TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

High-Speed Pose Tracking for Robotic Grasping Using an RGB Camera and Hybrid Vision Techniques

DIPLOMA THESIS

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

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Dipl.-Ing. Gerald Ebmer
Dr. techn. Minh Nhat Vu
Prof. Dr.-Ing. Wolfgang Kemmetmüller

submitted at the

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

by

Johannes König
Matriculation number 01507421

Vienna, May 2025

Complex Dynamical Systems Group
A-1040 Wien, Gußhausstr. 27–29, Internet: https://www.acin.tuwien.ac.at

Preamble

First and foremost, I would like to express my deepest gratitude to all those who made the completion of this thesis and my graduation possible.

Thank you, Minh, for granting me the opportunity to work on this thesis and for sharing your boundless enthusiasm for research. Your passion was truly inspiring. I am also deeply grateful to the entire team at the Austrian Institute of Technology for welcoming me so warmly. Professor Kemmetmüller, thank you for your insightful feedback, steady supervision, and for always being open to my questions. Huy Hoang (Eric), I am especially thankful for our invaluable discussions, your support, and the motivation you provided throughout this journey. To my colleagues at ACIN, especially Alexander, Guillaume, Marc-Philipp, Thies, and Florian, thank you for the countless hours you spent helping me. Your dedication and generosity made a huge difference. Gerald, I owe you a special thank you for lifting me up during a particularly difficult time and encouraging me to keep going. The lessons you shared during your supervision will stay with me forever.

To all my fellow students who became dear friends, I will always cherish the memories we made and look forward to what lies ahead for each of us. To all my flatmates, thank you for filling my life with friendship and laughter during my studies. Your presence gave me the much-needed energy to reach my goals. Valentina, thank you for having played a crucial role in pushing me over the line during our late-night writing sessions. To my brothers, Max, Mo, and Fabio, you've each been role models to me in your own way, and I'm incredibly thankful for your support. And last but never least: Mami and Papi, thank you for your unwavering love and support throughout my life. Everything I am, I owe to you.

This thesis was created at the Automation and Control Institute of the TU Wien in collaboration with the Austrian Institute of Technology.

Vienna, May 2025

Abstract

Object pose tracking is a crucial part in automated robotic manipulation, requiring reliably updating the pose estimate at high frequencies so to enable precise high-bandwidth control in real-time applications. This thesis presents a pipeline for model-based pose-tracking with an RGB camera, implementing deep-learning-based algorithms for object segmentation and initial pose estimation to guarantee flexibility, while classical computer vision methods are used to iteratively update the pose in each image frame. The pipeline achieves an average update rate of 50 Hz while demonstrating good tracking performance for objects in various scenarios, including a robotic grasping experiment. However, challenges remain in accurately estimating the orientation of symmetrical objects and handling objects with similar colours to the background or highly textured surfaces.

Kurzzusammenfassung

Posenschätzung ist ein wichtiger Bestandteil von automatisierter robotischer Objektmanipulation und erfordert eine zuverlässige Schätzung der Objektpose mit hoher Frequenz, um in Echtzeitregelungssystemen mit hoher Bandbreite Anwendung zu finden. Diese Arbeit stellt eine modellbasierte Pipeline für das Pose-Tracking mit einer RGB-Kamera vor. Sie integriert Deep-Learning-Algorithmen zur Objektsegmentierung und initialen Pose-Schätzung, um eine hohe Flexibilität zu gewährleisten, während klassische Methoden der Computer Vision zur iterativen Aktualisierung der Pose in jedem Bild genutzt werden. Die Pipeline erreicht eine durchschnittliche Aktualisierungsrate von 50 Hz und zeigt eine zuverlässige Tracking-Performance in verschiedenen Szenarien, einschließlich eines robotischen Greifexperiments. Dennoch bestehen Herausforderungen bei der genauen Schätzung der Orientierung symmetrischer Objekte sowie beim Tracking von Objekten mit geringen Farbkontrasten zum Hintergrund oder stark texturierten Oberflächen.

Contents

List of Figures

List of Tables

1 Introduction

Robots have become indispensable in modern industry, logistics and healthcare, taking on tedious, complex or dangerous tasks for humans. Many applications rely on vision-based robotic manipulation, where an RGB or RGB-D camera provides perception input. A prime example is given by automated surface cladding [1], in which a robotic arm continuously tracks the workpiece moving relative to the robot to apply the material on construction sites. In such dynamic environments, real-time perception is essential for adapting to motion and ensuring precise execution.

Vision-based robotic manipulation typically involves four interconnected subtasks [2]:

1. Locating the object of interest within the visual scene.
2. Estimating the object's six degrees of freedom (6-DoF) pose in 3D space from a single image. Doing this for each frame in a video sequence is called pose tracking.
3. Computing an appropriate grasping pose for the robot's end effector.
4. Planning and executing the robot's motion to grasp the object successfully.

In scenarios where objects move relative to the robot - such as conveyor belt systems or human-robot collaboration - steps 2 through 4 must be continuously updated. If tracking is lost, the system must restart from step 1.

A high update rate is critical for real-time robotic control. The pose tracking algorithm should operate at a frequency close to the camera's frame rate (e.g., 30 Hz for standard RGB cameras) to maintain high control bandwidth [3, 4]. A higher update rate enables the robot to react faster to moving objects, which allows precise and smooth motion planning, improving grasp success rates.

For widespread usability, pose tracking methods must be efficient, robust and implemented with common robotic software. This thesis explores real-time object pose tracking using only an RGB camera, focusing on applications in robotic grasping.

1.1 Related Work

A rigid body in three-dimensional space has six degrees of freedom (DoF), comprising three translational and three rotational components. Together, these define the 6-DoF pose of a rigid object. Pose estimation algorithms aim to determine an object's pose by processing information extracted from digital images. Tracking algorithms incrementally refine the object's initial pose over a sequence of images [5, 6].

The Benchmark for 6D Object Pose Estimation (BOP) challenges [7–10], which evaluate the performance of pose estimation and tracking algorithms, have highlighted the progression of techniques in this field. Until 2019, classic methods leveraging Point Pair Features (PPF) [11], utilizing RGB-D or depth-only data, dominated the leaderboard. However, with advancements in large-scale datasets and hardware capabilities, deep neural network (DNN)-based methods surpassed PPF-based approaches in the 2020 BOP challenge. Notably, recent entries [12–14] achieve high performance by relying solely on RGB images, demonstrating the versatility of DNN-based pose estimation compared to traditional computer vision techniques.

GigaPose [15] exemplifies this transition, achieving accurate pose estimations for unseen objects within an average of 50 ms per estimate. This method combines Vision Transformer (ViT) [16]-based feature extraction from scene images and object model renderings with RANSAC to determine the optimal pose. Despite its accuracy, GigaPose struggles to meet real-time requirements of 30 Hz. Its predecessor, MegaPose [17], refines coarse pose estimates at rates approaching 30 Hz, albeit with initial estimation times exceeding one second.

For applications requiring high update rates and constrained computational resources, traditional vision-based trackers remain a compelling choice. Model-based approaches leveraging handcrafted features, such as keypoints [18, 19] or edges [20, 21], compare features extracted from scene images with those from object renderings. These methods establish 2D-3D correspondences before solving an optimization problem to determine the pose. Keypoint-based methods [22] are effective for textured objects, while edge-based methods [23] perform better for textureless objects.

Stoiber et al. [24] introduced a method that leverages color information across correspondence lines to create a probabilistic model. By optimizing this model, the pose is efficiently estimated. The sparse representation of the region model allows the algorithm to run on a single CPU core with potential update times of up to 1.04 ms. Subsequent work added depth [25] and keypoint-based texture modalities [26], culminating in the modular M3T library [27], which supports kinematic chain tracking and enables users to combine multiple modalities as needed.

It is worth noting that most efficient pose-tracking methods are incremental in nature, relying on small pose updates and requiring an initial pose estimate. Furthermore, many pose-estimation algorithms, particularly those based on deep learning, depend on binary object segmentation masks, requiring the additional step of segmenting the object for end-to-end pose-estimation [2].

Traditional segmentation techniques, such as grayscale thresholding [28–30], or methods based on keypoints, edges, or regions, demonstrate varying performance depending on the specific application. Deep learning-based segmentation methods, however, offer greater adaptability. For instance, XMem [31], a convolutional neural network (CNN)-based approach, allows user feedback to refine segmentation masks by selecting points belonging to the foreground or background. While effective for simple shapes, XMem struggles with complex objects. Meta's Segment Anything (SAM) model [32], trained on an extensive

dataset of masked images, demonstrates superior performance by generating high-quality masks for diverse objects based on user input.

1.2 Aim and Structure of this Thesis

This thesis proposes an object-tracking pipeline with a fast update rate for vision-guided manipulation tasks, integrating existing object segmentation, object pose estimation and tracking algorithms. In many practical applications, the objects to be manipulated are known. Hence, the emphasis is on object-tracking based on CAD models.

Chapter 2 introduces the theoretical background for model-based pose tracking of rigid bodies and the required feature extraction from the images.

Chapter 3 describes the pipeline architecture and its implementation details.

Chapter 4 presents the setup and experiments that compare the pipeline's pose output with the measurement of a motion capture system. Furthermore, the pipeline is used in an experiment to track and grasp an object with a robotic gripper.

Finally, Chapter 5 concludes the thesis with a summary of key findings and suggests topics for future work.

2 Theoretical Foundations

To accurately estimate an object's pose in an image, it is essential to extract and process the relevant image information effectively. This chapter introduces the mathematical principles underlying image generation and processing, providing a foundation for understanding pose estimation techniques. A method for distinguishing between foreground and background regions is discussed, as this differentiation plays a crucial role in the proposed online tracking of an object's pose relative to the camera. Finally, the chapter explores data-driven feature extraction methods, which are widely employed in learning-based pose-tracking applications, increasing their adaptability,

2.1 Image Generation and Processing

Figure 2.1 illustrates the typical setup for estimating the pose of an object $\mathcal{O}$ using an RGB camera $\mathcal{C}$. Both the camera and the object are associated with right-handed orthonormal frames. The z-axis of the camera frame ($z_\mathcal{C}$) is parallel to the optical axis and the other axes are aligned with the image plane $\mathcal{P}_\mathcal{I}$. While the actual image plane, coinciding with the sensor plane, is positioned at the focal distance f behind the lens, it is depicted at f in front of the lens in Figure 2.1 for clarity. The pose of the object relative to the camera is described using a homogeneous transformation matrix [33]

$$\mathbf{H}_\mathcal{C}^\mathcal{O} = \begin{bmatrix} \mathbf{R}_\mathcal{C}^\mathcal{O} & \mathbf{x}_\mathcal{C}^\mathcal{O} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3) \ , \tag{2.1}$$

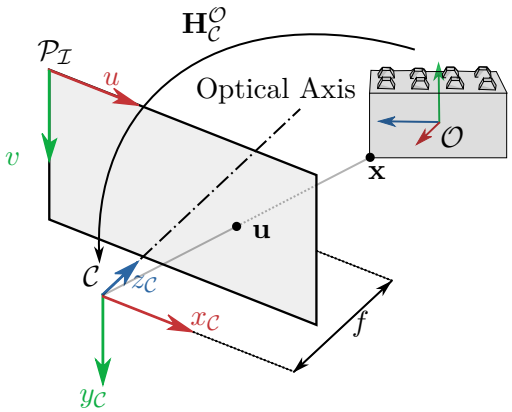


Figure 2.1: Image generation with perspective projection.

where $\mathbf{R}_{\mathcal{C}}^{\mathcal{O}} \in \mathrm{SO}(3)$ represents the rotation matrix, and $\mathbf{x}_{\mathcal{C}}^{\mathcal{O}} \in \mathbb{R}^3$, denotes the translation vector from the origin of $\mathcal{C}$ to the origin of $\mathcal{O}$. With

$$\begin{bmatrix} \mathbf{x}_{\mathcal{C}} \\ 1 \end{bmatrix} = \mathbf{H}_{\mathcal{C}}^{\mathcal{O}} \begin{bmatrix} \mathbf{x}_{\mathcal{O}} \\ 1 \end{bmatrix} \tag{2.2}$$

the coordinates $\mathbf{x}_{\mathcal{O}} \in \mathbb{R}^3$ relative to $\mathcal{O}$ of a point $\mathbf{x}$ are transformed into its coordinates $\mathbf{x}_{\mathcal{C}} \in \mathbb{R}^3$ with respect to $\mathcal{C}$.

Assuming $\mathbf{x}$ is in the camera's field of view and $\mathbf{x}_{\mathcal{C}} = \begin{bmatrix} x_{\mathcal{C}} & y_{\mathcal{C}} & z_{\mathcal{C}} \end{bmatrix}^{\mathrm{T}}$, the perspective projection model [34]

$$\begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} = \mathbf{\Phi}_{\mathbf{K}}(\mathbf{x}_{\mathcal{C}}) = \frac{1}{z_{\mathcal{C}}} \mathbf{K} \mathbf{x}_{\mathcal{C}} \ , \tag{2.3}$$

yields the position $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^{\mathrm{T}} \in \mathbb{U}_{H,W} = (0, H) \times (0, W)$ of the point's projection onto the image plane of height H and width W. The calibration matrix

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

groups the camera's known intrinsic parameters: the focal lengths f_x and f_y, and c_x and c_y indicating the image centre in pixels. Recovering the 3D coordinates from the 2D image coordinates is achieved via the inverse operation of (2.3)

$$\mathbf{x}_{\mathcal{C}} = {\mathbf{\Phi}_{\mathbf{K}}}^{-1}(\mathbf{u}, z_{\mathcal{C}}) = z_{\mathcal{C}} \mathbf{K}^{-1} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} . \tag{2.5}$$

The projection information for all points in the field of view is stored in an array $\mathcal{I} \in \mathbb{D}^{C \times H \times W}$, commonly referred to as a digital image. For RGB cameras, the number of channels is $C = 3$, where each channel represents the intensity of red, green, or blue colour components, digitally encoded as integer values between 0 and 255, such that $\mathbb{D} = \{0, \ldots, 255\}$. The red, green and blue colour values $\mathbf{y} \in \mathbb{D}^3$ of the pixel at $\mathbf{u} \in \mathbb{U}_{W,H}$ in the image $\mathcal{I}$ are extracted with the colour function

$$\mathbf{y} = \mathbf{f}_{\mathcal{I}}(\mathbf{u}) \ . \tag{2.6}$$

Scaling, rotating and shifting images is modelled using similarity transformation matrices [34]

$$\mathbf{M}_a^A = \begin{bmatrix} s_a^A \mathbf{R}_a^A\left(\alpha_a^A\right) & \mathbf{u}_a^A \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \ , \tag{2.7}$$

where $s_a^A > 0$ is the scaling factor, $\mathbf{R}_a^A\left(\alpha_a^A\right) \in \mathrm{SO}(2)$ denotes the in-plane rotation by an angle α_a^A, and $\mathbf{u}_a^A \in \mathbb{R}^2$ is the translation vector. This matrix transforms points

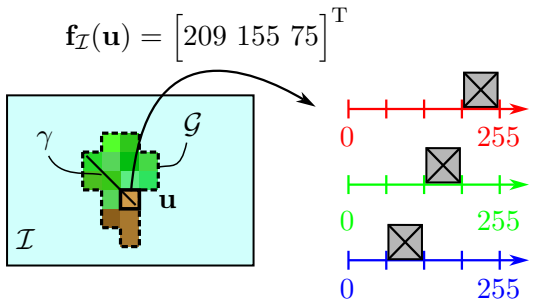


Figure 2.2: Filling up the colour histogram of a region $\mathcal{G}$ sparsely along the line γ. The region is delimited by dotted lines.

$\mathbf{u}_A \in \mathbb{U}_{H_A,W_A}$ from the original image $\mathcal{I}_A \in \mathbb{D}^{3\times H_A\times W_A}$ to their counterparts $\mathbf{u}_a \in \mathbb{U}_{H_a,W_a}$ on the transformed image $\mathcal{I}_a \in \mathbb{D}^{3\times H_a\times W_a}$ with

$$\begin{bmatrix}\mathbf{u}_a\\1\end{bmatrix} = \mathbf{M}_a^A \begin{bmatrix}\mathbf{u}_A\\1\end{bmatrix} . \tag{2.8}$$

The cropping and resizing operations are captured by setting $\alpha_a^A = 0$.

2.2 Region Model

Region-based methods partition the image into a foreground region $\mathcal{G}_f$ containing the object of interest, and a background region $\mathcal{G}_b$. Most region-based methods, like [35], extract features from the whole image, which is computationally expensive. In [24], Stoiber et al. proposed a region-based method, where colour histograms are created along lines perpendicular to the object's contour in the image. These so-called *correspondence lines* make sparse feature extraction possible and improve efficiency. The colour information is then used to compute the probability of the introduced contour distances to describe the observed colour distribution in the image.

2.2.1 Colour Histograms

A *colour histogram* for an image region $\mathcal{G} \subseteq \mathbb{U}_{W,H}$ partitions the RGB colour space $\mathbb{D}^3$ into discrete bins, recording the number of pixels in $\mathcal{G}$ that fall into each bin [34]. The histogram's accuracy depends on both the number of colour bins and the size of the region $\gamma \subseteq \mathcal{G}$ from which pixel colours are sampled. This region is called *extraction region*. The extraction region and bin size are design parameters for the colour histogram extraction. A higher number of bins and a larger extraction region improve accuracy but increase computational cost. If $\gamma = \mathcal{G}$, the colour histogram for $\mathcal{G}$ is said to be *densely* extracted, otherwise *sparsely* extracted.

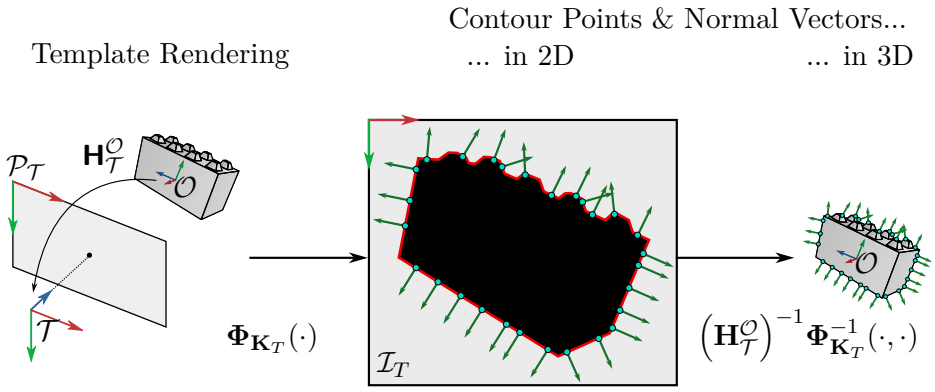


Figure 2.3: Creation of a sparse viewpoint model for a given pose $\mathbf{H}_\mathcal{T}^\mathcal{O}$ as in [24].

Figure 2.2 illustrates the extraction of the colour histogram for a region $\mathcal{G}$ of an image. In this example, the extraction region γ is defined along a line traversing the region's contour. In this example, a pixel at $\mathbf{u}$ belonging to γ is assigned to a bin in the histogram. Each colour channel is divided into four equidistant bins, resulting in a total of $4^3 = 64$ possible values. The colour of this pixel is determined using (2.6)

$$\mathbf{f}_\mathcal{I}(\mathbf{u}) = \begin{bmatrix} 209 \\ 155 \\ 75 \end{bmatrix} . \tag{2.9}$$

This corresponds to the fourth, third, and second bins of the red, green, and blue axes, respectively, incrementing the count of that specific colour bin. The colour bins to which $\mathbf{u}$ is assigned are marked with crossed boxes in the graphic.

Given the colour histograms for two disjoint regions $\mathcal{G}_f, \mathcal{G}_b \subset \mathbb{U}_{W,H}$ of an image $\mathcal{I}$ such that $\mathcal{G}_f \cup \mathcal{G}_b = \mathbb{U}_{W,H}$, the probability

$$p(\mathbf{y} \mid \mathcal{G}_i) \ , \quad \mathbf{y} \in \mathbb{D}^3 \ , \quad i \in \{f, b\} \ , \tag{2.10}$$

represents the likelihood of colour $\mathbf{y}$ occurring in a pixel within region $\mathcal{G}_i$. This probability is obtained by dividing the bin counts by the total number of pixels considered across both histograms. The probabilities for $\mathcal{G}_f$ and $\mathcal{G}_b$ are then combined to define the likelihood functions

$$p(\mathcal{G}_i \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathcal{G}_i)}{p(\mathbf{y} \mid \mathcal{G}_f) + p(\mathbf{y} \mid \mathcal{G}_b)} \tag{2.11}$$

which express the probability that a given colour $\mathbf{y}$ belongs to $\mathcal{G}_i$, $i = \{f, b\}$ [24].

2.2.2 Sparse Viewpoint Model and Correspondence Lines

Given an image $\mathcal{I}$ of an object $\mathcal{O}$, captured using a real camera $\mathcal{C}$ at a pose $\mathbf{H}_\mathcal{C}^\mathcal{O}$, Figure 2.3 and Figure 2.4 illustrate the creation process of a *sparse viewpoint model* $\mathcal{R}$ for a pose $\mathbf{H}_\mathcal{T}^\mathcal{O}$ where $\mathbf{R}_\mathcal{T}^\mathcal{O} = \mathbf{R}_\mathcal{C}^\mathcal{O}$, as in [24]. First, a virtual camera $\mathcal{T}$ with calibration matrix $\mathbf{K}_\mathcal{T}$

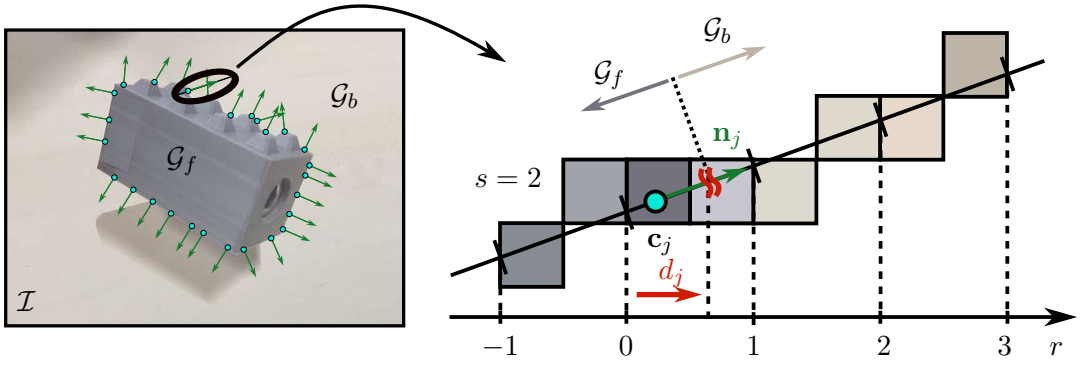


Figure 2.4: Close-up view of a projected correspondence line on a real image $\mathcal{I}$ as in [24].

renders a binary template image $\mathcal{I}_T$ of the CAD model of the object $\mathcal{O}$ at the pose $\mathbf{H}_{\mathcal{T}}^{\mathcal{O}}$ using (2.3). Then, using classical edge detection algorithms, n_c points along the object's silhouette contour are randomly sampled in $\mathcal{I}_T$, and their corresponding outward surface normal vectors are estimated. In Figure 2.3, the contour line is highlighted in red, while the contour points and normal vectors are drawn in blue and green, respectively. Finally, the contour points and normal vectors are transformed back into the 3D object frame $\mathcal{O}$ using $\left(\mathbf{H}_{\mathcal{T}}^{\mathcal{O}}\right)^{-1}$ and (2.5). These transformed contour points and normal vectors define the sparse viewpoint model $\mathcal{R}$ at $\mathbf{H}_{\mathcal{T}}^{\mathcal{O}}$.

The sparse viewpoint model is then projected onto $\mathcal{I}$ using $\mathbf{\Phi}_{\mathbf{K}}\left(\mathbf{H}_{\mathcal{C}}^{\mathcal{O}}(\cdot)\right)$, where $\mathbf{K}$ is the calibration matrix of $\mathcal{C}$. Figure 2.4 shows a magnified view of a *projected correspondence line*. Each correspondence line is defined by a *projected contour point* $\mathbf{c}_j$ and a *projected normal vector* $\mathbf{n}_j$, $j \in [1, \ldots, n_c]$. The *projected viewpoint model* Ω is defined as the union of all projected correspondence lines.

By grouping $s \in \mathbb{N}$ adjacent pixels on the correspondence line into *pixel segments*, a coordinate $r \in \mathbb{R}$ is introduced. It measures the distance between the contour point's closest pixel and another point along the correspondence line, expressed in pixel segments. Moving inward towards the object (the foreground $\mathcal{G}_f$), r decreases towards $-\infty$, whereas it increases towards $+\infty$ when moving into the background $\mathcal{G}_b$. The *signed contour distance* $d_j \in \mathbb{R}$ represents the signed separation between $\mathbf{c}_j$ and the actual boundary between foreground and background in the image. In Figure 2.4, the scale parameter is set to $s = 2$. The interested reader is referred to [24] for a more detailed explanation.

2.2.3 Probabilistic Model

In the presence of image noise or inaccuracies in the CAD model, the observed boundary between $\mathcal{G}_f$ and $\mathcal{G}_b$ on the j-th correspondence line may shift to a distance $r \neq d_j$, indicating a discrepancy between the template and real images. Given the contour distance d_j and the model $\mathcal{G} \in \{\mathcal{G}_f, \mathcal{G}_b\}$, the probability of a point at a distance r from the contour point

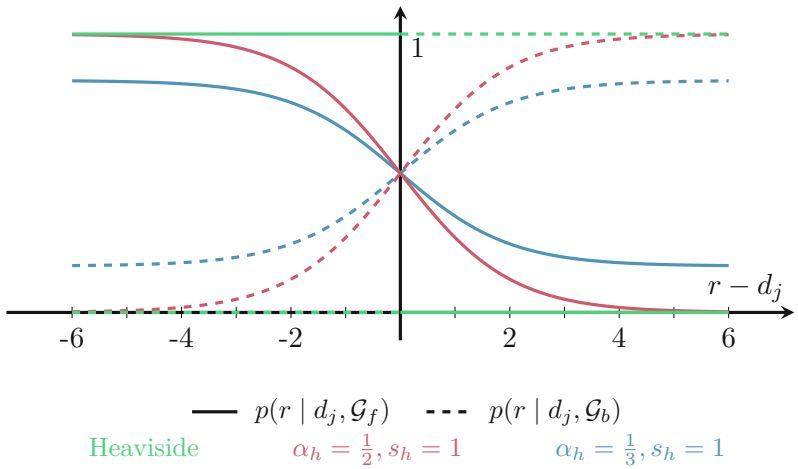


Figure 2.5: Different versions of the step functions $p(r \mid d_j, \mathcal{G}_f)$ and $p(r \mid d_j, \mathcal{G}_f)$ as shown in [24].

belonging to $\mathcal{G}$ can be described by the cumulative distribution functions

$$p(r \mid d_j, \mathcal{G}_f) = \sigma(d_j - r) \ , \tag{2.12a}$$
$$p(r \mid d_j, \mathcal{G}_b) = \sigma(r - d_j) \ , \tag{2.12b}$$

with the Heaviside step function

$$\sigma(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} . \tag{2.13}$$

In [24], the authors proposed modelling these probabilities using smoothed step functions

$$p(r \mid d_j, \mathcal{G}_f) = \frac{1}{2} - \alpha_h \tanh\left(\frac{r - d_j}{2 s_h}\right) \ , \tag{2.14a}$$
$$p(r \mid d_j, \mathcal{G}_b) = \frac{1}{2} + \alpha_h \tanh\left(\frac{r - d_j}{2 s_h}\right) \ , \tag{2.14b}$$

where the amplitude parameter $\alpha_h \in [0, 0.5]$ accounts for external effects such as noise, while the slope parameter s_h models local uncertainty. With $\alpha_h = \frac{1}{2}$ and $s_h \to 0$, these functions converge to the original step functions (2.14). Figure 2.5 demonstrates smoothed step functions with various parameter configurations.

To extract the colour histograms for foreground $\mathcal{G}_f$ and background $\mathcal{G}_b$ along the projected correspondence lines, a set-valued function $\mathbf{l}_j(r)$ is introduced which returns the set of s colours of pixels in the segment at distance r along the j-th correspondence line. Since the colour likelihoods (2.11) do not accept a set of colours as arguments, they are adapted to

$$p(\mathcal{G}_i \mid \mathbf{l}_j(r)) \propto \frac{\prod_{\mathbf{y} \in \mathbf{l}_j(r)} p(\mathbf{y} \mid \mathcal{G}_i)}{\prod_{\mathbf{y} \in \mathbf{l}_j(r)} p(\mathbf{y} \mid \mathcal{G}_f) + \prod_{\mathbf{y} \in \mathbf{l}_j(r)} p(\mathbf{y} \mid \mathcal{G}_b)} \ , \quad i \in \{f, b\} \ . \tag{2.15}$$

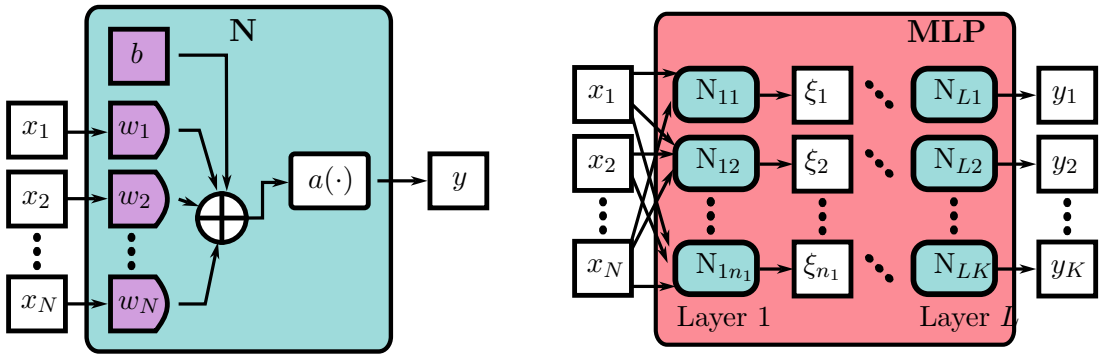


Figure 2.6: Single Perceptron (N) and a Multilayer Perceptron (MLP). Trainable parameters are coloured in violet.

They are combined with (2.14) to obtain the posterior density functions

$$p(d_j \mid \mathbf{l}_j) = \prod_{r \in \mathbb{L}_j} \left(\sum_{i \in \{f,b\}} p(\mathcal{G}_i \mid \mathbf{l}_j(r))\, p(r \mid d_j, \mathcal{G}_i) \right) , \tag{2.16}$$

where $\mathbb{L}_j$ is the discrete set ensuring that every segment along the correspondence line appears exactly once. This posterior describes the probability of observing the contour distance d_j given the colours extracted with $\mathbf{l}_j$ along its correspondence line. The full posterior, which models the probability of observing all contour distances $d_1, \dots, d_{n_c}$ given the entire colour information $\mathcal{D}$ extracted along the correspondence lines with the set valued functions $\mathbf{l}_j$, is given by

$$p(d_1 \wedge \cdots \wedge d_{n_c} \mid \mathcal{D}) \propto \prod_{j=1}^{n_c} p(d_j \mid \mathbf{l}_j)^{m_j s_h\, s^2} , \tag{2.17}$$

where m_j is a constant associated with the j-th correspondence line. This posterior will be used to track the object's pose in Section 3.4.2. A detailed derivation of (2.17) can be found in [27].

2.3 Feature Extraction with Deep Neural Networks

The human brain has an extraordinary ability to process and interpret diverse types of information, adapting to changing environments, which classical machine vision cannot do. Deep neural networks aim to replicate this behaviour. This section presents the foundations of *deep neural networks* (DNNs), before discussing feature extractors based on *convolutional neural networks* (CNNs) and *Vision Transformers* (ViTs).

2.3.1 Neuron to Deep Neural Networks

Neural networks aim to mimic the human brain by constructing interconnected computational units known as *perceptrons* or *neurons* (see Figure 2.6). The ultimate objective is

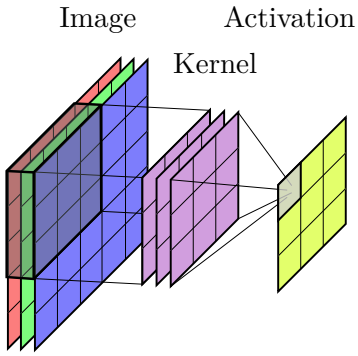


Figure 2.7: A convolutional layer of a 5×5 RGB image with a 3×3 kernel as shown in [36]. The learnable parameters are coloured in violet.

to approximate arbitrary functions using these networks. A single neuron is a function with a scalar output in the form of [36]

$$y = \mathrm{N}(\mathbf{x}, \mathbf{w}) = a\left(\mathbf{w}^{\mathrm{T}} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}\right), \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \\ b \end{bmatrix}. \tag{2.18}$$

The *weights* $w_1, \ldots, w_N$ and the *bias* b are learnable parameters, while the activation function $a(\cdot)$ introduces nonlinearity, enabling the neuron to model complex relationships.

A single neuron provides limited modelling capacity, but combining multiple neurons enables the approximation of vector-valued functions. By feeding an input $\mathbf{x}$ into K neurons, each with its own weight vector $\mathbf{w}_j$, $j = 1, \ldots, K$, a layer is formed, producing an output vector

$$\mathbf{y} = \mathbf{a}(\mathbf{W}\mathbf{x}), \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_K \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{w}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{w}_K^{\mathrm{T}} \end{bmatrix}. \tag{2.19}$$

Here, $\mathbf{a}(\cdot)$ denotes the element-wise application of the activation function. Stacking multiple layers forms a deep neural network and is commonly referred to as a *Multilayer Perceptron* (MLP) (see Figure 2.6). The number of neurons in the final layer determines the output dimensionality of the MLP.

Deeper networks can approximate more complex functions, but increasing depth comes at the cost of higher computational demands during both training and inference, and can lead to overfitting. For further details, see [36].

2.3.2 Convolutional Neural Networks

While fully connected neural networks, such as MLPs, are effective for many tasks, they become computationally inefficient when dealing with high-dimensional inputs like images.

Convolutional Neural Networks (CNNs) address this issue by leveraging spatial hierarchies and local connectivity patterns to efficiently extract relevant features from data.

A *convolutional layer* (see Figure 2.7) is the fundamental building block of a CNN. Instead of connecting each neuron to all input features, it applies a two-dimensional convolution operation, sliding a small matrix, known as a learnable filter or kernel $\mathbf{W} \in \mathbb{R}^{C \times M \times N}$, over the input $\mathcal{I} \in \mathbb{R}^{C \times H \times W}$. Here C represents the number of channels, $W \times H$ and $M \times N$ are the dimensions of the input and kernel for each channel, respectively. The outputs of the convolution are fed through a non-linear activation function and stored in a two-dimensional feature map or an activation map. Applying multiple filters results in a set of feature maps, each capturing different patterns such as edges, textures or complex structures.

After convolutional layers, *pooling layers* further reduce the spatial dimensions while retaining the most salient information. A common technique is *max pooling*, which takes the maximum value in a local region. This operation enhances translational invariance and reduces computational cost.

A typical CNN architecture consists of multiple convolutional and pooling layers followed by fully connected layers, which aggregate extracted features for final classification or regression. Given an input image, early convolutional layers detect low-level features (e.g., edges), while deeper layers capture high-level patterns (e.g., object parts).

CNNs have improved performance on tasks such as image classification, object detection, and segmentation. Their hierarchical structure efficiently extracts spatial features while reducing the number of trainable parameters compared to fully connected networks when applied directly to images. For further details, see [36].

2.3.3 Vision Transformers

While CNNs efficiently capture local spatial patterns using convolutional filters, they struggle with long-range dependencies and global context modeling. Vision Transformers (ViTs) [16] address this limitation by applying the *self-attention mechanism* from natural language processing (NLP) to image data [37]. Instead of using convolutional layers, ViTs process an image as a sequence of non-overlapping *patches*, enabling them to model global relationships between distant regions.

Figure 2.8 illustrates the workflow of a ViT. Given an input image $\mathcal{I}$, it is divided into N equal-sized patches $\mathcal{P}_i$, $i = 1, \ldots, N$. Each patch is flattened into a vector and projected into an embedding space via a learnable linear transformation, producing token vectors $\mathbf{e}_i \in \mathbb{R}^M$. These embeddings also include a positional encoding to retain spatial information, as the self-attention mechanism does not encode positional relationships [37].

The token vectors are grouped in the embedding matrix

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_1 & \ldots & \mathbf{e}_N \end{bmatrix} \in \mathbb{R}^{M \times N} . \tag{2.20}$$

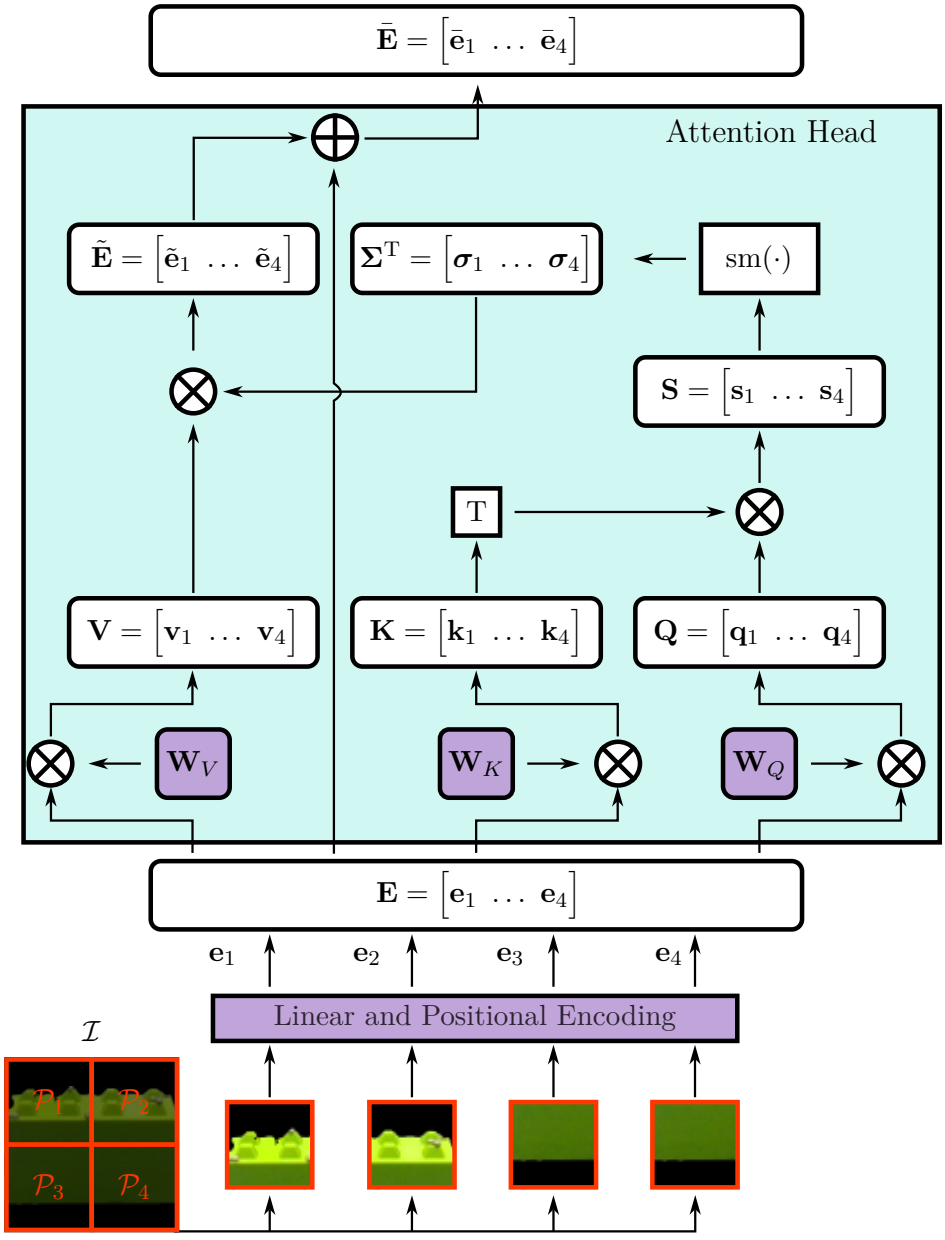


Figure 2.8: An Attention Head in a ViT [16]. The learnable parameters are coloured in violet.

The sequence of embeddings is then processed by a self-attention head, which learns how different patches relate to each other. Each token $\mathbf{e}_i$ is linearly transformed into a *query vector* $\mathbf{q}_i \in \mathbb{R}^m$ and into a corresponding *key vector* $\mathbf{k}_i \in \mathbb{R}^m$, $m < M$ and $i = 1, \ldots, N$, with

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \ldots & \mathbf{q}_N \end{bmatrix} = \mathbf{W}_Q \mathbf{E} \in \mathbb{R}^{m \times N} , \tag{2.21a}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_1 & \ldots & \mathbf{k}_N \end{bmatrix} = \mathbf{W}_K \mathbf{E} \in \mathbb{R}^{m \times N} , \tag{2.21b}$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{m \times M}$ are learnable matrices. The dot products $S_{ij} = \mathbf{k}_i^{\mathrm{T}} \mathbf{q}_j$ provide a measure of how much context $\mathcal{P}_i$ can give to $\mathcal{P}_j$, of how much the key patch $\mathcal{P}_i$ "attends" to the query patch $\mathcal{P}_j$. The dot products are gathered in the similarity matrix

$$\mathbf{S} = \mathbf{K}^{\mathrm{T}} \mathbf{Q} = \begin{bmatrix} \mathbf{s}_1 & \ldots & \mathbf{s}_N \end{bmatrix} \in \mathbb{R}^{N \times N} . \tag{2.22}$$

Each similarity vector $\mathbf{s}_i$ is fed into the softmax function $\mathrm{sm}(\cdot)$ to obtain the attention weight matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathrm{sm}(\mathbf{s}_1) & \ldots & \mathrm{sm}(\mathbf{s}_N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma}_1 & \ldots & \boldsymbol{\sigma}_N \end{bmatrix} \in \mathbb{R}^{N \times N} . \tag{2.23}$$

The entry Σ_{ij} indicates the probability that the key patch $\mathcal{P}_i$ can give context to the query patch $\mathcal{P}_j$. To encode the context a patch $\mathcal{P}_i$ might give another patch, each token $\mathbf{e}_i$ is transformed into a value vector $\mathbf{v}_i \in \mathbb{R}^M$ with

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_N \end{bmatrix} = \mathbf{W}_V \mathbf{E} \in \mathbb{R}^{N \times N} . \tag{2.24}$$

The variation of the query embedding

$$\tilde{\mathbf{e}}_i = \mathbf{V} \boldsymbol{\sigma}_i = \sum_{j=1}^{N} \mathbf{v}_j \Sigma_{ji} \tag{2.25}$$

represents the change in context for the query patch $\mathcal{P}_i$ due to the key patch $\mathcal{P}_j$, which is added to the original embedding

$$\bar{\mathbf{e}}_i = \mathbf{e}_i + \tilde{\mathbf{e}}_i . \tag{2.26}$$

The change for all embeddings is summarised in

$$\bar{\mathbf{E}} = \mathbf{E} + \mathbf{V} \boldsymbol{\Sigma} , \tag{2.27}$$

concluding the attention head.

In practice, multi-head self-attention is used, where multiple independent attention heads capture different aspects of the input relationships. This improves the model's ability to learn complex dependencies. For more details, the reader is referred to [16, 36, 37].

Unlike CNNs, which focus on local feature extraction, ViTs can model global interactions between distant image regions, making them highly effective for complex vision tasks. However, their main drawbacks include a large number of parameters, requiring substantial training data and a high computational cost during inference, due to the quadratic complexity of self-attention.

3 High-Speed Pose Tracking

This chapter presents the architecture of the high-speed pose tracking pipeline, as illustrated in Figure 3.1. To ensure broad applicability and real-time performance, the pipeline is implemented using Robot Operating System 2 Humble (ROS 2 Humble).

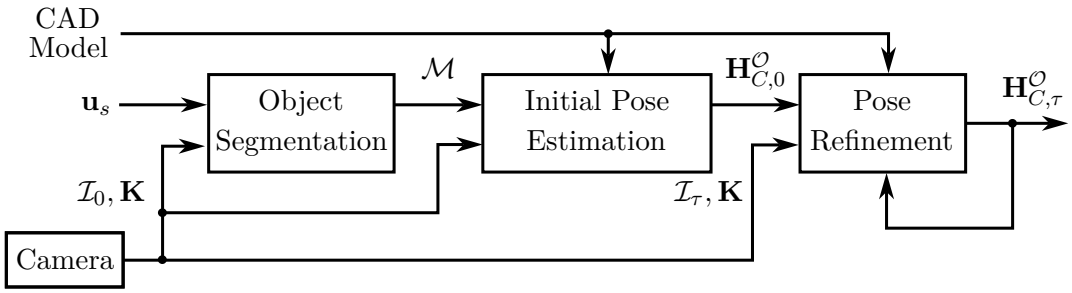


Figure 3.1: Pipeline architecture.

The pipeline assumes that a static object $\mathcal{O}$ with a known CAD model is within the camera's field of view. To initialize tracking, the user selects a point $\mathbf{u}_s \in \mathbb{U}_{H,W}$ on the object in a frame $\mathcal{I}_0 \in \mathbb{D}^{3\times H\times W}$. This selection, combined with the initial image, is used to generate a binary object mask $\mathcal{M} \in \{0,1\}^{H\times W}$.

The initial pose estimation module uses the mask, initial image, CAD model, and the camera calibration matrix $\mathbf{K}$ to compute the object's initial pose $\mathbf{H}^{\mathcal{O}}_{\mathcal{C},0}$ relative to the camera. From this point onward, an index $\tau \in \mathbb{N}$ is introduced to denote the sequence of incoming camera frames $\mathcal{I}_\tau$, allowing for object motion. The pipeline then iteratively refines the estimated pose $\mathbf{H}^{\mathcal{O}}_{\mathcal{C},\tau}$. The following sections describe the key pipeline components in detail.

3.1 RGB Camera and User Interface

The pipeline begins with an RGB camera with its coordinate frame $\mathcal{C}$, which continuously streams images $\mathcal{I}$ of resolution $H \times W$ alongside its intrinsic calibration matrix $\mathbf{K}$. The user can configure the image resolution and frame rate. A graphical user interface (GUI) in RViz2 [38] displays the camera feed in real-time across three separate windows: the first two show the raw camera feed, while the third presents the segmentation results.

The first GUI window (Figure 3.2a) serves as the user input interface, where the user

(a) Click Input for Segmentation. (b) Segmentation Mask. (c) Estimated Bounding Box.

Figure 3.2: Windows of the pipeline GUI implemented in RViz2.

selects a pixel $\mathbf{u}_s$ to mark an object for segmentation. The frame visible at the moment of selection serves as the initial image $\mathcal{I}_0$ for object segmentation and initial pose estimation. Once available, the resulting segmentation mask is visualised in the second window (Figure 3.2b).

Following initial pose estimation, the pipeline processes subsequent frames, indexed by $\tau = 1, 2, \ldots$, where pose refinement begins. A bounding box representing each new pose estimation is overlaid on the current video frame and displayed in the third window of the user interface (Figure 3.2c).

3.2 Object Segmentation

Many pose estimation algorithms require an object segmentation mask as input, as they do not independently locate objects. The object segmentation method must therefore be adaptable and capable of generating accurate masks for a diverse range of objects. Given an image $\mathcal{I}_0 \in \mathbb{D}^{3\times H\times W}$, an object *segmentation mask* $\mathcal{M} \in \{0,1\}^{H\times W}$ is a binary array where each pixel is classified as either belonging to the object of interest (1) or the background (0). To achieve this level of adaptability and precision, the pipeline employs Meta's neural network, Segment Anything (SAM).

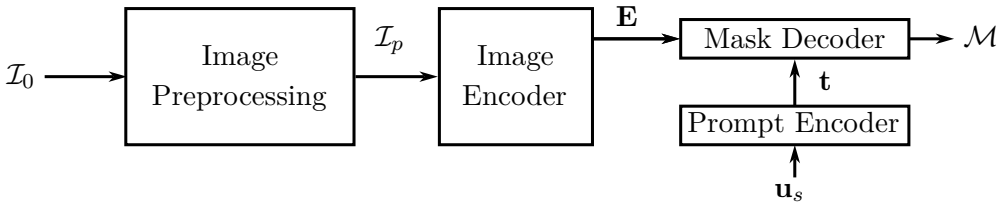


Figure 3.3: Overview of Segment Anything [32].

Figure 3.3 illustrates the workflow of SAM [32], which performs segmentation based on an input image $\mathcal{I}_0$ and optionally provided segmentation prompts. These prompts can take various forms, including foreground or background points, bounding boxes, or even textual descriptions. In this pipeline, the prompt consists of the selected pixel $\mathbf{u}_s$. Initially, SAM

resizes and pads the input image into $\mathcal{I}_p \in \mathbb{D}^{3\times D\times D}$ before employing a ViT-based encoder with windowed attention to generate feature embeddings $\mathbf{E} \in \mathbb{R}^{M\times d\times d}$ (2.27). Here, M denotes the dimensionality of the embeddings, while D and d represent the resolution of the processed image and image embedding, respectively. A separate prompt encoder processes the input prompts into a token vector $\mathbf{t} \in \mathbb{R}^M$. By applying a combination of self-attention, cross-attention layers, and multilayer perceptrons (MLPs), SAM predicts multiple mask candidates and assigns a confidence score to each. The highest-scoring mask $\mathcal{M}$ is selected. For further implementation details, the reader is referred to [32].

3.3 Initial Pose Estimation

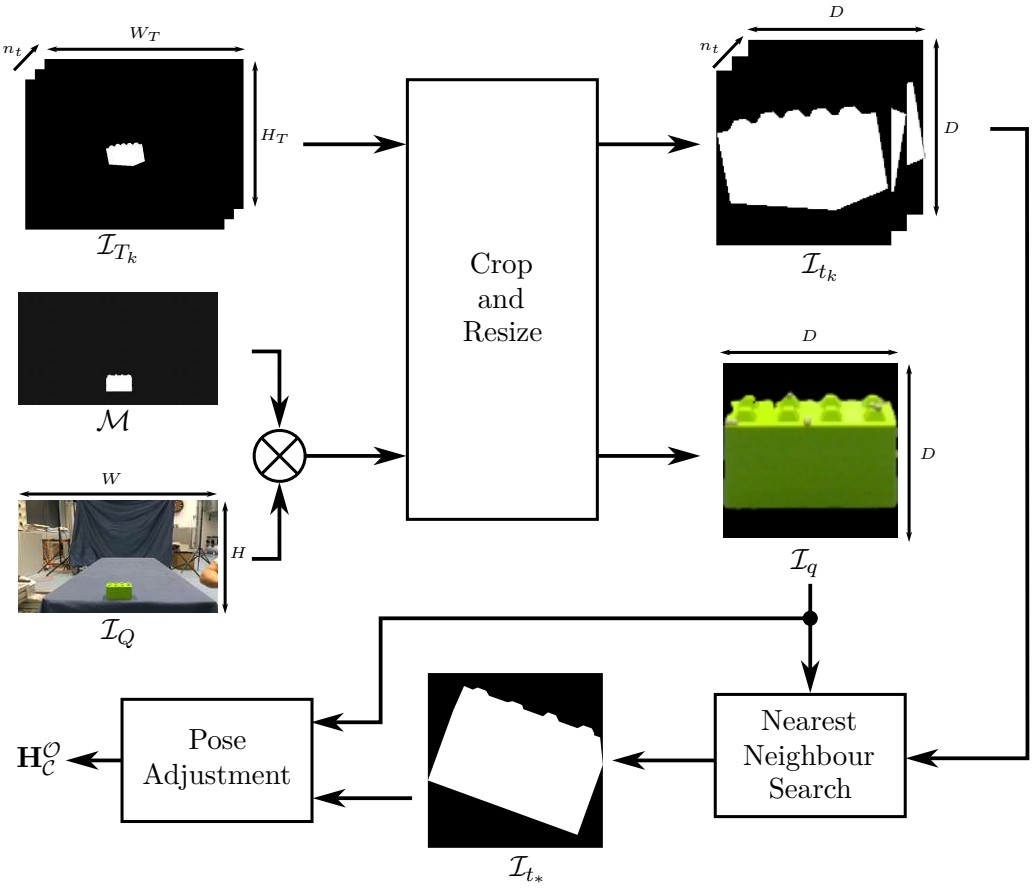


Figure 3.4: Model-based pose estimation with an RGB camera as done in [15].

A flexible pose tracking system requires a reliable method for initial pose estimation that can handle various object types. For this purpose, the CAD model-based neural network GigaPose [15] is chosen. This network generates a pose hypothesis when provided with the camera's intrinsic matrix $\mathbf{K}$, the query image $\mathcal{I}_Q = \mathcal{I}_0$, the object's binary segmentation mask $\mathcal{M}$, and the CAD model. Figure 3.4 illustrates the workflow of GigaPose, which is

summarised as follows.

3.3.1 Preprocessing

GigaPose first renders n_t template images $\mathcal{I}_{T_k} \in \mathbb{D}^{3\times H_T \times W_T}$, $k = 1, \ldots, n_t$ from the object's CAD model using predefined object poses $\mathbf{H}^{\mathcal{O}}_{\mathcal{T},k}$ relative to a virtual camera $\mathcal{T}$ with calibration matrix $\mathbf{K}_T$. The binary object mask $\mathcal{M}$ is applied to isolate the object in the query image. The template and the masked query images are then cropped and resized with the similarity transformations (2.7) $\mathbf{M}^{Q}_{q}$ and $\mathbf{M}^{T_k}_{t_k}$ to obtain images of the same dimensions $\mathcal{I}_q, \mathcal{I}_{t_k} \in \mathbb{D}^{3\times D\times D}$, in which the object is well framed.

3.3.2 Nearest Neighbour Search

A multi-head attention ViT-based network $\mathcal{F}_{\text{ae}}$ estimates the DOFs for out-of-plane rotation, being the **a**zimuth and **e**levation angles (hence the subscript $(\cdot)_{\text{ae}}$) by identifying the template $\mathcal{I}_{t_*}$ depicting the object from the most similar view to the query image. Therefore, $\mathcal{F}_{\text{ae}}$ divides the cropped images $\mathcal{I}_q$ and $\mathcal{I}_{t_k}$ each into N equally sized patches, to which a binary value is assigned, indicating whether each patch contains pixels belonging to the object. The ViT produces the output features (2.27)

$$\mathbf{E}_q = \begin{bmatrix} \mathbf{e}_{q,1} & \cdots & \mathbf{e}_{q,N} \end{bmatrix} \in \mathbb{R}^{M\times N} \tag{3.1}$$

for the cropped query image and

$$\mathbf{E}_k = \begin{bmatrix} \mathbf{e}_{k,1} & \cdots & \mathbf{e}_{k,N} \end{bmatrix} \in \mathbb{R}^{M\times N} \tag{3.2}$$

for each cropped template. In each query-template pair $(\mathbf{E}_q, \mathbf{E}_k)$, feature matching between the query image and the template is performed for each query feature $\mathbf{e}_{q,i}$ by maximizing the cosine similarity

$$\beta(\mathbf{e}_{q,i}, \mathbf{e}_{k,j}) = \frac{(\mathbf{e}_{q,i})^{\mathrm{T}}\mathbf{e}_{k,j}}{\|\mathbf{e}_{q,i}\|_2 \|\mathbf{e}_{k,j}\|_2}\,, \tag{3.3}$$

which is the cosine value of the angle between the vectors $\mathbf{e}_{q,i}$ and $\mathbf{e}_{k,j}$. The template $\mathcal{I}_{T_*}$ that achieves the highest average cosine similarity for the matching features is defined as the query's nearest neighbour. Figure 3.5 illustrates the relationship between the query image and its nearest neighbour. The main difference in the view between $\mathcal{I}_{T_*}$ and $\mathcal{I}_q$ lies in the scale and the in-plane rotation. Therefore, $\mathbf{R}^{\mathcal{O}}_{\mathcal{T}_*}$ of the nearest neighbour pose is a good initial estimation for the rotation of the real pose $\mathbf{H}^{\mathcal{O}}_{\mathcal{C}}$, as it typically approximates the 2 DOFs of the out-of-plane rotation. To achieve good results, $\mathcal{F}_{\text{ae}}$ is trained to be invariant to in-plane rotation, scale and translation.

3.3.3 Pose Adjustment

Figure 3.5 also shows the relationships between the different poses and images of the object. At this stage, the nearest neighbour pose $\mathbf{H}^{\mathcal{O}}_{\mathcal{T}_*}$ and the similarity transforms $\mathbf{M}^{T_*}_{t_*}$

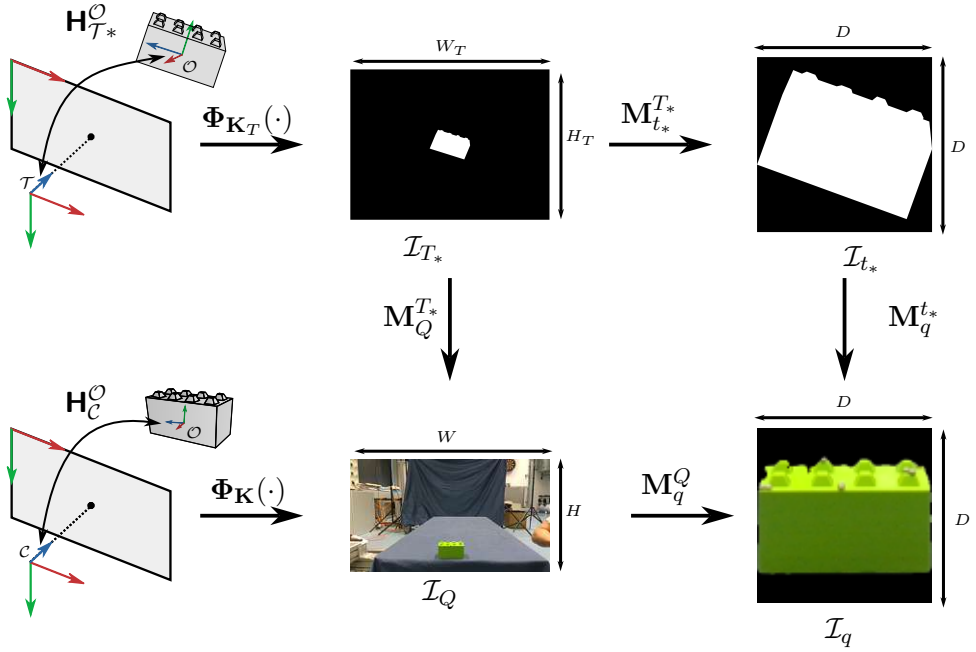


Figure 3.5: The relationship between the transformations.

and $\mathbf{M}^{Q}_{q}$ are known. It becomes apparent that finding an estimate for the similarity transformation $\mathbf{M}^{t_*}_{q}$ between the nearest neighbour and the cropped query image, which contains the remaining 4 DOFs, allows the computation of a pose estimate $\mathbf{H}^{\mathcal{O}}_{\mathcal{C}}$. These DoFs, being the **i**n-plane rotation angle, the **s**cale, and the two **t**ranslation parameters (resulting in the subscript $(\cdot)_{\text{ist}}$), are estimated by a CNN-based network $\mathcal{F}_{\text{ist}}$, which is trained to be invariant to out-of-plane rotation.

With

$$\mathbf{M}^{T_*}_{Q} = \left(\mathbf{M}^{Q}_{q}\right)^{-1}\mathbf{M}^{t_*}_{q}\mathbf{M}^{T_*}_{t_*} = \begin{bmatrix} s_* \cos(\alpha_*) & -s_* \sin(\alpha_*) & x_* \\ s_* \sin(\alpha_*) & s_* \cos(\alpha_*) & y_* \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

being the full similarity transformation (2.7) between the original template $\mathcal{I}_{T_*}$ and query image $\mathcal{I}_Q$, the estimation for the object-to-camera rotation is obtained by rotating around the camera's z-axis with

$$\mathbf{R}^{\mathcal{O}}_{\mathcal{C}} = \mathbf{R}_z(\alpha_*)\mathbf{R}^{\mathcal{O}}_{\mathcal{T}*} \ , \tag{3.5}$$

where $\mathbf{R}_z(\alpha_*)$ denotes the rotation by angle α_* around the z-axis. The frame of the nearest neighbour is (approximately) aligned with the real pose with $\mathbf{x}_{\text{temp}} = \mathbf{R}_z(\alpha_*)\mathbf{x}^{\mathcal{O}}_{\mathcal{T}*}$. The depth estimate is obtained with

$$z^{\mathcal{O}}_{\mathcal{C}} = \frac{z_{\text{temp}}}{s_*}\frac{f_x}{f_{x,T}} \ . \tag{3.6}$$

Here, z_{temp} denotes the z-component of $\mathbf{x}_{\text{temp}}$, while f_x and $f_{x,T}$ are the focal lengths of $\mathcal{C}$ and $\mathcal{T}$, respectively. The centre of the reoriented frame is projected onto the template

image plane and transformed onto the query image using the estimated similarity transform with

$$\begin{bmatrix} \mathbf{u}_{c,Q} \\ 1 \end{bmatrix} = \mathbf{M}_Q^{T_*} \mathbf{\Phi}_{\mathbf{K}_T}(\mathbf{x}_{\text{temp}}) \ . \tag{3.7}$$

Finally, the 3D translation is computed with the inverse projection of $\mathcal{C}$

$$\mathbf{x}_{\mathcal{C}}^{\mathcal{O}} = \mathbf{\Phi}_{\mathbf{K}}^{-1}\left(\mathbf{u}_{c,Q}, z_{\mathcal{C}}^{\mathcal{O}}\right) \tag{3.8}$$

completing the pose estimate $\mathbf{H}_{\mathcal{C}}^{\mathcal{O}}$ with (3.5), which serves as the initial pose estimation $\mathbf{H}_{\mathcal{C},0}^{\mathcal{O}}$, as depicted in Figure 3.1. A more detailed explanation is given in [15].

3.4 Pose Refinement

After obtaining the initial pose estimation $\mathbf{H}_{\mathcal{C},0}^{\mathcal{O}}$, the object can now move relative to the camera. The algorithm tracking the pose of the moving object in each subsequent image frame $\mathcal{I}_\tau$ for $\tau = 1, 2, \ldots$ must not only be accurate but also time efficient to keep up with camera frame rate and the object's speed.

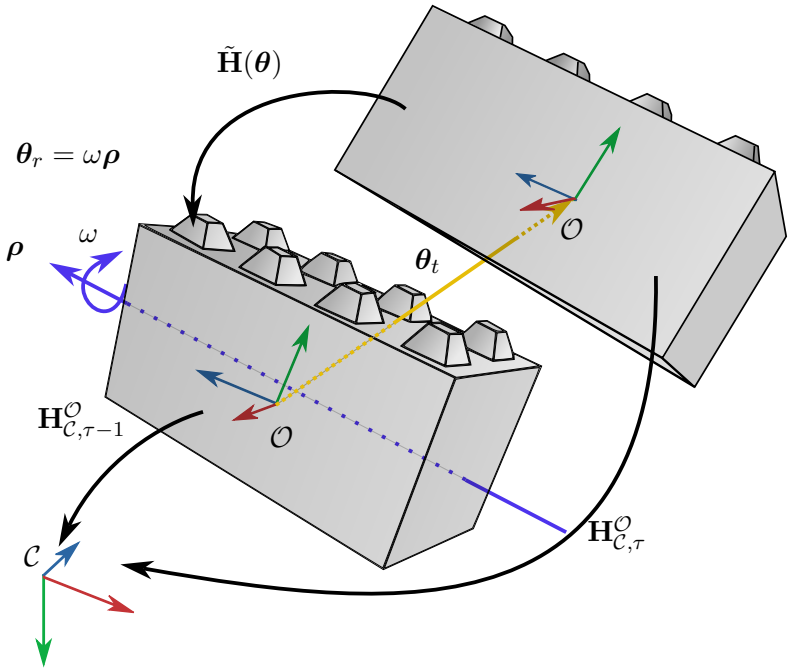


Figure 3.6: Pose Variation $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_t^{\mathrm{T}} & \boldsymbol{\theta}_r^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ between two consecutive frames $\mathcal{I}_{\tau-1}$ and $\mathcal{I}_\tau$ as defined in [24].

To achieve this, the tracker proposed in [24] was selected. It estimates the translation $\boldsymbol{\theta}_t \in \mathbb{R}^3$ and the rotation $\boldsymbol{\theta}_r = \omega\boldsymbol{\rho}$ of the object between consecutive frames $\mathcal{I}_{\tau-1}$ and $\mathcal{I}_\tau$. Here, $\omega \in \mathbb{R}$ is the rotation angle, and $\boldsymbol{\rho} \in \mathbb{R}^3$ is the normalized rotation axis. The

pose variation vector is defined as $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_t^{\mathrm{T}} & \boldsymbol{\theta}_r^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, which allows the computation of the object's pose transformation between consecutive frames as

$$\tilde{\mathbf{H}}(\boldsymbol{\theta}) = \begin{bmatrix}\exp(\mathbf{S}(\boldsymbol{\theta}_r)) & \boldsymbol{\theta}_t \\ \mathbf{0} & 1\end{bmatrix} , \tag{3.9}$$

where $\mathbf{S}(\mathbf{x})$ is the skew-symmetric matrix of a vector $\mathbf{x}$, defined as

$$\mathbf{S}(\mathbf{x}) = \begin{bmatrix}0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0\end{bmatrix} , \quad \mathbf{x} = \begin{bmatrix}x_1 \\ x_2 \\ x_3\end{bmatrix} . \tag{3.10}$$

The current pose estimate is then updated with

$$\mathbf{H}_{\mathcal{C},\tau}^{\mathcal{O}}(\boldsymbol{\theta}) = \mathbf{H}_{\mathcal{C},\tau-1}^{\mathcal{O}}\tilde{\mathbf{H}}(\boldsymbol{\theta}) , \quad \tau \in \mathbb{N}\backslash\{0\} . \tag{3.11}$$

This is illustrated in Figure 3.6, with the silhouette of the object from the previous frame and the current frame overlaid onto the image plane.

The following sections describe the steps involved in estimating the current pose $\mathbf{H}_{\mathcal{C},\tau}^{\mathcal{O}}$, given the previous and current frames $\mathcal{I}_{\tau-1}$ and $\mathcal{I}_\tau$, the pose estimate $\mathbf{H}_{\mathcal{C},\tau-1}^{\mathcal{O}}$ in $\mathcal{I}_{\tau-1}$, and the camera intrinsics $\mathbf{K}$.

3.4.1 Nearest Neighbour Search and Colour Histogram

Similar to Section 3.3.2, a nearest neighbour search is involved. The algorithm presented in [24] computes $n_v \in \mathbb{N}$ sparse viewpoint models $\mathcal{R}_k$ for different template poses $\mathbf{H}_{\mathcal{T},k}^{\mathcal{O}}$ before the tracking process starts, where $k = 1, \ldots, n_v$. With the estimation $\mathbf{H}_{\mathcal{C},\tau-1}^{\mathcal{O}}$ of the previous refinement step, the sparse viewpoint model $\mathcal{R}$ of the template pose $\mathbf{H}_{\mathcal{T}}^{\mathcal{O}}$ with the most similar orientation to $\mathbf{R}_{\mathcal{C},\tau-1}^{\mathcal{O}}$ is selected and projected onto $\mathcal{I}_{\tau-1}$ with $\boldsymbol{\Phi}_{\mathbf{K}}\left(\mathbf{H}_{\mathcal{C},\tau-1}^{\mathcal{O}}(\cdot)\right)$. The resulting projected viewpoint model Ω is illustrated in Figure 2.4. Colour histograms for $\mathcal{G}_f$ and $\mathcal{G}_b$ are extracted along all projected correspondence lines. Each colour channel is divided into $n_h \in \mathbb{N}$ equidistant bins. The extraction region γ_f of Figure 2.2 corresponds to a set of pixel segments along all correspondence lines lying on $\mathcal{G}_f$. This set of pixel segments is defined by local coordinates $r \in [-n_p, \ldots, -1]$ with $n_p \in \mathbb{N}$. Similarly, the extraction region γ_b is defined by all pixel segments on correspondence lines with $r \in [1, \ldots, n_p]$. With this, the colour probability (2.10) is computed and used to update a learned colour probability [27]

$$p_\tau(\mathbf{y} \mid \mathcal{G}_i) = \beta_i p(\mathbf{y} \mid \mathcal{G}_i) + (1 - \beta_i)p_{\tau-1}(\mathbf{y} \mid \mathcal{G}_i) , \quad \mathbf{y} \in \mathbb{D}^3 , \quad i \in \{f, b\} , \tag{3.12}$$

where $\beta_i \in [0, 1]$ are learning rates for foreground and background, and $p_{\tau-1}(\mathbf{y} \mid \mathcal{G}_i)$ is the probability distribution used in the previous frame.

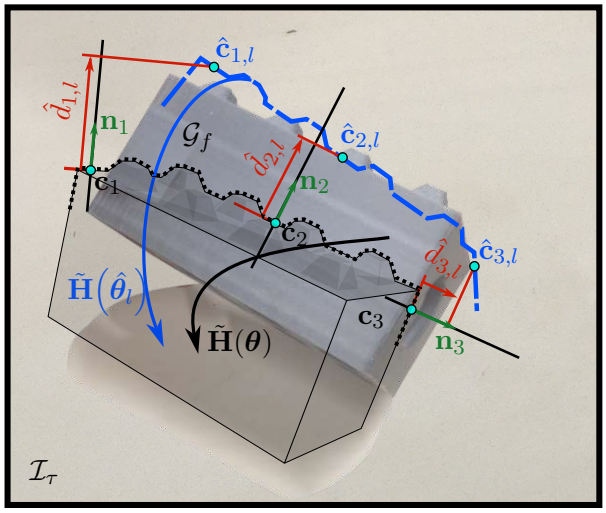


Figure 3.7: The variated contour distances on the current frame $\mathcal{I}_\tau$, as in[24]. The silhouette of the object in the previous frame is overlaid onto $\mathcal{I}_\tau$.

3.4.2 Optimizing for the Pose Variation

Upon receiving the current frame $\mathcal{I}_\tau$, where the object has moved relative to the last frame $\mathcal{I}_{\tau-1}$, the nearest neighbour viewpoint model $\mathcal{R}$ from $\mathcal{I}_{\tau-1}$ is projected onto $\mathcal{I}_\tau$. This projection, along with its contour points $\mathbf{c}_j$ and normal vectors $\mathbf{n}_j$, is then projected onto $\mathcal{I}_\tau$. The objective of [24] is to estimate the true pose variation $\boldsymbol{\theta}$ that best explains the colour distribution observed in the current frame. This process is illustrated in Figure 3.7.

The estimated variation $\hat{\boldsymbol{\theta}}$ is computed iteratively by solving an optimisation problem. In the l-th iteration, with $l = 0, 1, 2, \ldots$, the contour points of $\mathcal{R}$ are projected onto $\mathcal{I}_\tau$ using $\boldsymbol{\Phi}_{\mathbf{K}}\left(\hat{\mathbf{H}}_l(\cdot)\right)$. Here, $\hat{\mathbf{H}}_l$ is the object-to-camera pose due to the pose variation $\hat{\boldsymbol{\theta}}_l$ computed with (3.11). This projection results in variated contour points $\hat{\mathbf{c}}_{j,l} = \hat{\mathbf{c}}_j\left(\hat{\boldsymbol{\theta}}_l\right)$. The distances $\hat{d}_{j,l} = \hat{d}_j\left(\hat{\boldsymbol{\theta}}_l\right)$ between $\mathbf{c}_j$ and $\hat{\mathbf{c}}_{j,l}$ along the correspondence lines are combined with the updated histogram (3.12) and (2.17) to compute the log-posterior

$$f\left(\hat{\boldsymbol{\theta}}_l\right) = \ln\left\{p\left[\hat{d}_1\left(\hat{\boldsymbol{\theta}}_l\right) \wedge \cdots \wedge \hat{d}_{n_c}\left(\hat{\boldsymbol{\theta}}_l\right) \mid \mathcal{D}\right]\right\} \tag{3.13}$$

which is to be maximized. The optimisation iteration is performed using the Newton Method

$$\hat{\boldsymbol{\theta}}_{l+1} = \hat{\boldsymbol{\theta}}_l + \left(-\nabla^2 f\left(\hat{\boldsymbol{\theta}}_l\right) + \begin{bmatrix} \lambda_r \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \lambda_t \mathbf{I}_3 \end{bmatrix}\right)^{-1} \nabla f\left(\hat{\boldsymbol{\theta}}_l\right), \quad \hat{\boldsymbol{\theta}}_0 = \mathbf{0}\,. \tag{3.14}$$

Here, λ_r and λ_t are Tikhonov regularization parameters, $\nabla f(\boldsymbol{\theta}_l)$ is the gradient and $\nabla^2 f(\boldsymbol{\theta}_l)$ is the Hessian matrix evaluated at $\boldsymbol{\theta}_l$. Full implementation details are available in [27].

4 Setup and Experiments

In this chapter, the accuracy of the pose-tracking pipeline is tested by comparing its pose output to the pose measurement of the motion capture system Optitrack using two different objects in different setups. Finally, the tracker is used to grasp an object with a gripper mounted on a 7-DoF robotic arm.

4.1 Setup

Each block from the pipeline in Figure 3.1 is implemented as a node in ROS2 Humble on a computer with Ubuntu 24.04.2 and equipped with 128 GB of RAM, 32 Intel® Core™ i9 CPUs, and an NVIDIA GeForce RTX 4080 GPU. An RViz2 interface displays the current camera frame and lets the user select the object of interest by clicking on it, initiating the tracking process that starts with the object segmentation. After the tracking is started, the interface also depicts the estimated bounding box overlaid on the original camera image. The images, mask and estimated object-to-camera poses are published and saved in rosbags throughout the measurements.

For accuracy measurements, the pipeline's estimated pose is compared to the measurement provided by the motion capture system Optitrack that serves as ground truth. Six cameras are used to obtain measurements with submillimeter accuracy. Since Optitrack requires a Windows architecture, the computer used for Optitrack and the Ubuntu PC are connected via Ethernet. The motion capture measurements are published to the ROS2 domain with a specialized node[1].

The lightweight 7-DoF KUKA LBR iiwa robotic arm is equipped with a Robotiq 2-Finger Gripper and is used for experimental validation of the pose tracker by tracking and grasping an object with visual feedback provided by the pose-tracking-pipeline.

All experiments process the colour images from an Intel RealSense D435i camera with a fixed focal length.

4.2 Pose Tracking Accuracy

Figure 4.1 shows the setup for measuring the accuracy of the pipeline. The Intel RealSense D435i streams the colour images while tracking one of the two objects from this image. The object on the left is the Scrub Cleanser Bottle from the YCB Object Dataset [39],

[1]MOCAP4ROS2-Project `https://github.com/MOCAP4ROS2-Project`

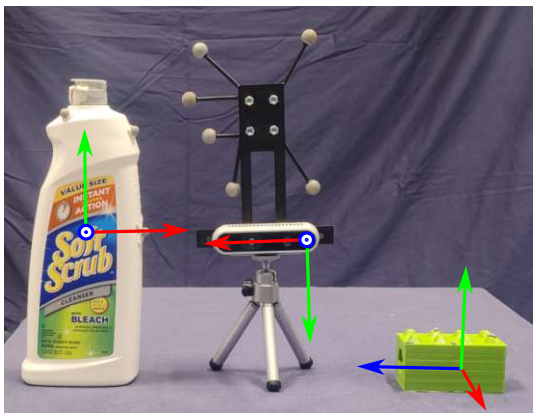


Figure 4.1: The two objects and the camera used in the accuracy experiments. The x, y and z axes of the intrinsic body-frames are drawn in red, green and blue, respectively.

while the object to the right is a 3D-printed miniature model of an interlocking concrete security block. These objects were chosen to compare the tracker's performance on objects with differing levels of symmetry and texture. The tracking pipeline provides the estimated pose $\hat{\mathbf{H}}_{\mathcal{C}}^{\mathcal{O}}(t)$ at time t.

Markers reflecting the infrared light emitted by the cameras of the motion capture system Optitrack are mounted on those objects. The position of the markers is then used to associate the objects and the camera to their respective Optitrack body frames $\mathcal{O}_b$ and $\mathcal{C}_b$. Optitrack measures the poses $\mathbf{H}_{\mathcal{W}}^{\mathcal{O}_b}(t)$ and $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}_b}(t)$ of these frames relative to a static world reference frame $\mathcal{W}$.

These frames differ from the object's intrinsic frames $\mathcal{O}$ and $\mathcal{C}$. They are related by

$$\mathbf{H}_{\mathcal{W}}^{\mathcal{O}}(t) = \mathbf{H}_{\mathcal{W}}^{\mathcal{O}_b}(t)\mathbf{H}_{\mathcal{O}_b}^{\mathcal{O}} \ , \tag{4.1a}$$

$$\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}(t) = \mathbf{H}_{\mathcal{W}}^{\mathcal{C}_b}(t)\mathbf{H}_{\mathcal{C}_b}^{\mathcal{C}} \ , \tag{4.1b}$$

where $\mathbf{H}_{\mathcal{C}_b}^{\mathcal{C}}$ and $\mathbf{H}_{\mathcal{O}_b}^{\mathcal{O}}$ are constant but unknown homogeneous matrices. Figure 4.2 illustrates the relationships between the different poses. The ground truth is then given by

$$\mathbf{H}_{\mathcal{C}}^{\mathcal{O}}(t) = \left(\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}(t)\right)^{-1}\mathbf{H}_{\mathcal{W}}^{\mathcal{O}}(t) \tag{4.2}$$

The accuracy is measured via the translational and angular errors

$$\mathbf{e}_t(t) = \mathbf{x}_{\mathcal{C}}^{\mathcal{O}}(t) - \hat{\mathbf{x}}_{\mathcal{C}}^{\mathcal{O}}(t) \in \mathbb{R}^3 \tag{4.3a}$$

$$\mathbf{e}_r(t) = \Delta\omega(t)\boldsymbol{\rho}(t) \in \mathbb{R}^3 \tag{4.3b}$$

of the pose estimation relative to the ground truth. Here, $\Delta\omega(t) \in \mathbb{R}$ is the error angle and $\boldsymbol{\rho}(t) \in \mathbb{R}^3$ denotes the normalized error rotation axis.

The following sections explain how the matrices $\mathbf{H}_{\mathcal{C}_b}^{\mathcal{C}}$ and $\mathbf{H}_{\mathcal{O}_b}^{\mathcal{O}}$ are determined and present the experimental results.

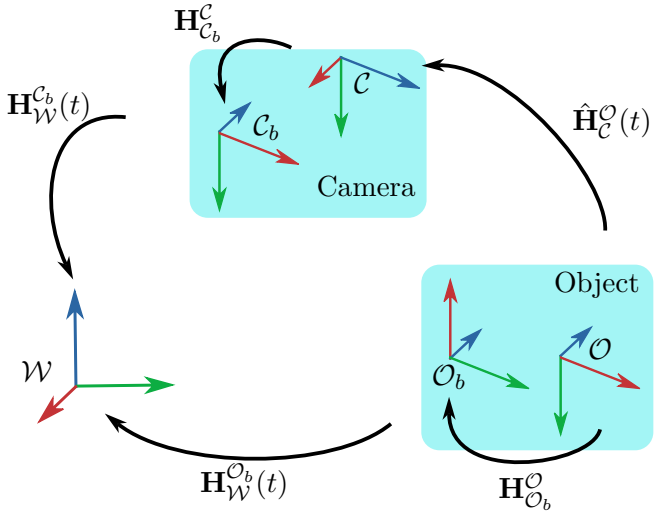


Figure 4.2: Transformations between the frames of Optitrack, object and camera models.

4.2.1 Calibrating the Pose Measurement

Given the setup shown in Figure 4.2, the matrices $\mathbf{H}^{\mathcal{C}}_{\mathcal{C}_b}\left(\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b}\right)$ and $\mathbf{H}^{\mathcal{O}}_{\mathcal{O}_b}\left(\mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b}\right)$ are estimated indirectly by optimizing the pose parameters

$$\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b} = \begin{bmatrix} \boldsymbol{\theta}^{\mathcal{C}}_{\mathcal{C}_b} \\ \mathbf{x}^{\mathcal{C}}_{\mathcal{C}_b} \end{bmatrix} \quad \text{and} \quad \mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b} = \begin{bmatrix} \boldsymbol{\theta}^{\mathcal{O}}_{\mathcal{O}_b} \\ \mathbf{x}^{\mathcal{O}}_{\mathcal{O}_b} \end{bmatrix}. \tag{4.4}$$

Here, $\boldsymbol{\theta}^{\mathcal{A}}_{\mathcal{A}_b} \in \mathbb{R}^3$, $\mathcal{A} \in \{\mathcal{C}, \mathcal{O}\}$, represents the Roll-Pitch-Yaw angles of the rotation matrix $\mathbf{R}^{\mathcal{A}}_{\mathcal{A}_b}\left(\boldsymbol{\theta}^{\mathcal{A}}_{\mathcal{A}_b}\right)$ [33]. The homogeneous transformation

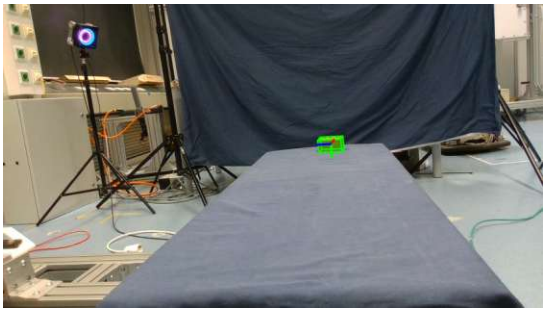
$$\bar{\mathbf{H}}\left(\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b}, \mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b}\right) = \left(\mathbf{H}^{\mathcal{C}_b}_{\mathcal{W}}\mathbf{H}^{\mathcal{C}}_{\mathcal{C}_b}\left(\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b}\right)\hat{\mathbf{H}}^{\mathcal{O}}_{\mathcal{C}}\right)^{-1}\mathbf{H}^{\mathcal{O}_b}_{\mathcal{W}}\mathbf{H}^{\mathcal{O}}_{\mathcal{O}_b}\left(\mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b}\right) \tag{4.5}$$

forms a closed kinematic chain, enabling the determination of the optimal pose vectors solving [40]

$$\begin{bmatrix} \mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b} \\ \mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b} \end{bmatrix} = \underset{\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b}, \mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b} \in \mathbb{R}^6}{\arg\min} \sum_{k=1}^{N} \left(\|\bar{\mathbf{x}}_k\|_2 + w \left\| \mathbf{I} - \bar{\mathbf{R}}_k \right\|_{\mathrm{F}} \right)^2, \tag{4.6}$$

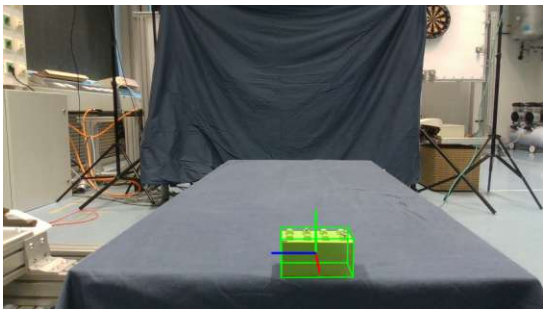
where N measurements $\bar{\mathbf{H}}_k$, $k = 1, \ldots, N$, are used. The terms $\|\cdot\|_2$ and $\|\cdot\|_{\mathrm{F}}$ denote the Euclidean and Frobenius norms, respectively.

To improve precision, the camera matrix $\mathbf{H}^{\mathcal{C}}_{\mathcal{C}_b}$ is first estimated independently of the object transformations using a standard pose estimation technique with ArUco markers [41]. In this step, a CharUco board replaces the object in Figure 4.2, with its reference frame manually aligned to the motion capture frame. The camera is then slowly moved around the board while estimating its pose $\hat{\mathbf{H}}^{\mathcal{O}}_{\mathcal{C},k}$ in each frame ($k = 1, \ldots, N$) using OpenCV's pose estimation algorithm. These measurements are used to optimize the pose parameters $\mathbf{p}^{\mathcal{C}}_{\mathcal{C}_b}$ for the camera and $\mathbf{p}^{\mathcal{O}}_{\mathcal{O}_b}$ for the CharUco board.



(a) 1.27 m

(b) 1.37 m



(c) 0.47 m

(d) 0.43 m

Figure 4.3: The block and the cleanser bottle placed at different distances from the camera for static pose estimation. The estimated bounding boxes and axes are projected onto the images.

For the actual objects of interest, the transformations $\mathbf{H}_{\mathcal{O}_b}^{\mathcal{O}}$ are computed by collecting training measurements $\hat{\mathbf{H}}_{\mathcal{C},k}^{\mathcal{O}}$ using the implemented tracking pipeline. The camera is again moved around the object after which (4.6) is solved, this time optimizing only for $\mathbf{p}_{\mathcal{O}_b}^{\mathcal{O}}$ while keeping $\mathbf{p}_{\mathcal{C}_b}^{\mathcal{C}}$ fixed to result of the camera optimization process.

4.2.2 Static Pose Tracking

Influence of Object-to-Camera Distance

Figure 4.3 illustrates the experimental setup, where objects are positioned at varying distances from the camera, which operates at a resolution of 720×1280. The estimated object poses are visualized by projecting their bounding boxes onto the images.

As shown in Table 4.1, the translation errors increase as the object moves farther from the camera, which aligns with expectations. The detailed measurement graphs Figure 4.4 and Figure 4.5 show that the pose tracker refines the initial pose estimate and converges after a few frames. The initial estimation error is random and independent of the object.

Interestingly, while the pose tracker improves positional accuracy for the cleanser bottle, the absolute position errors in depth estimation for the block increase after the initial pose

Table 4.1: Positional accuracies of static pose tracking for different object to camera distances.

		Near			Far		
		x	y	z	x	y	z
Block	Ground Truth Position [cm]	4.46	11.10	45.18	16.84	−2.25	126.15
	Mean Error [cm]	0.32	0.31	0.72	0.57	0.65	5.12
	Standard Deviation [cm]	0.01	0.02	0.06	0.13	0.04	0.74
	Relative Depth Error [%]	0.69	0.68	1.53	0.45	0.51	4.02
Bottle	Ground Truth Position [cm]	1.62	1.37	42.91	8.27	−6.60	136.87
	Mean Error [cm]	0.19	0.52	0.32	0.17	0.03	3.85
	Standard Deviation [cm]	0.01	0.05	0.29	0.06	0.05	1.01
	Relative Depth Error [%]	0.44	1.21	0.74	0.13	0.02	2.77

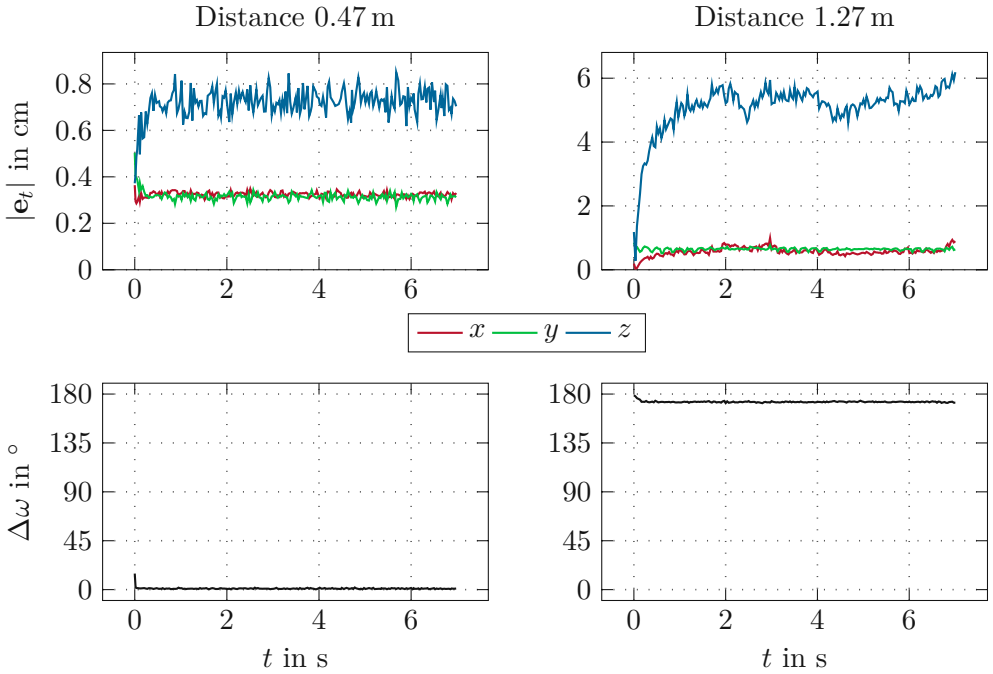


Figure 4.4: Static measurements with the block at different distances to the camera.

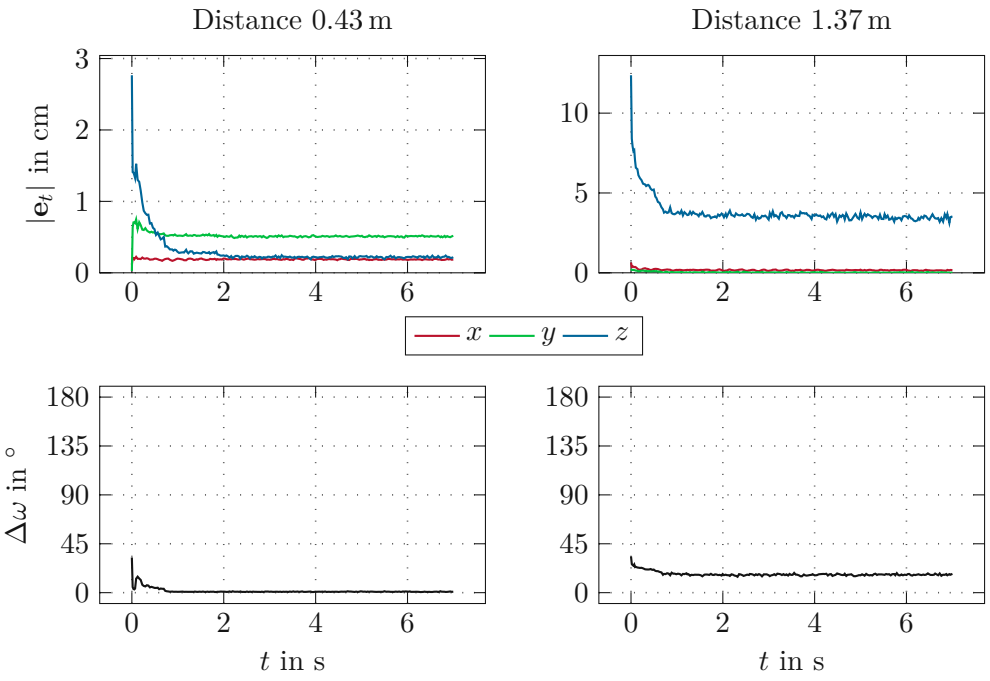


Figure 4.5: Static measurements with the cleanser bottle at different distances to the camera.

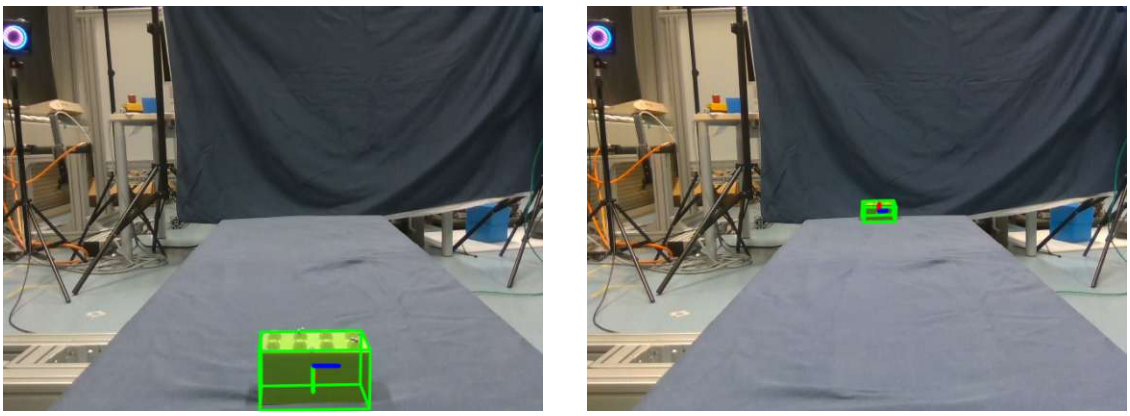
estimation. This effect is particularly pronounced when the block is positioned farther away. The likely cause is the object's small size in the image and the reduced contrast between foreground and background at greater distances, which diminish the effectiveness of the pose tracker.

Rotating the block by 180° around its x- or y-axis, or by $n \cdot 90°$, $n \in \mathbb{N}$, around its z-axis may lead to similar initial pose estimates. This explains the observed rotational error of nearly 180° around the estimated z-axis at a distance of 1.37 m (Figure 4.4). By rotating the estimated frame in Figure 4.3a adequately, the correct orientation can be restored. In contrast, the cleanser bottle, due to its asymmetry, yields a more stable and accurate orientation across different distances. Overall, static pose tracking performs more reliably for asymmetric objects.

Influence of the Camera's Resolution

When reducing the camera resolution to 480×640, one could expect the update rate to increase. However, the architecture of the neural networks in the initial pose estimator remains unchanged, while the computation of the region-based model and the optimization steps are resolution-independent so that the overall pipeline speed remains unaffected.

For static measurements, the cube was placed at distances of 0.42 m and 1.27 m at 480×640 (480p) (Figure 4.6) and the results were compared to those at 720×1280 (720p). As



(a) 0.42 m (b) 1.27 m

Figure 4.6: Static pose estimation of the block with a camera resolution of 480×640 at different distances. The estimated bounding boxes and axes are projected onto the images.

Table 4.2: Positional accuracies of static pose tracking for the block for different camera resolutions.

		Near			Far		
		x	y	z	x	y	z
720p	Ground Truth Position [cm]	4.46	11.10	45.18	16.84	−2.25	126.15
	Mean Error [cm]	0.32	0.31	0.72	0.57	0.65	5.12
	Standard Deviation [cm]	0.01	0.02	0.06	0.13	0.04	0.74
	Relative Depth Error [%]	0.69	0.68	1.53	0.45	0.51	4.02
480p	Ground Truth Position [cm]	2.81	12.0	39.77	3.68	1.91	126.94
	Mean Error [cm]	0.44	0.35	0.44	0.47	0.01	2.19
	Standard Deviation [cm]	0.05	0.08	0.31	0.06	0.05	0.85
	Relative Depth Error [%]	1.06	0.85	1.05	0.37	0.08	1.72

shown in Table 4.2, translational errors for the closer measurements are comparable across both resolutions. However, at a greater distance, translation estimates at 480p were more accurate than at 720p. This unexpected improvement may be because at lower resolutions, colour variations occur over fewer pixels along the correspondence line. As a result, the difference between foreground and background colour histograms becomes more pronounced under suboptimal lighting conditions. Finally, as in previous experiments, the block's symmetry causes orientation errors of 180° around an axis (Figure 4.7).

4.2.3 Dynamic Measurements

The tracking accuracy of the pipeline is evaluated through dynamic eye-in-hand tracking of two objects at resolutions of 480p and 720p. Figure 4.8 presents four frames from each

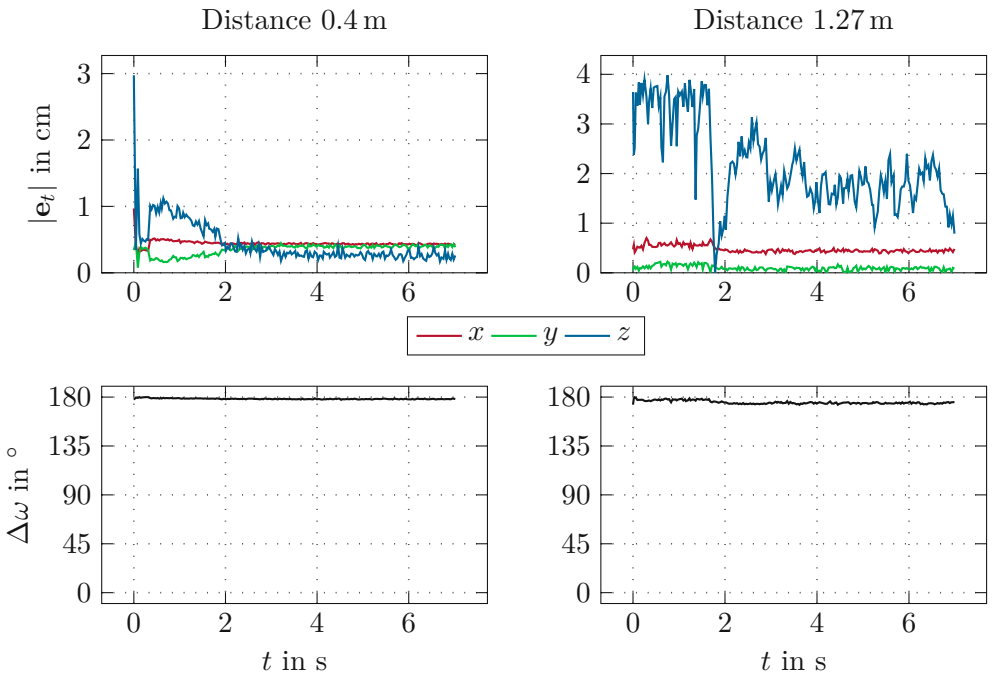
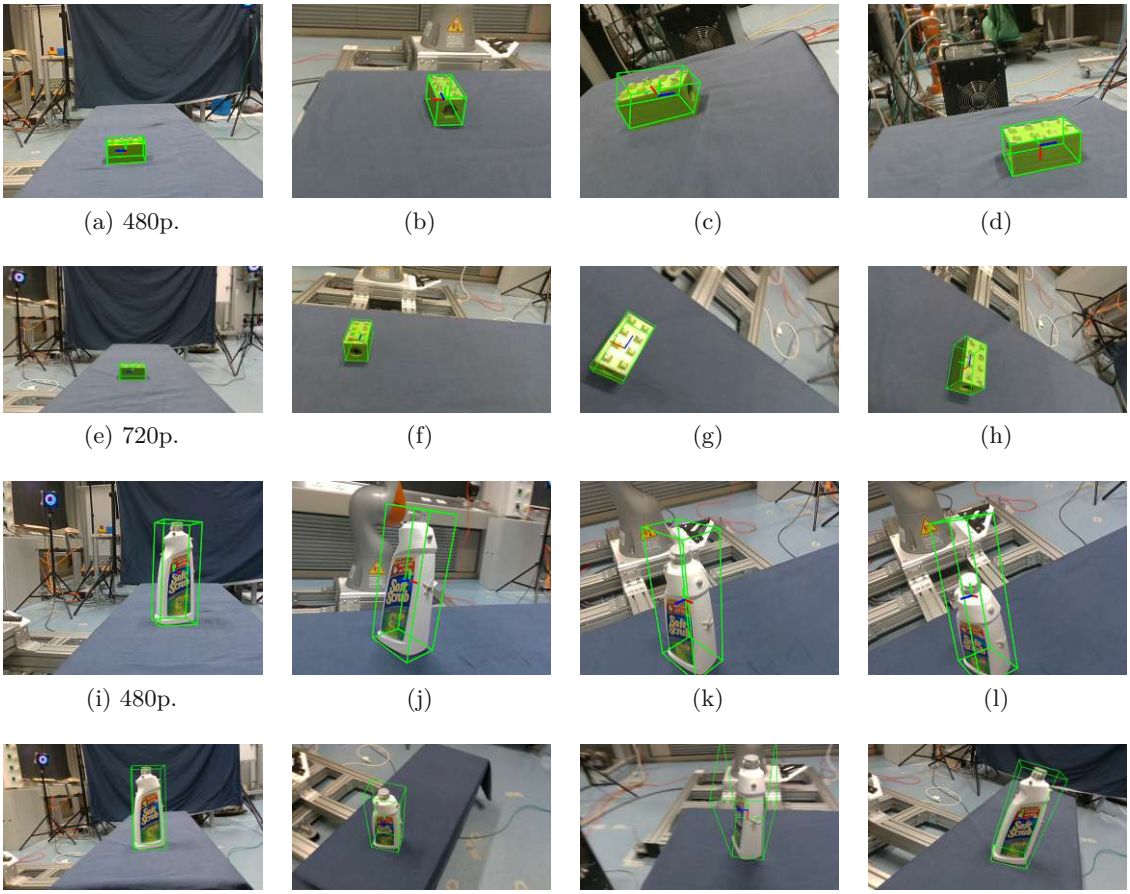


Figure 4.7: Static tracking error of the block with a resolution of 480p at different distances.

measurement in chronological order, while Figures 4.9 and 4.10 plot the measurement results for the block and the cleanser bottle, respectively. The positional error statistics are summarized in Table 4.3.

When tracking the block, the estimated orientation flips once during tracking, for both resolutions. At 480p, this occurs between the frames shown in Figures 4.8c and 4.8d, whereas at 720p, the flip happens between Figures 4.8g and 4.8h. These abrupt orientation changes are evident in the graphs depicting the error rotation axis components. This confirms that tracking the orientation of symmetrical objects remains unstable without additional adjustments. In terms of translational accuracy, the results indicate that higher resolution leads to lower average translation errors across all components.

When tracking the cleanser bottle, tracking failures occur in both measurements when the background colours closely resemble the object's colours (Figures 4.8k and 4.8o). However, at 480p, the object pose is permanently lost at this point (Figure 4.8l), leading to complete tracker failure. In contrast, at 720p, successfully reestablishes the pose. These resolution-dependent difficulties are observed at multiple instances, with tracking proving to be more stable at higher resolutions. The higher mean translation errors for the measurement with 720p are due to the instances where the pose is lost but reestablished afterwards.



(a) 480p. (b) (c) (d)

(e) 720p. (f) (g) (h)

(i) 480p. (j) (k) (l)

(m) 720p (n) (o) (p)

Figure 4.8: Dynamic eye-in-hand pose tracking of the objects at different resolutions.

Table 4.3: Positional accuracies of dynamic pose tracking for different resolutions.

		480p			720p		
		x	y	z	x	y	z
Block	Mean Error [cm]	0.57	1.40	1.09	0.22	0.11	0.32
	Standard Deviation [cm]	0.66	2.52	1.93	0.15	0.09	0.31
Bottle	Mean Error [cm]	0.40	0.62	0.49	0.48	0.27	0.82
	Standard Deviation [cm]	0.41	0.83	0.49	0.33	0.16	1.30

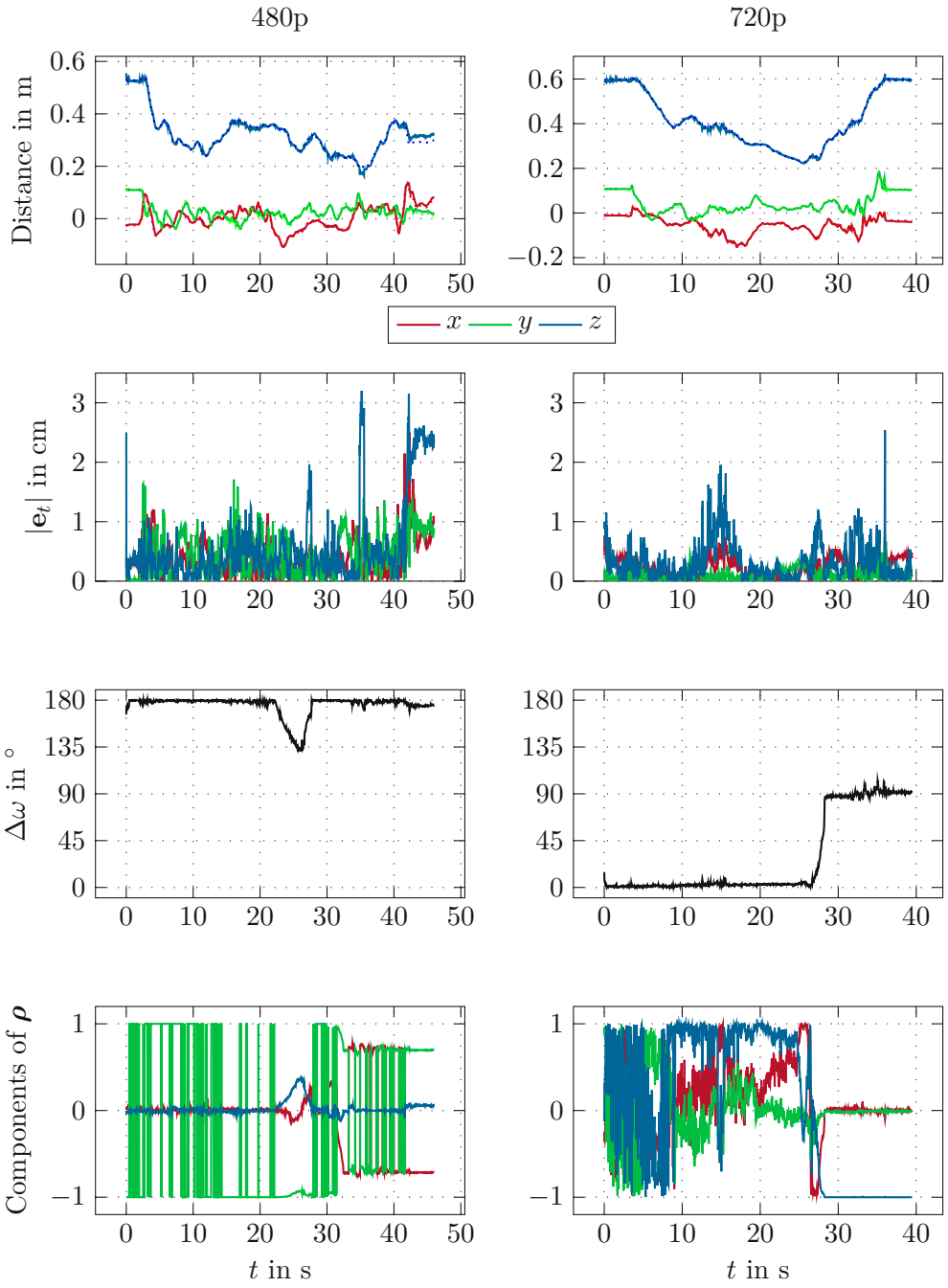


Figure 4.9: Measurements of dynamic eye-in-hand tracking of the block at different camera resolutions. The components of the ground truth translation are plotted with dotted lines.

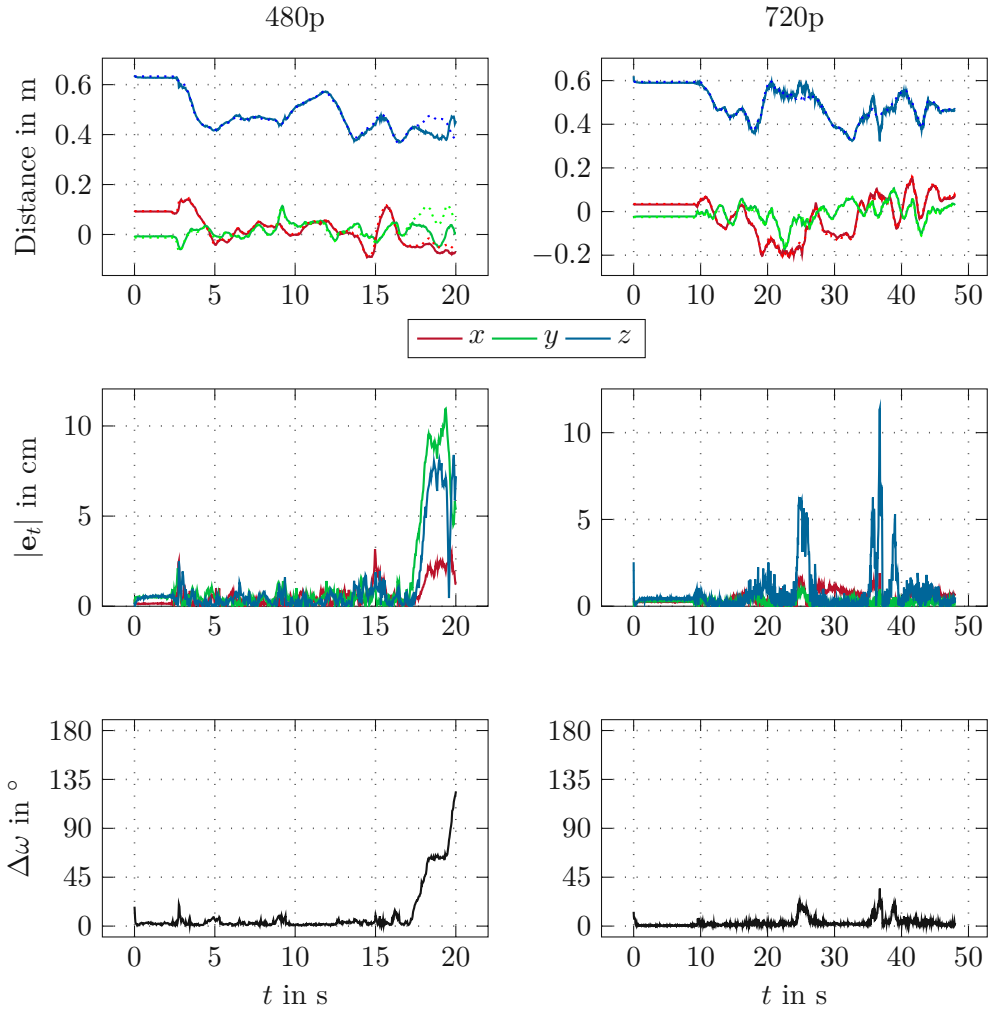
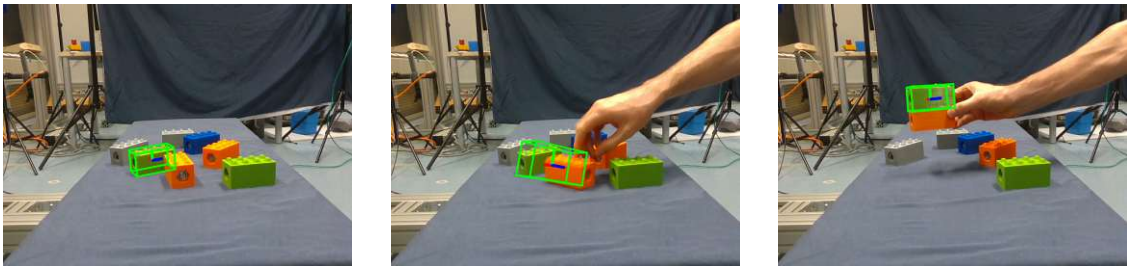


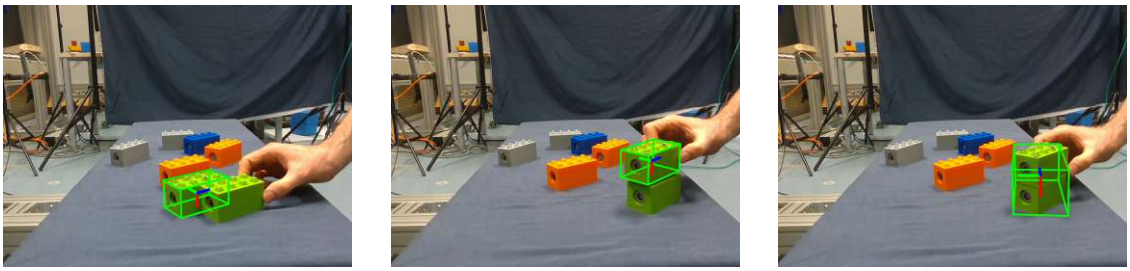
Figure 4.10: Measurements of dynamic eye-in-hand tracking of the cleanser bottle at different camera resolutions. The components of the ground truth translation are plotted with dotted lines.



(a) Initial Pose.

(b) Another block disrupts the tracking.

(c) Stacking blocks with different colours.



(d) Partial occlusion with block of same colour.

(e) Before stacking blocks with same colour.

(f) Tracking lost after stacking.

Figure 4.11: Tracking a block in an environment cluttered with similar objects.

4.2.4 Multiple Blocks

Figure 4.11 presents frames captured at a resolution of 480p during the eye-to-hand pose tracking of a block in an environment cluttered with identical blocks of different colours. The corresponding measurement results are plotted in Figure 4.12. Despite partial occlusion by a differently coloured block, the initial pose estimation (Figure 4.11a) remains accurate. The tracker performs reliably until the tracked block becomes excessively occluded by another object (Figure 4.11b).

The tracker successfully re-establishes the pose and continues tracking when the block is stacked on a differently coloured cube (Figure 4.11c) and even when partially occluded by a block of the same colour (Figure 4.11d). However, tracking fails when the block is stacked onto another block of the same colour (Figure 4.11f). As observed in previous experiments, the symmetry of the object results in an abrupt orientation flip, which is evident in the graphs depicting the rotational error.

4.2.5 Inference Times

For all measurements presented in this chapter, the average inference time of the initial pose estimation is 1.04 s. As previously discussed, this relatively high latency is primarily attributed to the use of computationally intensive ViTs and CNNs, which require execution on a GPU.

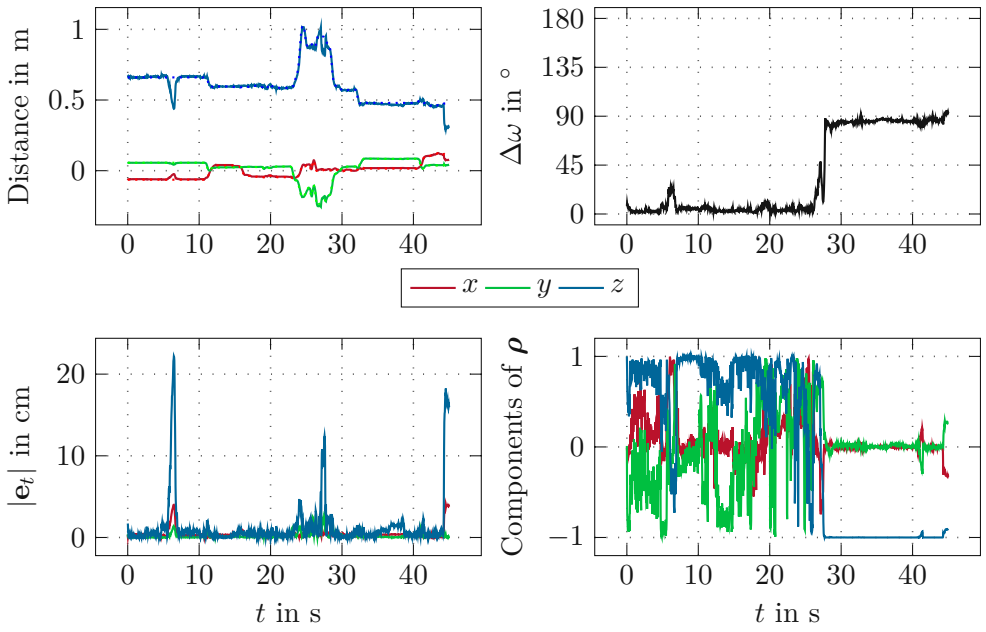


Figure 4.12: Dynamic eye-to-hand measurement results of the block in an environment cluttered with similar objects.

In contrast, the subsequent pose refinement performed by the tracker achieves a significantly lower average inference time of 20.4 ms, corresponding to a potential update rate of 49.01 Hz. Since the refinement step is based on classical computer vision techniques, it is executed efficiently on the CPU while maintaining a high update frequency.

4.3 Grasping an Object with a Robotic Gripper

In this experiment, the tracker is used to follow and grasp a block using the Robotiq 2-Finger Gripper 85, which is mounted on a 7-DoF KUKA LBR iiwa robotic arm. Figure 4.13 illustrates the experimental setup, showing the various reference frames and the transformations between them. The robotic gripper serves as the end-effector $\mathcal{E}$, while the robot is equipped with a fixed base frame $\mathcal{R}$.

The robot measures its joint angles, represented as the configuration vector $\mathbf{q}(t) \in \mathbb{R}^7$, at a sampling rate of 8 kHz, where t denotes the time. The pose $\mathbf{H}^{\mathcal{C}}_{\mathcal{W}}(t)$ of the camera $\mathcal{C}$ is tracked by a motion capture system relative to a fixed world frame $\mathcal{W}$. The transformation $\mathbf{H}^{\mathcal{R}}_{\mathcal{W}}$ between the robot base frame and the world frame remains constant and known. The tracking pipeline estimates the object's pose relative to the camera, denoted as $\mathbf{H}^{\mathcal{O}}_{\mathcal{C}}(t)$, which is then transformed into the robot's reference frame using

$$\mathbf{H}^{\mathcal{O}}_{\mathcal{R}}(t) = \left(\mathbf{H}^{\mathcal{R}}_{\mathcal{W}}\right)^{-1} \mathbf{H}^{\mathcal{C}}_{\mathcal{W}}(t) \mathbf{H}^{\mathcal{O}}_{\mathcal{C}}(t) \tag{4.7}$$

The desired transformation $\mathbf{H}^{\mathcal{O}}_{\mathcal{E}}(t)$ between the object and the end effector is typically

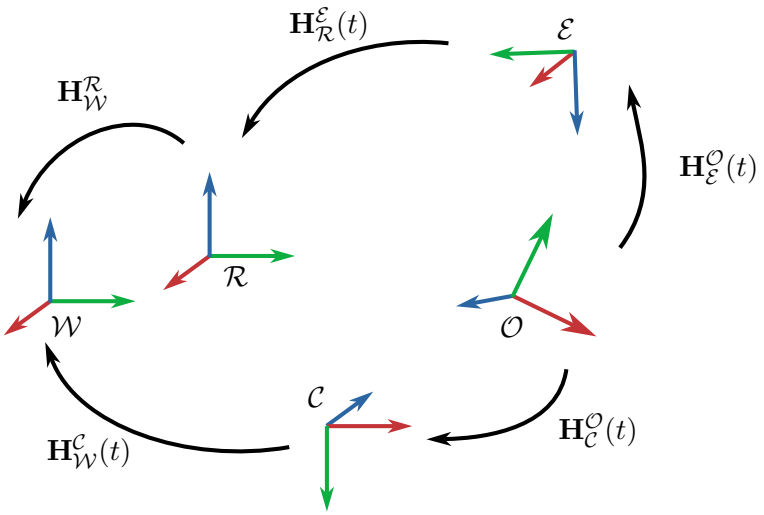


Figure 4.13: Setup for the grasping experiment.

constant and known when following the object, ensuring that the gripper hovers slightly above the object. When the user prompts the robot to grasp the object, the gripper lowers by reducing the z-component of the translation vector in $\mathbf{H}^{\mathcal{O}}_{\mathcal{E}}(t)$, and grasps it. This is modeled with

$$\mathbf{x}^{\mathcal{O}}_{\mathcal{E}}(t) = \begin{cases} \begin{bmatrix} 0 & 0 & d \end{bmatrix}^{\mathrm{T}}, & 0 < t < t_g \\ \begin{bmatrix} 0 & 0 & d - \Delta d \end{bmatrix}^{\mathrm{T}}, & t \geq t_g \end{cases}. \tag{4.8}$$

Here, $t_g > 0$ is the time at which the user prompts the robot to completely approach the object and close the gripper, and $0 < \Delta d < d$.

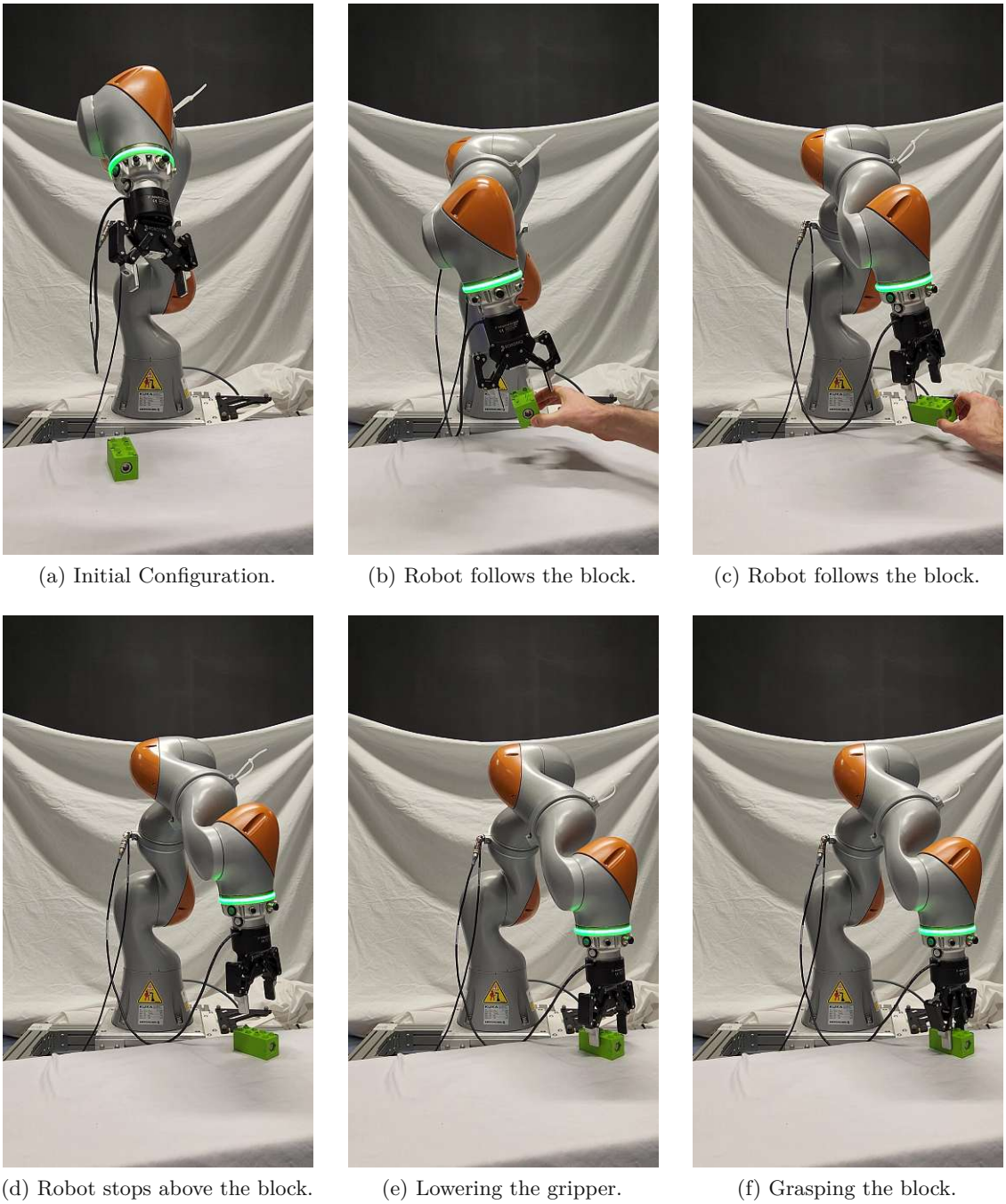
The block's frame $\mathcal{O}$ is defined as in Figure 4.1. The objective is to align the gripper's y-axis with the $\pm z$-axis of the block. To prevent tracking errors due to orientation flipping, an algorithm dynamically adjusts $\mathbf{R}^{\mathcal{O}}_{\mathcal{E}}(t)$.

The desired end effector pose relative to the robot frame to grasp the object at time t is computed with

$$\mathbf{H}^{\mathcal{E}}_{\mathcal{R},d}(t) = \mathbf{H}^{\mathcal{O}}_{\mathcal{R}}(t)\left(\mathbf{H}^{\mathcal{O}}_{\mathcal{E}}(t)(t)\right)^{-1} \tag{4.9}$$

and continuously fed into the task-space path-following controller described in [42].

Figure 4.14 illustrates the robot successfully tracking the object, adjusting its position and orientation in real-time, and finally grasping it when prompted by the user.



(a) Initial Configuration. (b) Robot follows the block. (c) Robot follows the block.

(d) Robot stops above the block. (e) Lowering the gripper. (f) Grasping the block.

Figure 4.14: Tracking and grasping a block with a robotic gripper.

5 Conclusion and Outlook

This thesis presented a model-based pipeline capable of tracking the pose of various objects with an update rate of up to 50 Hz using only an RGB camera. The pipeline's flexibility in selecting the object of interest is achieved by integrating the deep-learning-based methods Segment Anything [32] for object segmentation and GigaPose [15] for initial pose estimation. The high update rate is primarily attributed to the region-based tracking approach introduced in [24], which efficiently exploits classical computer vision techniques to differentiate between foreground and background to consecutively perform probabilistic pose refinement steps.

The pipeline demonstrates robust performance for asymmetric objects and stable translational estimates for symmetrical objects across various scenarios, including dynamic eye-in-hand pose tracking and eye-to-hand tracking in cluttered environments. Additionally, the pipeline has been successfully applied to a robotic grasping experiment, underscoring its potential for future applications in robotic manipulation.

Despite these strengths, the tracker exhibits limitations in reliably estimating the orientation of symmetrical objects, as reflected in the experimental results. Furthermore, the pipeline struggles in environments where the object's colour closely matches the background, a common challenge in region-based tracking methods. Another notable limitation is its inability to handle highly textured objects, which limits its applicability to more complex scenarios.

To address the latter issue, the integration of a texture modality could improve the robustness of the pipeline. In [27], the authors proposed combining their sparse region model with a modality based on keypoint descriptors, which could enhance the pipeline's performance in texture-rich environments. However, attempts to integrate this modality into the real-time pipeline were hindered by implementation flaws in the code provided by the authors, despite its promising results in the original framework.

Further improvements could be achieved by incorporating depth information, which would reduce the reliance on color contrasts between foreground and background. While this would require an additional sensor and likely increase computational complexity, depth information could improve the accuracy of pose estimation, especially in challenging lighting conditions or cluttered environments. Another promising direction is to enhance the refinement algorithm with a lightweight neural network, combining the efficiency of classical methods with the adaptability of deep learning-based approaches. To avoid abrupt changes in the estimated pose, additional constraints during the optimization process could improve the tracker's stability.

Future work could also focus on extending the pipeline to support multi-object tracking, enabling simultaneous pose estimation of multiple objects within the scene. Additionally, implementing a robust recovery mechanism to automatically reinitialize the tracker when the pose is lost would further enhance its reliability in real-world applications.

Bibliography

[1] D. Tish, N. King, and N. Cote, "Highly accessible platform technologies for vision-guided, closed-loop robotic assembly of unitized enclosure systems," *Construction Robotics*, vol. 4, no. 1–2, pp. 19–29, May 2020. DOI: 10.1007/s41693-020-00030-z.

[2] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, Aug. 2020. DOI: 10.1007/s10462-020-09888-5.

[3] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *Robotics: Science and Systems XIV*, ser. RSS2018, Robotics: Science and Systems Foundation, Jun. 2018. DOI: 10.15607/rss.2018.xiv.021.

[4] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, no. 2–3, pp. 183–201, Jun. 2019. DOI: 10.1177/0278364919859066.

[5] A. Trabelsi, M. Chaabane, N. Blanchard, and R. Beveridge, "A pose proposal and refinement network for better 6d object pose estimation," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Jan. 2021. DOI: 10.1109/wacv48630.2021.00243.

[6] Y. Xu, K.-Y. Lin, G. Zhang, X. Wang, and H. Li, "Rnnpose: 6-dof object pose estimation via recurrent correspondence field estimation and pose optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 7, pp. 4669–4683, Jul. 2024. DOI: 10.1109/tpami.2024.3360181.

[7] T. Hodaň *et al.*, "Bop: Benchmark for 6d object pose estimation," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 19–35. DOI: 10.1007/978-3-030-01249-6_2.

[8] T. Hodaň *et al.*, "Bop challenge 2020 on 6d object localization," in *Computer Vision – ECCV 2020 Workshops*. Springer International Publishing, 2020, pp. 577–594. DOI: 10.1007/978-3-030-66096-3_39.

[9] M. Sundermeyer *et al.*, "Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jun. 2023, pp. 2785–2794. DOI: 10.1109/cvprw59228.2023.00279.

[10] T. Hodan *et al.*, "Bop challenge 2023 on detection, segmentation and pose estimation of seen and unseen rigid objects," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jun. 2024, pp. 5610–5619. DOI: 10.1109/cvprw63382.2024.00570.

[11] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2010, pp. 998–1005. DOI: 10.1109/cvpr.2010.5540108.

[12] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 16 606–16 616. DOI: 10.1109/cvpr46437.2021.01634.

[13] Y. Hai, R. Song, J. Li, M. Salzmann, and Y. Hu, "Rigidity-aware detection for 6d object pose estimation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 8927–8936. DOI: 10.1109/cvpr52729.2023.00862.

[14] X. Liu *et al.*, "Gdrnpp: A geometry-guided and fully learning-based object pose estimator," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–17, 2025. DOI: 10.1109/tpami.2025.3553485.

[15] V. N. Nguyen, T. Groueix, M. Salzmann, and V. Lepetit, "Gigapose: Fast and robust novel object pose estimation via one correspondence," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2024, pp. 9903–9913. DOI: 10.1109/cvpr52733.2024.00945.

[16] A. Dosovitskiy *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929.

[17] Y. Labbé *et al.*, "Megapose: 6d pose estimation of novel objects via render & compare," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, Dec. 2023, pp. 715–725.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. DOI: 10.1023/b:visi.0000029664.99615.94.

[19] 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, Nov. 2011, pp. 2564–2571. DOI: 10.1109/iccv.2011.6126544.

[20] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986. DOI: 10.1109/tpami.1986.4767851.

[21] Z. Jin-Yu, C. Yan, and H. Xian-Xiang, "Edge detection of images based on improved sobel operator and genetic algorithms," in *2009 International Conference on Image Analysis and Signal Processing*, IEEE, 2009, pp. 31–35. DOI: 10.1109/iasp.2009.5054605.

[22] M. Lourakis and X. Zabulis, "Model-based pose estimation for rigid objects," in *Computer Vision Systems*. Springer Berlin Heidelberg, 2013, pp. 83–92. DOI: 10.1007/978-3-642-39402-7_9.

[23] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, Jul. 2002. DOI: 10.1109/tpami.2002.1017620.

[24] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer, "A sparse gaussian approach to region-based 6dof object tracking," in *Computer Vision – ACCV 2020*. Springer International Publishing, 2021, pp. 666–682. DOI: 10.1007/978-3-030-69532-3_40.

[25] M. Stoiber, M. Sundermeyer, and R. Triebel, "Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2022, pp. 6845–6855. DOI: 10.1109/cvpr52688.2022.00673.

[26] M. Stoiber, M. Elsayed, A. E. Reichert, F. Steidle, D. Lee, and R. Triebel, "Fusing visual appearance and geometry for multi-modality 6dof object tracking," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2023. DOI: 10.1109/iros55552.2023.10341961.

[27] M. Stoiber, "Closing the loop: 3d object tracking for advanced robotic manipulation," Ph.D. dissertation, Technische Universität München, Dec. 2023.

[28] X. Xu, S. Xu, L. Jin, and E. Song, "Characteristic analysis of otsu threshold and its applications," *Pattern Recognition Letters*, vol. 32, no. 7, pp. 956–961, May 2011. DOI: 10.1016/j.patrec.2011.01.021.

[29] R. Kohler, "A segmentation system based on thresholding," *Computer Graphics and Image Processing*, vol. 15, no. 4, pp. 319–338, Apr. 1981. DOI: 10.1016/s0146-664x(81)80015-9.

[30] D.-Y. Huang and C.-H. Wang, "Optimal multi-level thresholding using a two-stage otsu optimization approach," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 275–284, Feb. 2009. DOI: 10.1016/j.patrec.2008.10.003.

[31] H. K. Cheng and A. G. Schwing, "Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model," in *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 640–658. DOI: 10.1007/978-3-031-19815-1_37.

[32] A. Kirillov *et al.*, "Segment anything," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2023. DOI: 10.1109/iccv51070.2023.00371.

[33] K. Waldron and J. Schmiedeler, "Kinematics," in *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008, pp. 9–33. DOI: 10.1007/978-3-540-30301-5_2.

[34] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer International Publishing, 2022, ISBN: 9783030343729. DOI: 10.1007/978-3-030-34372-9.

[35] V. A. Prisacariu and I. D. Reid, "Pwp3d: Real-time segmentation and tracking of 3d objects," *International Journal of Computer Vision*, vol. 98, no. 3, pp. 335–354, Jan. 2012. DOI: 10.1007/s11263-011-0514-3.

[36] C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts*. Springer International Publishing, 2024. DOI: 10.1007/978-3-031-45468-4.

[37] A. Vaswani *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762.

[38] Open Source Robotics Foundation, *RViz - 3D Visualization Tool for ROS*, https://github.com/ros2/rviz, Accessed: 2025-05-13, 2017.

[39] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep. 2015. DOI: 10.1109/mra.2015.2448951.

[40] G. Ebmer *et al.*, "Real-time 6-dof pose estimation by an event-based camera using active led markers," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Jan. 2024, pp. 8122–8131. DOI: 10.1109/wacv57701.2024.00795.

[41] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, Mar. 2016. DOI: 10.1016/j.patcog.2015.09.023.

[42] G. Ebmer, *Robotische Assemblierung von Betonringsegmenten im Tunnelbau*, German. Wien: Technische Universität Wien, 2021, Diplomarbeit. DOI: 10.34726/hss.2021.74022.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct - Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Für die Erstellung dieser Arbeit wurde Künstliche Intelligenz (KI) unterstützend eingesetzt. Konkret kamen KI-basierte Werkzeuge ausschließlich zur Korrektur von Rechtschreibung, Grammatik sowie zur sprachlichen Überarbeitung einzelner Formulierungen zum Einsatz. Die inhaltliche Ausarbeitung, Argumentation und Struktur der Arbeit stammen vollständig vom Verfasser bzw. der Verfasserin. Die Verwendung erfolgte im Einklang mit den Richtlinien der TU Wien zur Nutzung von KI im Studium. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, May 2025

Johannes König