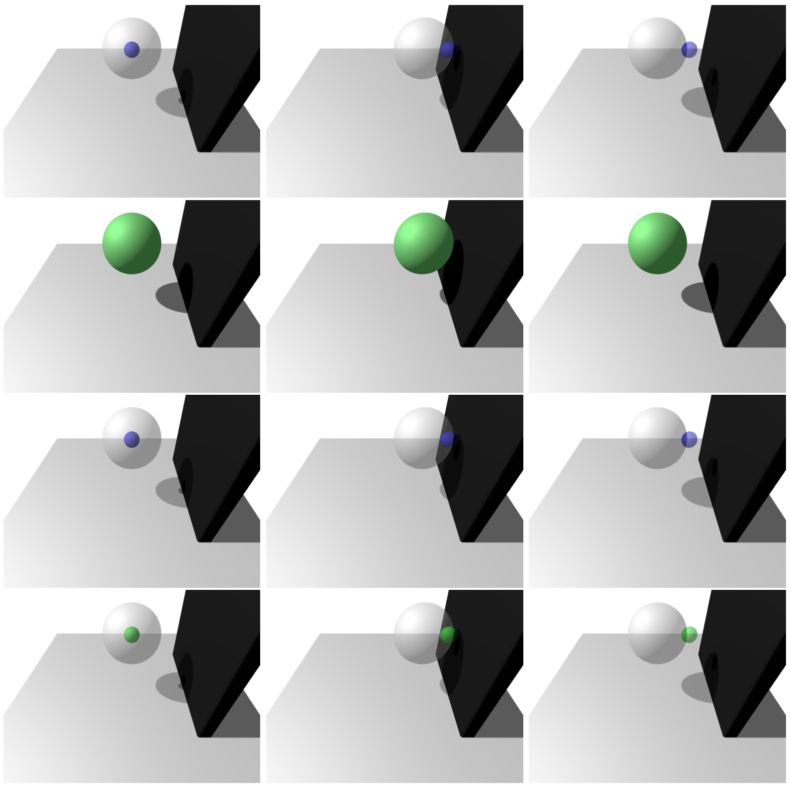


Figure 5.9: Trajectory fitting for a sphere. From left to right, each group shows 3 frames from the start, middle and end of the trajectory. We optimize the radius of a sphere from an initial value (blue), by comparing the simulated trajectory to that of a target sphere (gray, overlaid) and arrive at the result in green (also overlaid with gt in gray). Our formulation (top) works in the case of a head on-collision without gravity, while the engine without the time-of-contact differential fails.

Table 5.1: Resulting radius error for variants in the bouncing sphere scenarios. Time of contact differentials clearly improve the results and make the "no gravity" case work.

		resulting radius error		
scenario	variant	min	mean	max
w/ gravity	w/o toc	6e-5	0.038	0.219
	w/ toc	2e-6	0.007	0.046
w/o gravity	w/o toc	fails	fails	fails
	w/ toc	2e-4	0.002	0.006

to match a target position trajectory also generated with a random radius between 0.4 and 2.0 with the same initial conditions otherwise. This position depends on the shape parameters $\boldsymbol{\theta}$[9]. Trajectories are recorded as observation sequences of length 1.5 s. In each scenario, a new sphere radius is sampled randomly, and the predicted trajectory is fit to the observations using gradient descent on the objective function in Equation (5.58) with $\lambda = 0$. In Table 5.1 we compare accuracy for different variants. If gravity is disabled, the sphere hits the wall in a direction along the contact normal and the shape receives no gradient if the time of contact differential is not used (see Figure 5.9). If gravity is enabled, velocity at the contact has a downwards component, which transmits into a rotational velocity after the first bounce (see Figure 5.10). Together with the second bounce, this renders the trajectory more complex and we found it beneficial to optimize the trajectory in chunks including each contact separately by detaching poses and velocities before the second bounce to avoid exploding gradients that might build up by recursively computing gradients through this complex trajectory. The rotational velocity component leads to a gradient on the shape through the inertia tensor. In this case, using the time of contact differential improves convergence and accuracy of the radius estimate as displayed in Table 5.1 and Figure 5.11.

9: in this case, the radius

In Figure 5.12, we show quantitative results for optimizing the shape

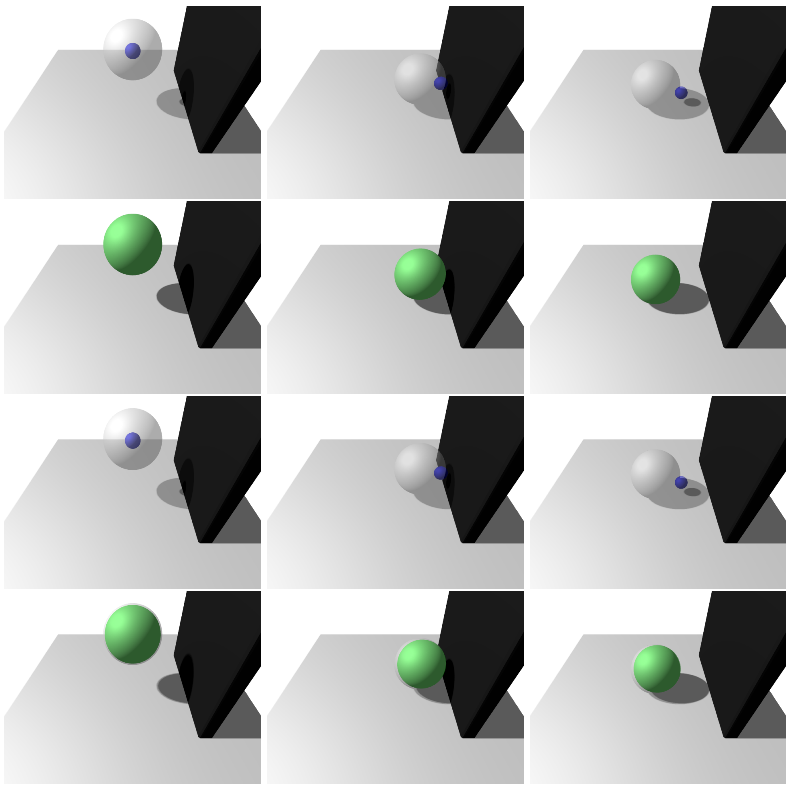


Figure 5.10: Trajectory fitting for a sphere. From left to right, each group shows 3 frames from the start, middle and end of the trajectory. We optimize the radius of a sphere from an initial value (blue), by comparing the simulated trajectory to that of a target sphere (gray, overlaid) and arrive at the result in green (also overlaid with gt in gray). In this setting with gravity enabled (bottom two rows), our formulation achieves more accurate results.

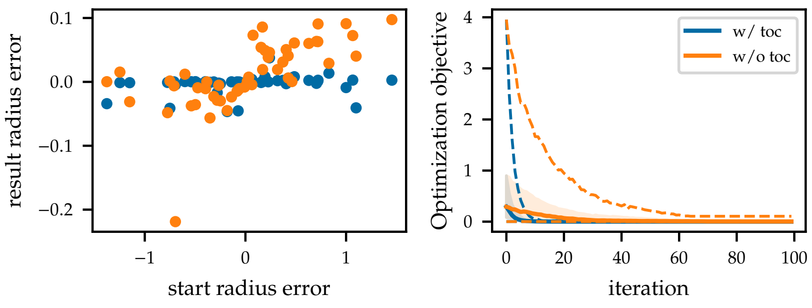


Figure 5.11: Bouncing sphere scenario with gravity. Left: start radius error vs. resulting radius error in sphere bounce experiment with gravity enabled with (blue) and without (orange) time of contact (toc) differential. The toc differential yields faster convergence and more accurate results. Right: median (solid lines), quartiles (shaded areas) and min/max (dashed lines) for the objective over time (blue: with toc, orange: without toc differential).

of several DeepSDF shape spaces (bob and spot trained with latent dimension 2; can, camera and mug with latent dimension 4) via a single bounce against the wall without gravity. For this experiment, λ in Equation (5.58) is set to 1×10^{-4}. The average Chamfer distance (measuring shape accuracy) over all objects is reduced from initial 0.016 to 0.010 by our approach. Without the time of contact differential, the resulting average is 0.017. One can see that while the optimization objective generally decreases by almost one order of magnitude in the median case, in some cases this does not hold for the shape accuracy. For these outlier runs, the objective exhibits a local optimum in this challenging scenario which does not contain the true shape. We further provide numeric results in Table 5.2 and include an ablation study for not using the time of contact differential for this experiment. One of the mug examples diverged and lead to an invalid state for the physics engine without the time of contact differential. It is thus excluded from the statistics in the w/o toc column and the overall average mentioned above. Qualitative results are shown in Figure 5.13.

We also test our approach for a challenging shape optimization scenario in which the DeepSDF shape space of bob and spot is used (see Figure 5.14).

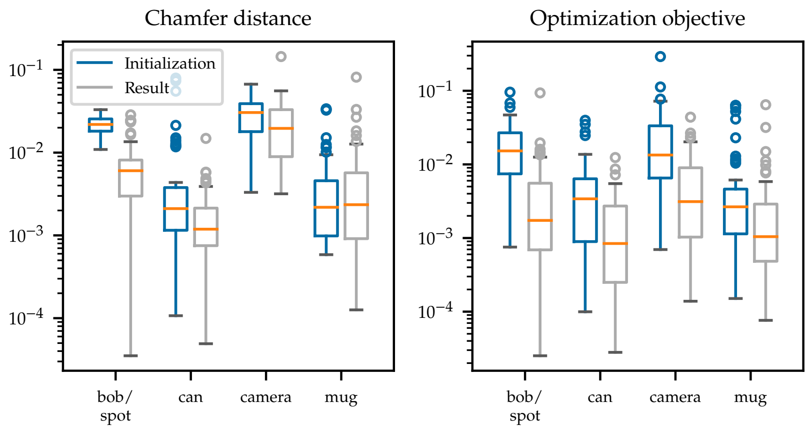


Figure 5.12: Trajectory fitting for learned shape spaces. Shape and trajectory accuracy improve in most cases. Note the log-scaling on the vertical axis.

The object is dropped on a stick. One of the objects is a genus-0 type cow-like shape (spot), while the other is genus-1 with a hole in the duck-like body (bob). Only the latter will fall through the stick and reach the target position. We optimize the shape latents in this scenario to reach the target position at the end of a 1.1 s sequence.

Shape from Inertia. Figure 5.15 gives quantitative results for shape optimization from inertia. We apply a random torque with unit norm on the objects in the first 0.3 s and simulate for 2 s. The optimization objective in this experiment is the mean-squared error of the object's rotational velocity to the ground truth:

$$E(\boldsymbol{\theta}) = \left\| \boldsymbol{\omega}(\boldsymbol{\theta})_e^{2s} - \boldsymbol{\omega}(\boldsymbol{\theta})_g^{2s} \right\|^2 + \lambda \left\| \boldsymbol{\theta} \right\|^2 , \tag{5.59}$$

where $\boldsymbol{\omega}(\boldsymbol{\theta})_e^{2s}$ and $\boldsymbol{\omega}(\boldsymbol{\theta})_g^{2s}$ are the angular velocities of the estimated and goal objects, respectively after 2 s of simulation. We additionally add the squared L^2 norm of the latent code with weight $\lambda = 1{\times}10^{-4}$ as regularizer for learned shape spaces and set $\lambda = 0$ for primitives. We sample 50 scenes with random initial and target shape parameters for each shape type and evaluate our approach for sphere, box and cylinder shapes, and the DeepSDF shape spaces from the bouncing objects experiment. We observe that the shape is well recovered in most of the runs. The average in Chamfer distance over all objects drops from 0.164 to 0.021. In a few outlier runs, the difference between result and target shape is large, while the optimization objective still achieves a very low value. This hints at local minima in the objective landscape, which could possibly be alleviated by choosing other torques. We further present numerical results in Table 5.3 and show qualitative results in Figure 5.16.

5.5.4 Friction and Mass Identification, Force Optimization

In Figure 5.17, we show quantitative results for friction, mass and force estimation in which either the duck-like "bob", or the cow "spot" is placed on a plane pushed with a constant force along the plane for 1 s. We generate 50 runs for each optimization target (mass, force and friction). We uniformly sample the force between 2 and 5 for each of the two dimensions along the plane, mass between 0.9 and 1.1, and friction between 0.01 and 0.25. Depending on the setting, one of these parameters

Table 5.2: Numerical results for trajectory fitting with learned shape spaces. “CD” denotes the Chamfer distance of the object to the target shape, “obj” denotes the optimization objective (Equation (5.58)) and “pos err” is the position error in the last step of the simulated trajectory. The time of contact differential shows improvements in accuracy in most cases over not using the differential. For the mug objects, the median accuracy is similar for using or not using the time of contact differential, while for Chamfer distance using time of contact differential makes the fitting more robust (see max measure).

Error		bob and spot			can			camera			mug		
		init	result	w/o toc	init	result	w/o toc	init	result	w/o toc	init	result	w/o toc
CD	mean	0.0216	**0.0081**	0.0157	0.0076	**0.0019**	0.0041	0.0290	**0.0244**	0.0285	**0.0052**	0.0064	0.0190
	min	0.0109	**0.0000**	0.0002	0.0001	**0.0000**	**0.0000**	0.0033	**0.0032**	0.0033	0.0006	**0.0001**	0.0002
	Q25	0.0182	**0.0030**	0.0044	0.0012	**0.0008**	0.0012	0.0179	**0.0089**	0.0137	0.0010	**0.0009**	0.0010
	median	0.0219	**0.0060**	0.0091	0.0021	**0.0012**	0.0017	0.0305	**0.0196**	0.0255	**0.0022**	0.0023	0.0022
	Q75	0.0255	**0.0081**	0.0199	0.0038	**0.0021**	0.0033	0.0390	**0.0330**	0.0337	**0.0046**	0.0057	0.0064
	max	0.0330	**0.0286**	0.1052	0.0794	**0.0148**	0.0382	**0.0671**	0.1447	0.1251	**0.0338**	0.0816	0.6794
obj	mean	0.0199	**0.0059**	0.0130	0.0058	**0.0020**	0.0027	0.0279	**0.0070**	0.0141	0.0087	**0.0042**	0.0063
	min	0.0008	**0.0000**	0.0002	0.0001	**0.0000**	0.0001	0.0007	**0.0001**	0.0004	0.0002	0.0001	**0.0000**
	Q25	0.0074	**0.0007**	0.0011	0.0009	**0.0002**	0.0004	0.0065	**0.0010**	0.0020	0.0011	0.0005	**0.0004**
	median	0.0153	**0.0017**	0.0044	0.0034	**0.0008**	0.0011	0.0134	**0.0031**	0.0061	0.0027	**0.0010**	0.0012
	Q75	0.0269	**0.0056**	0.0101	0.0064	**0.0027**	0.0032	0.0334	**0.0090**	0.0161	0.0046	**0.0029**	0.0037
	max	0.0957	**0.0936**	0.1254	0.0397	**0.0124**	0.0333	0.2917	**0.0438**	0.1112	**0.0633**	0.0647	0.0666
pos err	mean	0.3409	**0.1648**	0.2356	0.1601	**0.0962**	0.1065	0.3472	**0.2018**	0.2519	0.1799	**0.1399**	0.1404
	min	0.0794	**0.0091**	0.0249	0.0125	**0.0047**	0.0147	0.0344	0.0368	**0.0163**	0.0224	0.0145	**0.0127**
	Q25	0.2188	**0.0617**	0.0669	0.0721	**0.0435**	0.0447	0.2142	**0.0929**	0.1109	0.0746	0.0558	**0.0415**
	median	0.3212	**0.1215**	0.1709	0.1456	**0.0726**	0.0798	0.3336	**0.1531**	0.2038	0.1351	0.0908	**0.0890**
	Q75	0.4080	**0.2327**	0.2714	0.2291	**0.1430**	0.1556	0.4442	**0.2838**	0.3321	0.1945	**0.1721**	0.1756
	max	0.9399	**0.9301**	1.0076	0.5049	**0.3238**	0.3469	1.0767	**0.6934**	0.8276	0.8014	0.8032	**0.7997**

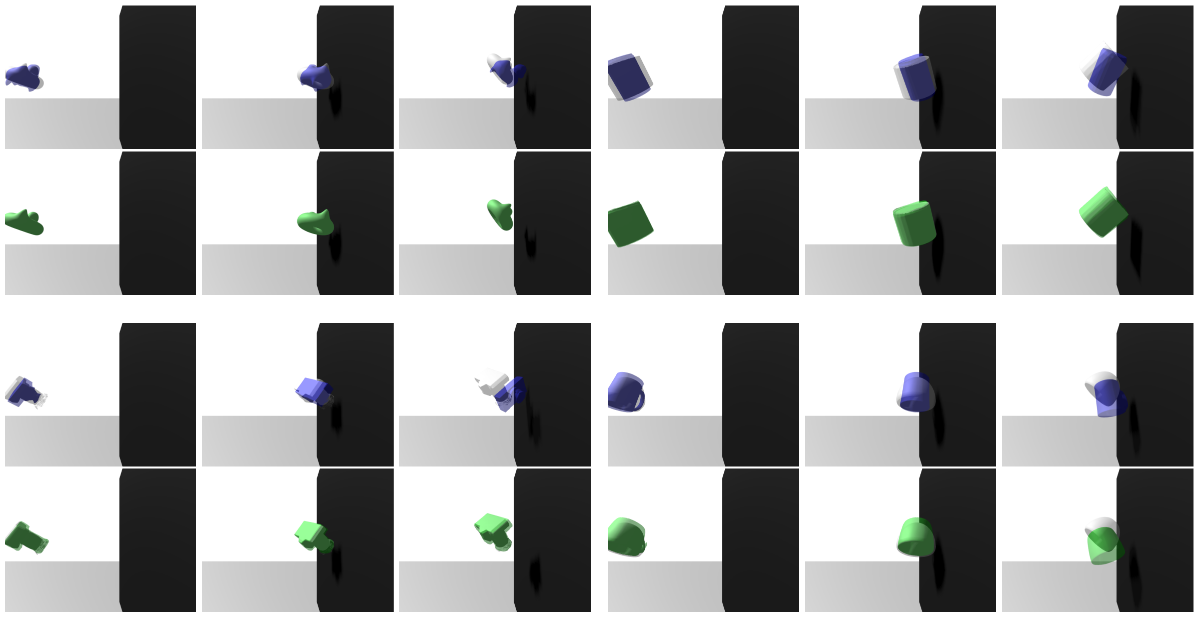


Figure 5.13: Trajectory fitting for learned shape spaces. From left to right, each group shows 3 frames from the start, middle and end of the trajectory. Initializations (blue) and results (green) overlaid with targets in gray. Each group shows one of the 4 learned shape spaces: bob and spot (top left), can (top right), camera (bottom left) and mug (bottom right). In most cases the estimated shapes are very accurate, except for the example shown for mugs.

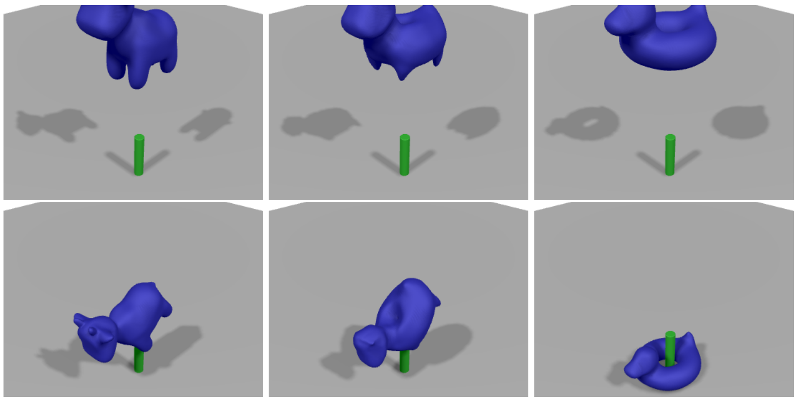


Figure 5.14: Collision-based shape optimization. Complex topological shape changes can be achieved by our approach. The shape is initialized with the genus-0 spot shape that falls onto a stick (left col.). The target pose for the object is on the floor through the stick (bottom right). This pose requires adapting the latent code to the genus-1 spot shape (middle col.: intermediate result (4 its.), right col.: final result (44 its.)).

Table 5.3: Numerical results for shape fitting by inertia. “CD” denotes the Chamfer distance of the object to the target shape and “obj” denotes the optimization objective (rotational velocity error).

Error		box		sphere		cylinder		bob and spot		can		camera		mug	
		init	result	init	result	init	result	init	result	init	result	init	result	init	result
CD	mean	1.4e-01	2.6e-02	6.3e-01	4.0e-02	3.1e-01	6.2e-02	2.2e-02	1.7e-03	7.9e-03	5.6e-04	3.0e-02	1.4e-02	5.4e-03	2.9e-03
	min	5.0e-04	1.0e-05	9.3e-04	2.3e-08	1.8e-03	1.2e-05	1.3e-02	1.3e-06	2.3e-04	9.1e-06	3.1e-03	7.5e-04	5.4e-04	6.5e-05
	Q25	6.3e-02	2.8e-03	7.3e-02	8.2e-06	1.1e-01	5.5e-04	1.9e-02	8.2e-06	1.1e-03	1.8e-04	1.9e-02	3.7e-03	1.0e-03	4.1e-04
	median	1.2e-01	1.1e-02	3.9e-01	1.5e-04	2.4e-01	8.5e-03	2.1e-02	1.3e-05	2.5e-03	5.3e-04	3.2e-02	8.3e-03	2.3e-03	1.3e-03
	Q75	1.9e-01	2.7e-02	8.0e-01	1.7e-03	4.2e-01	9.2e-02	2.4e-02	3.3e-05	4.0e-03	9.0e-04	4.0e-02	2.0e-02	3.9e-03	2.7e-03
	max	4.5e-01	1.9e-01	2.9e+00	7.8e-01	1.3e+00	3.7e-01	3.5e-02	4.9e-02	8.4e-02	1.3e-03	6.7e-02	1.1e-01	3.9e-02	4.5e-02
obj	mean	1.1e+00	5.2e-03	7.6e-01	7.8e-03	2.9e-01	2.1e-03	3.6e-01	4.2e-03	4.1e-01	7.3e-04	4.3e-01	4.0e-03	1.7e-02	4.0e-04
	min	8.0e-05	1.2e-05	6.1e-05	2.3e-08	2.4e-04	8.4e-08	1.4e-02	9.9e-07	1.6e-04	2.7e-06	5.1e-03	8.8e-06	7.6e-04	4.0e-06
	Q25	1.1e-01	3.8e-04	1.7e-02	4.6e-06	1.2e-02	4.2e-05	1.6e-01	4.6e-06	3.1e-03	3.5e-05	3.2e-02	4.6e-04	2.0e-03	6.5e-05
	median	4.0e-01	1.3e-03	1.1e-01	1.2e-04	7.0e-02	2.3e-04	2.5e-01	3.4e-05	1.4e-02	1.2e-04	1.3e-01	1.2e-03	4.3e-03	1.1e-04
	Q75	1.2e+00	3.4e-03	8.2e-01	6.6e-04	4.0e-01	8.1e-04	4.0e-01	1.2e-04	6.2e-02	1.1e-03	5.8e-01	2.9e-03	1.8e-02	3.5e-04
	max	7.5e+00	1.2e-01	6.7e+00	1.6e-01	2.6e+00	2.3e-02	1.3e+00	2.0e-01	1.0e+01	4.9e-03	2.3e+00	3.6e-02	1.5e-01	3.5e-03

is sampled anew to create a varied initialization. We optimize the mean squared position error of the simulation towards the ground-truth. The physical quantities are recovered with high accuracy in most settings. The average error drops from 0.07 to 0.003 for mass, 1.6 to 0.015 for force, and 0.08 to 1×10^{-4} for friction. Except for a few outliers, the optimization can recover the ground-truth parameters and trajectories almost perfectly, as we can see in Figure 5.17 and the numerical results in Table 5.4. Qualitative results are shown in Figure 5.18.

5.5.5 Fitting to Depth Image Observations

Synthetically Rendered Objects. We also test our method for fitting the physics simulation based on synthetic depth image observations from a static camera. We render depth images and object segmentation masks at resolution of 640×480 using pyrender[10]. The depth images are augmented with synthetic per-pixel Gaussian noise with mean $\mu = d$ and standard deviation $\sigma = 0.0001d^2$. We evaluate using the bounce scenarios (see Subsection 5.5.3) with spheres and cubes with rounded edges in randomized poses and sizes. We run the scenarios 20 times each for different settings (sphere and cube, each with and without gravity). The ground truth pose is set by adding Gaussian noise with mean 0 and standard deviation 0.1 to a standard rotation (represented as unit quaternion) and position. We initialize the first pose by adding the same noise distribution to the ground-truth pose. Spheres and cubes are

10: https://github.com/mmatl/pyrender

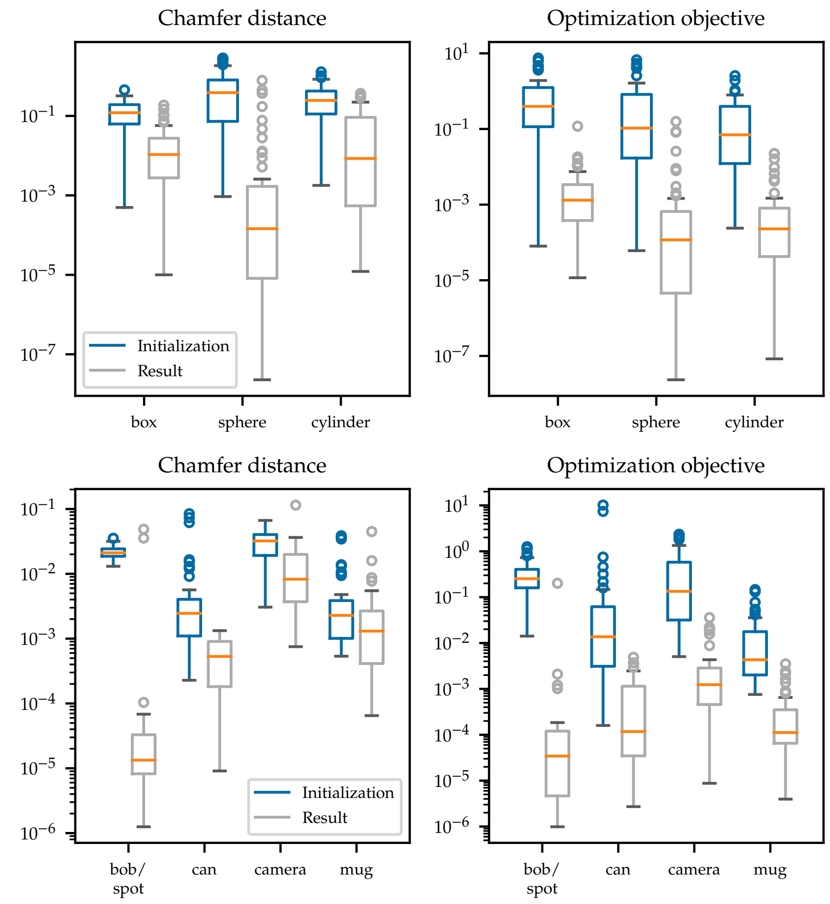


Figure 5.15: Shape from inertia. Left: Chamfer distance. Right: optimization objective value. Note the log-scale on the vertical axis.

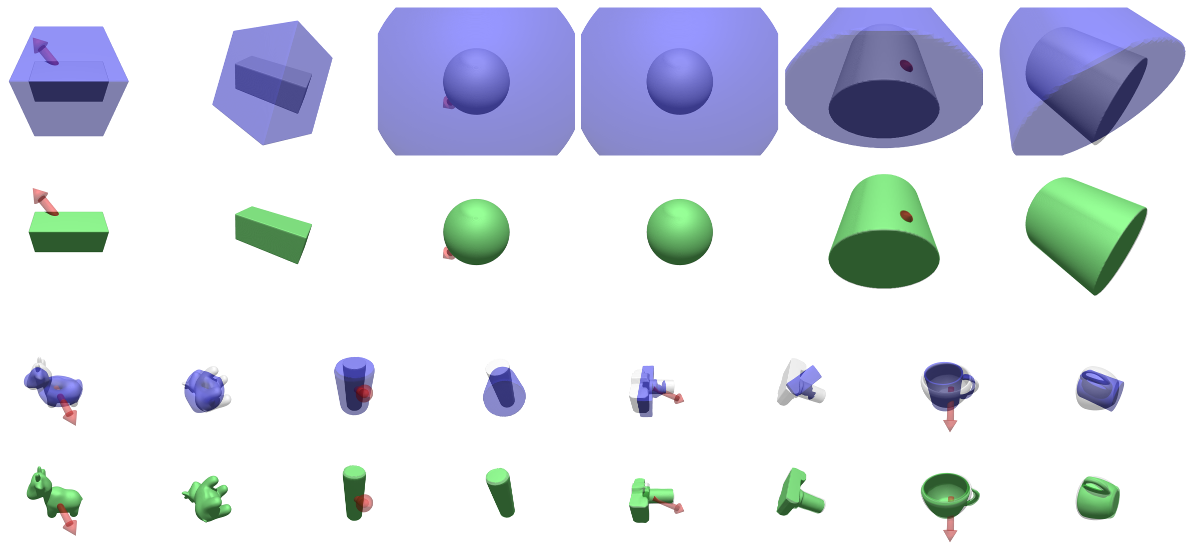


Figure 5.16: Shape from inertia. For each object, we show the initialization (blue) and result (green) overlaid with the ground truth in gray. The difference in inertia results in a different pose after 2 seconds of simulation (right rendering for each object). The red arrows indicate the torque that is applied at the beginning of the trajectory.

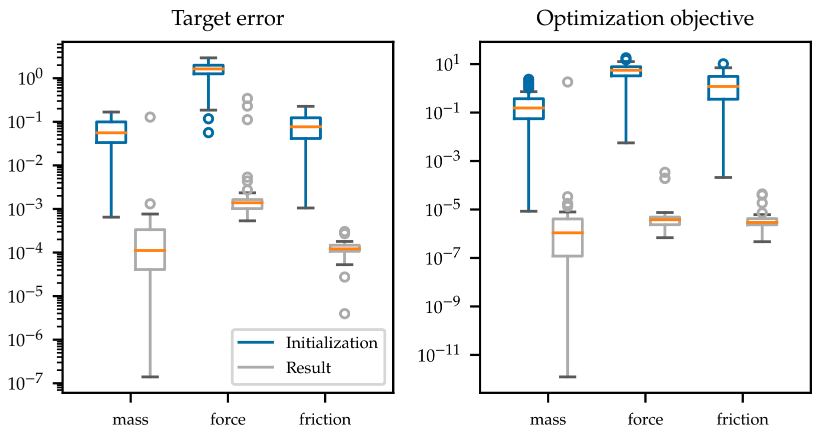


Figure 5.17: System identification results. Mass, force and friction are estimated with high accuracy. Note the log-scale on the vertical axis.

Table 5.4: Numerical results for system identification. The target error is the difference to the target value and "objective" denotes the optimization objective (position trajectory error).

Error		mass		force		friction	
		init	result	init	result	init	result
target error	mean	6.6e-02	2.8e-03	1.6e+00	1.5e-02	8.3e-02	1.3e-04
	min	6.5e-04	1.4e-07	5.6e-02	5.3e-04	1.1e-03	4.0e-06
	Q25	3.3e-02	4.1e-05	1.3e+00	1.0e-03	4.1e-02	1.1e-04
	median	5.6e-02	1.1e-04	1.6e+00	1.4e-03	7.7e-02	1.2e-04
	Q75	1.0e-01	3.4e-04	2.0e+00	1.6e-03	1.2e-01	1.5e-04
	max	1.7e-01	1.3e-01	2.9e+00	3.4e-01	2.3e-01	3.0e-04
objective	mean	3.3e-01	3.7e-02	5.7e+00	1.8e-05	1.9e+00	5.0e-06
	min	8.5e-06	1.2e-12	5.6e-03	6.9e-07	2.1e-04	4.7e-07
	Q25	5.6e-02	1.2e-07	3.3e+00	2.4e-06	3.5e-01	2.3e-06
	median	1.5e-01	1.1e-06	5.6e+00	3.8e-06	1.2e+00	2.9e-06
	Q75	3.7e-01	4.1e-06	7.8e+00	4.9e-06	3.1e+00	4.2e-06
	max	2.3e+00	1.8e+00	1.8e+01	3.4e-04	1.0e+01	4.4e-05

sampled with radii/edge lengths $s \in [0.5, 1.5]$. We sample initialization object sizes $s' = s + \varepsilon$ with $\varepsilon \in [0, 1.0]$. This initialization larger than the target size is chosen as segmented depth measurements from RGB-D cameras typically overestimate the size of the object due to inaccurate segmentation masks. For pose and trajectory fitting, we first segment the depth map using the given segmentation mask and compute the set of 3D points $\mathcal{P}$ belonging to the object from it as explained in Section 2.3. We then use the mean squared SDF value of the points in $\mathcal{P}$ after transforming them to the object frame using its pose estimate and the known camera view pose:

$$E_{\text{sdf}}(s, \mathbf{T}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \phi(\mathbf{T}\mathbf{p}, s)^2, \tag{5.60}$$

where $\mathbf{T} \in \mathrm{SE}(3)$ is the transformation matrix transforming the points in $\mathcal{P}$ from camera to object frame and $\phi(\mathbf{p}, s)$ is the object SDF (see also Subsection 2.2.1). Fitting of pose and object size in a single frame is prone to local minima and typically overestimates the object size, as can be seen in the middle rows of Figures 5.19 to 5.21. Figure 5.22 illustrates this problem. Fitting the shape and position of the objects (gray) to the point clouds (yellow) yields wrong local optima. For the cube, this local optimum is a perfect fit, while the actual object would be smaller, ending where the point cloud ends and having a position closer to the camera. For the sphere, the local optimum ends up with some points outside and some inside the object.

We compare the single frame pose fit with its refinement through fitting the trajectory on all depth image observations (cumulating Equation (5.60)

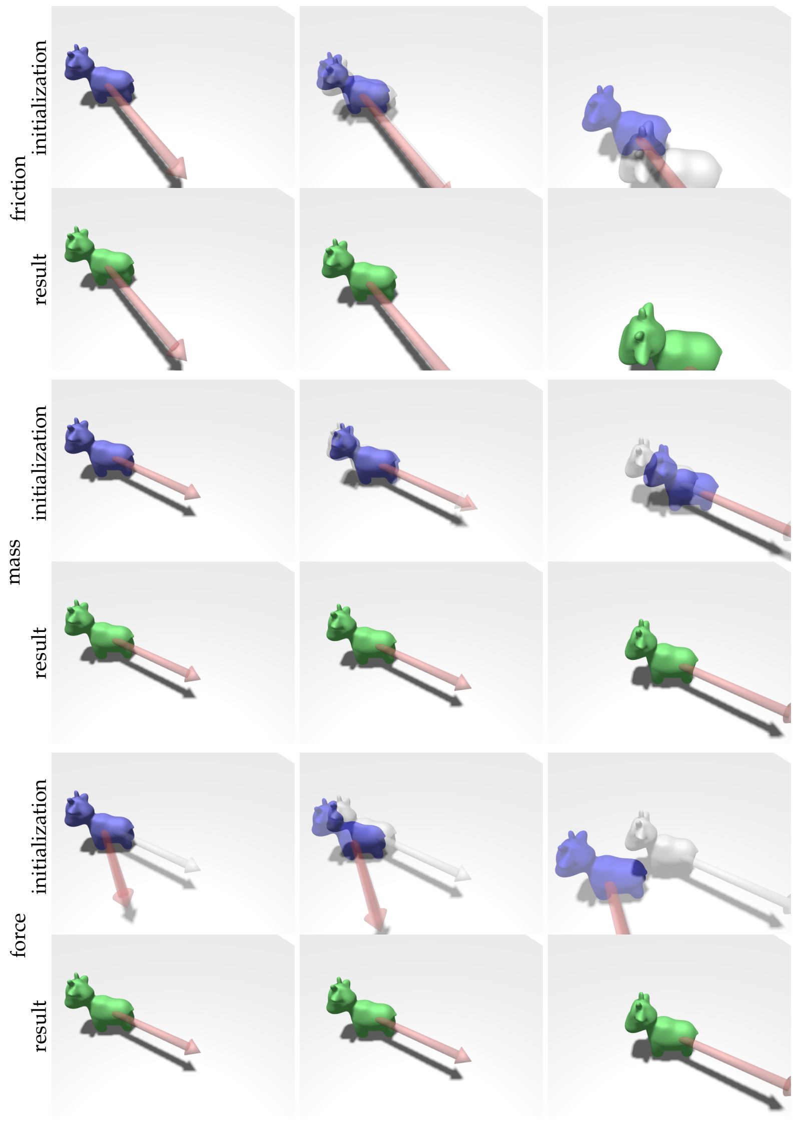


Figure 5.18: Friction (top) and mass (middle) identification and force optimization (bottom). Each group shows 3 frames from the beginning, middle, and end of a 1s trajectory. With the initial estimates (blue), the simulated trajectory differs from the target one (overlay in gray). By optimizing the parameters to match the trajectory, we manage to recover them well. The red arrows indicate the pushing force parallel to the plane that is applied to the object (scaled by a factor of 0.5 for better visibility).

over all frames) in each sequence using the differentiable simulation. Table 5.5 shows quantitatively that fitting pose and size of spheres and cubes to a depth map falls into a local optimum, making the pose estimate worse while improving the shape estimates. Refining this result via trajectory optimization improves radius and pose estimates, as can also be seen qualitatively in Figures 5.19 to 5.21.

Real-World Experiment. In Figure 5.23 we provide results for a real data experiment, in which we reproduce a setting similar to the synthetic depth fitting experiment by throwing a tennis ball against a wall and recording it with an Intel RealSense D455 camera at a resolution of 640×480 with 30 FPS. The camera comes with an inertial measurement unit (IMU), which gives the gravity direction. In contrast to the synthetic examples, we do not have known ground-truth parameters for restitution, friction, velocity or position and need to optimize for all of these parameters. We first compute the point cloud from the depth map as explained in Section 2.3. Then, we segment the planes and the ball from the point cloud by combined geometric and color segmentation.

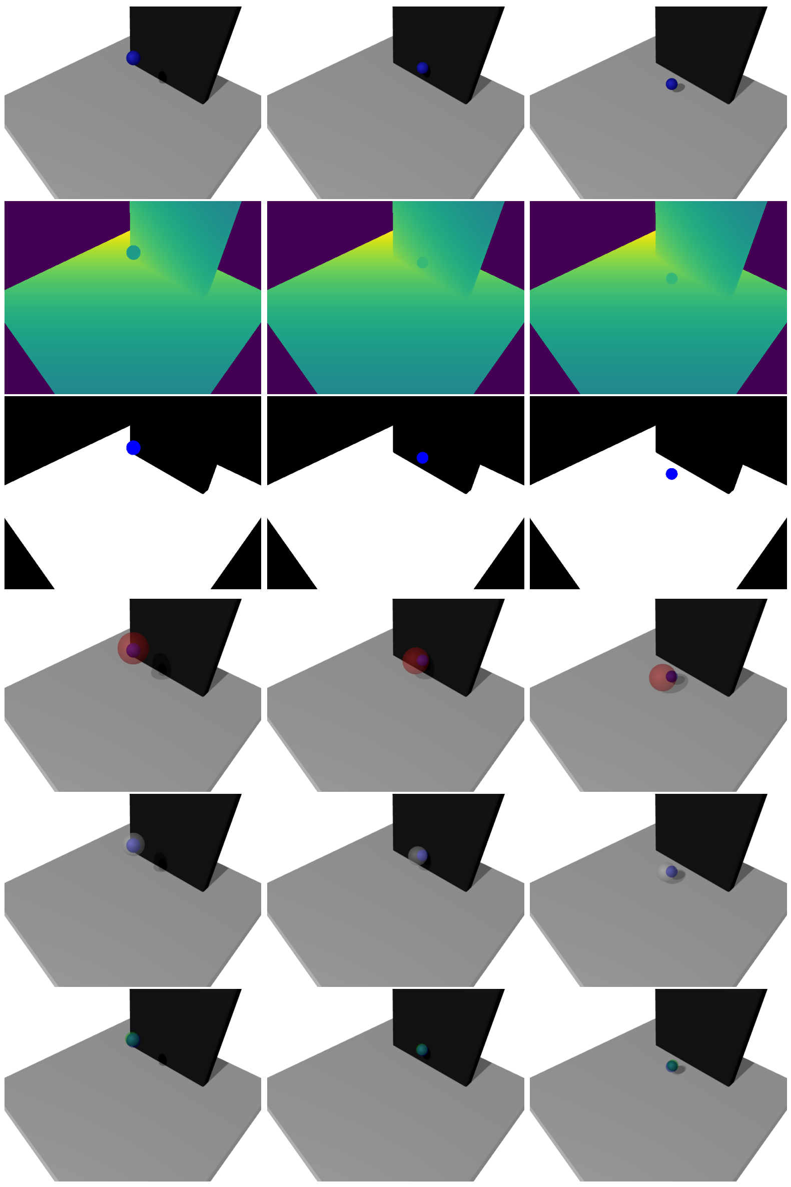


Figure 5.19: Fitting to depth observations for spheres with gravity. The inputs for this experiment are depth and segmentation masks (illustrated in the second and third rows below a rendering of the scene) at 3 time steps from the beginning, middle and end of a 1.5 s trajectory. Optimization of initialization pose and the size of the object is then carried out in 2 stages. From an initialization (red overlay over the blue target), pose and shape of the SDF are first fit to the first depth frame (gray overlay over blue target). This optimization can fall into local optima, *e. g.*, by overestimating the size and putting the object further back as can be seen in the fifth row. In the last row, we can see that this error is recovered by our optimization using our differentiable simulation (green overlay over blue target).

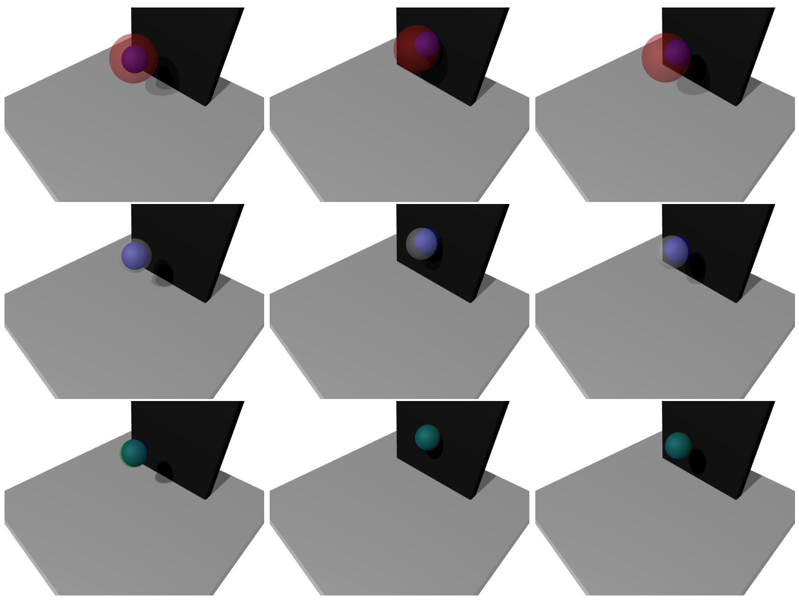


Figure 5.20: Fitting to depth observations for spheres without gravity. The inputs for this experiment are the same as in Figure 5.19. Optimization of initialization pose and the size of the object is then carried out in 2 stages. From an initialization (red overlay over the blue target), pose and shape of the SDF are first fit to the first depth frame (gray overlay over blue target). This optimization can fall into local optima, *e. g.*, by overestimating the size and putting the object further back as can be seen in the second row. In the last row, we can see that this error is recovered by our optimization using our differentiable simulation (green overlay over blue target).

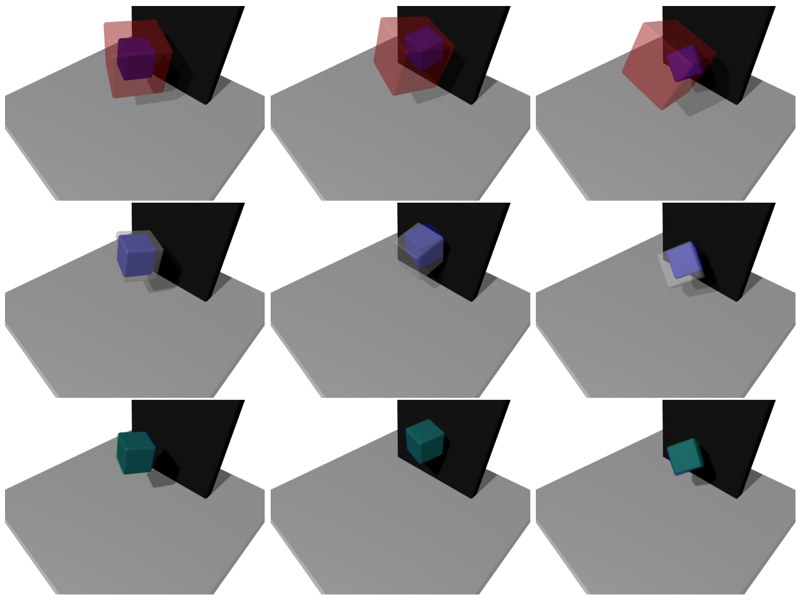


Figure 5.21: Fitting to depth observations for cubes without gravity. The inputs for this experiment are the same as in Figure 5.19. Optimization of initialization pose and the size of the object is then carried out in 2 stages. From an initialization (red overlay over the blue target), pose and shape of the SDF are first fit to the first depth frame (gray overlay over blue target). This optimization can fall into local optima, *e.g.*, by overestimating the size and putting the object further back as can be seen in the second row. In the last row, we can see that this error is recovered by our optimization using our differentiable simulation (green overlay over blue target).

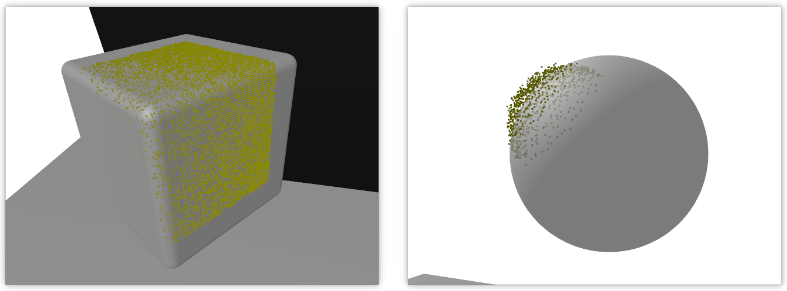


Figure 5.22: Fitting an object (gray) to a point cloud (yellow) can yield a wrong local optimum for both pose and shape estimates since the point cloud does not constrain the unseen side of the object. For the box (left) this local optimum yields a perfect fit, for the sphere (right), some of the points end up outside the object (darker) and some inside it (lighter/gray points).

As the initial velocity of the ball is unknown, we try to estimate it using finite differences from the first two frames. For initialization, we thus first compute the positions in these frames as the centroids of the segmented point clouds and the radius of the ball as half the diameter of the segment in the first frame, resulting in an estimate of 2.96 cm.

In a first optimization stage[11], we refine the previously initialized positions in the first two frames and the radius of the ball by optimizing them using Equation (5.60). The resulting radius (3.68 cm) is a slight over-estimate of the ball's radius (gt: 3.24 cm). As we can see in the second row of Figure 5.23, simulating the ball (blue) with this radius, the initial velocity computed by finite differences from the positions, and empirically set friction and restitution coefficients, fits the depth measurements (red) well in the beginning but deviates after the collision with the wall.

11: Similar to the first-frame fit in the synthetic examples

We then optimize restitution, friction, initial velocity, initial position, and radius using the depth fitting objective from Equation (5.60). This optimization problem has many degrees of freedom, hence, fitting these parameters on a single trajectory is prone to local minima. Still, our approach is able to recover the radius and trajectory well in this experiment. The initial radius is improved to 3.13 cm by our approach and we can see a better fit of the ball simulated with the optimized parameters (green) and the segmented point cloud (red) in the last row of Figure 5.23. The result demonstrates that trajectory and radius estimates can be improved by our physics-based approach in this challenging scenario.

	sphere		cube	
error	w/o gravity	w/ gravity	w/o gravity	w/ gravity
init pos	0.040	0.040	0.040	0.040
pos frame fit	0.056	0.056	0.077	0.077
pos traj. fit	0.031	0.044	0.023	0.029
init rot	–	–	0.135	0.135
rot frame fit	–	–	0.001	0.001
rot traj. fit	–	–	0.002	0.000
init size	0.512	0.512	0.512	0.512
size frame fit	0.163	0.163	0.137	0.137
size traj. fit	0.022	0.014	0.029	0.030

Table 5.5: Position and shape parameter errors for single-frame fitting and trajectory fitting to depth observations.

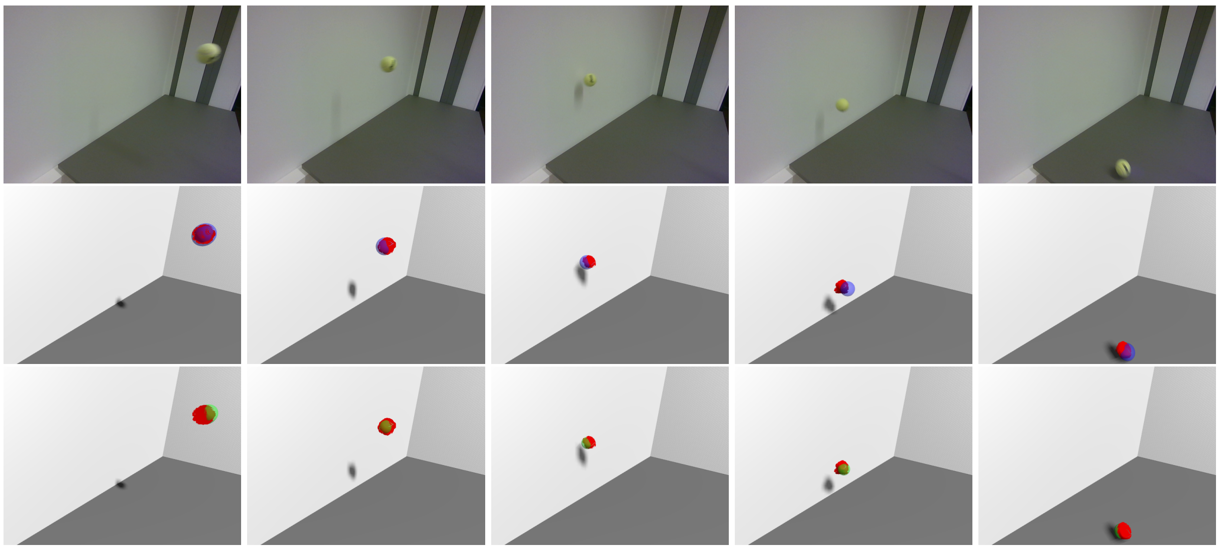


Figure 5.23: Results on a real-world scene (frame 0, 2, 4, 6 and 9). Blue: fit to first two frames. Green: fit of simulated trajectory. Red: Recorded point cloud segment. Our result (green) fits the trajectory better overall, especially after the bounce. The radius is improved from 3.68 cm (initial 2 frames fit) to 3.13 cm (trajectory fit, ground truth 3.24 cm).

5.5.6 Runtime

We implemented our method in PyTorch (Paszke et al. 2019) and have not yet tuned our implementation for efficiency. Currently, our method requires several seconds of computation time for one second in simulation. Major room for run-time improvements is in collision detection and contact point estimation which we have not yet optimized on the GPU. Note that Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse (2020) have demonstrated that these steps can be significantly sped up with timings in the microseconds. The LCP optimization could be sped up by approaches such as (Shao et al. 2021). Implementing more efficient C++ implementations instead of the current ones in Python could further improve the efficiency of contact detection and the solution of the LCP.

5.5.7 Limitations

We observed that bouncing boxes with sharp corners and edges can have strong variations in contact points and number of contacts which makes system identification challenging due to varying contact situations (*e.g.*, a box collides with different corners/edges). First-order gradient based optimization can be difficult in such cases for the trajectory alignment. In the shape from inertia experiment, we observed that in some outlier cases, depending on the start and target configurations, the torque and the shape space, the velocity loss can be reduced while the shape is converging to a wrong local minimum. Non-convex shapes can render the system identification problem itself non-convex for which gradient descent will retrieve local minima.

5.6 Conclusion

We propose a novel approach for differentiable rigid-body physics simulation that models arbitrary watertight shapes using SDF representations. We devise differentiable inertia tensors and time of contact in a velocity-based constraint-based time-stepping method. Our experimental results demonstrate that physical system identification including shape inference is possible for several challenging scenarios with non-convex shapes and collisions via gradient descent. We fit our model on sample trajectories and depth image observations of synthetic scenes. Further scaling our approach for system identification, 3D vision, and control in more complex scenarios through more elaborate optimization methods than gradient descent is an interesting direction for future research.

Conclusion 6

In this thesis, we present several approaches for 3D reconstruction in dynamic scenes to enable autonomous agents to perceive the 3D world around them. While 3D reconstruction in general has been an active area of research in computer vision over the last decades, most previous works assumed static environments, in which the camera or the autonomous agent is the only moving entity. This assumption limits the applicability of these algorithms, as we often want autonomous agents to interact with objects or other agents that move, *i.e.*, in *dynamic environments*. Some approaches have tried to address this issue by making the tracking and reconstruction algorithms *robust* to dynamic scene elements, *i.e.*, treating dynamic parts of the scene as outliers. By contrast, we reconstruct and track the dynamic parts of the scene and the static background in separate models. Further, we demonstrate that dynamic scenes exhibit more information than just the geometry observed by the camera. We reason about physically plausible scene configurations and physically plausible object motion and show that we can infer more information about object shapes.

In Chapter 3, we present an approach for tracking and mapping dynamic objects after detecting them using semantic instance segmentation. We choose to represent the objects and the static background by individual volumetric truncated signed distance function models. This representation allows for efficiently formulating the data association problem of pixels to the different moving objects in a geometric fashion, as signed distance functions trivially provide distances to the closest surface for 3D points. We compute the distances of points measured in depth images to the different models and formulate a data association probability of pixels to object models based on these distances. After *estimating* the probability under the scene configuration from the previous frame, we then *maximize* it by updating the poses and geometries using the estimated probability to attribute the pixel contributions to the different models. This probabilistic formulation allows us to run the comparatively expensive semantic instance segmentation only on a relatively sparse set of frames for improved efficiency. We further demonstrate state-of-the-art performance in object tracking accuracy compared to related approaches.

We extend our work in Chapter 4 for completing the reconstructed object geometries from Chapter 3 by reasoning about physical plausibility of the scene configurations. Our approach collects depth measurements in key frames as oriented surface point clouds and attributes them to the individual object models by the association likelihood from Chapter 3. We then combine these point clouds with two physical plausibility constraints in an offline optimization method. The first of these constraints reasons about *observed empty space* and constrains the objects to only occupy *unobserved* space, *i.e.*, to lie within a hull. While this constraint can and has been used to improve reconstructions in static environments, we propose a second constraint, exploiting the *dynamics* of the scene. We reason that observed surfaces do not disappear just because they are occluded by another object and that they constrain the possible geometries of the occluding object as no two objects can occupy the same space at the same time. Signed distance functions are especially suited for formulating this constraint, as negative distances indicate how far *inside* an object a certain point is. We reason that if a point in 3D is a certain distance inside one object, it has to be *outside* all other objects by at least the same distance. Our experiments demonstrate that incorporating these constraints leads to physically plausible shape completions. Most other approaches to shape completion use learning-based shape priors, requiring 3D training data for the object one expects to observe. As our approach is based on reasoning about physical plausibility, it provides an orthogonal direction as we do not need to make such assumptions about the present 3D shapes. While our reconstructions are not very accurate on the surfaces estimated by the constraints, we anticipate that our watertight models still provide useful cues for applications like robotic grasping.

In Chapter 5, we present another approach for using physical plausibility cues in object reconstruction. Different to Chapters 3 and 4, we now employ learned parametric signed distance function models to represent shapes. This allows us to distinguish different shapes by low-dimensional parameters and to optimize these parameters when fitting to observations as was shown in previous work. We combine these parametric models with a differentiable physics simulation method by extending it to use signed distance functions for contact detection. To optimize the shape of objects through the physics simulation, we observe that the trajectory of objects after collisions is affected by the object shape, as, *e.g.*, a ball moving towards a wall would hit the wall earlier or later depending on its radius. In other words, the *time of contact* depends on the shape parameters of the objects. As the differentiable simulation we build upon assumes the time step size to be constant when computing gradients, we propose a novel method for computing gradients for the time of contact at collisions. Our experiments demonstrate that our contributions enable accurate shape estimation by reasoning about physically plausible object motion in the simulation. The optimized shapes further recover the target trajectories and depth observations well, indicating possible applications in real-to-sim transfer. We anticipate that our contributions can be useful for estimating object shapes and reasoning about possible interactions with the objects in robotic applications.

While the works presented in this thesis provide several approaches for improving robotic 3D perception in dynamic environments, more work is needed to make them applicable in practice. We will now summarize the limitations of our approaches and discuss possible directions for future work.

6.1 Limitations

Segmentation Methods for EM-Fusion. While we do not make assumptions about the reconstructed 3D shapes in Chapters 3 and 4, both approaches are limited to objects that can be detected by the semantic instance segmentation method Mask R-CNN (He et al. 2017). While we observed that Mask R-CNN typically manages to create segmentation masks (sometimes not with the correct classification) for moving objects, this limits our methods to object categories in the training set for Mask R-CNN. The Mask R-CNN module can easily be replaced by other approaches that provide segmentation masks for the objects. One way to avoid having to train on known object classes are recent advances in instance segmentation that allow class-agnostic segmentation for instances of any object (Kirillov et al. 2023; X. Wang et al. 2022). Other approaches can perform instance segmentation much faster than Mask R-CNN (Bolya et al. 2019), or focus more on moving or potentially moving objects by motion segmentation (Bao et al. 2022; Tokmakov et al. 2018; Xie et al. 2022). Another interesting direction would be to use the geometric distances after the robust alignment with the Huber norm as segmentation cues.

Invalid Object Configurations in Differentiable Simulation. The differentiable simulation we use in Chapter 5 cannot resolve penetrations and assumes a valid scene configuration without penetration at the start. Initialization configuration from observations (*e.g.*, object pose estimates) might not satisfy this condition and might thus require additional processing to resolve the penetration. Further, gradient-based optimization might also lead to such invalid states, requiring a similar failure case handling strategy. Apart from resolving penetrations "manually", penalty-based differentiable simulation approaches (Geilinger et al. 2020; J. Xu, T. Chen, et al. 2021; J. Xu, S. Kim, et al. 2022) model contacts by soft penalties instead of the hard constraints in the LCP formulation (Avila Belbute-Peres et al. 2018) and might be able to handle these cases better.

Computation Times. One aspect currently limiting the applicability of the presented methods is computation time. Robotics applications typically have strict real-time requirements, *i.e.*, by the time the next frame arrives from the camera, the previous frame should already be processed, and the robot might need to react to its environment depending on that processing result. While our works' focus lies on demonstrating what is possible with our approaches, we still want to summarize the computational performance of our implementations and possible paths for improvement.

As reported in Chapter 3, we achieve average computation times per frame between 100 ms and 260 ms for EM-Fusion. The videos in the experiments are captured at 30 Hz, capturing a new frame every 33 ms, which is about one third of our best-case average processing time. While we did implement the approach presented in Chapter 3 in C++ for computational efficiency and performed many computations on the GPU using CUDA, we anticipate further run time improvements, like parallel (instead of sequential) processing of the different object models or multithreading for the semantic instance segmentation part, will yield closer to real-time computation times.

Co-Section in Chapter 4 is designed as an offline optimization approach to be run at a lower frame rate. Generally the key frame rate at which the optimization refines the previous object models can be seen as a parameter which can be tuned for a trade-off between updating models more often and lower overall run times. As explained in Chapter 4, the computation times are in the ranges of several seconds, while Schroers et al. (2014) report lower computation times at higher resolutions. We anticipate that a more efficient implementation of the optimization algorithm can improve the performance.

DiffSDFSim in Chapter 5 is quite far from real-time applications. The simulation takes up to several seconds for one second of simulation time and needs to be rolled out for each iteration of the optimization algorithm. We implemented DiffSDFSim in Python and PyTorch (Paszke et al. 2019). While PyTorch allows the parallelization of some operations on the GPU, the Python interface comes at the cost of computational overhead. We anticipate that more efficient implementations, *e.g.*, in C++, for the contact detection algorithm and solving the LCP, will improve the run time. Especially the contact detection algorithm, which takes up a large part of the computation time in our approach, was demonstrated with much lower computation times in an efficient implementation on the GPU (Macklin, Erleben, Müller, Chentanez, Jeschke, and Corse 2020).

6.2 Future Work

Combining the Presented Methods. Apart from addressing the limitations mentioned in Section 6.1, one path for future work includes combinations of the presented approaches. One interesting direction would be to use learned parametric shape spaces like (Park et al. 2019) in online tracking like EM-Fusion (Chapter 3), possibly further optimizing the shape parameters using the physical plausibility constraints from Co-Section (Chapter 4). While the object poses in EM-Fusion were arbitrarily aligned with the world frame, this combination requires first estimating the object pose as the shape spaces are typically trained on objects in a fixed coordinate frame. As the shape spaces are usually trained on categories of objects, this introduces the problem of *category-level object pose estimation*. This problem was first addressed by H. Wang et al. (2019), who also published a data set for this problem. Subsequent work has tried to combine category level pose and shape estimation (K. Chen and Dou 2021; Irshad, Kollar, et al. 2022; Tian et al. 2020), recently even by predicting shape latent codes for DeepSDF models (Irshad, Zakharov, et al. 2022). As mentioned before, such pose estimation models could be used as initialization for tracking DeepSDF models with approaches like EM-Fusion (Chapter 3)[1]. Including the plausibility losses from Co-Section (Chapter 4) or the trajectory optimization from DiffSDFSim (Chapter 5) can then further improve the shape estimation.

1: Other approaches proposed category-level object tracking from known initial poses, (*e.g.*, C. Wang et al. 2020; Wen and Bekris 2021) and could also be used here.

Interactive Perception in Dynamic Scenes. Thinking further, while vision is one important cue in human perception, we also include other modalities like sound or touch when perceiving our environment. Especially for touch, we interact with objects from a very young age and collect experiences about how our interactions affect objects. Interactive perception approaches for robotic agents could build upon the reconstructions recovered by the approaches presented in this thesis and refine the shape and motion estimates further. Moreover, physical models like the simulation model used in DiffSDFSim only provide approximations and do not model all effects we observe in our complex physical world.

Simulating interactions and comparing the result with actual interactions with the real world might provide interesting cues for improving the physical model. We hope that the approaches presented in this thesis will provide useful building blocks in more general approaches for dynamic scene perception.

Bibliography

Abdulla, Waleed (2017). *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN (cited on pages 37, 38).

Agarwal, Sameer, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski (2011). 'Building Rome in a day'. In: *Communications of the ACM* 54.10, pp. 105–112. DOI: 10.1145/2001269.2001293 (cited on page 2).

Agarwal, Sameer, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski (2009). 'Building Rome in a day'. In: *2009 IEEE 12th International Conference on Computer Vision*. 2009 IEEE 12th International Conference on Computer Vision. ISSN: 2380-7504. IEEE, pp. 72–79. DOI: 10.1109/iccv.2009.5459148 (cited on page 2).

Alexa, Marc, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva (2003). 'Computing and rendering point set surfaces'. In: *IEEE Transactions on Visualization and Computer Graphics* 9.1 (1), pp. 3–15. DOI: 10.1109/tvcg.2003.1175093 (cited on page 47).

Amos, Brandon and J. Zico Kolter (Aug. 2017). 'OptNet: Differentiable Optimization as a Layer in Neural Networks'. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 136–145 (cited on pages 65, 70, 71).

Anitescu, Mihai and Florian A. Potra (1997). 'Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems'. In: *Nonlinear Dynamics* 14.3, pp. 231–247. DOI: 10.1023/a:1008292328909 (cited on pages 65, 66).

Avila Belbute-Peres, Filipe de, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter (2018). 'End-to-End Differentiable Physics for Learning and Control'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. Curran Associates, Inc., pp. 7178–7189. (Visited on 08/02/2022) (cited on pages 6, 64–66, 70, 71, 97).

Bao, Zhipeng, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert (2022). 'Discovering Objects that Can Move'. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 11779–11788. DOI: 10.1109/cvpr52688.2022.01149 (cited on page 97).

Baraff, David (1996). 'Linear-time dynamics using Lagrange multipliers'. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*. ACM Press. DOI: 10.1145/237170.237226 (cited on page 65).

Bender, Jan, Kenny Erleben, and Jeff Trinkle (2013). 'Interactive Simulation of Rigid Body Dynamics in Computer Graphics'. In: *Computer Graphics Forum* 33.1, pp. 246–270. DOI: 10.1111/cgf.12272 (cited on pages 5, 65).

Bender, Jan and Alfred A. Schmitt (2006). 'Fast dynamic simulation of multi-body systems using impulses'. In: *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, pp. 81–90 (cited on page 65).

Besl, Paul J. and Ramesh C. Jain (1986). 'Invariant surface characteristics for 3D object recognition in range images'. In: *Computer Vision, Graphics, and Image Processing* 33.1, pp. 33–80. DOI: 10.1016/0734-189x(86)90220-3 (cited on page 22).

Besl, Paul J. and Neil D. McKay (1992). 'A method for registration of 3-D shapes'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2, pp. 239–256. DOI: 10.1109/34.121791 (cited on pages 25, 33).

Bishop, Christopher M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer (cited on pages 4, 23, 32).

Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee (2019). 'YOLACT: Real-Time Instance Segmentation'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00925 (cited on page 97).

Bolza, Oskar (1904). *Lectures on the calculus of variations: Repr. of ed. 1904*. Vol. 14. University of Chicago Press (cited on page 49).

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press (cited on pages 49, 68, 70).

Bylow, Erik, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers (2013). 'Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions'. In: *Robotics: Science and Systems IX*. Berlin, Germany: Robotics: Science and Systems Foundation. DOI: 10.15607/rss.2013.ix.035 (cited on pages 3, 4, 23, 25, 28, 34, 35).

Calakli, Fatih and Gabriel Taubin (2011). 'SSD: Smooth Signed Distance Surface Reconstruction'. In: *Computer Graphics Forum* 30.7, pp. 1993–2002. DOI: 10.1111/j.1467-8659.2011.02058.x (cited on pages 5, 47, 49, 50).

Canelhas, Daniel R., Todor Stoyanov, and Achim J. Lilienthal (2013). 'SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images'. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. DOI: 10.1109/iros.2013.6696880 (cited on page 28).

Chen, Kai and Qi Dou (2021). 'SGPA: Structure-Guided Prior Adaptation for Category-Level 6D Object Pose Estimation'. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, pp. 2773–2782. DOI: 10.1109/iccv48922.2021.00277 (cited on page 98).

Cifuentes, Cristina Garcia, Jan Issac, Manuel Wuthrich, Stefan Schaal, and Jeannette Bohg (2017). 'Probabilistic Articulated Real-Time Tracking for Robot Manipulation'. In: *IEEE Robotics and Automation Letters (RA-L)* 2.2, pp. 577–584. DOI: 10.1109/lra.2016.2645124 (cited on page 25).

Cline, Michael Bradley (2002). 'Rigid body simulation with contact and constraints'. en. In: Retrospective Theses and Dissertations, 1919-2007. DOI: 10.14288/1.0051676 (cited on pages 67–69).

Coumans, Erwin (2010). *Bullet physics engine*. Open Source Software: https://bulletphysics.org (cited on page 66).

Cremers, Daniel and Kalin Kolev (2011). 'Multiview Stereo and Silhouette Consistency via Convex Functionals over Convex Domains'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.6, pp. 1161–1174. DOI: 10.1109/tpami.2010.174 (cited on pages 47, 48).

Curless, Brian and Marc Levoy (1996). 'A volumetric method for building complex models from range images'. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. SIGGRAPH '96. New York, NY, USA: ACM Press, pp. 303–312. DOI: 10.1145/237170.237269 (cited on pages 4, 17, 26, 27, 35, 47, 48, 53).

Dai, Angela, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner (2018). 'ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2018.00481 (cited on pages 4, 45, 48).

Davison, Andrew J., Ian D. Reid, Nicholas Molton, and Olivier Stasse (2007). 'MonoSLAM: Real-Time Single Camera SLAM'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6, pp. 1052–1067. DOI: 10.1109/tpami.2007.1049 (cited on page 3).

Firman, Michael, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow (2016). 'Structured Prediction of Unobserved Voxels from a Single Depth Image'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2016.586 (cited on pages 4, 45, 48).

Frank, Marguerite and Philip Wolfe (1956). 'An algorithm for quadratic programming'. en. In: *Naval Research Logistics Quarterly* 3.1-2, pp. 95–110. DOI: 10.1002/nav.3800030109. (Visited on 02/01/2021) (cited on pages 72, 74).

Geilinger, Moritz, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros (2020). 'ADD: analytically differentiable dynamics for multi-body systems with frictional contact'. In: *ACM Transactions on Graphics* 39.6, pp. 1–15. DOI: 10.1145/3414685.3417766 (cited on pages 5, 63, 65, 97).

Girshick, Ross B. (2015). 'Fast R-CNN'. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, pp. 1440–1448. DOI: 10.1109/iccv.2015.169 (cited on page 3).

Girshick, Ross B., Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). 'Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation'. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 580–587. DOI: 10.1109/cvpr.2014.81 (cited on page 3).

Gropp, Amos, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman (July 2020). 'Implicit Geometric Regularization for Learning Shapes'. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 3789–3799 (cited on pages 15, 16, 72, 79).

Hachiuma, Ryo, Christian Pirchheim, Dieter Schmalstieg, and Hideo Saito (2019). 'DetectFusion: Detecting and Segmenting Both Known and Unknown Dynamic Objects in Real-time SLAM'. English. In: *Proceedings British Machine Vision Conference (BMVC)* (cited on pages 48, 53).

Hart, John C. (1996). 'Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces'. In: *The Visual Computer* 12.10, pp. 527–545. DOI: 10.1007/s003710050084 (cited on page 19).

Hartley, Richard and Andrew Zisserman (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press (cited on pages 1, 2, 21).

Hassan, Mohamed, Vasileios Choutas, Dimitrios Tzionas, and Michael Black (2019). 'Resolving 3D Human Pose Ambiguities With 3D Scene Constraints'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00237 (cited on page 48).

Hasson, Yana, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid (2019). 'Learning Joint Reconstruction of Hands and Manipulated Objects'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2019.01208 (cited on page 48).

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). 'Mask R-CNN'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 2980–2988. DOI: 10.1109/iccv.2017.322 (cited on pages 3, 4, 25, 29–31, 39, 53, 56, 96).

Hu, Yuanming, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand (Oct. 2020). 'DiffTaichi: Differentiable Programming for Physical Simulation'. In: *International Conf. on Learning Representations (ICLR)* (cited on pages 5, 63).

Hu, Yuanming, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik (2019). 'ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics'. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. DOI: 10.1109/icra.2019.8794333 (cited on page 65).

Irshad, Muhammad Zubair, Thomas Kollar, Michael Laskey, Kevin Stone, and Zsolt Kira (2022). 'CenterSnap: Single-Shot Multi-Object 3D Shape Reconstruction and Categorical 6D Pose and Size Estimation'. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. DOI: 10.1109/icra46639.2022.9811799 (cited on page 98).

Irshad, Muhammad Zubair, Sergey Zakharov, Rares Ambrus, Thomas Kollar, Zsolt Kira, and Adrien Gaidon (July 2022). 'ShAPO: Implicit Representations for Multi-object Shape, Appearance, and Pose Optimization'. en. In: *Lecture Notes in Computer Science*. Ed. by Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, pp. 275–292. DOI: 10.1007/978-3-031-20086-1_16 (cited on page 98).

Jaimez, Mariano, Christian Kerl, Javier Gonzalez-Jimenez, and Daniel Cremers (2017). 'Fast odometry and scene flow from RGB-D cameras based on geometric clustering'. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3992–3999. DOI: 10.1109/icra.2017.7989459 (cited on page 39).

Kandukuri, Rama Krishna, Jan Achterhold, Michael Moeller, and Joerg Stueckler (2020). 'Learning to Identify Physical Parameters from Video Using Differentiable Physics'. In: pp. 44–57. DOI: 10.1007/978-3-030-71278-5_4 (cited on page 66).

Kandukuri, Rama Krishna, Michael Strecke, and Joerg Stueckler (2024). 'Physics-Based Rigid Body Object Tracking and Friction Filtering From RGB-D Videos'. In: *International Conference on 3D Vision (3DV)*. accepted, preprint arXiv: 2309.15703. DOI: 10.1109/3DV62453.2024.00111 (cited on page 7).

Kato, Hiroharu, Yoshitaka Ushiku, and Tatsuya Harada (2018). 'Neural 3D Mesh Renderer'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE Computer Society, pp. 3907–3916. DOI: 10.1109/cvpr.2018.00411 (cited on pages 6, 64, 66).

Kazhdan, Michael, Matthew Bolitho, and Hugues Hoppe (2006). 'Poisson Surface Reconstruction'. In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, pp. 61–70 (cited on pages 5, 47, 49, 50).

Kazhdan, Michael and Hugues Hoppe (2013). 'Screened poisson surface reconstruction'. In: *ACM Transactions on Graphics* 32.3, pp. 1–13. DOI: 10.1145/2487228.2487237. (Visited on 05/16/2023) (cited on pages 5, 50).

Keller, Maik, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb (2013). 'Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion'. In: *2013 International Conference on 3D Vision*. IEEE, pp. 1–8. DOI: 10.1109/3dv.2013.9 (cited on pages 23, 25).

Kerl, Christian, Jurgen Sturm, and Daniel Cremers (2013). 'Dense visual SLAM for RGB-D cameras'. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2100–2106. DOI: 10.1109/iros.2013.6696650 (cited on pages 3, 23, 25).

Keselman, Leonid, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik (2017). 'Intel(R) RealSense(TM) Stereoscopic Depth Cameras'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. DOI: 10.1109/cvprw.2017.167 (cited on page 21).

Kirillov, Alexander, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick (Oct. 2023). 'Segment Anything'. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv51070.2023.00371 (cited on page 97).

Krishna Murthy, Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler (2021). 'gradSim: Differentiable simulation for system identification and visuomotor control'. In: *International Conference on Learning Representations (ICLR)* (cited on pages 5, 63, 66).

Levenberg, Kenneth (1944). 'A method for the solution of certain non-linear problems in least squares'. In: *Quarterly of Applied Mathematics* 2.2, pp. 164–168. DOI: 10.1090/qam/10666 (cited on pages 28, 35).

Lidec, Quentin Le, Igor Kalevatykh, Ivan Laptev, Cordelia Schmid, and Justin Carpentier (2021). 'Differentiable Simulation for Physical System Identification'. In: *IEEE Robotics and Automation Letters* 6.2, pp. 3413–3420. DOI: 10.1109/LRA.2021.3062323 (cited on page 66).

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick (2014). 'Microsoft COCO: Common Objects in Context'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48 (cited on page 38).

Lorensen, William E. and Harvey E. Cline (1987). 'Marching cubes: A high resolution 3D surface construction algorithm'. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*. ACM Press. DOI: 10.1145/37401.37422 (cited on pages 19, 57, 72, 74, 80).

Ma, Yi, Stefano Soatto, Jana Košecká, and S. Shankar Sastry (2004). *An Invitation to 3-D Vision*. Springer New York (cited on pages 9, 10, 12).

Macklin, Miles, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse (2020). 'Local Optimization for Robust Signed Distance Field Collision'. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3.1, pp. 1–17. DOI: 10.1145/3384538 (cited on pages 13, 72–74, 93, 98).

Macklin, Miles, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim (2020). 'Primal/Dual Descent Methods for Dynamics'. In: *Computer Graphics Forum* 39.8, pp. 89–100. DOI: 10.1111/cgf.14104 (cited on page 65).

Magnus, Jan R. and Heinz Neudecker (1988). *Matrix differential calculus with applications in statistics and econometrics*. English. Wiley Ser. Probab. Math. Stat. Chichester (UK) etc.: John Wiley & | Sons (cited on page 71).

Marquardt, Donald W. (1963). 'An Algorithm for Least-Squares Estimation of Nonlinear Parameters'. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2, pp. 431–441. DOI: 10.1137/0111030 (cited on pages 28, 35).

Mattingley, Jacob and Stephen Boyd (Mar. 2012). 'CVXGEN: a code generator for embedded convex optimization'. In: *Optimization and Engineering* 13.1, pp. 1–27. DOI: 10.1007/s11081-011-9176-9. (Visited on 05/09/2023) (cited on page 70).

McCormac, John, Ronald Clark, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger (2018). 'Fusion++: Volumetric Object-Level SLAM'. In: *2018 International Conference on 3D Vision (3DV)*. IEEE, pp. 32–41. DOI: 10.1109/3dv.2018.00015 (cited on pages 3, 25, 28–31).

Mescheder, Lars, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger (2019). 'Occupancy Networks: Learning 3D Reconstruction in Function Space'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2019.00459 (cited on pages 6, 14, 66).

Mirtich, Brian (1996). 'Fast and Accurate Computation of Polyhedral Mass Properties'. In: *Journal of Graphics Tools* 1.2, pp. 31–50. DOI: 10.1080/10867651.1996.10487458 (cited on page 74).

Mirtich, Brian and John Canny (1995). 'Impulse-based simulation of rigid bodies'. In: *Proceedings of the 1995 symposium on Interactive 3D graphics - SI3D '95*. ACM Press. DOI: 10.1145/199404.199436 (cited on page 65).

Müller, Matthias, Bruno Heidelberger, Marcus Hennix, and John Ratcliff (2007). 'Position based dynamics'. In: *J. Vis. Commun. Image Represent.* 18.2, pp. 109–118. DOI: 10.1016/j.jvcir.2007.01.005 (cited on page 65).

Mur-Artal, Raul, J. M. Martínez Montiel, and Juan D. Tardós (2015). 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System'. In: *IEEE Transactions on Robotics* 31.5, pp. 1147–1163. DOI: 10.1109/tro.2015.2463671 (cited on page 3).

Mur-Artal, Raul and Juan D. Tardós (2017). 'ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras'. In: *IEEE Transactions on Robotics* 33.5, pp. 1255–1262. DOI: 10.1109/tro.2017.2705103 (cited on page 3).

Newcombe, Richard A. (2012). 'Dense visual SLAM'. PhD thesis. Imperial College London, UK (cited on pages 17, 21, 27, 53).

Newcombe, Richard A., Andrew Fitzgibbon, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, and Steve Hodges (2011). 'KinectFusion: Real-time dense surface mapping and tracking'. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE. DOI: 10.1109/ismar.2011.6092378 (cited on pages 3, 4, 17–19, 23–25, 27, 28, 53).

Newman, Timothy S. and Hong Yi (2006). 'A survey of the marching cubes algorithm'. In: *Computers & Graphics* 30.5, pp. 854–879. DOI: 10.1016/j.cag.2006.07.021 (cited on page 19).

Nicastro, Andrea, Ronald Clark, and Stefan Leutenegger (2019). 'X-Section: Cross-Section Prediction for Enhanced RGB-D Fusion'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00160 (cited on page 48).

Nießner, Matthias, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger (2013). 'Real-time 3D reconstruction at scale using voxel hashing'. In: *ACM Transactions on Graphics* 32.6, pp. 1–11. DOI: 10.1145/2508363.2508374 (cited on page 25).

Oleynikova, Helen, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto (2017). 'Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. DOI: 10.1109/iros.2017.8202315 (cited on page 13).

Pan, Yue, Yves Kompis, Luca Bartolomei, Ruben Mascaro, Cyrill Stachniss, and Margarita Chli (2022). 'Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, pp. 5331–5338. DOI: 10.1109/IROS47612.2022.9981318 (cited on page 13).

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). 'DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2019.00025 (cited on pages 6, 14–16, 64, 66, 72, 79, 98).

Parker, Steven, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan (1998). 'Interactive ray tracing for isosurface rendering'. In: *Proceedings Visualization '98 (Cat. No.98CB36276)* (Proceedings Visualization '98 (Cat. No.98CB36276)). IEEE. DOI: 10.1109/visual.1998.745713 (cited on page 18).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 8024–8035 (cited on pages 93, 98).

Pollefeys, Marc, Reinhard Koch, Maarten Vergauwen, and Luc Van Gool (Oct. 1999). 'Hand-held acquisition of 3D models with a video camera'. In: *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*. Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062). IEEE Comput. Soc, pp. 14–23. DOI: 10.1109/im.1999.805330 (cited on page 2).

Qiao, Yi-Ling, Junbang Liang, Vladlen Koltun, and Ming C. Lin (July 4, 2020). 'Scalable Differentiable Physics for Learning and Control'. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)* (cited on page 65).

Remelli, Edoardo, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua (June 6, 2020). 'MeshSDF: Differentiable Iso-Surface Extraction'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33, pp. 22468–22478 (cited on pages 6, 20, 64, 66, 72, 74).

Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun (2017). 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, pp. 1137–1149. DOI: 10.1109/tpami.2016.2577031 (cited on page 3).

Rindler, Filip (2018). *Calculus of Variations*. Springer International Publishing (cited on page 49).

Rockafellar, Ralph Tyrell (1970). *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press (cited on page 49).

Rockafellar, Ralph Tyrrell and Roger J. B. Wets (1998). *Variational Analysis*. Vol. 317. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg (cited on page 49).

Rünz, Martin and Lourdes Agapito (2017). 'Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects'. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4471–4478. DOI: 10.1109/icra.2017.7989518 (cited on pages 3, 21, 23, 25, 36–42, 46, 48, 53, 57).

Rünz, Martin, Maud Buffier, and Lourdes Agapito (2018). 'MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects'. In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, pp. 10–20. DOI: 10.1109/ismar.2018.00024 (cited on pages 3, 23, 25, 38–40, 48, 53).

Rusinkiewicz, Szymon, Olaf Hall-Holt, and Marc Levoy (2002). 'Real-time 3D model acquisition'. In: *ACM Transactions on Graphics* 21.3, pp. 438–446. DOI: 10.1145/566654.566600 (cited on page 21).

Salas-Moreno, Renato F., Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly, and Andrew J. Davison (2013). 'SLAM++: Simultaneous Localisation and Mapping at the Level of Objects'. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1352–1359. DOI: 10.1109/cvpr.2013.178 (cited on page 25).

Sarbolandi, Hamed, Damien Lefloch, and Andreas Kolb (2015). 'Kinect range sensing: Structured-light versus Time-of-Flight Kinect'. In: *Computer Vision and Image Understanding* 139, pp. 1–20. DOI: 10.1016/j.cviu.2015.05.006. (Visited on 05/24/2023) (cited on page 21).

Schmidt, Tanner, Katharina Hertkorn, Richard Newcombe, Zoltan Marton, Michael Suppa, and Dieter Fox (2015). 'Depth-based tracking with physical constraints for robot manipulation'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 1050-4729. IEEE, pp. 119–126. DOI: 10.1109/icra.2015.7138989 (cited on page 25).

Schroers, Christopher, Simon Setzer, and Joachim Weickert (2014). 'A Variational Taxonomy for Surface Reconstruction from Oriented Points'. In: *Computer Graphics Forum* 33.5, pp. 195–204. DOI: 10.1111/cgf.12445 (cited on pages 5, 46, 47, 49–52, 54, 55, 60, 98).

Scona, Raluca, Mariano Jaimez, Yvan R. Petillot, Maurice Fallon, and Daniel Cremers (2018). 'StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments'. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1–9. DOI: 10.1109/icra.2018.8460681 (cited on pages 23, 25, 39).

Shao, Han, Tassilo Kugelstadt, Torsten Hädrich, Wojciech Pałubicki, Jan Bender, Sören Pirk, and Dominik L. Michels (June 2021). 'Accurately Solving Rod Dynamics with Graph Learning'. In: *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 93).

Shen, Chen, James F. O'Brien, and Jonathan R. Shewchuk (2004). 'Interpolating and approximating implicit surfaces from polygon soup'. In: *ACM Transactions on Graphics* 23.3, p. 896. DOI: 10.1145/1015706.1015816 (cited on pages 47, 50).

Sheth, Bhavin R. and Ryan Young (2016). 'Two Visual Pathways in Primates Based on Sampling of Space: Exploitation and Exploration of Visual Information'. In: *Frontiers in Integrative Neuroscience* 10. DOI: 10.3389/fnint.2016.00037. (Visited on 05/19/2023) (cited on page 1).

Sommer, Christiane, Lu Sang, David Schubert, and Daniel Cremers (2022). 'Gradient-SDF: A Semi-Implicit Surface Representation for 3D Reconstruction'. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr52688.2022.00618 (cited on page 17).

Song, Shuran, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser (2017). 'Semantic Scene Completion from a Single Depth Image'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2017.28 (cited on pages 4, 45, 48).

Stewart, David E. and Jeffrey C. Trinkle (1996). ‘An Implicit Time-Stepping Scheme For Rigid Body Dynamics With Inelastic Collisions and Coulomb friction’. In: *International Journal for Numerical Methods in Engineering* 39.15, pp. 2673–2691. DOI: 10.1002/(sici)1097-0207(19960815)39:15<2673::aid-nme972>3.0.co;2-i (cited on pages 64, 65).

Strecke, Michael and Joerg Stueckler (2019). ‘EM-Fusion: Dynamic Object-Level SLAM With Probabilistic Data Association’. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2019.00596 (cited on pages 3, 5–7, 23, 46, 48, 53, 56).

– (2020). ‘Where Does It End? - Reasoning About Hidden Surfaces by Object Intersection Constraints’. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr42600.2020.00961 (cited on pages 4, 6, 7, 45).

– (2021). ‘DiffSDFSim: Differentiable Rigid-Body Dynamics With Implicit Shapes’. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. DOI: 10.1109/3dv53792.2021.00020 (cited on pages 5–7, 63).

Stückler, Jörg and Sven Behnke (2013). ‘Hierarchical Object Discovery and Dense Modelling from Motion Cues in RGB-D Video’. In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI ’13. Beijing, China: AAAI Press, pp. 2502–2509 (cited on page 25).

– (2015). ‘Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video’. In: *International Journal of Computer Vision (IJCV)* 113.3, pp. 233–245. DOI: 10.1007/s11263-014-0796-3 (cited on page 25).

Sturm, Jürgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers (2012). ‘A benchmark for the evaluation of RGB-D SLAM systems’. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 573–580. DOI: 10.1109/iros.2012.6385773 (cited on pages 21, 37, 39).

Stutz, David and Andreas Geiger (2018). ‘Learning 3D Shape Completion from Laser Scan Data with Weak Supervision’. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2018.00209 (cited on page 57).

Sulsky, Deborah, Shi-Jian Zhou, and Howard L. Schreyer (1995). ‘Application of a particle-in-cell method to solid mechanics’. In: *Computer Physics Communications* 87.1, pp. 236–252. DOI: 10.1016/0010-4655(94)00170-7 (cited on page 65).

Szeliski, Richard (2011). *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer (cited on page 1).

Taylor, Jonathan, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, Jamie Shotton, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, and Julien Valentin (2016). ‘Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences’. In: *ACM Transactions on Graphics* 35.4, pp. 1–12. DOI: 10.1145/2897824.2925965 (cited on page 25).

Terzopoulos, Demetri and Kurt Fleischer (1988). ‘Modeling inelastic deformation’. In: *Proceedings of the 15th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’88*. ACM Press. DOI: 10.1145/54852.378522 (cited on page 65).

Tian, Meng, Marcelo H. Ang, and Gim Hee Lee (2020). ‘Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation’. In: *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 530–546. DOI: 10.1007/978-3-030-58589-1_32 (cited on page 98).

Tokmakov, Pavel, Cordelia Schmid, and Karteek Alahari (2018). ‘Learning to Segment Moving Objects’. In: *International Journal of Computer Vision* 127.3, pp. 282–301. DOI: 10.1007/s11263-018-1122-2 (cited on page 97).

Tomasi, Carlo and Takeo Kanade (1992). ‘Shape and motion from image streams under orthography: a factorization method’. In: *International Journal of Computer Vision* 9.2, pp. 137–154. DOI: 10.1007/bf00129684. (Visited on 05/12/2023) (cited on page 2).

Tzionas, Dimitrios and Juergen Gall (2016). ‘Reconstructing Articulated Rigged Models from RGB-D Videos’. In: *Lecture Notes in Computer Science*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, pp. 620–633. DOI: 10.1007/978-3-319-49409-8_53 (cited on page 25).

Ummenhofer, Benjamin and Thomas Brox (2015). ‘Global, Dense Multiscale Reconstruction for a Billion Points’. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2015.158 (cited on page 47).

Wang, Chen, Roberto Martin-Martin, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu (2020). ‘6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints’. In: *2020 IEEE International*

Conference on Robotics and Automation (ICRA). IEEE. doi: 10.1109/icra40945.2020.9196679 (cited on page 98).

Wang, He, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas (2019). 'Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 2642–2651. doi: 10.1109/cvpr.2019.00275 (cited on page 98).

Wang, Xinlong, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M. Alvarez (2022). 'FreeSOLO: Learning to Segment Objects without Annotations'. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 14156–14166. doi: 10.1109/cvpr52688.2022.01378 (cited on page 97).

Weickert, Joachim, Sven Grewenig, Christopher Schroers, and Andrés Bruhn (2015). 'Cyclic Schemes for PDE-Based Image Analysis'. In: *International Journal of Computer Vision* 118.3, pp. 275–299. doi: 10.1007/s11263-015-0874-1 (cited on pages 52, 55, 57).

Weinstein, Rachel, Joseph Teran, and Ron Fedkiw (May 2006). 'Dynamic simulation of articulated rigid bodies with contact and collision'. In: *IEEE Transactions on Visualization and Computer Graphics* 12, pp. 365–74. doi: 10.1109/TVCG.2006.48 (cited on page 65).

Weiss, Sebastian, Robert Maier, Daniel Cremers, Rudiger Westermann, and Nils Thuerey (2020). 'Correspondence-Free Material Reconstruction using Sparse Surface Constraints'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 4685–4694. doi: 10.1109/cvpr42600.2020.00474 (cited on page 66).

Wen, Bowen and Kostas Bekris (2021). 'BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models'. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. doi: 10.1109/iros51168.2021.9635991 (cited on page 98).

Whelan, Thomas, Michael Kaess, Maurice F. Fallon, Hordur Johannsson, John J. Leonard, and John B. McDonald (July 2012). 'Kintinuous: Spatially Extended KinectFusion'. In: *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Sydney, Australia (cited on pages 25, 37, 38).

Whelan, Thomas, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison (2015). 'ElasticFusion: Dense SLAM Without A Pose Graph'. In: *Robotics: Science and Systems XI*. Rome, Italy: Robotics: Science and Systems Foundation. doi: 10.15607/rss.2015.xi.001 (cited on pages 25, 38).

Wu, Jiajun, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman (2016). 'Physics 101: Learning physical object properties from unlabeled videos'. In: *Proceedings of the British Machine Vision Conference (BMVC)*. doi: 10.5244/c.30.39 (cited on page 66).

Wu, Jiajun, Erika Lu, Pushmeet Kohli, William T Freeman, and Joshua B Tenenbaum (2017). 'Learning to See Physics via Visual De-animation'. In: *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 66).

Wu, Jiajun, Ilker Yildirim, Joseph J Lim, William T Freeman, and Joshua B Tenenbaum (2015). 'Galileo: Perceiving physical object properties by integrating a physics engine with deep learning'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., pp. 127–135 (cited on page 66).

Xie, Junyu, Weidi Xie, and Andrew Zisserman (2022). 'Segmenting Moving Objects via an Object-Centric Layered Representation'. In: *NeurIPS* (cited on page 97).

Xu, Binbin, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger (2019). 'MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM'. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. doi: 10.1109/icra.2019.8794371 (cited on pages 3, 23, 25, 38, 39, 48, 53).

Xu, Jie, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal (July 15, 2021). 'An End-to-End Differentiable Framework for Contact-Aware Robot Design'. In: *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation. doi: 10.15607/rss.2021.xvii.008 (cited on page 97).

Xu, Jie, Sangwoon Kim, Tao Chen, Alberto Rodriguez Garcia, Pulkit Agrawal, Wojciech Matusik, and Shinjiro Sueda (2022). 'Efficient Tactile Simulation with Differentiability for Robotic Manipulation'. In: *6th Annual Conference on Robot Learning* (cited on page 97).

Yang, Bo, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen (2019). ‘Dense 3D Object Reconstruction from a Single Depth View’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.12, pp. 2820–2834. doi: 10.1109/tpami.2018.2868195 (cited on pages 4, 45, 48).