TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Cable Manipulation by a Mobile Robot

DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
David Schweighofer
Registration Number 11771131

Vienna, September 2025

Vision for Robotics Group
A-1040 Wien, Gusshausstr. 27, Internet: http://www.acin.tuwien.ac.at

Acknowledgements

I would like to express my sincere gratitude to Jean-Baptiste Weibel for generously sharing his expertise and guiding me through the technical challenges of this work. I am deeply thankful to Markus Vincze for giving me the opportunity to carry out this thesis at ACIN and for providing valuable feedback and reviews throughout the process. My special thanks go to Daniel Zimmer for his continuous emotional support during this project, as well as for his technical assistance with Sasha. Finally, I am grateful to my father and mother for their unwavering encouragement and support throughout this thesis.

Portions of the writing were refined with the help of language-editing tools (ChatGPT and Paperpal). These tools were used for rephrasing and style suggestions only; the technical content, analyses, and conclusions are my own. I am responsible for all remaining errors.

Abstract

Grasping transparent and flexible objects remains a significant challenge in robotics due to their poor visibility in conventional RGB or depth sensing. This thesis addresses the problem of reliably grasping a transparent tube using a mobile, eye-in-hand robot. We propose a novel pipeline that reconstructs the 3D shape of the tube and determines a suitable grasp pose using only the robot's hand-mounted RGB camera in combination with its known poses.

Our method first combines Depth Anything v2 for monocular depth estimation with Grounded Segment Anything 2 for robust tube segmentation to generate an initial 3D B-spline representation. As the robot approaches the object, the pose of the B-spline is iteratively refined using multiview observations from the moving camera. The resulting 3D reconstruction enables the computation of a grasp pose that is executed on a physical robot.

Experimental evaluations in real-world scenarios demonstrated that the proposed approach can accurately localize and successfully grasp transparent tubes without the need for additional depth sensors or specialized hardware. These results demonstrate that integrating state-of-the-art monocular depth estimation and segmentation techniques can effectively address the long-standing challenge of transparent-object manipulation.

Kurzzusammenfassung

Das Greifen transparenter und flexibler Objekte stellt aufgrund ihrer schlechten Sichtbarkeit in herkömmlichen RGB- oder Tiefensensoren nach wie vor eine große Herausforderung in der Robotik dar. Diese Arbeit befasst sich mit dem Problem des zuverlässigen Greifens eines transparenten Schlauches mit einem mobilen Eye-in-Hand-Roboter. Wir schlagen eine neuartige Methode vor, die die 3D-Form des Schlauches rekonstruiert und eine geeignete Greifpose bestimmt, wobei nur die an der Hand des Roboters montierte RGB-Kamera in Kombination mit ihrer bekannten Pose verwendet wird.

Unsere Methode kombiniert zunächst Depth Anything v2 für die monokulare Tiefenschätzung mit Grounded Segment Anything 2 für eine robuste Segmentierung, um eine erste 3D B-Spline-Darstellung zu erzeugen. Wenn sich der Roboter dem Objekt nähert, wird die Form und Pose der B-Spline anhand von mehreren Ansichten der beweglichen Kamera iterativ verfeinert. Die resultierende 3D-Rekonstruktion ermöglicht die Berechnung einer stabilen Greifpose, die auf einem physischen Roboter ausgeführt wird.

Experimentelle Bewertungen in realen Szenarien haben gezeigt, dass der vorgeschlagene Ansatz transparente Röhren ohne zusätzliche Tiefensensoren oder spezielle Hardware genau lokalisieren und erfolgreich greifen kann. Diese Ergebnisse zeigen, dass die Integration modernster monokularer Tiefenschätzung und Segmentierungstechniken, die seit langem bestehende Herausforderung der Manipulation transparenter Objekte effektiv bewältigen kann.

Contents

List of Figures

List of Tables

1 Introduction

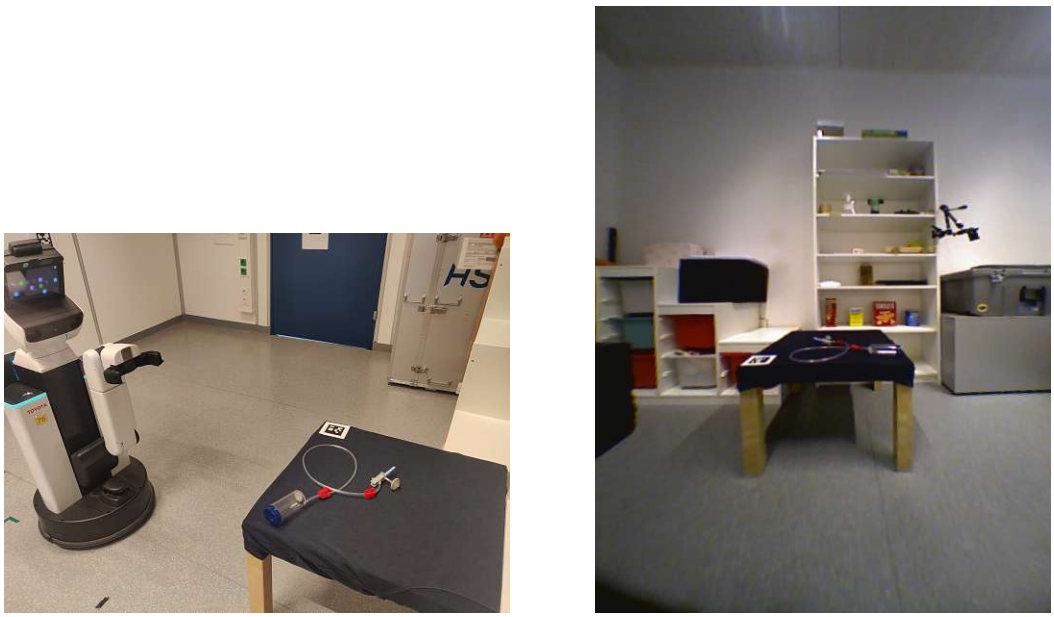
The rapid growth of mobile robots in homes, hospitals, and service or industrial settings marks a new phase in robotics: machines are moving beyond the structured factory floor into messy real-world spaces. In these locations, robots must recognize not only rigid, clearly shaped items but also deformable linear objects (DLOs), such as thin, flexible cables, hoses, and transparent plastic tubes. Handling these objects is important because they are common in everyday tasks and professional workspaces. A robot that can manage tubes can assist nurses by preparing or clearing IV lines, supporting older or mobility-impaired people with cable handling at home, and improving productivity in industrial wiring or hose routing. By taking over these tasks, robots can reduce the physical workload and repetitive strain on humans while allowing them to focus on higher-value tasks or more demanding care work. Such capabilities also enable safer work in hazardous or hygiene-critical areas, such as chemical laboratories or hospital isolation rooms. Overall, reliable tube handling by mobile robots offers clear social and economic benefits. This improves efficiency, lowers operational costs, and increases the independence and usefulness of service robots outside traditional factories.

1.1 Challenge

Grasping transparent tubes is difficult for three main reasons.

Flexible shape. A tube is a deformable object with no fixed configuration. Each time it is placed, it may bend, loop, or coil in a new way. This variability means that a tube rarely appears the same between two scenes or even between two camera viewpoints in the same scene. Standard object detectors and pose estimation methods rely on the assumptions of rigidity and a predictable geometric model. When the tube changes its configuration, template-based detection fails, and matching the 3D points becomes unreliable. In practice, a robot cannot depend on a single representation to recognize a tube, which complicates grasp planning and tracking over time.

No Texture. Transparent tubes have smooth and shiny surfaces with almost no visible texture. This lack of distinctive surface features prevents classical feature detectors, such as SIFT [1] and ORB [2], from identifying reliable keypoints. Without these keypoints, multi-view matching is extremely challenging because there are no correspondences to establish the depth or alignment between views. Even modern learning-based methods can fail because the visual signal from a clear tube is weak and dominated by background or reflections. Consequently, stereo reconstruction, structure-from-motion, and SLAM methods [3] struggle to recover the 3D shape of the tube.



(a) (b)

Figure 1.1: Problem scenario illustrating the grasping challenge. (a) Workspace photograph with the deformable linear object (DLO) laid out for manipulation. (b) Corresponding view from the robot-mounted camera, showing the low-texture, translucent appearance.

Transparency. The appearance of a tube is highly dependent on its environment and lighting conditions. It often takes on the color of the background, may partially disappear in complex scenes, and produces highlights or glare where light is reflected. This variability renders the simple color-based segmentation unreliable. Moreover, changes in illumination, shadows, and background textures can lead to large appearance shifts across frames, further complicating vision algorithms. Transparent materials also refract and reflect incoming light, creating ambiguous edges and ghost images that can mislead both traditional detectors and deep neural networks. Depth sensors also struggle with transparency because light can pass straight through, refract out of the field of view, or return as bright specular highlights that create invalid or missing depth data. This makes 3D reconstruction from RGB-D sensors unreliable in practical applications. Together, these effects make consistent detection in real-world environments difficult without careful control of lighting or the addition of artificial markers on the target.

Because these effects remove the usual cues for geometry and appearance, many research systems add color markers or work in tightly controlled lighting, which is impractical in homes, hospitals, and workshops. This explains why grasping transparent tubes remains largely unsolved in uncontrolled real-world environments.

Additionally, rigid components or connectors that are not part of the DLO are present at both ends of many tubes. These attachments complicate detection and grasp planning because they change the apparent geometry of the tube, add extra edges, and cause reflections or partial occlusions of the flexible segment. This increases the overall complexity

of perception and manipulation.

1.2 Approach

In this thesis, we present a vision-guided, eye-in-hand approach that enables a mobile robot to reconstruct the 3D configuration of a transparent tube and subsequently grasp it. We rely solely on the RGB camera mounted on the robot's gripper, together with its pose information for sensing. Multiple images of the scene are captured sequentially from different viewpoints. After each image is captured, the tube estimation is refined, and a temporary grasp pose is determined. A control loop moves the gripper in a discrete step towards the temporary grasp pose before the next image is captured. An overview of this iterative workflow is provided in Fig. 1.2.

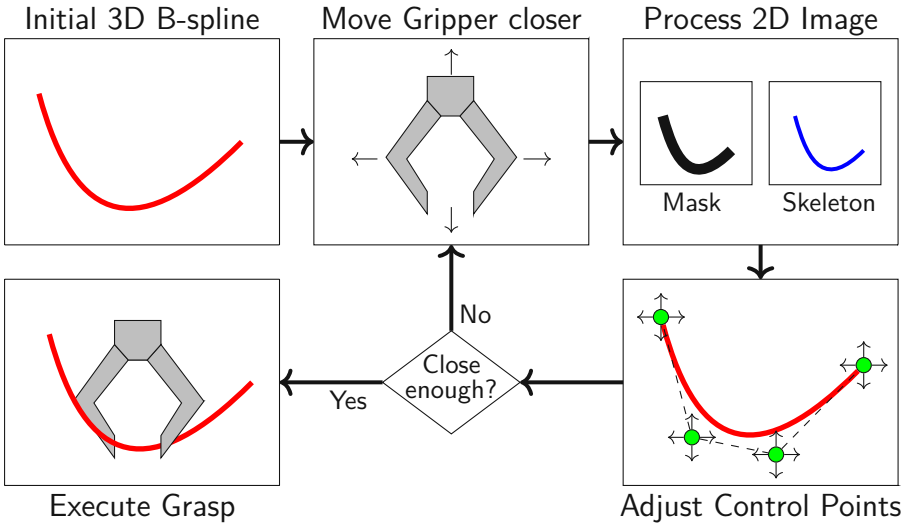


Figure 1.2: High-level block diagram of the proposed pipeline. The process begins with an initial 3D B-spline (▬). A grasp point is selected on the spline, and the gripper is advanced in a discrete step toward this target. A new RGB image is captured by the eye-in-hand camera and processed to obtain a tube mask (▬) and its skeleton (▬). A nonlinear optimizer updates the 3D B-spline's control points (●) such that the projected spline aligns with all skeletons observed to date. With the refined B-spline, the cycle of motion and refinement is repeated until the final grasp is executed.

Initial estimation. Starting from the initial pose (see Fig. 1.1), we acquire an initial image and first submit it to Depth Anything v2 [4] to generate a relative depth map. The resulting depth map is then cropped using the masked tube generated from Grounded Segment Anything 2 (Grounded SAM 2) [5] to isolate the tube region. Next, the camera is moved to a predefined offset pose, and a second image is captured and processed with Grounded SAM 2 to obtain the tube mask from the new view. Because the depth map is only defined up to an unknown scale and shift, we convert it into a 3D point cloud and

project it onto the image plane of the second view. We then adjust the scale and shift until the projected points align with the second-view mask, producing an optimal depth map and a 3D point cloud of the tube. Finally, we fit a 3D B-spline to this optimal point cloud to form the initial 3D representation of the DLO.

Grasp point and control law Using the initial 3D B-spline, we take its parameter midpoint as a stable intermediate grasp target. The gripper advances a fixed fraction toward this point under a linear position-error controller, and a new RGB image is then captured from the updated viewpoint to drive the next refinement cycle.

Refining the B-spline. We skeletonize the DLO masks from each view to obtain the 2D centerlines. The 3D B-spline is projected onto each image plane and compared with their skeletons. A nonlinear solver adjusts the B-spline control points to satisfy all views. After this refinement step is complete, we select a temporary grasp point and move the gripper as described in the previous paragraph. A new image is then captured from the updated position, and the process is repeated iteratively as the camera moves closer to the tube.

Grasp execution. After the refinement-and-movement cycle was completed four times and the estimated B-spline was sufficiently accurate, the gripper moves to the grasp pose and closes the fingers to grasp the tube.

In summary, our method incrementally refines the 3D B-spline of the tube as the camera advances toward the object, integrating new views at each step. After four refinement cycles, the robot executes the grasp by moving to the planned grasp pose and closing the fingers to secure the DLO.

1.3 Results Preview

The experiments demonstrate both the potential and current limitations of the proposed method. Across 42 trials with three types of deformable linear objects (DLOs) and multiple table textures, the overall grasp success rate was 64%. The reconstruction accuracy, measured as the deviation between the estimated and reference B-splines, showed median errors in the range of 15 mm–22 mm.

Under stable conditions with repeated experiments of a cable on a black surface, the grasp success rate increased to 70%, indicating that the pipeline can achieve a more reliable performance when segmentation is consistent and viewpoint variation is limited. Nevertheless, this method remains sensitive to incomplete initial detections and tends to shorten the reconstructed spline during refinement.

In summary, the results confirm that incremental refinement of a 3D B-spline enables successful grasps of transparent tubes under certain conditions but also highlights the need for improvements in sensing accuracy, regularization, and view selection to achieve robust real-world deployment.

1.4 Thesis Outline

In Chapter 2, we provide a comprehensive review of state-of-the-art methods for detecting and grasping deformable linear objects (DLOs). We examine existing methods, highlight successful approaches, identify their limitations, and situate our contribution within the current research landscape.

Chapter 3 presents the theoretical foundations and existing algorithms that form the basis of the proposed method. It introduces relevant concepts in vision-based perception and B-spline modeling, ensuring that the reader has the necessary background to understand the proposed pipeline.

In Chapter 4, we describe our complete pipeline for reconstructing and grasping a DLO using multiple sequential 2D images. We describe a step-by-step process, from image acquisition and segmentation to B-spline refinement, motion control, and grasp execution.

Finally, Chapter 5 presents a thorough evaluation of the proposed method in various experimental scenarios. We quantify the performance under different conditions, analyze the robustness of the approach, and discuss the results in the context of prior work.

2 Related Work

This chapter surveys methods that enable grasping of transparent, deformable linear objects (DLOs) with an eye-in-hand camera. The core challenges are twofold: perception is brittle because tubes are thin, weakly textured, and often transparent (commodity RGB-D depth is missing or corrupted). Control must cope with non-rigid geometry whose appearance changes with viewpoint and lighting. We organize prior work by how each line of research addresses these failure modes and by how its outputs can be consumed in a grasping pipeline: (i) transparent-object perception that repairs or replaces unreliable depth, (ii) 2D DLO detection that yields masks and ordered centerlines, (iii) 3D DLO reconstruction that recovers a curve model for planning, and (iv) eye-in-hand visual servoing during the final approach.

2.1 Transparent Objects

Transparent and specular objects violate the sensing assumptions of commodity RGB-D cameras, which breaks depth-based grasp stacks. One pragmatic response is single-view depth repair. **ClearGrasp** [6] predicts transparent masks, surface normals, and occlusion *and* contact boundaries from a single RGB-D frame and combines them in a global optimization that replaces the sensor's depth only over glassy regions. Plugging this refined depth into an existing grasp network yields large gains in *grasp success rate* without retraining. Reported improvements for the gripper types are: suction (64%→86%) and parallel-jaw (12%→72%). **ClueDepth Grasp** [7] preserves valid "clue" points (geometrically filtered raw depth within a reflection-angle threshold) and completes the rest end-to-end with a DenseFormer (DenseNet+Swin) multi-modal U-Net that fuses RGB, ClueDepth, surface normals, and occlusion boundaries, achieving state-of-the-art (SOTA) accuracy on the ClearGrasp benchmarks and improved robot grasping.

A second line uses multi-view or light-field sensing to sidestep unreliable single-view depth. **GlassLoc** [8] mounts a plenoptic (light-field) camera—a camera with a micro-lens array that captures many slightly different sub-aperture views across the main aperture—and builds a depth-likelihood volume with reflection suppression from those sub-aperture images, then classifies graspable locations directly. They report 81% success over 220 grasps. **MVTrans** [9] discards the sensor depth channel and jointly infers depth, instance segmentation, and 6-DoF pose from multi-view RGB via a plane-sweep matching volume and Feature Pyramid Network (FPN) heads that exploit multi-scale features.

Recently, NeRF-based perception has connected difficult transparent visuals to graspable geometry. Neural Radiance Fields (NeRFs) are neural scene models that render images from novel viewpoints by integrating density and color along rays. **Dex-NeRF** [10] trains a per-scene NeRF from many calibrated images (small camera grids, not truly sparse) and

renders a transparency-aware depth by taking the first ray sample whose NeRF density σ exceeds a threshold m (a tunable cutoff that treats the first sufficiently dense sample as the surface). The resulting depth feeds Dex-Net and achieves very high grasp success on real glassware. Lighting is augmented to produce strong specular cues that help training. **GraspNeRF** [11] instead learns a generalizable NeRF and a 6-DoF grasp head that, from about six calibrated RGB views, predicts a truncated signed distance field (TSDF) volume and grasp quality, orientation, and width at roughly 11 FPS, transferring across materials without per-scene optimization.

When vision alone is brittle, multi-modal strategies help. **Weng et al.** [12] transfer supervision from a depth-based grasp model to train RGB(-D) grasp scorers without grasp labels, with early and late fusion variants improving grasping on transparent and specular objects. **Li et al.** [13] couple an RGB grasp CNN (TGCNN, Gaussian-mask labeling) with tactile calibration and fusion modules—tactile height sensing (THS) and tactile position exploration (TPE)—to lift success in cluttered, irregular, and even underwater scenes.

Takeaway. Drop-in depth repair (ClearGrasp, ClueDepth) is the lowest-friction fix. Multi-view (GlassLoc, MVTrans) and NeRF approaches (Dex-NeRF, GraspNeRF) offer stronger geometry when short capture sequences are feasible. Multi-modal fusion is a safety net in visually degenerate scenes.

2.2 2D Reconstruction of DLOs

For grasping and reconstruction, 2D methods must output ordered centerlines that survive clutter, crossings, and weak texture.

FASTDLO [14] segments DLO pixels with DeepLabV3+ (a modern encoder–decoder semantic segmentation network), skeletonizes the mask, handles intersection pixels, scores endpoint-pair connections with a shallow similarity network using local color, thickness, and direction features, and merges pairs to form per-instance centerlines. A simple RGB-variation heuristic gives top/bottom ordering at crossings. The system runs in real time and outperforms prior graph-walk methods (for example, Ariadne+ [15]) and general instance-segmentation baselines on their benchmarks.

mBEST [16] performs topology cleanup and then resolves crossings by selecting the continuation that minimizes bending energy. This yields robust centerlines in tangled scenes at 30–50 FPS. A blurred-RGB standard deviation heuristic estimates crossing order.

RT-DLO [17] builds a graph over distance-transform maxima instead of walking the skeleton. Vertices are farthest-point sampled peaks with local orientation from a light CNN. Edges are selected from the k-nearest neighbors (KNN) and scored by cosine similarities (vertex–vertex and vertex–edge) with length-aware weighting. Intersections are solved in local subgraphs and crossing order again via RGB standard deviation. The method is both faster and more accurate than FASTDLO and prior graph-walk methods.

DLOFTBs [18] focuses on speed and generality without learning. It opens and skeletonizes each mask, greedily orders segments by endpoint distance and orientation, and fits (re-fits)

a B-spline per frame rather than maintaining an incremental temporal update rule. This achieves tens of milliseconds per frame while handling occlusions and self-intersections.

Cross-visual-field route estimation. For large or branched wiring harnesses, **CVF-DLO**[19] segments and skeletonizes each view, resolves crossings and bifurcations with an endpoint cost combining HSV color, direction, and curvature, projects pixel paths onto known surfaces using camera poses, fits quasi-uniform cubic B-splines, and stitches routes across overlapping fields of view using a depth-first search (DFS) over a path graph.

2.3 3D Reconstruction of DLOs

DLO reconstruction aims to recover a 3D centerline (and optionally width) under sparse or unreliable sensing.

Multi-view RGB Eye-in-hand approaches lift per-view 2D splines or masks from a small set of calibrated views to a consistent 3D spline, then plan grasps on that curve. **Caporali et al.** [20] run ARIADNE-style 2D B-splines per view and triangulate via least-squares ray intersection to form a 3D spline in a static scene. They report that about five views can suffice, though many runs use 15–20. **DLO3DS** [21] improves the pipeline with epipolar matching of spline samples, online view-planning (baseline and distance optimization), sliding-window stitching with overlap, and LOWESS (LOcally WEighted Scatterplot Smoothing). It reports sub-millimeter mean error on synthetic cases and real demos including semi-transparent hoses, with triangulation taking roughly 7–27 ms.

Single RGB-D frame A robust pipeline by **Sun et al.** [22] segments with Grounding-DINO+SAM, applies mask clean-up by size, skeleton topology, and width checks, performs 3D completion by K-means over the closest-depth quartile to reduce spurious depth outliers ("flying pixels"), bridges gaps with quadratic Bézier curves, fits a B-spline and downsamples to about 60 keypoints, and finally applies a Discrete Elastic Rod smoother that emphasizes bending. It is accurate but not real-time, with segmentation dominating runtime at roughly 10 s for 1080p. **Zhu et al.** [23] segment the point cloud with PointSIFT+PointNet++ (point-cloud neural networks for semantic segmentation), split it via adaptive K-means with an automatically chosen K, fit local cylinders, take the cylinder axes' centers, and order them using a principal component analysis (PCA) bounding box plus an octree-style directional constraint before interpolation. They report less than 1 mm average centerline error and sub-second reconstruction.

Learning a 3D state from a single point cloud **Lv et al.** [24] assume a pre-segmented DLO point cloud and learn two branches (global end-to-end regression and local point-wise voting) fused by a modified Coherent Point Drift (CPD) registration with known correspondences gated by a visibility heatmap. This preserves uniform spacing and is robust to 0–60% occlusion at about 34 ms per frame.

Model and sensor estimators When vision is weak or occluded, model-based estimators integrate physics with internal robot sensing such as joint and force measurements. **Nakagawa & Mochiyama** [25] recover the 3D shape of an elastic rod in real time from a wrist force/torque sensor and end-effector orientation only, using a discretized Kirchhoff-rod model with linear-time recursion. Assumptions include fixed length, known stiffness, and no unknown external contacts.

Differentiable multi-view tracking **DLO-Splatting** [26] closes the loop by optimizing a rope-like 3D representation directly against images. A prediction–update filter uses position-based dynamics (Verlet, length constraints, gravity, surface normal reaction, friction) to predict, and a 3D Gaussian-splatting rendering loss over multi-view RGB with known poses to update (spherical Gaussians sized by rope diameter). It currently runs at about 1 Hz and improves tip tracking in knot-tying, but struggles with heavy self-occlusions and topology changes.

2.4 Eye-in-Hand Visual Servoing

Visual servoing (VS) regulates the end-effector directly from visual errors. Foundational surveys define the taxonomy: image-based VS (IBVS) drives 2D features with an interaction matrix, while pose-based VS (PBVS) regulates a 6-DoF pose estimate. Most practical systems adopt a dynamic look-and-move architecture [27]–[29]. IBVS is generally less sensitive to calibration when driving image error to zero but exhibits coupling effects (for example, "retreat" on pure rotations). PBVS yields straight camera trajectories under perfect pose but is more sensitive to calibration and pose errors [28]. The companion survey *Visual Servo Control, Part II* [29] reviews advanced schemes: estimating required 3D quantities via epipolar or homography relations, or directly updating the interaction matrix (for example, with a Broyden update), hybrid 2½-D controllers that decouple translation and rotation using features such as $(x, \log Z)$, and partitioned IBVS using features like (α,σ) or (ρ,θ) to reduce coupling. It also discusses joint-space VS and Lyapunov-based switching between IBVS and PBVS. Moments-based IBVS [30] selects image-moment features whose interaction matrix is near block-triangular (constant translational block via normalization), with a virtual camera rotation to handle non-parallel desired poses. Dense and discrete objects lead to different interaction matrices, and the rotational invariants are chosen per object.

Learning has expanded VS. **Levine et al.** [31] learn closed-loop grasping directly from monocular RGB, training a CNN to score candidate task-space motions at each control cycle from about 1.7 M autonomous grasp attempts across many robots. The policy requires no precise hand–eye calibration and outperforms open-loop and depth-plus-calibration baselines. **EARL** [32] combines an eye-in-hand RGB-D pose-tracking frontend (Transformer tracker → Alpha-Refine → R2D2 → BundleTrack) with Proximal Policy Optimization (PPO) control that outputs joint velocities and triggers a grasp from a pool generated by Contact-GraspNet. It achieves roughly 15 FPS and strong sim-to-real dynamic grasping without explicit motion prediction.

3 Background

This chapter provides the theoretical foundation for the methods employed in this study. It introduces key concepts and techniques from computer vision and robotics, which are essential for the reconstruction and grasping of transparent deformable linear objects (DLOs). Understanding these principles is crucial for comprehending the design choices and implementation details of the proposed approach.

3.1 Open-Set Object Detection/Segmentation

Traditional detectors are limited to a closed list of classes, making them brittle when novel objects appear. Vision–language models address this by unifying text prompts with image features, allowing *open-set* (open-vocabulary) detection and segmentation. A user can simply ask for "the transparent silicone tube," even if that label never existed during training. Grounded Segment Anything 2 (Grounded SAM 2) [5] exemplifies this paradigm, combining powerful text-conditioned grounding with the high-resolution masks of the Segment Anything decoder. The following section outlines Grounded SAM 2's architecture and explains why it underpins our perception pipeline.

Grounded SAM 2 is a state-of-the-art model for object detection, tracking, and segmentation that extends the capabilities of its predecessor, Grounded SAM. By combining vision-language grounding with image segmentation, precise segmentation masks can be generated based on natural language descriptions.

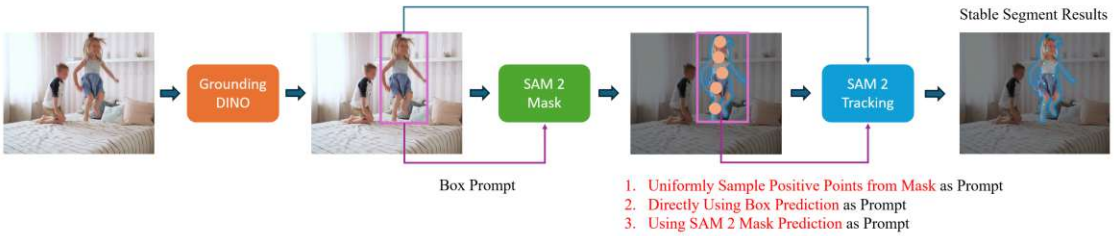


Figure 3.1: Grounded Segment Anything 2 (Grounded SAM 2) pipeline[1]. An input image and a natural language prompt are first processed by Grounding DINO (DINO: Self-Distillation with No Labels), which performs *grounding*—the alignment of text queries with corresponding visual regions—to generate a bounding box around the described object. The bounding box serves as a prompt for the SAM 2 decoder, producing a high-resolution segmentation mask. Finally, the SAM 2 tracking module propagates the segmentation across subsequent frames, enabling consistent multi-frame object tracking.

The model comprises two main components: the grounding module and the segmentation module. The grounding module interprets the text prompt and identifies the relevant regions in the image by generating bounding boxes around the potential objects of interest as seen in Fig. 3.1. This step is powered by a vision-language transformer trained to associate natural language descriptions with visual features in the images.

The segmentation module SAM 2 refines the output by producing pixel-level masks from the bounding boxes. It performs exceptionally well without requiring dataset-specific fine-tuning. Moreover, SAM 2 includes a tracking mechanism that allows objects to be followed across multiple images of the same scene, which reduces the computational cost because the grounding module only needs to run once.

By combining these two modules, Grounded SAM 2 can detect, segment, and track arbitrary objects solely from a textual description, providing a versatile and efficient tool that is directly applicable to the robust detection of challenging objects in our pipeline.

3.2 Monocular Depth Estimation

Monocular depth estimation has become a key component in vision-based robotic manipulation, especially when hardware constraints or workspace geometry don't alllow for stereo or active sensors. For our setup, where a hand-mounted RGB camera must perceive thin, transparent tubes, traditional monocular methods struggle with faint edges and specular backgrounds. Recent learning-based approaches address these shortcomings by leveraging large, diverse training datasets. Among them, Depth Anything V2 [4] stands out for its ability to recover fine-grained relative depth from a single image without requiring manually annotated real-world depth.

Depth Anything V2 is a monocular depth-estimation model that predicts the relative depth from a single RGB image without the need for manually annotated real-world depth data. Instead, it follows a two-stage teacher-student strategy. First, a large teacher network was trained on 595 000 synthetic images with known depth. Then, the teacher is applied to a much larger collection of approximately 62 million unlabeled real images, producing pseudo-depth maps. In the second stage, lighter student models learn solely from those pseudo-labels, resulting in models that run faster while preserving most of the teachers' accuracy.

The authors reported that Depth Anything V2 produces finer details and is more robust to challenging content such as transparent or reflective surfaces. V2 outperformed its predecessor V1 on a dedicated benchmark and qualitative comparison. These qualities are particularly relevant to our work. An example is shown in Fig. 3.2.

Because the resulting depth map is only correct up to an unknown global scale and shift, these parameters must be determined to obtain an absolute depth estimation.

[1] Source: `https://github.com/IDEA-Research/Grounded-SAM-2/blob/main/assets/g_sam2_tracking_pipeline_vis_new.png`

(a) Original image (b) Predicted depth map

Figure 3.2: Qualitative results of Depth Anything V2 [4]. (a) Input RGB images. (b) Corresponding monocular depth predictions, where red denotes closer regions and blue denotes farther regions relative to the camera. The examples illustrate the model's ability to recover fine-grained depth structure from a single image.

3.3 Pinhole Camera Model

The pinhole camera model describes the basic geometry of an ideal camera without lens distortion. It is widely used in computer vision due to its simplicity and analytical tractability. It is fully defined by the focal lengths f_x, f_y and the principal point $[c_x, c_y]^{\mathrm{T}}$. The focal length represents the distance between the focal point and the image plane, with f_x and f_y potentially differing due to non-square pixels on the camera sensor. The principal point corresponds to the projection of the optical center onto the image plane, expressed in pixel coordinates.

Figure 3.3 illustrates the pinhole camera model, which serves as the foundation for projecting 3D points onto 2D image coordinates in many computer vision tasks.

We define

$$x = \frac{X}{Z}, \qquad y = \frac{Y}{Z}$$

as the normalized image coordinates of the 3D point $P = [X, Y, Z]^{\mathrm{T}}$ given in the camera coordinate system. The projection onto the image plane is defined as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

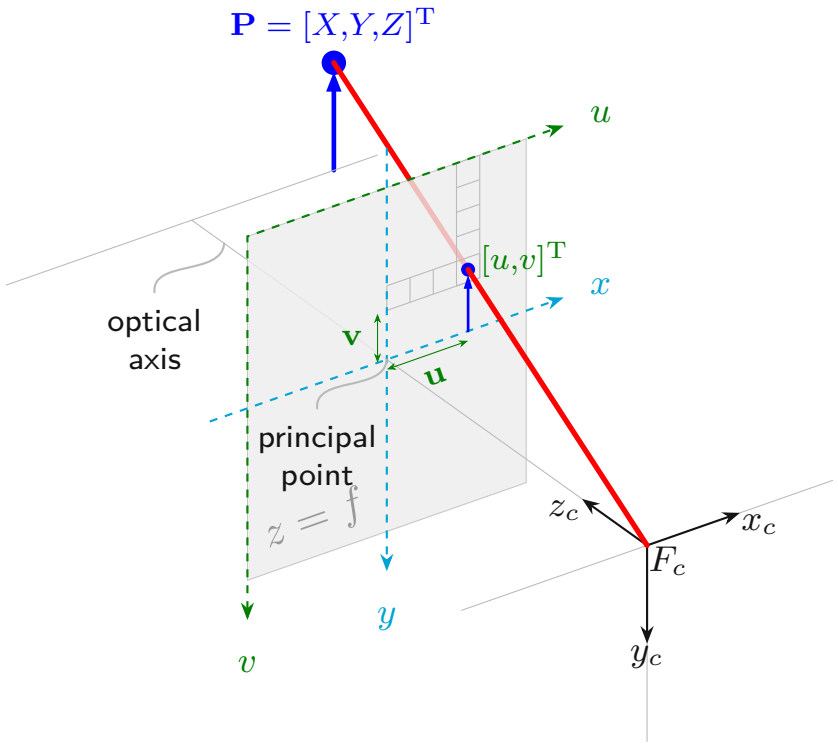


Figure 3.3: Pinhole camera model[2]. A 3D point $\mathbf{P} = [X, Y, Z]^{\mathrm{T}}$ in the camera coordinate frame F_c is first mapped to normalized image coordinates $[x, y]^{\mathrm{T}} = [X/Z, Y/Z]^{\mathrm{T}}$ and then projected onto the image plane as pixel coordinates $[u, v]^{\mathrm{T}}$ using the focal length f.

with the camera matrix

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

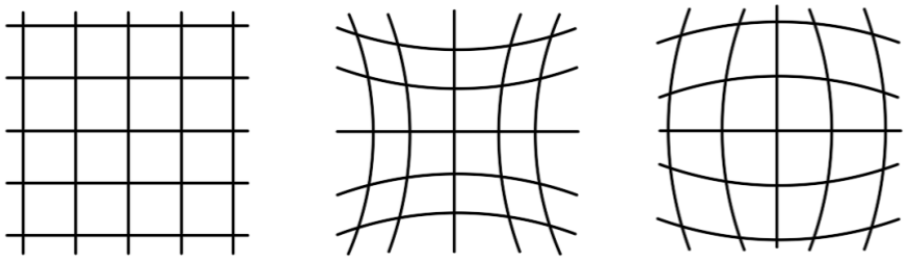
where $[u, v]^{\mathrm{T}}$ is the projected point in pixel space.

Typical photographic lenses introduce radial and tangential distortions to the image due to imperfections and small misalignment to the image sensor. Figure 3.4 shows the typical radial distortions.

To work with the pinhole camera model, we need to un-distort the original camera image. This is achieved by utilizing the *Brown–Conrady* distortion model [33].

1. Using the pinhole camera equations, we calculate the distorted image coordinates $x_d = \frac{u_d - c_x}{f_x}, y_d = \frac{v_d - c_y}{f_y}$ where $[u_d, v_d]^{\mathrm{T}}$ denotes distorted pixel coordinates. We also define the radial distance of the undistorted image coordinates $[x_u, y_u]^{\mathrm{T}}$ as $r_u^2 = x_u^2 + y_u^2$.

[2]Source: `https://tex.stackexchange.com/a/323778`



(a) No distortion (b) Pincushion Distortion (c) Barrel Distortion

Figure 3.4: Examples of radial lens distortion.[3] These effects arise from imperfections in lens geometry: (a) No distortion. (b) Pincushion distortion from lenses that bend light rays too strongly at the edges, causing lines to bow inward. (c) Barrel distortion from wide-angle lenses, where edge rays are bent less, causing lines to bow outward.

2. With three radial coefficients k_1, k_2, k_3 and two tangential coefficients p_1, p_2, the *Brown–Conrady* model is defined as

$$x_d = x_u(1 + k_1 r_u^2 + k_2 r_u^4 + k_3 r_u^6) + 2p_1 x_u y_u + p_2(r_u^2 + 2x_u^2) \tag{3.1a}$$

$$y_d = y_u(1 + k_1 r_u^2 + k_2 r_u^4 + k_3 r_u^6) + p_1(r_u^2 + 2y_u^2) + 2p_2 x_u y_u \ . \tag{3.1b}$$

   To retrieve the undistorted image coordinates $[x_u, y_u]^{\mathrm{T}}$, Eq. 3.1 is solved using the *Newton–Raphson* method [34].

3. The undistorted image coordinates $[x_u, y_u]^{\mathrm{T}}$ are re-projected to the pixel space $u_u = f_x x_u + c_x, v_u = f_y y_u + c_y$. Because $[u_u, v_u]^{\mathrm{T}}$ is generally a non-integer, the source image is resampled at this sub-pixel coordinate using bilinear interpolation.

This procedure is applied to every pixel of the original image.

The mapping from original to undistorted pixel coordinates can be reused for every subsequent image, as long as the camera's intrinsic parameters (focal lengths, principal point, distortion coefficients) and the chosen output image geometry remain unchanged.

3.4 Skeletonize

Skeletonization is the process of reducing a binary mask to a one-pixel-wide skeleton while preserving connectivity. In our application, the skeleton of a DLO mask is used to represent the 2D centerline of the DLO. We employed the classic *Zhang–Suen* algorithm [35] implemented in `scikit-image`'s `skeletonize` [36].

Let $\mathbf{M} : \mathbf{\Omega} \subset \mathbb{Z}^2 \to \{0, 1\}$ be a binary image and let

$$P_1 = \mathbf{M}(i,j), \quad P_2 = \mathbf{M}(i-1,j), \quad P_3 = \mathbf{M}(i-1,j+1), \ldots, P_9 = \mathbf{M}(i-1,j-1)$$

be the pixels listed in Table 3.1.

[3] Source: `https://images.wondershare.com/filmora/article-images/2024/09/lens-distortion-photography-2.png`

Table 3.1: Designations of the nine pixels in a 3×3 window.

P_9 $(i-1, j-1)$	P_2 $(i-1, j)$	P_3 $(i-1, j+1)$
P_8 $(i, j-1)$	P_1 (i,j)	P_4 $(i, j+1)$
P_7 $(i+1, j-1)$	P_6 $(i+1, j)$	P_5 $(i+1, j+1)$

Let

$$B(P_1) = \sum_{k=2}^{9} P_k$$

and $A(P_1)$ be the number of 01 patterns in the ordered set $P_2, P_3, \ldots, P_9, P_2$ (wrapping back to P_2).

The algorithm iterates until convergence. Each full iteration consists of two sub-iterations:

Sub-iteration 1 Mark a foreground point $P_1 = 1$ for deletion if all the following conditions apply.

(a) $2 \leq B(P_1) \leq 6$

(b) $A(P_1) = 1$

(c) $P_2 \cdot P_4 \cdot P_6 = 0$

(d) $P_4 \cdot P_6 \cdot P_8 = 0$

Remove all marked points simultaneously.

Sub-iteration 2 Mark a foreground point $P_1 = 1$ for deletion if all following conditions apply.

(a) $2 \leq B(P_1) \leq 6$

(b) $A(P_1) = 1$

(c') $P_2 \cdot P_4 \cdot P_8 = 0$

(d') $P_2 \cdot P_6 \cdot P_8 = 0$

Again, remove all marked points simultaneously and repeat both sub-iterations until no point can be deleted.

This procedure guarantees that the endpoints and connectivity are preserved, and only the true contour points are removed, resulting in a one-pixel-wide skeleton.

3.5 B-Spline

B-splines are widely used when it comes to representing DLOs due to their many advantages. They are robust and flexible methods due to their inherent smoothness and local control. Adjusting a single control point affects only a local segment of the B-spline. This property means that it is highly suited for incremental adjustments and real-time updates. The smoothness of the B-spline is advantageous because it reflects the properties of real-world DLOs. Due to these properties, a B-spline is well suited for our method.

A B-spline $\mathbf{g}(u)$ with n_c control points is defined as

$$\mathbf{g}(u) = \sum_{i=0}^{n_c-1} \mathbf{c}_i B_{i,k}(u), \quad u \in [t_k, t_{n_c}] \ , \tag{3.2}$$

where $\mathbf{c}_i$ is the i-th control point and $B_{i,k}$ the i-th B-spline basis function of degree $k-1$, defined recursively by the Cox-de Boor formula

$$B_{i,0}(u) = \begin{cases} 1, & t_i \leq u < t_{i+1}, \\ 0, & \text{otherwise}, \end{cases}$$

$$B_{i,k}(u) = \frac{u - t_i}{t_{i+k} - t_i} B_{i,k-1}(u) \ + \ \frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(u), \quad k \geq 1 \ .$$

The basis functions are defined by the knot vector $\mathbf{t} = [t_0, t_1, \ldots, t_{n+k}]$, which consists of non-descending scalar values.

Throughout this work, we model the DLO by a 3D B-spline curve, that is, $\mathbf{c}_i, \mathbf{g}(u) \in \mathbb{R}^3$. We employ a cubic spline ($k = 3$) because a third-order basis provides C^2 continuity (continuous position, tangent, and curvature), capturing the smooth bending behavior of real DLOs while remaining computationally lightweight and locally controllable—properties that are crucial for real-time manipulation.

B-spline fitting Fitting a B-spline to ordered 3D samples is the process of determining a smooth parametric curve $\mathbf{g}(u)$ in Eq. (3.2), which approximates the observed points $\{\mathbf{p}_j\}_{j=1}^m$.

The two goals of fitting are to (i) represent the data faithfully (data fidelity) and (ii) keep the curve smooth (regularity). These objectives are balanced by a smoothing parameter: for small smoothing, the spline interpolates the data; for larger smoothing, the spline trades fit for increased regularity.

Algorithmically, popular solvers (e. g. FITPACK [37]) formulate the problem in the B-spline basis and solve a regularized least-squares problem: they minimize a fidelity term (weighted squared residuals at u_j) subject to a constraint on the roughness of the spline. The solver automatically selects knot positions (unless fixed by the user) and returns both control points and the knot vector, enabling efficient evaluation and differentiation of $\mathbf{g}(u)$ [37]–[39]:

> Given a set of m data points in D dimensions, $\mathbf{p}_j$, with $j = 1, \ldots, m$ and $\mathbf{p}_j = (p_{j;1}, \ldots, p_{j;D})$, this routine constructs the parametric spline curve $g_a(u)$ with $a =$

$1, \ldots, D$, to minimize the sum of jumps, $D_{i;a}$, of the k-th derivative at the internal knots ($u_b < t_i < u_e$), where

$$D_{i;a} = \lim_{\varepsilon \downarrow 0} \mathbf{g}_a^{(k)}(t_i + \varepsilon) - \lim_{\varepsilon \downarrow 0} \mathbf{g}_a^{(k)}(t_i - \varepsilon) \ .$$

Specifically, the routine constructs the spline function $\mathbf{g}(u)$ which minimizes

$$\sum_i \sum_{a=1}^{D} |D_{i;a}|^2 \to \min$$

provided that

$$\sum_{j=1}^{m} \sum_{a=1}^{D} \Big(\mathbf{w}_j \times \big(\mathbf{g}_a(u_j) - p_{j;a} \big) \Big)^2 \leq s$$

where u_j is the value of the parameter corresponding to the data point $(p_{j;1}, \ldots, p_{j;D})$ and $s > 0$ is the input parameter. In other words, we balance maximizing the smoothness (measured as the jumps of the derivative, the first criterion) and the deviation of $\mathbf{g}(u_j)$ from the data $\mathbf{p}_j$ (the second criterion).

In this work, the extracted 3D centerline samples from the depth map are converted into a smooth B-spline representation. This parametrization yields a compact and differentiable model of the object's geometry, which can be efficiently evaluated, sampled, and optimized in subsequent stages of the pipeline.

3.6 Chamfer Distance

This section defines the score metric used to measure the agreement between binary masks: the (one-sided) Chamfer distance. We use this distance to compare the binary segmentation produced from an observed image to the projection of a 3D model (or point set) onto the image plane. The Chamfer formulation is computationally simple, robust to small localization errors, and, after applying sub-pixel bilinear interpolation of the Euclidean distance transform, yields smooth, differentiable residuals suitable for gradient-based refinement of the B-spline model parameters.

Let $\mathbf{M} : \Omega \subset \mathbb{Z}^2 \to \{0, 1\}$ be a binary image. We denote the set of foreground pixels by

$$\mathcal{M} = \{ \mathbf{x} \in \Omega \mid \mathbf{M}(\mathbf{x}) = 0 \} \ .$$

The exact Euclidean distance transform (EDT) assigns to every pixel $\mathbf{x} \in \Omega$ the distance to the nearest foreground pixel

$$\mathrm{DT}_{\mathcal{M}}(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{M}} ||\mathbf{x} - \mathbf{y}||_2 \ . \tag{3.3}$$

Given a second binary image $\mathbf{P}$, its foreground set $\mathcal{P} = \{\mathbf{x} \in \Omega \mid P(\mathbf{x}) = 1\}$, the (one-sided) Chamfer distance of $\mathbf{P}$ to $\mathbf{M}$ is

$$d_{\mathrm{ch}}(\mathbf{P} \to \mathbf{M}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} \mathrm{DT}_{\mathcal{M}}(\mathbf{x}) \ . \tag{3.4}$$

This yields a metric that is defined purely on binary images. When a set of 3D points is projected onto an image plane, the resulting pixel coordinates are generally non-integer and fall between discrete pixel centers. A simple approach is to round each subpixel location to the nearest integer pixel and sample the Euclidean distance transform $\mathrm{DT}(\mathbf{x})$ at that location. Although simple, this quantises the residual and produces discrete steps, which is detrimental to gradient-based optimization.

To obtain a smooth residual, we employ bilinear interpolation. Let the projected point be $\mathbf{p}_{uv} = [u,\, v]^{\mathrm{T}}$. Decompose its coordinates as

$$u = u_0 + \gamma, \quad v = v_0 + \eta, \quad \gamma, \eta \in [0, 1) \ ,$$

where $u_0 = \lfloor u \rfloor$ and $v_0 = \lfloor v \rfloor$ are the neighboring integer coordinates of $\mathbf{p}_{uv}$. Denote the four neighboring integer coordinates by

$$\begin{aligned} \mathbf{p}_{00} &= [u_0, v_0]^{\mathrm{T}}, & \mathbf{p}_{10} &= [u_0 + 1, v_0]^{\mathrm{T}}, \\ \mathbf{p}_{01} &= [u_0, v_0 + 1]^{\mathrm{T}}, & \mathbf{p}_{11} &= [u_0 + 1, v_0 + 1]^{\mathrm{T}} \ . \end{aligned}$$

The bilineary interpolated distance transform is then defined with Eq. (3.3) by

$$\begin{aligned} \widetilde{\mathrm{DT}}(\mathbf{p}_{uv}) = (1-\eta)\Big[(1-\gamma)\,\mathrm{DT}(\mathbf{p}_{00}) + \gamma\,\mathrm{DT}(\mathbf{p}_{10})\Big] \\ + \eta\Big[(1-\gamma)\,\mathrm{DT}(\mathbf{p}_{01}) + \gamma\,\mathrm{DT}(\mathbf{p}_{11})\Big] \ . \end{aligned} \tag{3.5}$$

Because $\widetilde{\mathrm{DT}}(\mathbf{p}_{uv})$ varies smoothly with $\mathbf{u}, \mathbf{v}^{\mathrm{T}}$, it provides continuous, differentiable residuals that are well suited for gradient-based solvers.

In summary, the one-sided Chamfer distance combined with the bilinear interpolation of the Euclidean distance transform provides a compact, differentiable data term for fitting two binary masks. Its simplicity makes it attractive as the core matching term in our optimizer; however, depending on the application, one may consider symmetric variants or complementary penalties to mitigate bias from systematic false positives or negatives in the masks.

4 DLOGrasp Pipeline

In this chapter, we present our complete perception-and-control pipeline that estimates the 3D configuration of a deformable linear object (DLO) from a monocular RGB camera mounted on an articulated robotic arm. Throughout this chapter, we assume that the images from the camera have been rectified according to the pinhole camera model described in Section 3.3 and that the intrinsic and extrinsic parameters are known.

Our pipeline is organized into five sequential stages. Section 4.1 explains how an initial 3D B-spline representation of the DLO is obtained. Section 4.2 introduces the grasp point selection strategy. Section 4.3 presents the closed-loop control law that drives the gripper towards the determined grasp point, while Section 4.4 explains how the initial B-spline is iteratively refined with the masks collected during that control loop. In Section 4.5 the final grasp pose is determined.

4.1 Initialize 3D B-Spline

To obtain the first approximation of the DLO, we first recover a correct depth map and subsequently fit a smooth 3D B-spline curve to the centerline of that map. The resulting spline constitutes a coarse yet physically plausible representation that constrains later optimization, thereby accelerating convergence and preventing degenerate solutions.

4.1.1 Estimate Correct Depth Map

Depth estimation is performed using two RGB images acquired from different viewpoints. From the first image, a monocular depth map is predicted and re-projected in 3D. The resulting point cloud is then projected onto the image plane of the second viewpoint, where it is compared to the corresponding DLO mask. The scale and shift parameters of the initial depth map are optimized such that the projection error is minimized (Fig. 4.1). In the following, we detail how the initial depth map and segmentation mask are obtained.

A dense depth map is first predicted using Depth-Anything v2 [4] (Fig. 4.2b). To isolate the deformable linear object, we apply a binary segmentation mask obtained with Segment-Anything 2 [5] (Fig. 4.2c). Depth values outside the mask are discarded, resulting in a masked depth map that contains only the DLO pixels (Fig. 4.2d) and serves as the input for subsequent processing.

Because the predicted depth is defined only up to an unknown global scale and shift, these two parameters must be recovered to obtain a correct depth estimation. The masked

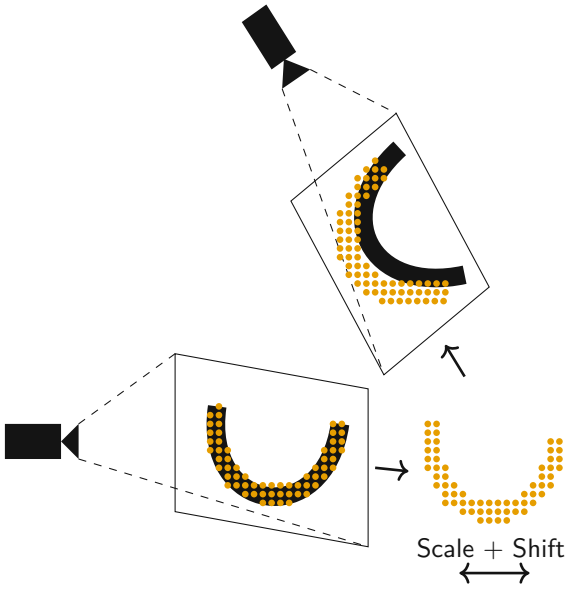


Figure 4.1: Depth-correction pipeline using two RGB views. The masked depth map (●) predicted in the first camera plane is re-projected into 3D, where the 3D points are projected onto the second image plane and compared to the DLO mask (▬). The depth map is then scaled and shifted to minimize the alignment error, thereby resolving the scale ambiguity of the monocular depth estimation.

depth map (Fig. 4.2d) is back-projected into three-dimensional space according to

$$\mathbf{p} = \mathbf{D}_{v,u}\,\mathbf{K} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} , \tag{4.1}$$

resulting in a sparse 3D point cloud of unscaled points, $\mathbf{p} = (x, y, z)^{\mathrm{T}}$ expressed in the corresponding camera frame.

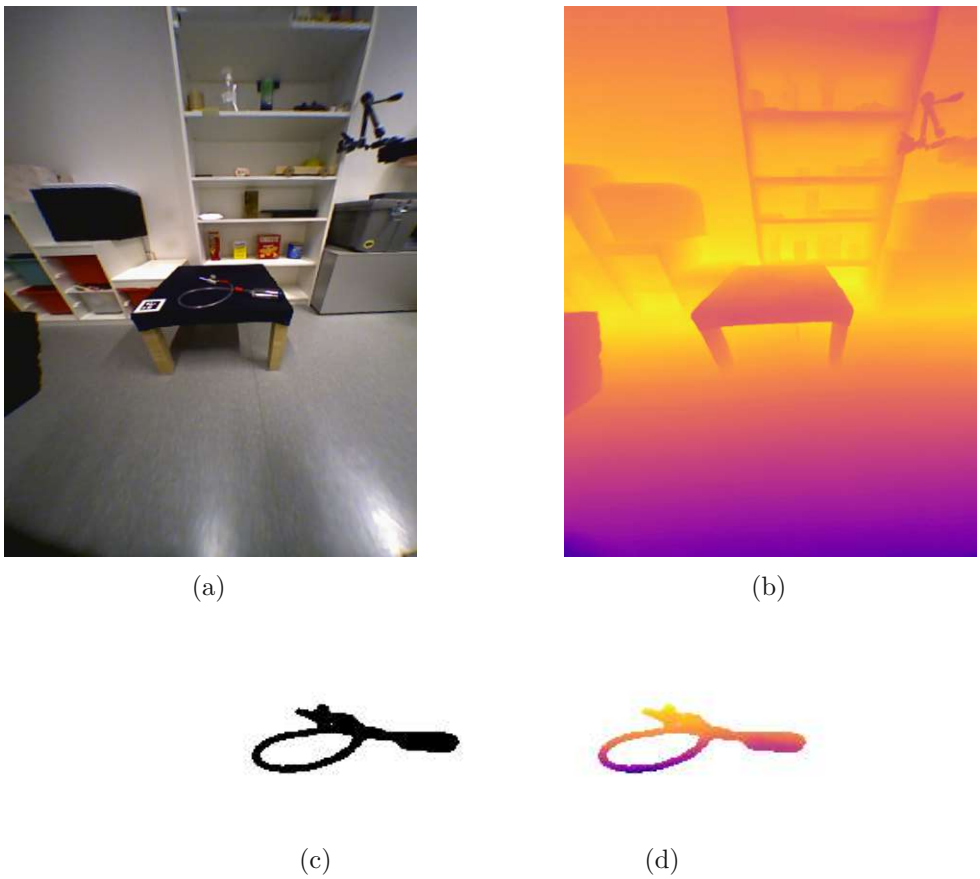
A second RGB image is then captured after the manipulator executes a predefined relative motion of the gripper, moving the eye-in-hand camera to a known offset pose whose optical axis is tilted downward as shown in Fig. 4.3. This elevated viewpoint provides a more top-down look at the scene, whereas the first image was recorded from a predominantly frontal perspective.

Each point $\mathbf{p}$ is transformed into the coordinate system of the second camera using relative rotation $\mathbf{R} \in \mathbf{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbf{R}^3$ between the two camera poses

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \; . \tag{4.2}$$

The transformed points $\mathbf{p}' = [x', y', z']^{\mathrm{T}}$ are then projected onto the second image plane

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} f_x \frac{x'}{z'} + c_x \\ f_y \frac{y'}{z'} + c_y \end{bmatrix} , \tag{4.3}$$



(a) (b)

(c) (d)

Figure 4.2: Initial depth estimation pipeline for the deformable linear object (DLO). (a) Rectified RGB frame acquired by the eye-in-hand camera at the start of the experiment. (b) Raw monocular depth map predicted by Depth-Anything v2. (c) Binary DLO segmentation obtained with Grounded-SAM. (d) Depth map from (b) element-wise masked with (c), isolating the DLO's depth values.

where $[u',v']^{\mathrm{T}}$ denotes the sub-pixel image coordinates associated with the original 3D point.

Let $\mathbf{P}'$ be the set of all projected points $[u',v']^{\mathrm{T}}$ and let $\mathbf{M}$ be the binary mask in the second view. Their alignment is measured using the one-sided Chamfer distance $d_{\mathrm{ch}}(\mathbf{P}' \to \mathbf{M})$ introduced in Section 3.6. This metric averages the bilaterally interpolated Euclidean distance transform at every projected location, yielding a smooth, differentiable residual whose magnitude decreases as the projection aligns better with the mask.

To resolve the global scale ambiguity of monocular depth, we apply the affine depth re-scaling

$$\mathbf{D}' = \alpha \mathbf{D} + \beta \ , \tag{4.4}$$

where $\mathbf{D}$ and $\mathbf{D}'$ denote the original and corrected depth maps, and α and β are the scalar scale and shift parameters, respectively. The optimal parameters are obtained by minimising $d_{\mathrm{ch}}(\mathbf{P}' \to \mathbf{M})$ with the bound-constrained quasi-Newton method L-BFGS-B [40].

(a) (b)

Figure 4.3: First two gripper poses during the experiment, where an image is captured at each pose. (a) Initial pose at the start of the experiment. (b) Predefined pose reached to capture the scene from an alternative viewpoint.

The resulting depth map is the starting point for the subsequent B-spline refinement. Despite this correction, local inaccuracies inherited from the single-image learning-based estimator persist and are addressed in the next stage of the pipeline.

4.1.2 Fit Initial 3D B-Spline

In this step, we convert the transformed depth map $\mathbf{D}'$ from Eq. (4.4) into a parametric curve. Specifically, we aim to fit a smooth 3D B-spline $g(u)$ to the DLO centerline. This requires us to extract an ordered set of 3D centerline points to estimate the spine coefficients using the procedure described in Section 3.5.

We begin by computing a one-pixel-wide skeleton of the binary mask in the first view (Fig. 4.4a), as explained in Section 3.4.

The raw skeleton typically contains spurs and branch points (junctions), which would hinder ordering. To obtain branch-free pieces, we detect junction pixels as those with three or more skeleton neighbors and then excise a small circular neighborhood around each junction (see Fig. 4.4b).

The excision radius is set from the Euclidean distance transform of the foreground mask evaluated at the junction location. The Euclidean distance transform assigns to each foreground pixel its shortest straight-line distance to the nearest background pixel. This value approximates half of the local tube thickness, so the cut scales with width and reliably separates the branches. After excision, we compute the connected components on the remaining skeleton and discard components shorter than the minimum length threshold (see Fig. 4.4c).

For each surviving component, we construct an ordered pixel sequence by locating an endpoint (a pixel with a single skeleton neighbor) and walking along the neighbors until the component is exhausted. Among these ordered sequences, we select the longest by

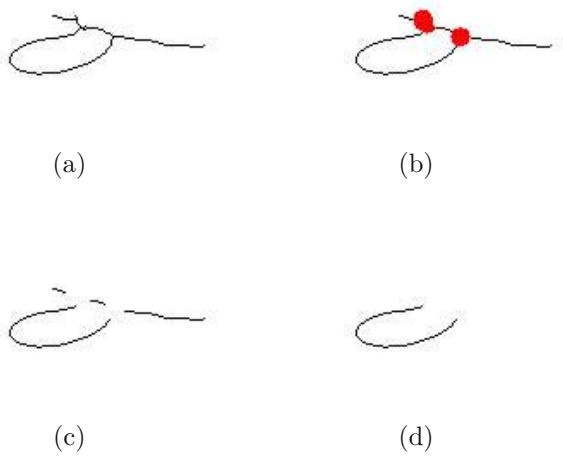


Figure 4.4: Skeleton refinement workflow: (a) One-pixel-wide skeleton extracted from the cropped depth mask (see Fig. 4.2c). (b) Detected junction pixels on the raw skeleton. (c) Skeleton after removal of junctions and short dead-end branches. (d) Longest remaining branch retained as the ordered centerline for subsequent 3D B-spline fitting.

pixel count as the initial centerline candidate.

This procedure ensures that the extracted centerline corresponds to the longest continuous branch, which we interpret as the target DLO. By doing so, the method remains robust, even in the presence of overlapping or intersecting DLOs, as shorter branches are discarded.

We then back-project the pixels of the longest branch into 3D using the corresponding depth map and camera intrinsics according to Eq. (4.1). These ordered 3D points are used for B-spline fitting, as explained in Section 3.5. The resulting fitted B-spline is shown in Fig. 4.5.

Figure 4.5: Back-projected point cloud (●) obtained from the corrected depth map, together with the fitted 3D B-spline (▬), which provides the initial centerline estimate of the DLO.

4.2 Intermediate Grasp Point

In this Section, we determine an intermediate grasp point that, in combination with the next Section 4.3, is used to drive the robotic gripper closer to the DLO.

The midpoint of the reconstructed B-spline was selected as the grasp point. Empirically, this choice provided consistent results because the midpoint showed minimal displacement across iterative refinements, thereby maintaining a stable grasp target. In contrast, selecting the highest point along the B-spline introduced variability because this location often shifted following refinement. Furthermore, grasping near the center of a DLO is intuitively advantageous and aligns with practical handling strategies, whereas the segmentation output from Grounded SAM was observed to be more reliable in this central region.

For simplicity, we selected the parameter midpoint of the B-spline instead of the geometric midpoint. It is computed by evaluating the B-spline $\mathbf{g}(u_{\text{mid}})$ in Eq. (3.2) at the middle of the knot vector

$$u_{\text{start}} = t_k, \quad u_{\text{end}} = t_{n-1}, \quad u_{\text{mid}} = \frac{u_{\text{start}} + u_{\text{end}}}{2} \ . \tag{4.5}$$

4.3 Control Law

To move the robot's gripper to the target point, we implement a proportional, discrete control law.

The target point $\mathbf{p}_1$ is determined in Section 4.2. To obtain the rotation of the target pose $\mathbf{q}_1$, we rotate the gripper's initial pose about its y-axis such that its z-axis is in alignment with the negative z-axis of the world coordinate system, thereby orienting the gripper to point directly downward. In practice, we obtain this rotation by rotating the `base_footprint` frame 90° about its x-axis. Figure 4.6 shows the robots initial pose $(\mathbf{p}_0, \mathbf{q}_0)$ and target pose $(\mathbf{p}_1, \mathbf{q}_1)$

Given the step size $\lambda \in [0, 1]$, we define the grippers next position $\mathbf{p}_{next}(\lambda)$ in the control loop to be

$$\mathbf{p}_{next}(\gamma) = (1 - \gamma)\mathbf{p}_0 + \gamma\mathbf{p}_1 \ ,$$

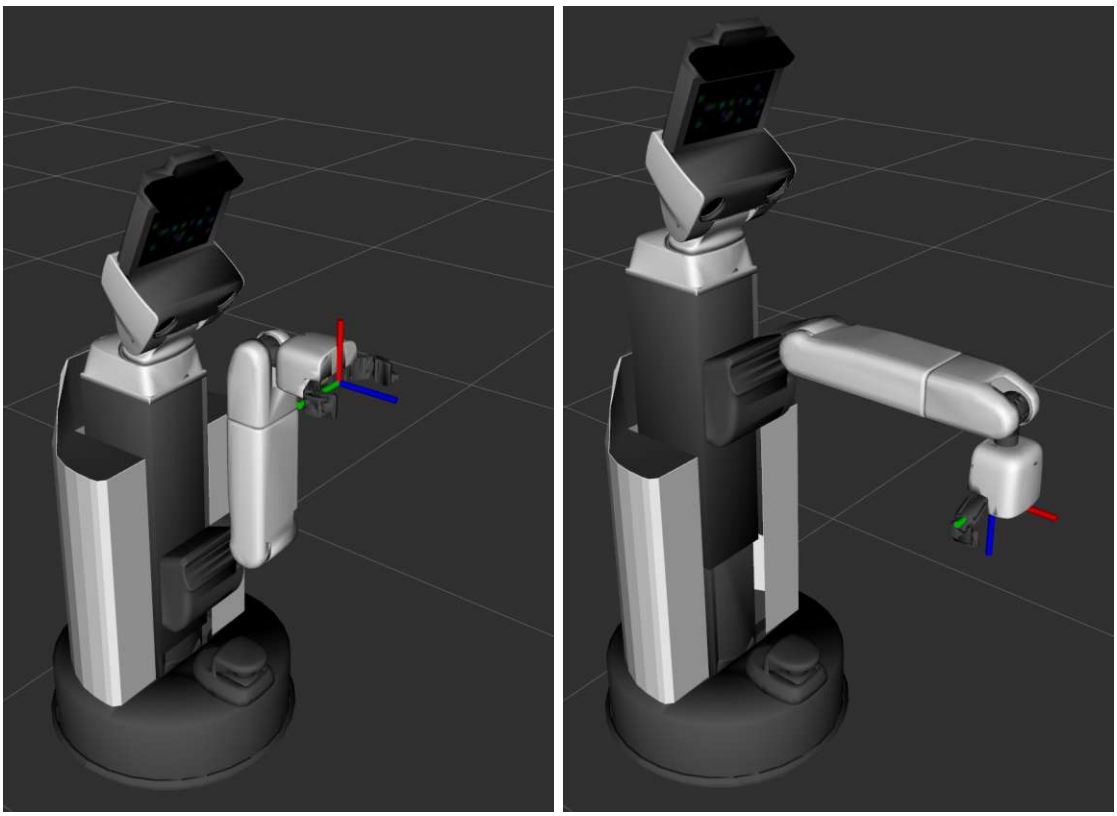
where $\mathbf{p}_0$ is the current position of the gripper, and $\mathbf{p}_1$ is the target point.

Similarly, we define the grippers next rotation $\mathbf{q}_{next}(\gamma)$ as

$$\mathbf{q}_{next}(\gamma) = \mathbf{q}_0 \otimes (\mathbf{q}_0^{-1} \otimes \mathbf{q}_1)^{\gamma} \ ,$$

where $\mathbf{q}_0$ is the current orientation of the gripper and $\mathbf{q}_1$ is the target orientation, both in quaternions. The operand $\otimes$ represents the Hamilton product [41], and $\mathbf{q}_0^{-1}$ represents the quaternion inverse.

The gripper is moved to the next pose $(\mathbf{p}_{next}, \mathbf{q}_{next})$ and the pipeline continues by refining the B-spline in Section 4.4. A wrist rotation, corresponding to a rotation about the gripper's local z-axis, is not applied during the approach phase. It is only executed when the robot performs the final grasp as described in Section 4.5.



(a) Gripper start pose (b) Gripper target (hover) pose

Figure 4.6: Gripper start and target pose visualized in the `hand_palm_link` coordinate frame used for controlling the gripper. (a) Initial pose at the start of the experiment. (b) Target (hover) pose where the gripper is facing downward before wrist rotation. This is the target pose $(\mathbf{p}_1, \mathbf{q}_1)$ of the control law.

4.4 Refine 3D B-Spline

After each control-law step, the camera captures an image from the new viewpoint. A fresh mask of the DLO is extracted with Grounded SAM 2 and skeletonized, as described in Section 3.4. Our objective is to update the control points of the current 3D B-spline such that when the curve is projected into every available image, the resulting 2D points coincide with the corresponding skeletons. The fit quality is captured by a score function (Section 5.5) and minimized with a nonlinear optimizer using the L-BFGS-B method [40]. An overview of this projection–refinement workflow is illustrated in Fig. 4.7.

Projection loss. For each of the n_f captured images, we perform the following steps:

1. **Distance Transform.** Compute the skeleton and exact its Euclidean distance transform (see Fig. 4.8). Each pixel stores its distance to the nearest skeleton pixel.

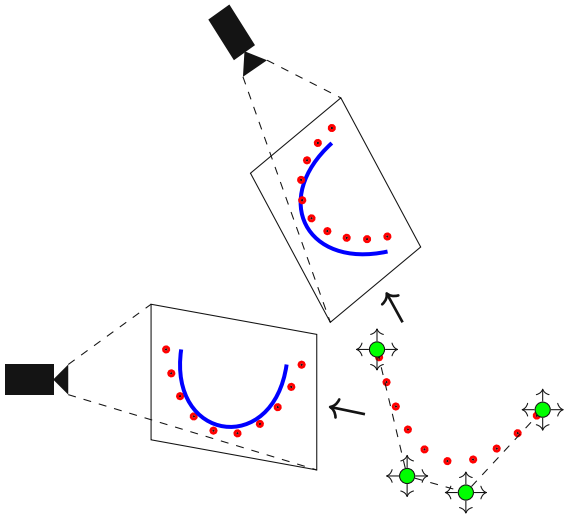


Figure 4.7: Illustration of the refinement step. The sampled B-spline (●), defined by its control points (●), is projected from 3D into all camera views. In the image planes, the projections are compared to the corresponding skeletons (▬) using the Chamfer distance. The resulting error is minimized by adjusting the control points, thereby aligning the spline more closely with the observed DLO.

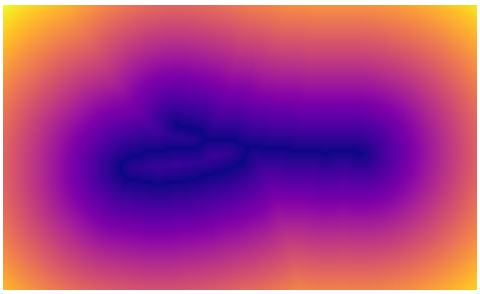


Figure 4.8: Euclidean distance transform of the skeleton for the first captured image. Each pixel encodes the shortest distance to the nearest skeleton pixel, with small distances shown in dark blue/purple and large distances in orange to yellow.

2. **B-Spline Sampling.** Select n_s parameter values

$$u_j = u_{\text{start}} + \frac{j}{n_{\text{s}} - 1}(u_{\text{end}} - u_{\text{start}}), \qquad j = 0, \ldots, n_{\text{s}} - 1 \ ,$$

where u_{start} and u_{end} from Eq. (4.5), and evaluate the B-spline $\mathbf{g}(u_j)$. The sampled B-spline is shown in Fig. 4.9.

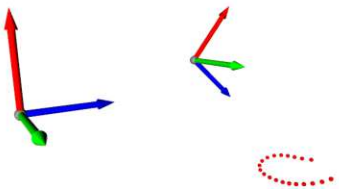


Figure 4.9: First two camera poses and the sampled 3D B-spline before refinement. The coordinate frames indicate the camera poses during image capture. The blue z-axis corresponds to the optical axis. Red dots denote sampled B-spline points. Their projections are compared against the image skeletons.

3. **Projection.** Transform every sample point into the camera frame of the all views using Eq. (4.2) and project them onto the image planes using Eq. (4.3). The first two camera frames are shown in Fig. 4.9 and the first two projections are shown in Fig. 4.10.

Because projected coordinates are generally non-integers, the distance at each point is obtained by bilinear interpolation of the distance-transform image (Section 3.6). Averaging these values yields the one-sided Chamfer distance $\mathrm{d}_{\mathrm{ch}}(\mathbf{P}_l \rightarrow \mathbf{S}_l)$ for view l, where $\mathbf{P}_l$ is the binary image of the projected points and $\mathbf{S}_l$ is the skeleton mask.

The one-sided Chamfer distance is chosen because the initial B-spline may not represent the entire physical DLO, as discussed in Section 4.1.2. In such cases, a symmetric Chamfer distance penalizes parts of the skeleton that are not covered by the spline, effectively encouraging the spline to stretch.

Because our robotic platform is mobile, odometric errors accumulate as the robot moves; therefore, the pose associated with earlier frames is progressively less reliable. We weight older images with an exponential decay factor $w_l = \eta_{\mathrm{d}}^{n_{\mathrm{f}}-l}$ The weighted mean projection error is then

$$\mathcal{L}_{\mathrm{proj}} = \frac{1}{W} \sum_{l=1}^{n_{\mathrm{f}}} w_l \, \mathrm{d}_{\mathrm{ch}}(\mathbf{P}_l \rightarrow \mathbf{S}_l)$$

where $W = \sum_{k=1}^{n_{\mathrm{f}}} w_l$.

Curvature regularisation. To suppress unrealistic sharp bends, we penalize the squared Euclidean norm of the curve's second derivative, which acts as a discrete approximation of its curvature. The loss value for of the curvature is expressed as

$$\mathcal{L}_{\mathrm{curve}} = \lambda_{\mathrm{curve}} \frac{1}{n_{\mathrm{s}}} \sum_{j=1}^{n_{\mathrm{s}}} \left\| \mathbf{g}''(u_j) \right\|^2 .$$

Here, $\mathbf{g}''(u_j)$ is obtained from the analytic second derivative of the B-spline basis function from Eq. (3.2) and $\lambda_{\mathrm{curv}} \geq 0$ controls the strength of the regularizer.

Minimizing this term encourages smoothly varying tangents that mirror the limited bending stiffness of the physical DLO while still permitting local adjustments where image evidence demands.

Drift regularisation. Large displacements of the B-splines' control points would distort the coarse shape obtained from earlier views, especially in regions where the image evidence is weak or ambiguous. To prevent such excessive translations or shrinkage, we penalize these displacements.

Let $\mathbf{c}_i^{(0)}$ denote the i-th control point of the initial spline and $\mathbf{c}_i$ denote the current estimate. The drift cost is defined as

$$\mathcal{L}_{\text{drift}} = \lambda_{\text{drift}} \frac{1}{n_{\text{c}}} \sum_{i=1}^{n_{\text{c}}} \left\| \mathbf{c}_i - \mathbf{c}_i^{(0)} \right\|^2$$

where $\lambda_{\text{drift}} \geq 0$ controls the penalty strength.

Total objective and optimization. The total loss is the combination of the projection-, curvature-, and drift loss

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{proj}} + \mathcal{L}_{\text{curve}} + \mathcal{L}_{\text{drift}} \ .$$

Starting from the spline obtained in the previous cycle, we minimize $\mathcal{L}_{\text{total}}$ using a coarse-to-fine strategy that balances speed and accuracy:

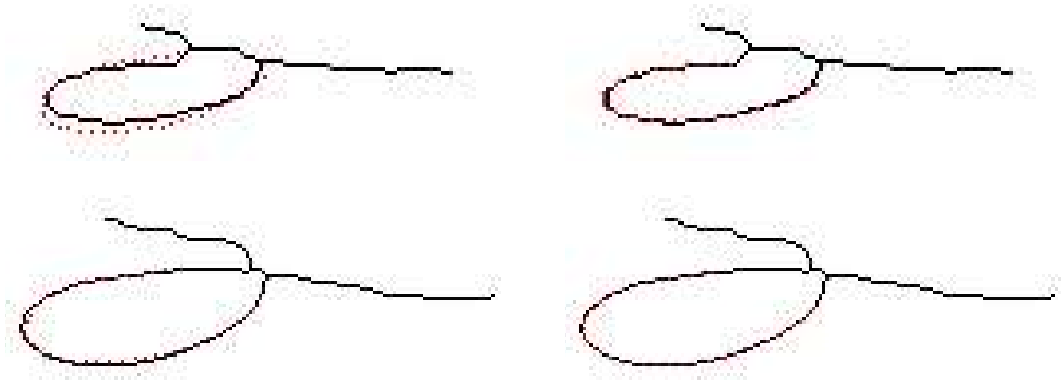
1. **Rigid alignment.** We first restrict the parameter space to a single translation vector $\mathbf{t} \in \mathbb{R}^3$ which is added to every control point. Optimizing this three-parameter problem rapidly moves the entire B-spline into a better starting position for the subsequent refinement steps.
2. **Coarse shape refinement.** All control points are then released, but the projection loss is evaluated on a sparsely sampled set of spline points. This allows for large, smooth deformations while maintaining a low computational cost.
3. **Fine shape refinement.** Finally, the optimization is repeated with denser sampling along the spline, capturing local deviations that are critical for precise grasping. A comparison between the B-spline projections before refinement and after the fine shape refinement is shown in Fig. 4.10.

Owing to the one-sided Chamfer metric, the refinement concentrates on bringing the spline onto the observed centerline without being misled by unmatched portions of the skeleton, while the curvature and drift terms preserve the smoothness and coarse shape of the splines.

After convergence, the pipeline returns to Section 4.2 (Intermediate Grasp Point). Four movement–refinement cycles are executed before the robot proceeds to the final grasp.

4.5 Final Grasp Point

After completing four refinement–movement cycles—an empirically chosen trade-off between runtime and accuracy—the robot executes the grasp. The target pose of the palm frame, $(\mathbf{p}_1, \mathbf{q}_1)$, was determined in Section 4.2. Only the grippers' final writs rotation (z-axis rotation) must be aligned with the local DLO tangent as seen in Fig. 4.11.



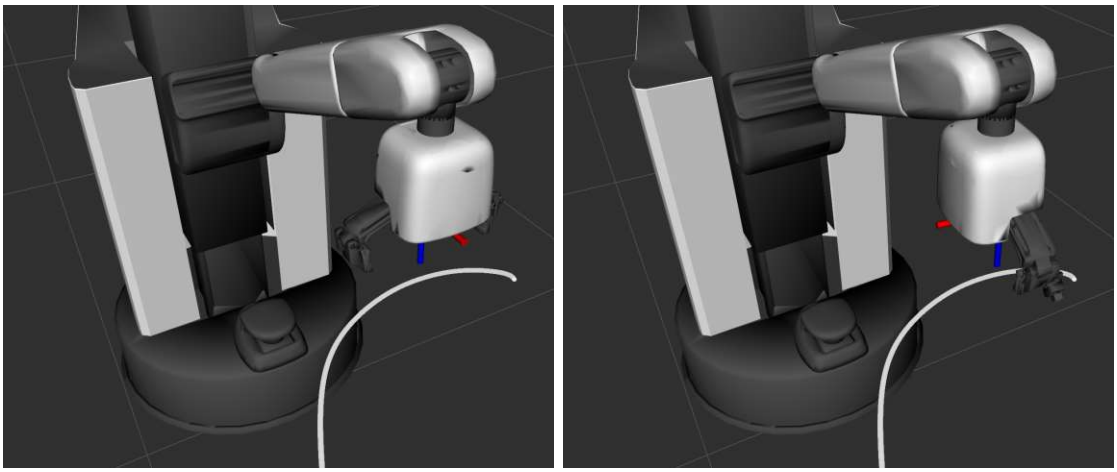
(a) Before refinement (b) After refinement

Figure 4.10: Projection of the B-spline samples (●) onto the first two camera views compared with the corresponding skeletons (▬). (a) Initial projection before refinement. (b) Refined spline after optimization with Chamfer, curvature, and drift terms, yielding closer alignment with the observed centerline while preserving smoothness.

At the parameter midpoint u_{mid} from Eq. (4.5), we evaluate the first derivative of the B-spline in Eq. (3.2)

$$\mathbf{g}'(u) = [x', y', z']^\top.$$

Discarding the vertical component yields the horizontal projection $\mathbf{t}_{xy} = [x', y', 0]^\top$, whose orientation is $\vartheta = \text{atan2}(y', x')$. The corresponding quaternion about the palm z-axis is $\mathbf{q}_\vartheta = \left[0, 0, \sin(\frac{\vartheta}{2}), \cos(\frac{\vartheta}{2})\right]^\mathrm{T}$. Finally, the angle correction is applied by quaternion composition, $\tilde{\mathbf{q}}_1 = \mathbf{q}_\vartheta \otimes \mathbf{q}_1$, and the gripper is moved to the final pose $(\mathbf{p}_1, \tilde{\mathbf{q}}_1)$ and closes it's fingertips, see Fig. 4.12.



(a) (b)

Figure 4.11: Final wrist rotation adjustment. (a) Gripper positioned at the target pose. (b) Same pose after the in-plane (z-axis) rotation that aligns the gripper's fingertips with the local DLO tangent.

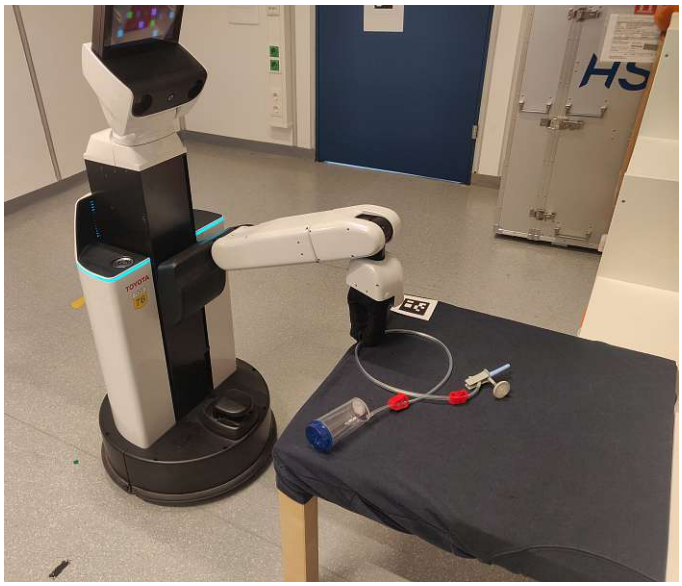


Figure 4.12: Execution of a grasp on the Toyota HSR using a 5 mm diameter transparent tube on a black surface.

5 Experiments and Results

This chapter presents an experimental evaluation of the proposed pipeline for grasping deformable linear objects (DLOs) with a mobile robot. The objective of the experiments is twofold: first, to verify that the perception–and–control loop described in Chapter 4 can reliably reconstruct the three-dimensional configuration of transparent and textureless DLOs under realistic conditions; and second, to assess whether the estimated model enables accurate and robust grasp execution across different objects and surface textures.

In Section 5.1, we describe the robotic platform and its sensor suite, which provide the hardware basis for all experiments. Section 5.2 details the experimental setup, including the choice of DLOs, the workspace, and the scene configurations that introduce visual variability. The experimental procedure, including segmentation prompts, spline fitting parameters, control law settings, and refinement hyperparameters, is presented in Section 5.3. To enable quantitative evaluation, a ground-truth reference is established by voxel carving and fitting a B-spline to the carved volume (Section 5.4). In Section 5.6, the reconstructed splines are compared against this ground truth, and the grasp outcomes are reported. Finally, Section 5.7 discusses the limitations of the approach, outlines possible improvements, and concludes the chapter.

5.1 Experimental Platform

Robot platform All experiments were carried out using Toyota's Human Support Robot (HSR), a compact mobile robot originally conceived to assist people with limited mobility in domestic environments. The HSR combines an omnidirectional caster-drive base (430 mm diameter), a telescopic torso (1.00 m–1.35 m height range), and a lightweight 4-DoF manipulator equipped with a two-finger parallel gripper. Despite its modest mass ($\approx$ 37 kg), the arm can handle objects up to 1.2 kg and reach from floor level to standard work-surface height, enabling typical fetch-and-carry tasks inside a household. The wrist hosts a 6-axis force/torque sensor for compliant interaction, while the sensor suite mounted on the head and forearm includes an RGB-D camera, a wide-angle laser scanner, microphones, and inertial sensing. This hardware layout was deliberately kept simple to maximize safety and fit through narrow domestic passages ($<$ 50 cm) while still offering the perception and manipulation capabilities required for the present study.

Software architecture Our platform runs entirely on the Robot Operating System (ROS) middleware (ROS 1 Noetic). ROS is an open-source robotics framework whose more than 16 000 community-maintained packages cover everything from low-level device drivers to state-of-the-art perception and planning algorithms. Thanks to its publish/subscribe messaging, distributed parameter server, and unified build system, ROS has become

the de-facto standard for academic and industrial robot development, with recent download statistics indicating that over 80% of all professional deployments rely on a ROS distribution.

Hand-camera perception pipeline Our grasp-planning module relies on rectified images streamed by the wrist-mounted hand camera, together with the corresponding 6-DoF camera pose. Prior to further processing, every raw image frame is rectified online by the `image_proc` nodelet from the `image_pipeline` meta-package. During an offline calibration session, we estimated the pinhole intrinsics $\mathbf{K}$ and the radial–tangential distortion vector $\mathbf{d}$ with the standard ROS `camera_calibration tool`. The resulting parameters are published on the `/hand_camera/camera_info` topic and consumed by `image_proc`, which applies an OpenCV–based undistortion and resampling step in real time. This guarantees that the subsequent modules operate on metric, lens-distortion-free imagery.

Manipulator control Finally, the gripper is positioned through Toyota's proprietary whole-body controller and exposed to ROS as an action server that accepts the target poses for the `hand_palm_link` frame. A Cartesian goal $(\mathbf{p}, \mathbf{q})$ expressed in the map frame is converted to a joint trajectory by the controller's integrated inverse kinematics and executed under dynamic constraint supervision to guarantee collision-free motion. This abstraction allowed us to focus on grasp strategy development without delving into low-level actuation details.

To achieve fingertip grasps on the DLO, we must account for the fact that the controller accepts Cartesian goals only for the frame `hand_palm_link`, whose origin lies in the center of the gripper palm. The fingertips were displaced by a constant 83 mm along the local z-axis of `hand_palm_link`. Because at the target rotation $\mathbf{q}_1$, this axis is parallel to the global $-z$-axis of the `map` frame, the offset can be applied directly to the world coordinates. Thus, for a desired grasp point $\mathbf{p}_1$ on the DLO we command the intermediate target

$$\mathbf{p}_{\text{grasp}} = \mathbf{p}_1 + \begin{bmatrix} 0 \\ 0 \\ 83 \end{bmatrix} \text{mm}$$

such that closing the gripper brings the fingertips into proper alignment with the DLO, see Fig. 5.1.

5.2 Experimental Setup

Our evaluation was carried out on a controlled tabletop workspace that combines three representative work surfaces with three types of deformable linear objects (DLOs). The resulting matrix of scenes stresses the vision–grasping pipeline with substantial appearance variation while keeping the geometric scale fixed.

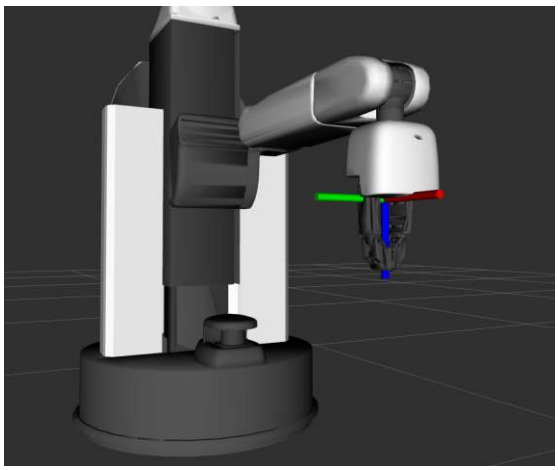


Figure 5.1: Toyota HSR gripper with fingers fully closed. The drawn coordinate frame marks the origin of `hand_palm_link`, located at the center of the palm. Because the frame's z-axis is parallel to the global $-z$-axis, the fingertips are displaced by a constant 83 mm straight "downwards" in world coordinates. During grasp planning we therefore add this fixed offset to the desired grasp point so that, once the gripper closes, the fingertips coincide with the target location on the DLO.

Workpieces. Three DLOs were considered:

(a) **Electrical cable** (grey/white PVC): length 0.8 m, diameter 7.5 mm.

(b) **Single sterile tube** with drip chamber and two roller clamps: length 0.8 m, outer diameter 5 mm.

(c) **Dual sterile tubes**: the tube from (b) plus a second tube of length 0.5 m and identical diameter.

Work surfaces. All trials were carried out on an *Ikea Lack* table (footprint 0.55×0.55 m, height 0.45 m). We evaluated three visual textures:

- a plain black cotton tablecloth,
- the table's own light-brown wood–grain laminate (bare surface, no covering),
- a densely patterned multicoloured tablecloth.

Scene configurations. Each DLO was evaluated in **six** distinct starting poses. Three illustrative examples—a *coil*, a *spiral*, and a *crossed* layout—are shown in Fig. 5.2. Combining the six configurations with the three work surfaces yields $3 \times 3 \times 6 = 54$ unique scenes.

Robot pose. At the beginning of every trial, the mobile base is positioned 0.3 m in front of the table center, facing the table orthogonally, such that the manipulator's workspace fully covers the tabletop while remaining collision-free.

This protocol ensures that the grasp-success rates obtained on one surface remain comparable across the entire matrix of scenes while exposing the perception system to realistic visual variability.

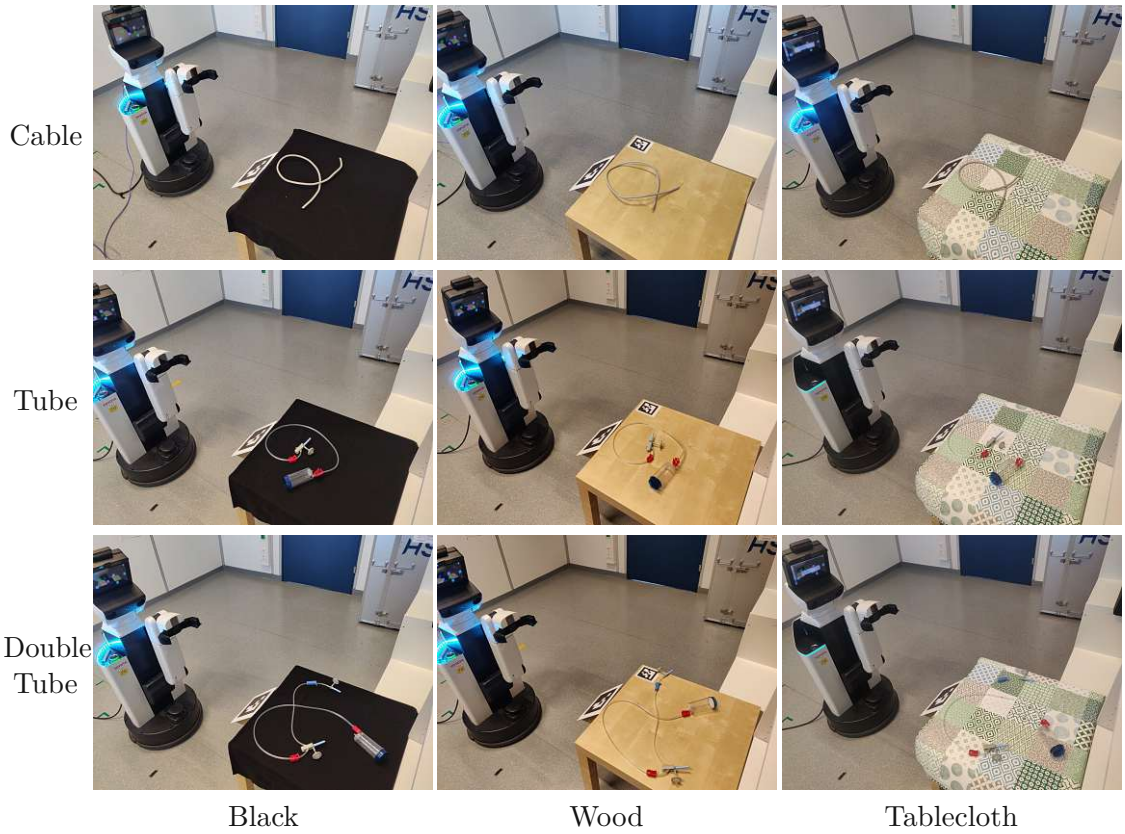


Figure 5.2: Photographs of the initial experimental setups, displaying each deformable linear object (*Cable, Tube, Double-Tube*) on the three table textures (black, wood, tablecloth).

5.3 Experimental Procedure

The experiments were conducted using the perception–and–control pipeline described in Chapter 4. The method was validated under realistic tabletop scenarios involving different deformable linear objects (DLOs) and textures (see Section 5.2). Each scenario was evaluated in six trials. *Tube* and *Double-Tube* on tablecloth were skipped due to failed detection (total $n = 42$). An additional $n = 30$ cable–black trials were run under a fixed configuration to reduce variability.

Segmentation prompts. For segmentation, Grounded Segment Anything 2 (SAM2) was conditioned on textual prompts tailored to the objects. The prompt *"white bent string."*

was used for the *Cable*, whereas *"transparent intravenous line . medical plastic hose ."* was employed for both the *Tube* and the *Double-Tube*. These short descriptive phrases provided the most consistent masks across scenes and were, therefore, fixed throughout all trials.

Spline fitting. To represent the centerline of the DLOs, we fitted a three-dimensional B-spline to the ordered 3D samples obtained in Section 3.5. For this task, we employ SciPy's `make_splprep` routine, which provides a Python interface to P. Dierckx's FITPACK library [37]. This routine solves a regularized least-squares problem to balance fidelity and smoothness, thereby returning a compact parametric spline representation that can be evaluated and differentiated using `splev`. In all experiments, we fit cubic splines ($k = 3$) with $n_c = 20$ control points and a smoothing parameter $s = 10^{-5}$, a configuration that we found to provide stable convergence without overfitting local noise.

Control law. The motion of the gripper toward the intermediate grasp point (Section 4.3) was executed with a discrete interpolation step size of $\gamma = 0.25$. This value balances progress toward the target with sufficient opportunity to incorporate new viewpoints for refinement.

Spline refinement. During each refinement cycle (Section 4.4), the loss function combines the projection term with the curvature and drift regularizers. All experiments were conducted with hyperparameters

$$\lambda_{\text{curve}} = 0.05, \qquad \lambda_{\text{drift}} = 10, \qquad w_l = 1\,,$$

where λ_{curve} controls the strength of curvature penalization, λ_{drift} anchors the spline against large deviations from the initial shape, and w_l denotes the exponential decay factor for older views. Here, $w_l = 1$ corresponds to equal weighting of all images.

Protocol. Each experimental run proceeds as follows: Starting from the initial pose in front of the table, the system acquires the first two images, estimates the depth-scaled point cloud, and fits the initial B-spline. The robot then performs four movement–refinement cycles, each consisting of (i) advancing the gripper toward the current midpoint grasp candidate, (ii) acquiring a new image and segmentation mask, and (iii) updating the B-spline via nonlinear optimization. After the final cycle, the grasp pose is computed, as described in Section 4.5 and executed on the physical robot.

This setup ensures reproducibility across all DLO–texture combinations and provides a consistent basis for quantitative evaluation, which we report in Section 5.6.

5.4 Ground Truth

To quantitatively evaluate the reconstruction accuracy, we established a ground-truth centerline by performing voxel carving [42] on the images captured during each experiment

(the same views used by our method) together with two additional views and subsequently fitting a B-spline to the carved volume.

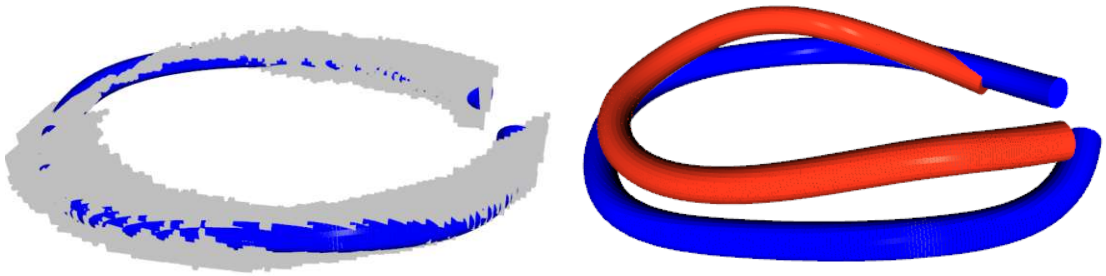
All camera poses were registered using an ArUco marker that was visible in every image (see Fig. 5.3). The marker was not used by our method during the experiments; it only served to establish the ground truth afterward. We took the final image recorded during the experiment (see Fig. 5.3a) as the reference and assume its camera pose to be correct. From this image, we computed the rigid transform from the marker to the global `map` frame, thereby fixing the absolute pose of the marker. Given this absolute marker pose, the remaining camera poses are recovered by chaining the known relative marker–camera transforms with the marker–to–`map` transform.

(a) Final pre-grasp image (b) Additional view (c) Additional view

Figure 5.3: A subset of images used for camera-pose alignment and voxel carving were captured by the robot's hand camera. (a) Final pre-grasp frame from the experiment (used by our method and as the reference for pose alignment). (b,c) Additional views not used by the method; employed only to provide extra silhouettes for voxel carving.

For voxel carving, we obtained ground truth segmentation masks by manually verifying and correcting the segmentation masks obtained during the experiments. The resulting carved occupancy grid served as the basis for the reference B-spline centerline. The two additional hand camera views, used exclusively for carving, are shown in Figs. 5.3b and 5.3c. An example of a carved volume is shown in Fig. 5.4a.

To obtain a parametric centerline from the carved DLO volume, we fitted a cubic B-spline to a union of uniformly subsampled occupied voxels and a small ordered set of guide points in the world frame. A representative result, showing the fitted spline overlaid on the carved voxel grid, is shown in Fig. 5.4a. The spline endpoints were constrained to coincide with the first and last guide points. We estimate the curve via weighted least squares using SciPy's `make_lsq_spline`, assigning larger weights to guide points—especially the endpoints—to anchor the trajectory, while the voxel samples preserve the global shape.



(a) Carved voxel grid ( ) with fitted ground-truth B-spline ( ).

(b) Reconstructed B-spline from our method ( ) overlaid with the ground-truth B-spline ( ).

Figure 5.4: Ground-truth centerline and method comparison. (a) Carved voxel grid with fitted B-spline (ground truth). (b) Reconstructed B-spline from the proposed method overlaid with the same ground-truth spline from (a). All B-splines were visualized as 7.5 mm diameter tubes.

5.5 Evaluation Metrics

To assess the quality of our method, we evaluated both the accuracy of the reconstructed spline and the outcome of the physical grasp.

Reconstruction accuracy. We compared the B-spline estimated by our method with the reference B-spline obtained via voxel carving (Section 5.4). To quantify this difference, the estimated spline was uniformly sampled at 1 mm intervals along its arc length. For each sampled point, the Euclidean distance to the closest point on the reference spline was computed. The collection of these distances was then summarized by reporting the minimum, first quartile (Q25), median, third quartile (Q75), and maximum values. This procedure yielded a robust statistical description of the reconstruction error distribution for each DLO type and table texture.

Grasp-point error. In addition to evaluating the overall spline reconstruction, we specifically assessed the accuracy of the predicted grasp location. The grasp point estimated by our method was defined as the midpoint of the reconstructed B-spline (see Section 4.2). We computed its error as the Euclidean distance between this estimated point and the closest point on the reference B-spline obtained from voxel carving. This metric directly reflects how precisely the method identifies a feasible grasp location independent of the global reconstruction error.

Grasp success. A grasp is deemed successful if the robot can close its fingers around the DLO and lift it from the table surface. If the fingertips contact the table with excessive force during closure, the attempt is aborted and counted as a failure. Light contact with the table is permissible, provided that it still allows secure closure and does not damage the table. This binary outcome metric directly reflects the practical effectiveness of the proposed pipeline in executing reliable grasps.

5.6 Results

In total, we conducted six trials for every combination of DLO type and table texture, resulting in $6 \times 3 \times 3 = 54$ possible experiments. Because Grounded Segment Anygthing 2 (Grounded SAM 2) could not reliably detect *Tube* and *Double-Tube* on the patterned tablecloth, these trials could not be executed. This reduced the total number of experiments to 42.

Reconstruction coverage. Our method does not always recover the complete geometry of the DLO. Because the initial centerline is extracted from the first captured image, only the portion visible in that view is considered for the subsequent reconstruction. If the initial image contains a loop or crossing, the procedure retains only the longest non-intersecting branch and discards the remainder, as described in Section 4.1.2. Similarly, when multiple DLOs are present, as in the case of the *Double-Tube*, only one tube is retained for initialization. As a result, single DLOs (*Cable, Tube*) are generally reconstructed more completely than *Double-Tube*, where coverage is inherently limited to one of the two branches.

Reconstruction accuracy. The reconstruction errors across all the scenarios are summarized in Fig. 5.5. Overall, the medians of the absolute distances between the estimated and ground-truth splines were approximately 15 mm–22 mm. Across all trials, the error values span a range from nearly 0 mm to approximately 85 mm, while the typical spread between the first quartiles (Q25) and third quartiles (Q75) is approximately 13 mm. The largest errors occurred near the ends of the reconstructed B-splines, typically in cases where the voxel-carving reference spline was shorter than the spline produced by the proposed method.

Grasp-point error. Figure 5.6 shows the error distribution at the planned grasping point. The error components along the x- and y-axes remained relatively small (median $\leq$ 5 mm), whereas the z-axis error was larger (median $\approx$ 16 mm), leading to overall Euclidean errors in the range of a few centimeters. These direction-dependent errors reflect the systematic biases introduced by our method and will be revisited in the Discussion section (Section 5.7).

Grasp success. The grasp success rates are shown in Tab. 5.1. Out of 42 trials, 27 resulted in a successful grasp, corresponding to an overall success rate of 64%. Success was highest for *Cable* on the tablecloth surface (83%), and lowest for *Tube* and *Double-Tube* on the black surface (33%). The results also showed that grasping was generally more successful on wood and tablecloth textures than on a black background, despite the latter providing more reliable segmentation.

Extended evaluation. To further examine the reliability under stable conditions, we conducted 30 additional trials of the *Cable* on the black surface using the same cable

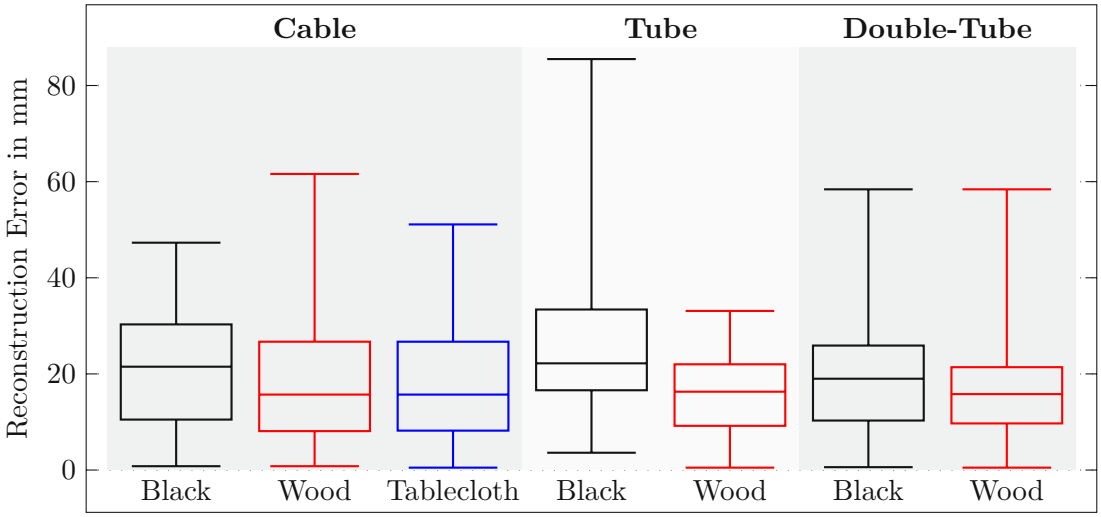


Figure 5.5: Box-plots of the absolute, Euclidean reconstruction error between the B-spline generated by the proposed method and the reference B-spline obtained via voxel carving. The *Cable* was tested on black, wood, and tablecloth textures, whereas *Tube* and *Double-Tube* were tested on black and wood textures only, yielding seven conditions. Each box summarizes six trials ($n = 6$).

Table 5.1: Number of successful grasps and corresponding success rate for each DLO–texture scenario and overall. Each entry was based on six trials.

	Black	Wood	Tablecloth	Total
Cable	4 / 6 (67%)	4 / 6 (67%)	5 / 6 (83%)	13 / 18 (72%)
Tube	2 / 6 (33%)	5 / 6 (83%)	—	7 / 12 (58%)
Double-Tube	2 / 6 (33%)	5 / 6 (83%)	—	7 / 12 (58%)
Overall	8 / 18 (44%)	14 / 18 (78%)	5 / 6 (83%)	27 / 42 (64%)

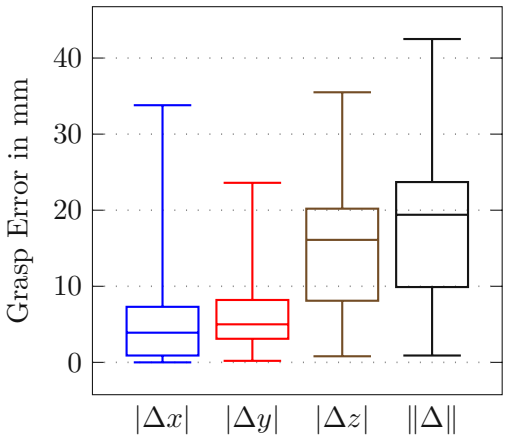


Figure 5.6: Box-plots of the absolute reconstruction error between the B-spline generated by the proposed method and the reference B-spline obtained via voxel carving evaluated at the grasp point. The first three boxes depict the error components along the x-, y-, and z-axes, respectively, whereas the fourth box shows the total Euclidean error $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$. Each box summarizes all 42 experiments ($n = 42$).

configuration in every run. In this controlled scenario, the grasp succeeded in 21 of the 30 cases, corresponding to a success rate of 70% (Tab. 5.2).

Table 5.2: Extended evaluation with 30 additional grasp attempts on the Cable–Black scenario. The table reports the number of successful grasps and the corresponding success rate.

Scenario	Successes / Trials (Rate)
Cable–Black (extended)	21 / 30 (70%)

5.7 Discussion

The experiments revealed several limitations of the proposed method in terms of both reconstruction accuracy and grasp reliability.

Reconstruction accuracy. The reconstruction accuracy was modest. For a grasp to succeed, the error at the grasp point must remain below the diameter of the deformable linear object (DLO) along the gripper's z-axis (5 mm for *Tube* and 7.5 mm for *Cable*). In our trials, this condition was not consistently met, with the median absolute error in the z-direction being approximately 16 mm. Nevertheless, the robot arm is not perfectly stiff, meaning that the gripper can tolerate positioning a few millimeters too low. In practice, the compliance of the arm allows it to give way slightly when in contact with the table

surface. Therefore, some grasps still succeeded despite the z-error exceeding the nominal DLO diameter.

Segmentation on black surfaces was more reliable than that on wood or tablecloth backgrounds, yet paradoxically, the reconstruction error tended to be higher for the black background. A likely explanation is that masks on dark surfaces were more complete, leading to longer splines whose accumulated error was larger, whereas incomplete masks (wood and tablecloth) often produced shorter splines that our method fitted more tightly. This suggests that the pipeline performs better when operating on shorter visible spline segments.

Another limitation arises from the eye-in-hand viewing strategy. As the gripper moves closer to the DLO, later images provide more precise information. However, they also offer less leverage to correct errors introduced in earlier views. Because the camera rotates from a frontal to a top-down view during the approach, the initial frames captured from farther away produce larger reprojection errors along the global z-axis. This explains the axis-dependent spread of errors, with the vertical component being the most affected. In particular, the median absolute grasp error in z was approximately 16 mm, well above the DLO diameters considered here, highlighting the need for explicit compensation in future work. Consequently, millimeter-level accuracy figures reported elsewhere (e.g., [21]) are not directly comparable to our real-world setup and evaluation protocol.

Grasp success. The overall grasp success rate of 64% indicates that the method is not yet reliable for deployment. Even under controlled conditions with a cable on a black surface, the performance plateaued at 70%. The main factor limiting grasp success is the reconstruction error along the global z-axis, which often exceeded the DLO diameter and caused the gripper to be positioned too high or too low relative to the object.

Limitations. A central weakness of the pipeline lies in its dependence on the initial B-spline. If the object detector produces an incomplete mask, the initial spline will only represent the visible portion of the DLO and cannot later expand to the full geometry. One possible remedy would be to regenerate the spline once more of the DLO becomes visible, while still retaining earlier segmentations for refinement.

Another limitation observed during the experiments is that the reconstructed spline systematically shrinks during optimization. Preserving the DLO's physical length would prevent such artifacts and should be enforced via a dedicated regularizer.

6 Conclusion

This thesis presents a novel perception-and-control pipeline for reconstructing and grasping transparent deformable linear objects (DLOs) using only a wrist-mounted RGB camera on Toyota's Human Support Robot (HSR). By incrementally refining a 3D B-spline representation through sequential eye-in-hand views, the system aims to overcome the challenges posed by low-texture transparent materials that defeat classical sensing approaches.

Summary of contributions. The key contribution of this study is the integration of modern open-vocabulary segmentation, monocular depth estimation, and spline-based optimization into a closed-loop grasping pipeline. We showed that it is feasible to initialize a plausible 3D DLO model from only two images, refine it iteratively as new views are acquired, and use it to guide the gripper toward a stable grasp point. To enable quantitative evaluation, a voxel-carving procedure was introduced to establish a B-spline reference for each experiment.

Main findings. The experimental results across 42 trials revealed both the potential and limitations of this method. The reconstruction accuracy reached median errors of 15 mm–22 mm. At the grasp point, the median z-axis error was approximately 16 mm, exceeding the diameters of the tested DLOs (5 mm and 7.5 mm). Nevertheless, compliance with the HSR arm allowed some grasps to succeed, despite such errors. Overall, 64% of all grasps were successful, with the performance improving to 70% in repeated trials with a cable on a black surface under controlled conditions. A qualitative trend indicated that shorter DLOs were reconstructed more robustly, as fewer global compromises in smoothness were required.

Limitations. The analysis identified several bottlenecks. First, the method depends heavily on the initial B-spline: incomplete masks in the first image irreversibly limit coverage. Second, refinement tends to shorten the reconstructed curve, deviating from the true DLO length. Third, the eye-in-hand strategy introduced systematic depth errors along the global z-axis, as early oblique views carried the most ambiguity, while later top-down views had less corrective power. These factors jointly constrain the grasp reliability.

Future improvements. Future research could address these shortcomings in several ways. Improving the accuracy of camera pose estimation, for example, through image-based refinement, would reduce drift and improve alignment between views. Introducing explicit penalties against spline shortening would help preserve the true DLO length across

refinement cycles. The weighting scheme for refinement could be made more selective, reducing the influence of later views captured from nearly redundant perspectives and contributing little new information. Finally, re-initializing the spline dynamically once additional parts of the DLO become visible could substantially improve coverage beyond the limitations imposed by the initial detection.

Outlook. Despite its modest performance, this thesis demonstrates that incremental 3D B-spline refinement from monocular eye-in-hand images is a viable approach for handling transparent DLOs. With improved sensing accuracy, better regularization, and more adaptive refinement strategies, this approach can mature into a reliable tool for robotic applications in healthcare, domestic, and industrial environments. By advancing toward robust tube and cable handling, such methods will enable service robots to perform a wider range of tasks in real-world settings, reducing human workload and extending robotic assistance into domains where it is currently impractical.

Bibliography

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

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, „Orb: An efficient alternative to sift or surf,“ in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.

[3] T. Bailey and H. Durrant-Whyte, „Simultaneous localization and mapping (slam): Part ii,“ *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[4] L. Yang, B. Kang, Z. Huang, *et al.*, „Depth anything v2,“ *arXiv:2406.09414*, 2024.

[5] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, *Sam 2: Segment anything in images and videos*, 2024. arXiv: 2408.00714 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.00714.

[6] S. S. Sajjan, M. Moore, M. Pan, *et al.*, *Cleargrasp: 3d shape estimation of transparent objects for manipulation*, 2019. arXiv: 1910.02550 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1910.02550.

[7] Y. Hong, J. Chen, Y. Cheng, *et al.*, „Cluedepth grasp: Leveraging positional clues of depth for completing depth of transparent objects,“ *Frontiers in Neurorobotics*, vol. Volume 16 - 2022, 2022, ISSN: 1662-5218. [Online]. Available: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2022.1041702.

[8] Z. Zhou, T. Pan, S. Wu, H. Chang, and O. C. Jenkins, „Glassloc: Plenoptic grasp pose detection in transparent clutter,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4776–4783.

[9] Y. R. Wang, Y. Zhao, H. Xu, *et al.*, *Mvtrans: Multi-view perception of transparent objects*, 2023. arXiv: 2302.11683 [cs.RO].

[10] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg, *Dex-nerf: Using a neural radiance field to grasp transparent objects*, 2021. arXiv: 2110.14217 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2110.14217.

[11] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang, *Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf*, 2023. arXiv: 2210.06575 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2210.06575.

[12] T. Weng, A. Pallankize, Y. Tang, O. Kroemer, and D. Held, „Multi-modal transfer learning for grasping transparent and specular objects,“ *IEEE Robotics and Automation Letters*, vol. 5, no. 3, 3791–3798, Jul. 2020, ISSN: 2377-3774. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2974686.

[13] S. Li, H. Yu, W. Ding, *et al.*, „Visual–tactile fusion for transparent object grasping in complex backgrounds,“ *IEEE Transactions on Robotics*, vol. 39, no. 5, 3838–3856, Oct. 2023, ISSN: 1941-0468. [Online]. Available: http://dx.doi.org/10.1109/TRO.2023.3286071.

[14] A. Caporali, K. Galassi, R. Zanella, and G. Palli, „Fastdlo: Fast deformable linear objects instance segmentation,“ *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9075–9082, 2022.

[15] A. Caporali, R. Zanella, D. D. Greogrio, and G. Palli, „Ariadne+: Deep learning–based augmented framework for the instance segmentation of wires,“ *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8607–8617, 2022.

[16] A. Choi, D. Tong, B. Park, D. Terzopoulos, J. Joo, and M. K. Jawed, „Mbest: Realtime deformable linear object detection through minimal bending energy skeleton pixel traversals,“ *IEEE Robotics and Automation Letters*, vol. 8, no. 8, 4863–4870, Aug. 2023, ISSN: 2377-3774. [Online]. Available: http://dx.doi.org/10.1109/LRA.2023.3290419.

[17] A. Caporali, K. Galassi, B. L. Žagar, R. Zanella, G. Palli, and A. C. Knoll, „Rt-dlo: Real-time deformable linear objects instance segmentation,“ *IEEE Transactions on Industrial Informatics*, vol. 19, no. 11, pp. 11 333–11 342, 2023.

[18] P. Kicki, A. Szymko, and K. Walas, *Dloftbs – fast tracking of deformable linear objects with b-splines*, 2023. arXiv: 2302.13694 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2302.13694.

[19] C. Yu, J. Wang, P. Feng, D. Yu, and J. Zhang, „Cvf-dlo: Cross-visual-field branched deformable linear objects route estimation,“ *IEEE Robotics and Automation Letters*, vol. 10, no. 8, pp. 8332–8339, 2025.

[20] A. Caporali, K. Galassi, and G. Palli, „3d dlo shape detection and grasp planning from multiple 2d views,“ in *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2021, pp. 424–429.

[21] A. Caporali, K. Galassi, and G. Palli, „Deformable linear objects 3d shape estimation and tracking from multiple 2d views,“ *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3852–3859, 2023.

[22] S. Zhaole, H. Zhou, L. Nanbo, L. Chen, J. Zhu, and R. B. Fisher, „A robust deformable linear object perception pipeline in 3d: From segmentation to reconstruction,“ *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 843–850, 2024.

[23] Y. Zhu, X. Xiao, W. Wu, and Y. Guo, „3d reconstruction of deformable linear objects based on cylindrical fitting,“ *Signal, Image and Video Processing*, vol. 17, no. 5, pp. 2617–2625, 2023.

[24] K. Lv, M. Yu, Y. Pu, X. Jiang, G. Huang, and X. Li, *Learning to estimate 3-d states of deformable linear objects from single-frame occluded point clouds*, 2023. arXiv: 2210.01433 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2210.01433.

[25] N. Nakagawa and H. Mochiyama, „Real-time shape estimation of an elastic rod using a robot manipulator equipped with a sense of force,“ in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 8067–8073.

[26] H. Dinkel, M. Büsching, A. Longhini, *et al.*, *Dlo-splatting: Tracking deformable linear objects using 3d gaussian splatting*, 2025. arXiv: 2505.08644 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2505.08644.

[27] S. Hutchinson, G. Hager, and P. Corke, „A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[28] F. Chaumette and S. Hutchinson, „Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[29] F. Chaumette and S. Hutchinson, „Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[30] O. Tahri and F. Chaumette, „Point-based and region-based image moments for visual servoing of planar objects," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1116–1127, 2005.

[31] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, „Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[32] B. Huang, J. Yu, and S. Jain, *Earl: Eye-on-hand reinforcement learner for dynamic grasping with active pose estimation*, 2023. arXiv: 2310.06751 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2310.06751.

[33] C. B. Duane, „Close-range camera calibration," *Photogramm. Eng*, vol. 37, no. 8, pp. 855–866, 1971.

[34] B. Benligiray and C. Topal, „Lens distortion rectification using triangulation based interpolation," in *International Symposium on Visual Computing*, Springer, 2015, pp. 35–44.

[35] T. Y. Zhang and C. Y. Suen, „A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[36] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, *et al.*, „Scikit-image: Image processing in Python," *PeerJ*, vol. 2, e453, Jun. 2014, ISSN: 2167-8359. [Online]. Available: https://doi.org/10.7717/peerj.453.

[37] P Dierckx, „Algorithms for smoothing data with periodic and parametric splines," *Computer Graphics and Image Processing*, vol. 20, no. 2, pp. 171–184, 1982, ISSN: 0146-664X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0146664X82900430.

[38] SciPy Developers, *`Scipy.interpolate.make_splprep` function — scipy documentation (version 1.16.0)*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.make_splprep.html, SciPy 1.16.0 documentation; DOI: 10.5281/zenodo.15716342; accessed 2025-07-04, 2025. (visited on 07/04/2025).

[39] P. Dierckx, *Curve and surface fitting with splines*. Oxford University Press, 1995.

[40] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, „A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[41] J. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality* (Princeton paperbacks). Princeton University Press, 1999, ISBN: 9780691102986. [Online]. Available: `https://books.google.at/books?id=_Og9DwAAQBAJ`.

[42] S. M. Seitz and C. R. Dyer, „Photorealistic scene reconstruction by voxel coloring,“ *International journal of computer vision*, vol. 35, no. 2, pp. 151–173, 1999.

Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und, dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Vienna, September 2025

David Schweighofer